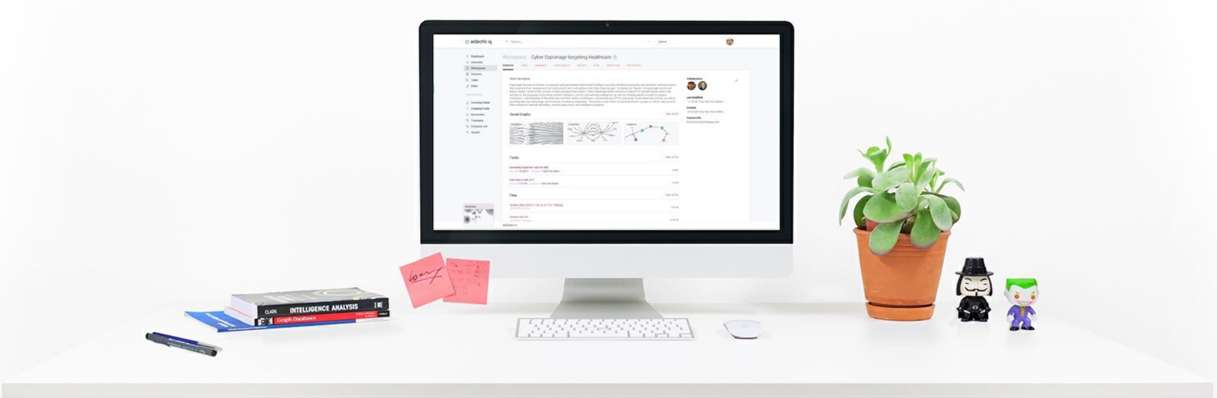
EclecticIQ Platform user guide

Ingest data with incoming feeds and enrichers — 1/4

Last generated: October 20, 2017



Last generated on Oct 20, 2017

Table of contents

Table of contents 2
User guide to EclecticIQ Platform 6
  Scope 6
  Goal 6
  Audience 6
  Feedback 6
About enrichment 8
  Ingestion 8
  Deduplication 9
  Enrichment 11
    Filtering and enriching 12
    idref resolution at entity level 13
    idref resolution at nested object level 14
      Example of an empty placeholder entity 17
    Saving data 17
      Saving data URIs and raw artifacts 18
Configure incoming feeds 20
  Configure the general options 20
  Set a schedule 21
  Set a TLP override 22
  Set half-life values 22
  Save options 23
  Configure transport and content for specific incoming feeds 23
Start and stop feeds 24
  Manually start a feed 24
  Manually stop a feed 25
    Suspend and disable a running feed 25
    Stop and terminate a running feed 26
Delete and purge feeds 27
  Delete an incoming feed 27
  Purge an incoming feed 27
Configure IMAP email fetcher transport and content 29
  About IMAP email fetcher 29
  Configure the transport type 29
    Email body or attachment download known issues 30
Configure IMAP email attachment fetcher transport and content 31
  About IMAP email attachment fetcher 31
  Configure the transport type 31
    Email body or attachment download known issues 32
Configure FTP download transport and content 34
  About FTP download 34
  Configure the transport type 34
Configure HTTP download transport and content 36
  About HTTP download 36
  Configure the transport type 36
Configure mount point download transport and content 39
  About Mount point download 39
  Configure the transport type 39
Configure TAXII inbox transport and content 41
  About TAXII inbox 41
  Configure the transport type 41
  Push data to a TAXII inbox 42
Configure TAXII poll transport and content 45
  About TAXII poll 45
  Configure the transport type 45
Configure Anubis Cyberfeed transport and content 49
  About Anubis Cyberfeed 49
  Configure the transport type 50

Configure BFK transport and content 52
  About BFK API 52
  Configure the transport type 52
Configure Cisco Threat Grid Curated Feed transport and content 53
  About Cisco Threat Grid Curated Feed 53
  Configure the transport type 54
Configure Cisco Threat Grid Samples API transport and content 55
  About Cisco Threat Grid Samples API 55
  Configure the transport type 55
Configure Crowdstrike Falcon Intelligence Indicator transport and content 57
  About Crowdstrike Falcon Intelligence Indicator 57
  Configure the transport type 57
Configure Crowdstrike Falcon Intelligence Reports transport and content 59
  About Crowdstrike Falcon Intelligence Reports 59
  Configure the transport type 59
Configure CVE Search API transport and content 61
  About CVE Search API 61
  Configure the transport type 61
Configure FireEye iSIGHT Intelligence Report API transport and content 62
  About FireEye iSIGHT Intelligence Report API 62
  Configure the transport type 62
Configure Group-IB transport and content 64
  About Group-IB 64
  Configure the transport type 64
Configure Intel 471 transport and content types 67
  About Intel 471 API 67
  Configure the transport type 67
Configure PhishMe Intelligence transport and content 68
  About PhishMe Intelligence 68
  Configure the transport type 68
Configure SpyCloud Breach API transport and content 70
  About SpyCloud Breach API 70
  Configure the transport type 70
Configure Threat Recon transport and content 71
  About Threat Recon JSON API 71
  Configure the transport type 71
Incoming feeds reference 72
  Available incoming feeds 72
  Content types 73
  Transport types 75
Configure enrichers 77
  Configure the general options 77
    Edit enricher tasks 78
  Configure enricher rules 79
    Add enricher rules 79
      Save options 80
    Edit enricher rules 80
    Delete enricher rules 81
  Configure specific enricher parameters 81
Enable and disable enrichers 83
  View enricher tasks 83
  Enable and disable enrichers 83
Run enrichers 85
  Enrich entities 85
    Enrich automatically 85
    Enrich manually 86
  Review enrichment observables 89
    Review enrichment observables on the graph 89
  Search for enrichment observables 92
Censys enricher 96

  Configure the Censys enricher parameters 96
    Query the Censys API 98
CIRCL IPs related to SSL certificate enricher 101
  Configure the CIRCL IPs related to SSL certificate enricher parameters 101
CIRCL SSL Certificate Fetcher enricher 103
  Configure the CIRCL SSL Certificate Fetcher enricher parameters 103
Cisco Threat Grid enricher 105
  Configure the Cisco Threat Grid enricher parameters 105
Crowdstrike Falcon Intelligence Indicator enricher 107
  Configure the Crowdstrike Falcon Intelligence Indicator enricher parameters 107
CVE Search enricher 109
  Configure the CVE Search enricher parameters 109
DomainTools Hosted Domains enricher 110
  Configure the DomainTools Hosted Domains enricher parameters 110
DomainTools Malicious Server Domains enricher 111
  Configure the DomainTools Malicious Server Domains enricher parameters 111
DomainTools Reputation enricher 113
  Configure the DomainTools Reputation enricher parameters 113
DomainTools Parsed Whois enricher 115
  Configure the DomainTools Parsed Whois enricher parameters 115
DomainTools Suspicious Domains enricher 117
  Configure the DomainTools Suspicious Domains enricher parameters 117
Elasticsearch sightings enricher 119
  Configure the Elasticsearch sightings enricher parameters 119
Farsight DNSDB enricher 122
  Configure the Farsight DNSDB enricher parameters 122
FireEye iSIGHT enricher 124
  Configure the FireEye iSIGHT enricher parameters 124
Flashpoint AggregINT enricher 126
  Configure the Flashpoint AggregINT enricher parameters 126
Flashpoint Blueprint enricher 128
  Configure the Flashpoint Blueprint enricher parameters 128
Flashpoint Thresher enricher 130
  Configure the Flashpoint Thresher enricher parameters 130
Fox-IT InTELL Portal enricher 132
  Configure the Fox-IT InTELL Portal enricher parameters 132
Intel 471 enricher 134
  Configure the Intel 471 enricher parameters 134
OpenDNS OpenResolve enricher 136
  Configure the OpenDNS OpenResolve enricher parameters 136
PassiveTotal IP/Domain enricher 137
  Configure the PassiveTotal IP/Domain enricher parameters 137
PassiveTotal Malware enricher 139
  Configure the PassiveTotal Malware enricher parameters 139
PassiveTotal Passive DNS enricher 141
  Configure the PassiveTotal Passive DNS enricher parameters 141
PassiveTotal Whois enricher 143
  Configure the PassiveTotal Whois enricher parameters 143
PyDat enricher 145
  Configure the PyDat enricher parameters 145
Recorded Future enricher 147
  Configure the Recorded Future enricher parameters 147
RIPEstat GeoIP enricher 149
  Configure the RIPEstat GeoIP enricher parameters 149
RIPEstat Whois enricher 150
  Configure the RIPEstat Whois enricher parameters 150
Shodan enricher 151
  Configure the Shodan enricher parameters 151
Splunk sightings enricher 153
  Configure the Splunk sightings enricher parameters 153

SpyCloud Breach Data enricher 156
  Configure the SpyCloud Breach Data enricher parameters 156
ThreatCrowd enricher 158
  Configure the ThreatCrowd enricher parameters 158
Unshorten-URL enricher 160
  Configure the Unshorten-URL enricher parameters 160
VirusTotal enricher 161
  Configure the VirusTotal enricher parameters 161
Create custom enrichers 163
  About extensions 163
  Create enricher extensions 163
  Prepare the boilerplate 164
    Edit the setup file 164
      Include dependencies 165
      Set the entry point 165
    Define the enricher initialization 166
  Build the enricher 167
    Import dependencies 169
    Set the UI schema definition 170
    Define the UI schema 171
      Field attributes 172
    Define the enricher behavior 173
    Define transport and content handlers 176
      Define the data provider 176
      Define the data transformer 178
  Package and deploy the enricher 181
    Restart processes 182
  Check that the enricher is registered 182
    Make an API call with HTTPie 182
    API call response 184
  Enable the enricher 184
  Initialize the enricher 185
    Create and run the fixtures 185
  Test the enricher 185
    Run extension validation 186
      Error handling 186
    Test the enricher with a test file 187
    Test the enricher through the platfom UI 189
  Disable extensions 193
Enrichers reference 194
  Enrichers 194
  Default enrichers 194
    Enricher types 196
    Enricher input 201
    Enricher output 203
    Enrichment data types 205

User guide to EclecticIQ Platform

This user guide helps you configure the main options of the platform, as well as familiarize with EclecticIQ Platform, so that you can start collecting and analyzing potential threats efficiently.

Scope

The user guide to EclecticIQ Platform aims at providing clear and to-the-point help to get you acquainted with the threat intelligence platform, so that you can configure it as needed, and you can use it to collect and analyze intelligence on potential threats, as well as share it and collaborate with other analysts.

Although it is not a complete reference manual, this guide shows end-users how they can use the platform and its rich feature set to collect data, to analyze and investigate potential threats, and to collaborate and share intelligence with other analysts.

Goal

Learn how to incorporate the platform in your daily workflow as a poweful tool to:

- Automate data ingestion
- View, edit, create, and delete platform entities
- Enrich entities with additional contextual details
- Analyze entities on the graph to identify potential threats and their relationships
- Search, filter, and slice data using rules
- Share your findings and collaborate

Audience

This document targets the following audience:

- Cyber threat intelligence analysts
- Cyber threat intelligence specialists

Feedback

No one reads manuals, ever. We know.
Yet, we strive to give you clear, concise, and complete documentation that helps you get stuff done neatly.

We are committed to crafting good documentation, because life is too short for bad doc.
We appreciate your comments, and we’d love to hear from you: if you have questions or suggestions, drop us a line and share your thoughts with us!

✋ The Product Team

Last generated on Oct 20, 2017

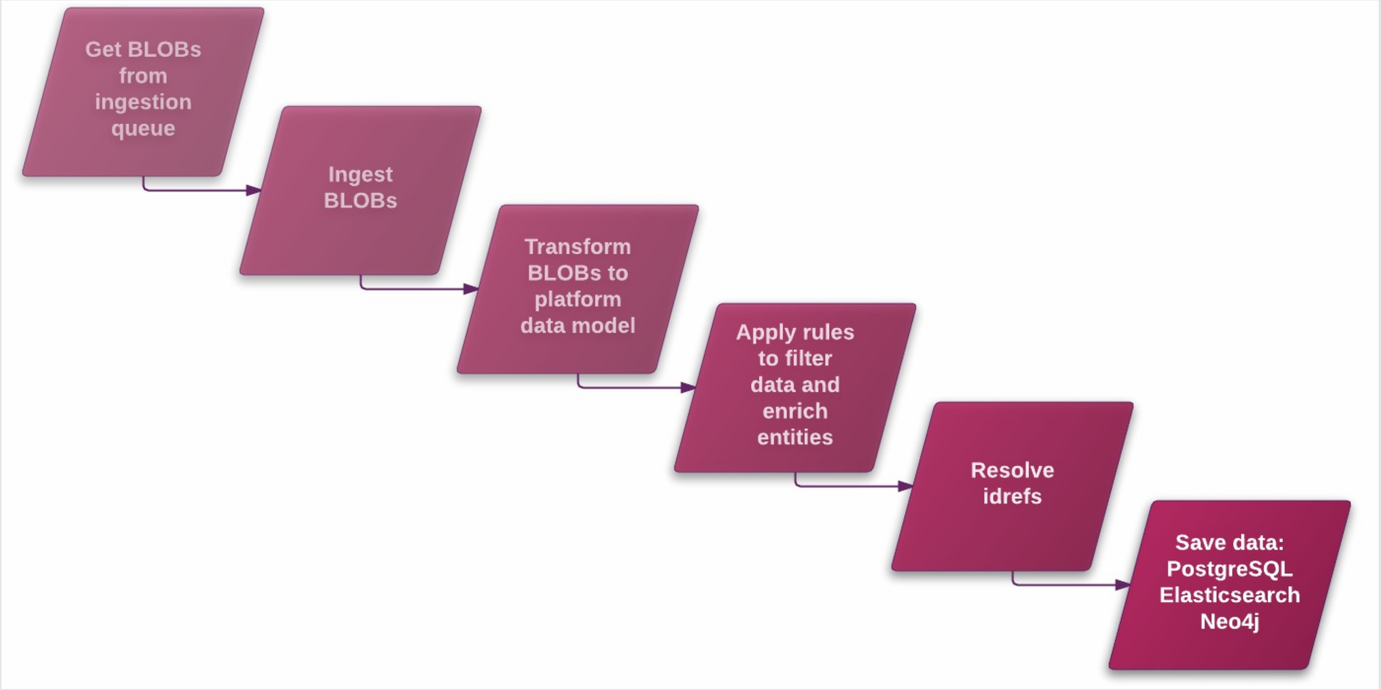
About enrichment

Enrichment augments existing cyber threat intelligence value by adding contextual information. Enrichers use rules and tasks to automatically enrich entities, so that you can explore a broader and more granular cyber threat intelligence landscape.

Before introducing enrichment, let’s have a look at how the platform ingests and processes incoming data.

Ingestion

Data ingestion into the platform is a multi-step process:


Get BLOBs from ingestion queue
Ingest BLOBs
Transform BLOBs to platform data model
Apply rules to filter data and enrich entities
Resolve idrefs
Save data: PostgreSQL Elasticsearch Neo4j

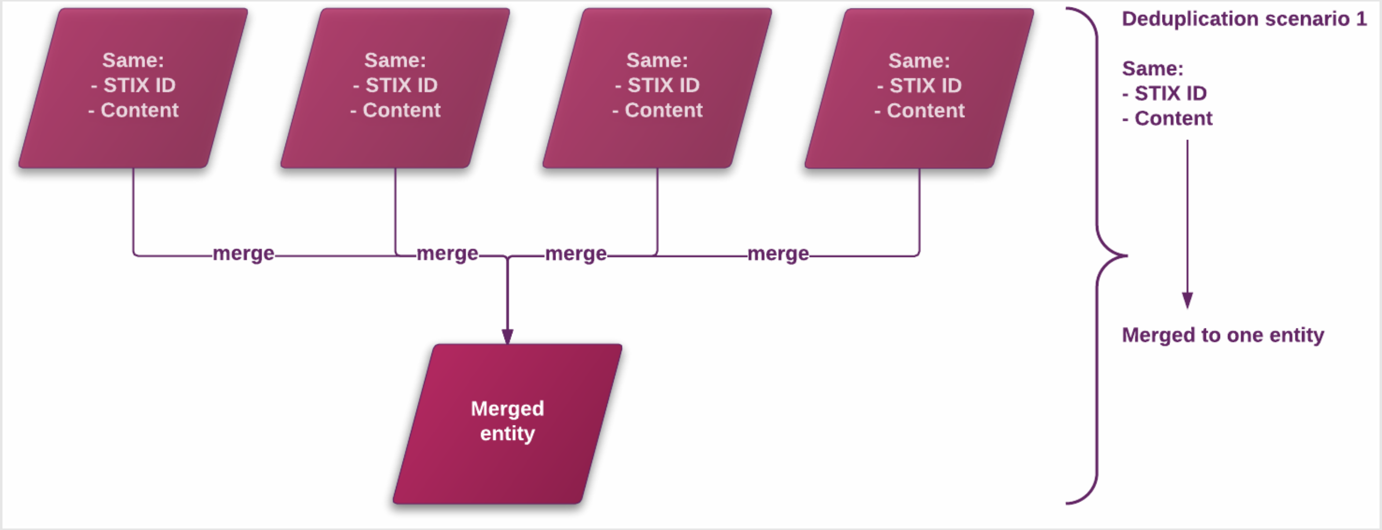
- Incoming data flows in and it is added to the ingestion queue.
- BLOBs are fetched from the ingestion queue to be processed.
- BLOBs are processed:
  - Data is deduplicated;
  - Data is normalized;
  - Data is transformed to the platform internal data model:
    - Entities
    - Observables
    - Relationships.
- Rules and filters enrich entities, create relationships, flag and tag entities, and so on.
- The platform tries to resolve ID references by looking for the data the IDs refer to.
- The ingested entities are saved to the databases.

Deduplication

When checking newly ingested entities vs. existing ones, entity-level deduplication handles the following scenarios:

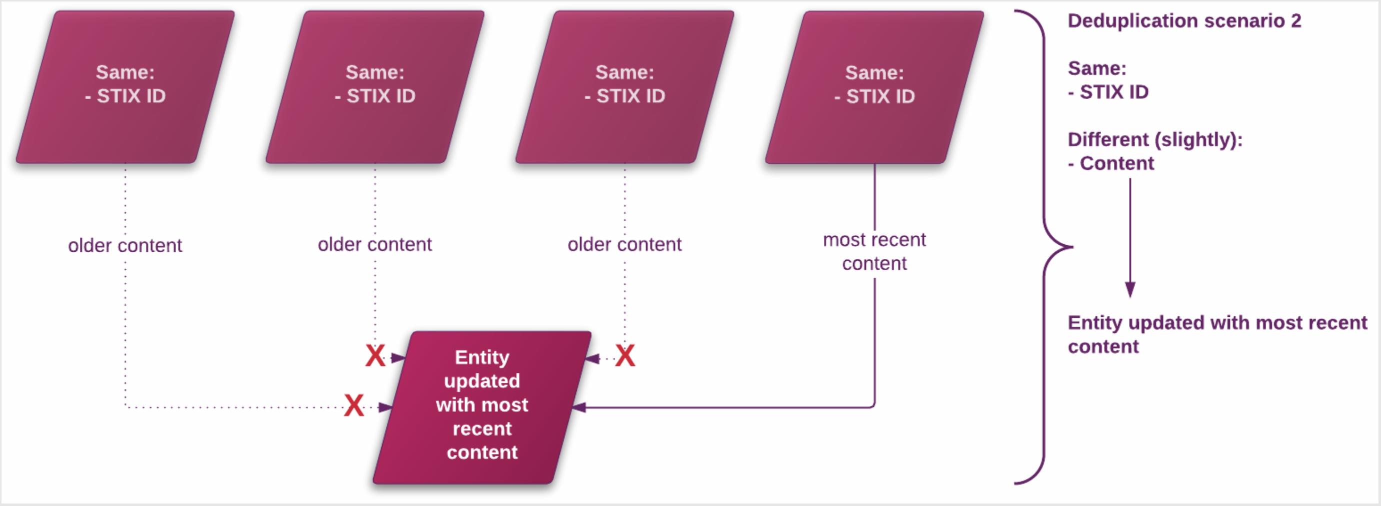
- Multiple entities sharing the *same STIX ID* and having *identical content* are handled like identical copies.
  In this case, identical copies of the same entity are merged to one entity.
- Multiple entities sharing the *same STIX ID* and having *slightly different content* are handled like versions of the same entity.
  In this case, the most recent content is used to update the existing entity. This avoids creating redundant copies of the same entity in the system.
- Multiple entities sharing the *same STIX ID* and having *identical content*, but *different timestamps* are handled like chronological versions of the same entity.
  In this case, the existing entity timestamp is updated to the most recent value without creating a new version of the entity.

Before deduplication	After deduplication
Multiple entities	They are merged to one entity
Same STIX ID	
Identical content	
(identical copies)	

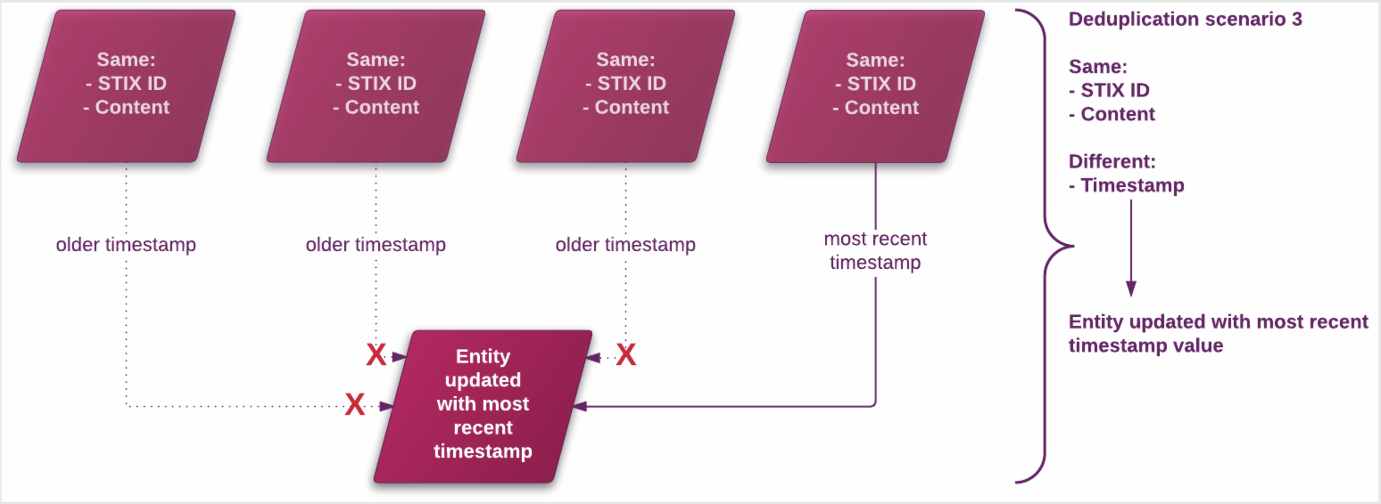


Before deduplication	After deduplication
Multiple entities	The existing entity is updated with the most recent content
Same STIX ID	
Slightly different content	

Before deduplication	**After deduplication**
(versions of the same entity)	



Before deduplication	**After deduplication**
Multiple entities	The existing entity timestamp is updated to the most recent value
Same STIX ID	
Identical content	
Different timestamps	
(chronological versions of the same entity)	

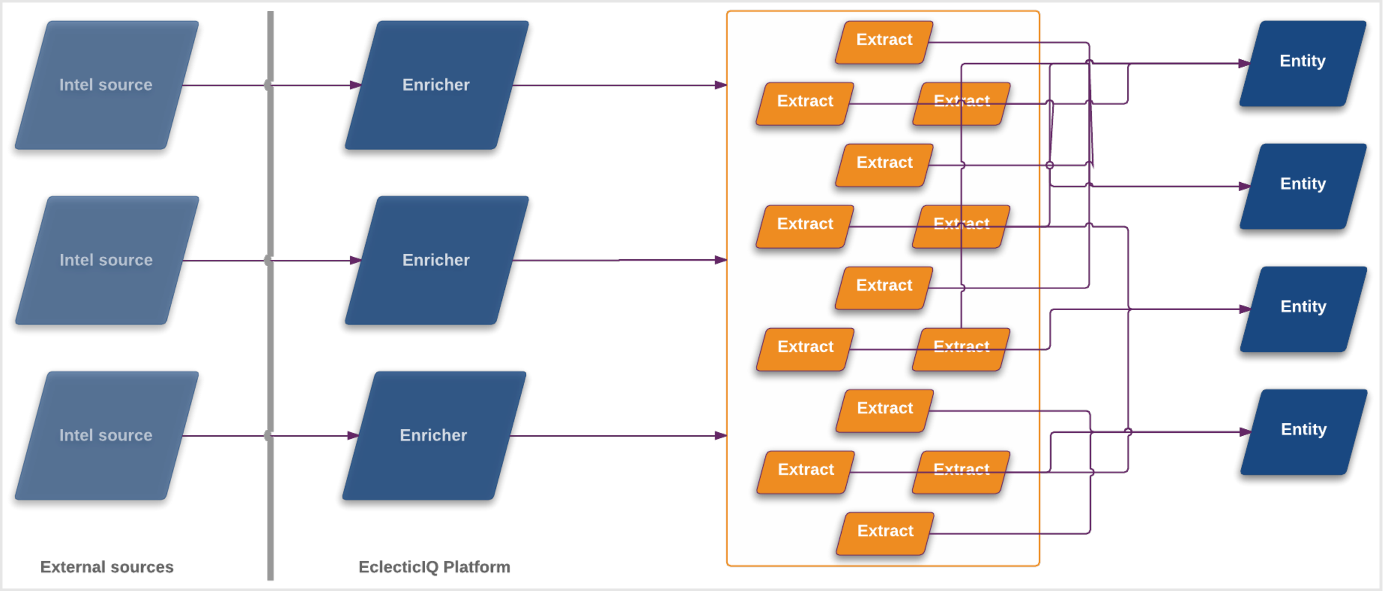


Enrichment

The platform can ingest cyber threat intelligence through incoming feeds, by manually uploading one or more files, or by creating an entity in the entity editor.

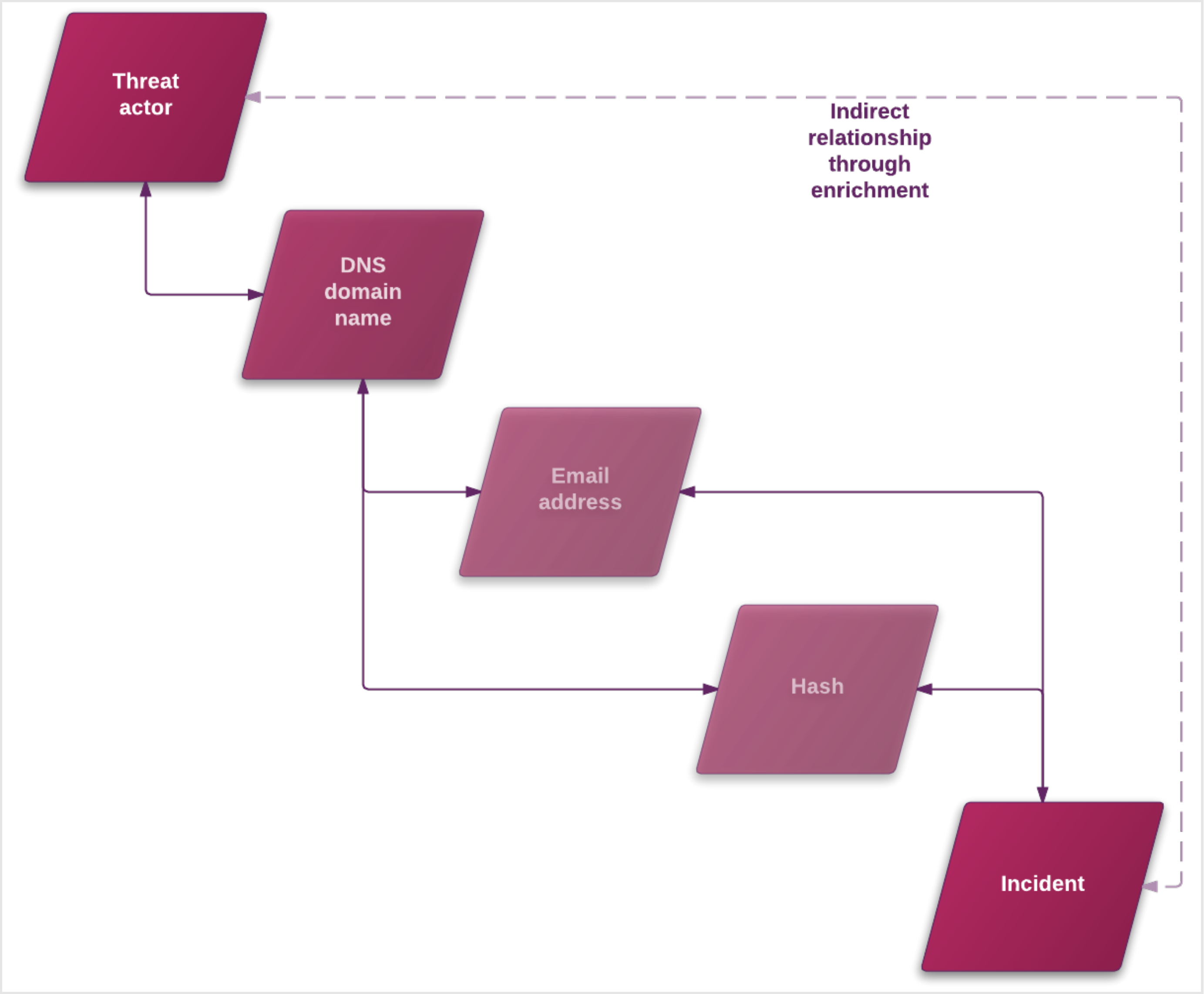
After ingesting and saving entities to the database, you can integrate the existing information with additional details. The extra information is raw data that augments the entity intelligence value by adding more context and meaning to it. The data is extracted from different sources such as feeds, reports, database searches, curated intel distribution lists, and so on.

The platform uses enrichers to fetch and extract the data. Enricher rules sift through the data to link it to relevant entities as enrichment observables.
This process does not alter core entity data: each bit of enriching information is saved to observables, which are related to entities.

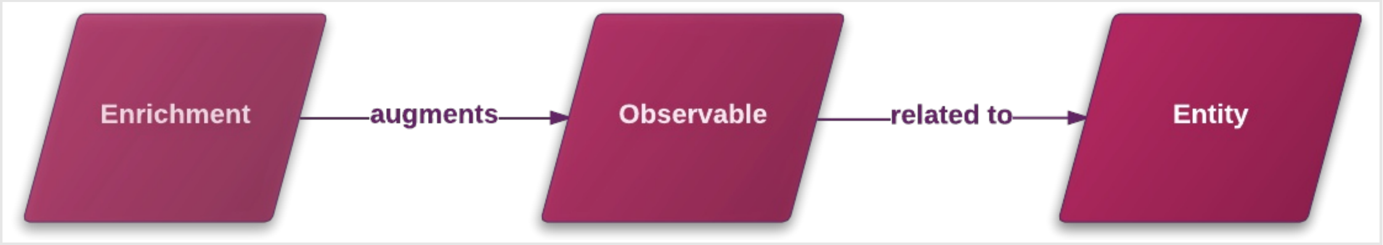


For example, let’s assume a scenario where an analyst is investigating a threat actor entity. The entity includes some observables, and one of them is a DNS domain name.
The analyst looks up the domain name by running it through a whois service. The lookup results include an email address. During the investigation, the analyst retrieves also a file hash related to the domain name. An examination reveals that the hash is related to an incident. Information about the incident includes the same email address detail the DNS domain name returned.
There is an indirect relationship between the threat actor and the incident that would not have been noticeable without extra context, which in this example is provided by the hash.

Enrichments help get a broader and sharper picture: by adding meaningful context, they help discover broader, indirect relationships that are not immediately visible.



Enrichments augment observables with raw data information related to entities:



Filtering and enriching

Data extraction produces observables from data retrieved in embedded CybOX objects. This process contribute to data fusion across the whole platform dataset.

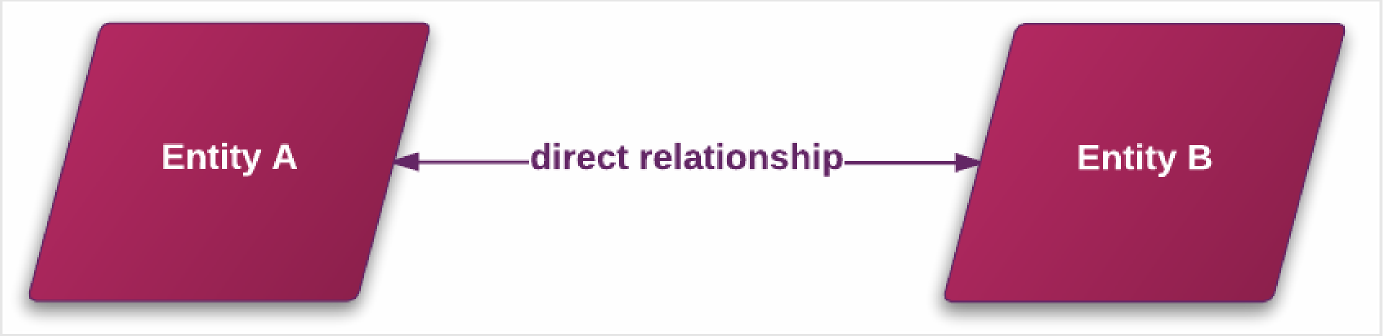
- Enrichment rules sift through data to augment it with enrichment observables, that is, observables, obtained from the available enrichment sources, and to exclude specific data based on the defined filtering rules.
- You can further refine the quality of the extracted data by applying ad-hoc rules to classify observables as safe or malicious.

STIX and CyBOX objects can both include placeholder references to external objects and data in the `idref` field.
By default, the platform attempts to resolve `idref`s at entity level and at nested object level by looking for the data matching the `idref` value.
If it finds matching data, it creates either a relationship between entities — entity-level, STIX `idref` resolution — or it replaces the `idref` placeholder value with the corresponding actual data — nested object-level, CybOX `idref` resolution.

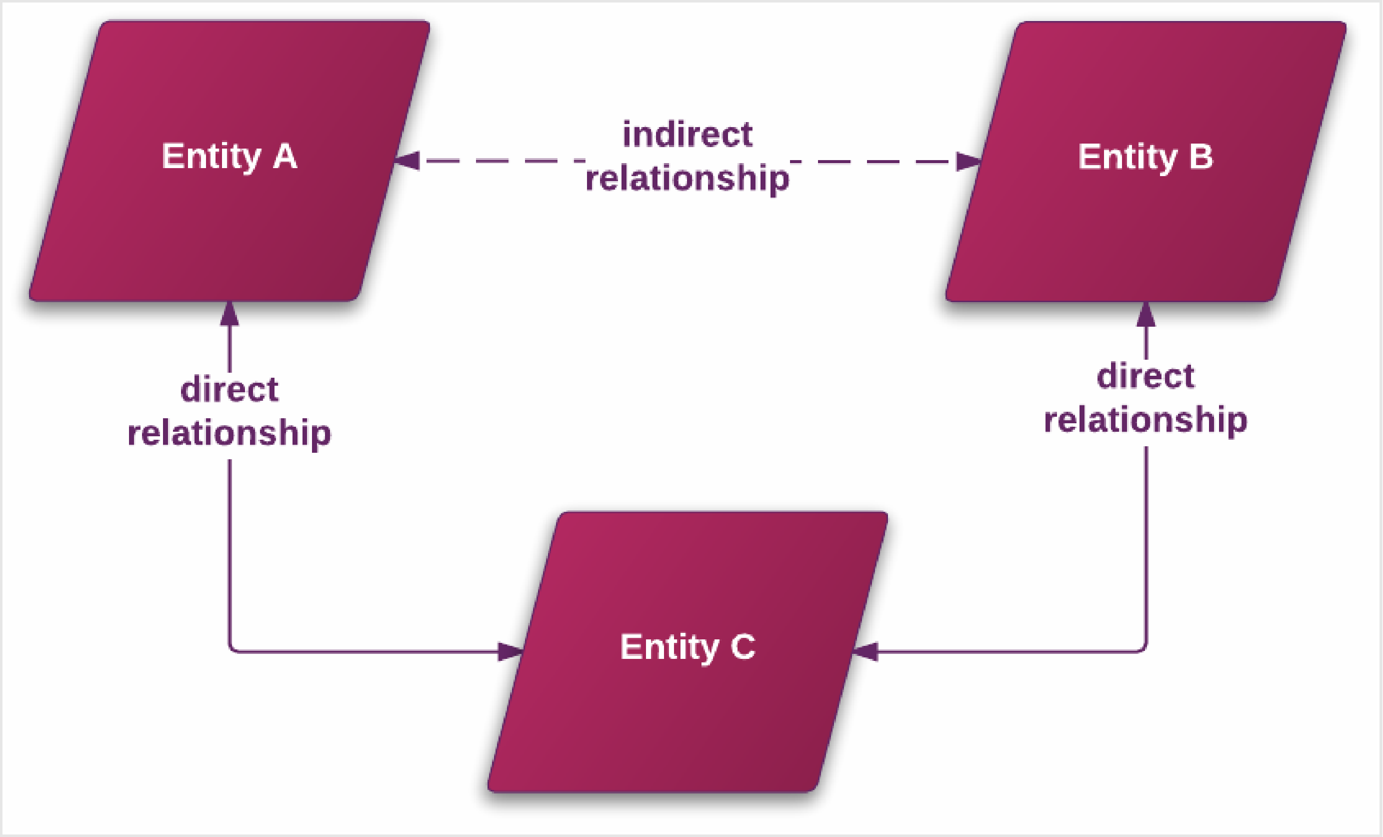
idref resolution at entity level

Entities and observables can reference external data through the STIX and CybOX `idref` field.
When the `idref --> id`reference is at entity level (STIX), the platform creates an entity relationship between the entities.

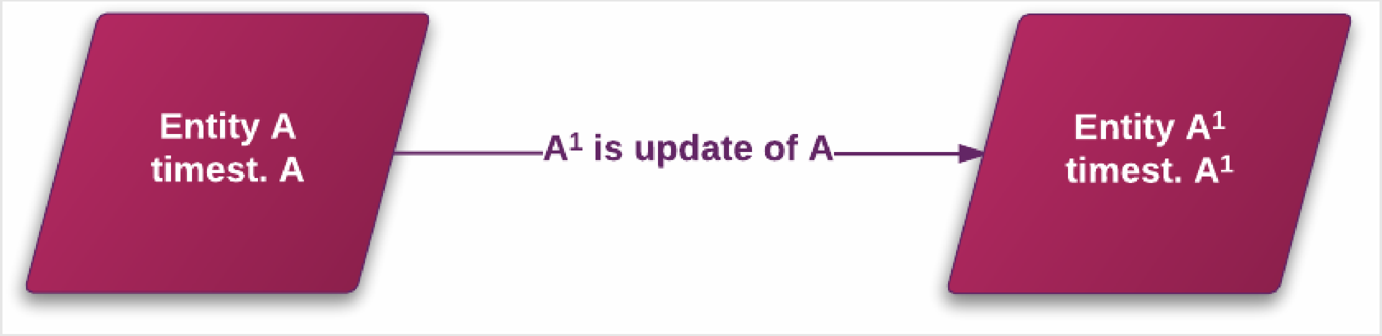
- When a STIX `idref` directly references an entity, the platform creates a direct relationship to associate the two entities.



- When a STIX `idref` indirectly references an entity, for example by establishing a relationship with the target entity through a connecting entity or an observable, the platform creates an indirect relationship to associate the two entities.



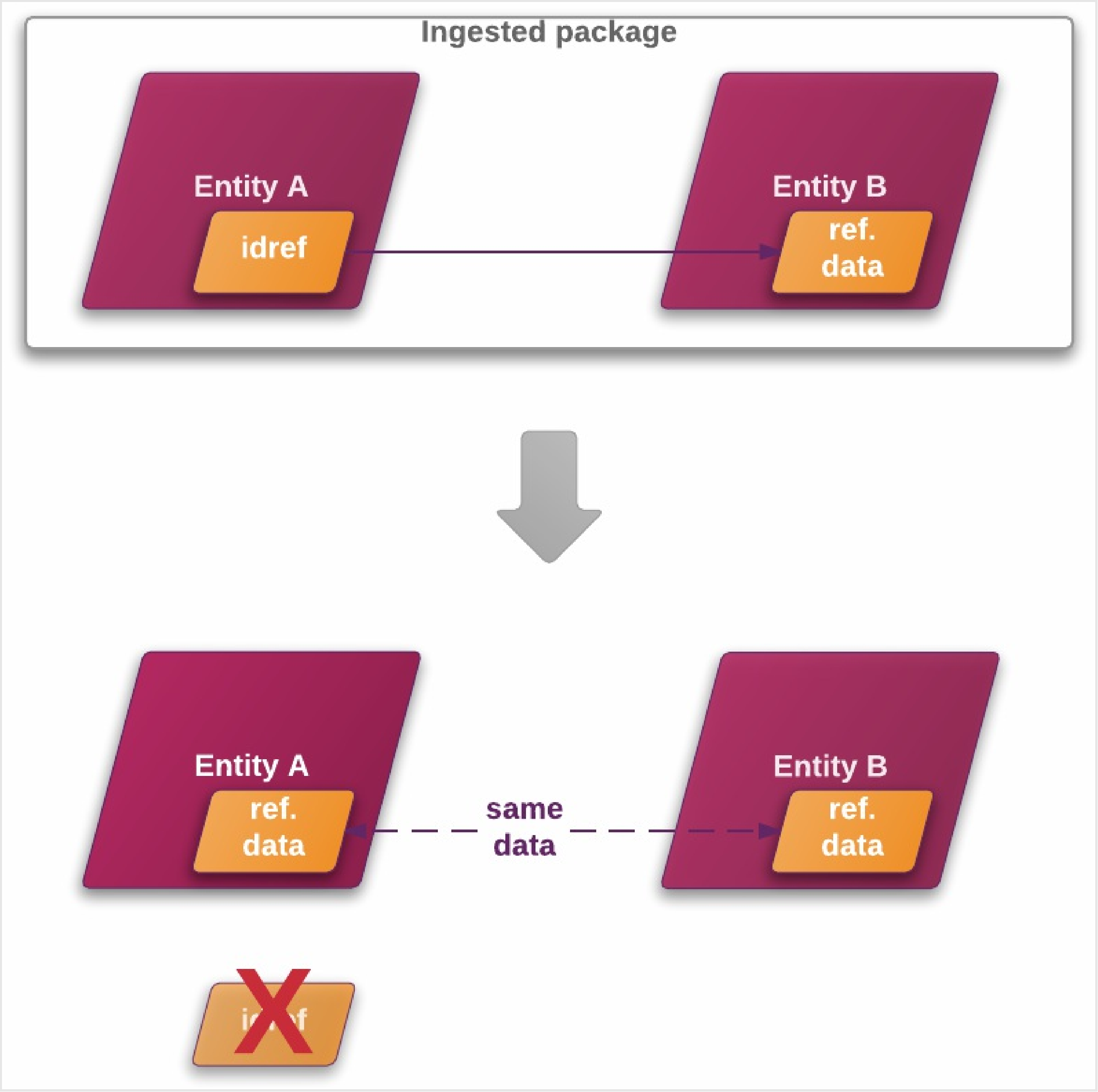
- When a STIX `idref` references an entity of the same type and content as the entity it belongs to, the only differences being either the timestamp, or the version reference values between the two entities, the platform processes the entity with the more recent timestamp or with the higher version value as an update of the other entity.



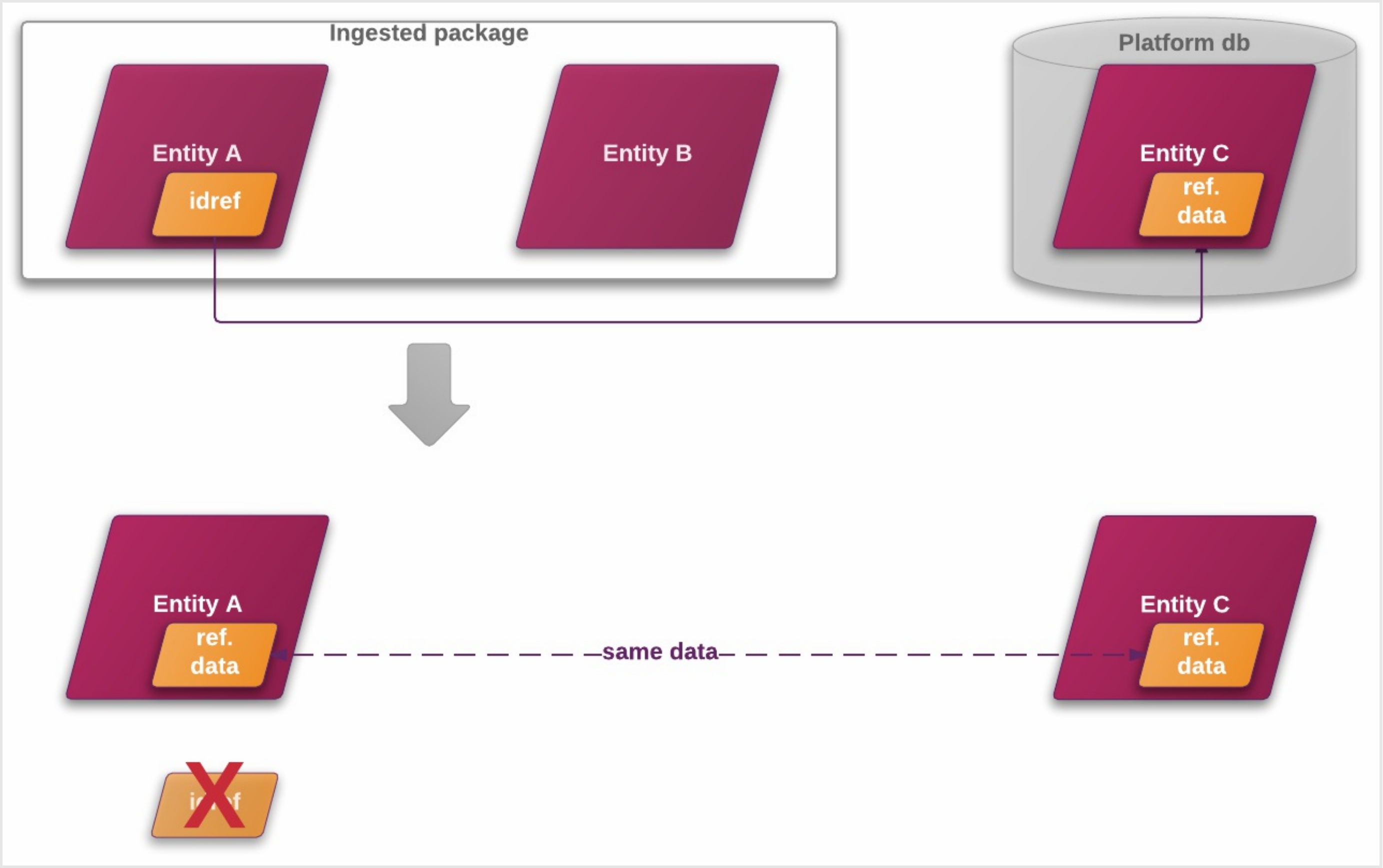
idref resolution at nested object level

When the `idref --> id`reference is at nested object level (CybOX), that is, when an entity includes an embedded CybOX observable object with an `idref`, the platform attempts to resolve the CybOX `idref` by looking for the referenced data:

- If the CybOX `idref` references data available inside the same ingested package or data stored in the platform database, the newly ingested `idref` is removed and it is replaced with the existing referenced data.

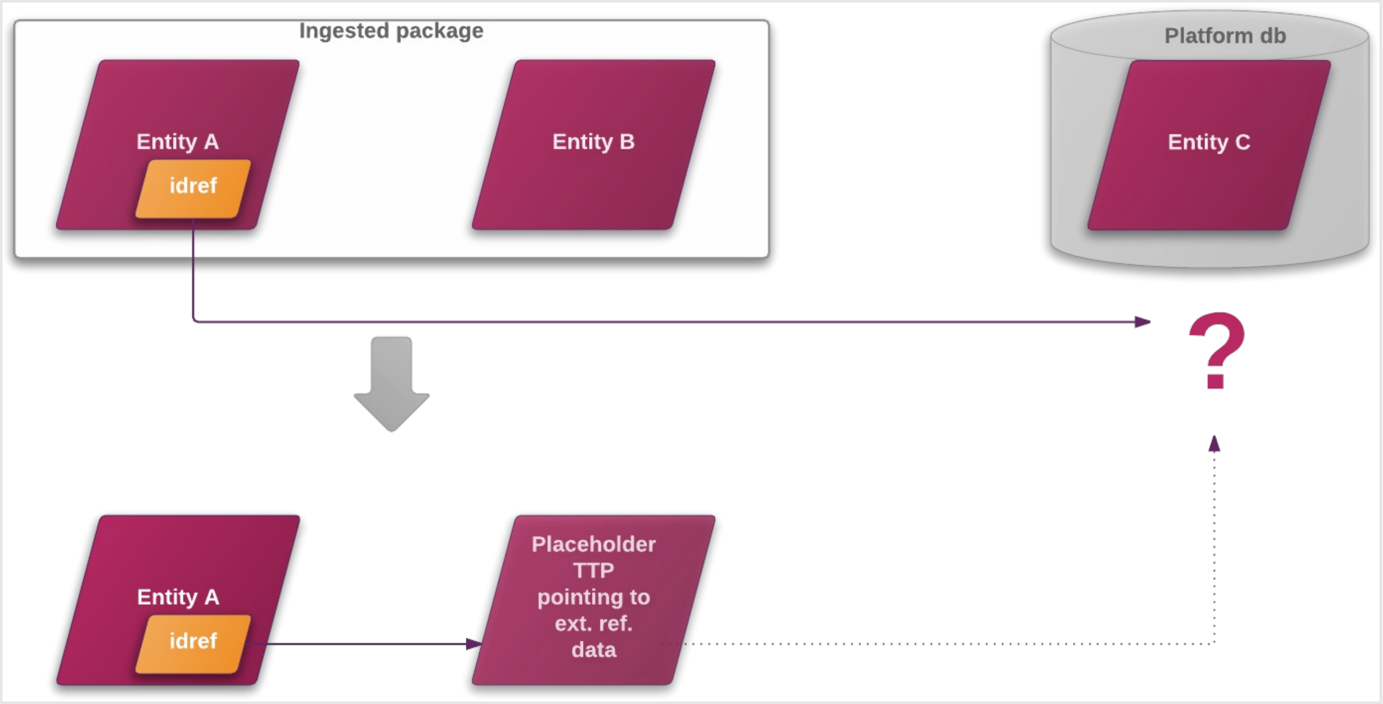


Referenced data is inside the same ingested package



Referenced data is outside the same ingested package, but available in the platform database

- If the `idref` references unavailable data at the moment, the reference is resolved if/when the referenced data becomes available:
    - The `idref` is converted to an empty placeholder TTP or indicator entity that is populated when/if the originally referenced data becomes available.
    - The empty placeholder entity is indexed; therefore, it is searchable, and it can be loaded on the graph.



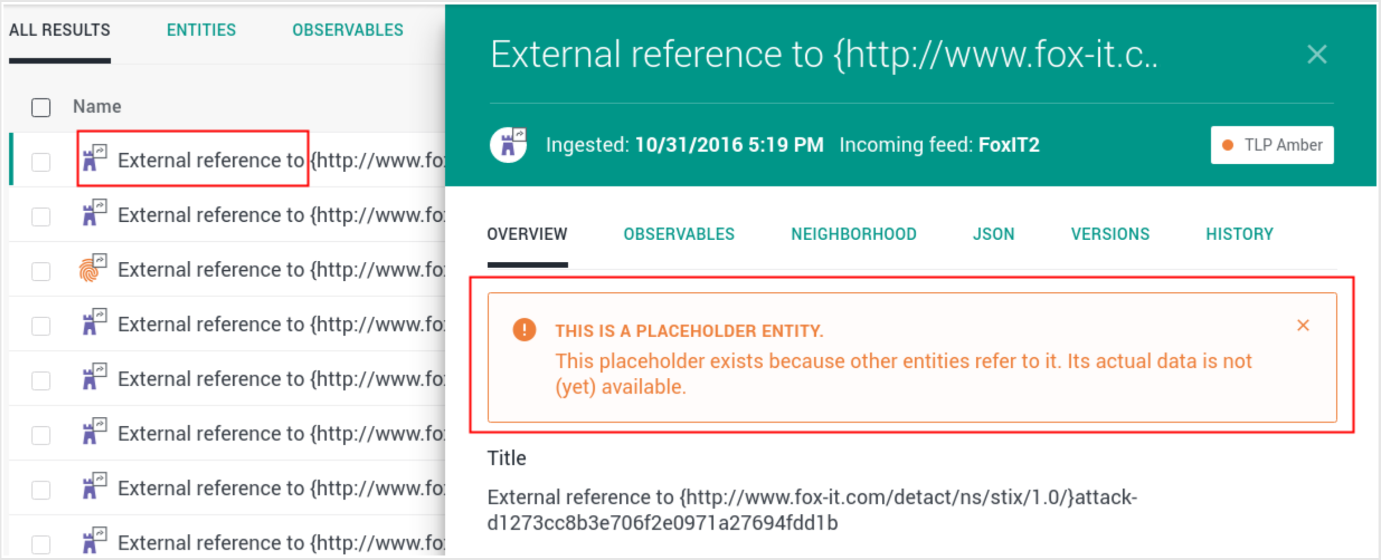
Referenced data is unavailable, an empty placeholder entity points to it in case it becomes available in the future

The process works identically in the opposite direction: if CybOX data is ingested, the platform looks for an existing `idref` pointing to it.

- If it finds a matching `idref`, the existing `idref` is removed and it is replaced with the newly ingested referenced data.
- If no matching `idref` is available, nothing happens. If a matching `idref` becomes available at a later time, it will be resolved then by replacing it with the corresponding referenced data.

Example of an empty placeholder entity

This is an example of an empty TTP placeholder, as shown in the entity detail pane:


ALL RESULTS
ENTITIES
OBSERVABLES
☐ Name
☐ External reference to {http://www.fo
☐ External reference to {http://www.fo
☐ External reference to {http://www.fo
☐ External reference to {http://www.fo
☐ External reference to {http://www.fo
☐ External reference to {http://www.fo
☐ External reference to {http://www.fo
☐ External reference to {http://www.fo
External reference to {http://www.fox-it.c..
×
TLP Amber
OVERVIEW
OBSERVABLES
NEIGHBORHOOD
JSON
VERSIONS
HISTORY
×

Ingested: **10/31/2016 5:19 PM** Incoming feed: **FoxIT2**

THIS IS A PLACEHOLDER ENTITY.

This placeholder exists because other entities refer to it. Its actual data is not (yet) available.

Title

External reference to {http://www.fox-it.com/detact/ns/stix/1.0/}attack-d1273cc8b3e706f2e0971a27694fdd1b

This is the corresponding JSON representation:

```
{
  "entities": [

    ...

      "data": {
        "id": "{http://www.example.com/stix/1.0/}attack-d1273cc8b3e108h2e0971a27694fdd1e",
        "type": "ttp",
        "title": "External reference to {http://www.example.com/stix/1.0/}attack-
d1273cc8b3e108h2e0971a27694fdd1e"
      }
    }
  ],

    ...

}
```

Saving data

Last but not least, the platform saves the ingested entities to the databases in the following order:

- Entity store (PostgreSQL)
- Search store (Elasticsearch)

- Graph store (Neo4j)

Saving data URIs and raw artifacts

The extracted entity data that is stored inside observables ranges from short, simple data such as email addresses, domain names, IP addresses, and so on, to binary data. When an entity contains binary data, the data can be included as either a *data URI* or a *CybOX raw artifact* element.

During ingestion, extraction logic handles binary data URI and raw artifact objects embedded in CybOX objects in the following way:

- **data URIs** `(https://en.wikipedia.org/wiki/data_uri_scheme)` are extracted and stored as entity attachments and new hash values:
    - The data URI value is recalculated to a new hash: `uri-hash-sha256`.
      The SHA-256 hash value for `uri-hash-sha256` is calculated over the UTF-8 encoding of the data URI string.
      The `uri-hash-sha256` hash substitute allows for entity correlation among entities containing the same data URI.
    - The binary data/raw content embedded in the data URI is decoded and processed:
        - The extracted binary data content is stored as an entity attachment similar to the CybOX `Raw_Artifact` object.
        - The extracted content is hashed using SHA-512, SHA-256, SHA-1, and MD5.
          Each resulting hash is added to the relevant entities as an observable.

Example

A data URI with image content nested inside a CybOX object generates the following output:

- 1 `uri-hash-sha256` hash to facilitate entity correlation
- 4 calculated hash observables: `hash-sha512`, `hash-sha256`, `hash-sha1`, and `hash-md5`
- 1 embedded JSON entity attachment (`raw-artifact`) with the extracted binary data

The following example shows a sample input along with the corresponding output.

```
dataUriExtractionSample(

        input={
            data:image/gif;base64,R0lGODlhAQABAAAAACH5BAEKAAEALAAAAAABAAEAAAICTAEAOw==
        },

        output={

            # Recalculated hash of the original URI:
            ('uri-hash-sha256:'
             'd16ae5d51dda6f58995171aa23c0fa5e'
             '6dcd9c777cf9c251c4be3b1d62fdf670'),

            # Multiple hashes of the decoded content:
            'hash-md5:3eacd0132310ea44cad756b378a3bc07',

            'hash-sha1:e2216a7e9b73f5cb0279351c78ce61c33475cea7',

            ('hash-sha256:'
             'bb229a48bee31f5d54ca12dc9bd960c6'
             '3a671f0d4be86a054c1d324a44499d96'),

            ('hash-sha512:'
             'bd9ab35dde3a5242b04c159187732e13'
             'b0a6da50ddcff7015dfb78cdd68743e1'
             '91eaf5cddedd49bef7d2d5a642c21727'
             '2a40e5ba603fe24ca676a53f8c417c5d'),

            # (Attachment) Raw artifact as embedded JSON with the content:
            ('raw-artifact:{"content": '
             '"R0lGODlhAQABAAAAACH5BAEKAAEALAAAAAABAAEAAAICTAEAOw==", '
             '"content_encoding": "base64", "type": "image/png"}'),
        }),
```

Configure incoming feeds

Configure incoming feeds to ingest data from selected sources in many different formats.

EclecticIQ Platform can ingest cyber threat intelligence made available in multiple formats. You can configure incoming feeds to retrieve cyber threat data from many different sources. Incoming feeds provide data, context, and additional information to help analysts draw an accurate map of the threat landscape they are examining. This knowledge is essential to making informed decisions quickly and efficiently, and to promoting a proactive behavior when monitoring IT infrastructures and managing IT security.

You can populate the platform with data by defining one or more incoming feed sources.
Once it is set up and it is running, an incoming feed provides a data stream that the platform ingests and processes automatically.

A minimal incoming feed configuration includes:

- A *data source*: the intel origin the incoming fed fetches data from.
  For example, a URI, a path pointing to a network location, or an IP address to an API endpoint.
- A *transport type*: the vehicle carrying the data.
  Typically, this is a communications protocol like TAXII, HTTP, FTP, or IMAP.
- A *content type*: the incoming data format the platform should expect from the incoming feed.
  For example, STIX, JSON, CSV, or PDF.

This article describes how to configure the **general options** for incoming feeds to ingest data into EclecticIQ Platform. These options are identical for all incoming feeds.

To configure transport type and content type, as well as any other specific options for a particular incoming feed, follow the links under Configure transport and content for specific incoming feeds.

Configure the general options

✔ Input fields marked with an asterisk are required.

The **Incoming feeds** page displays an overview of the configured incoming feeds ingesting data from the specified intel providers and data sources.

- On the top navigation bar, select **Data configuration > Incoming feeds**.
- On the top-left corner of the page click the ✚ icon to open the incoming feed editor.
- On the **Create incoming feed** form you can populate the input fields to define the intel provider/data source for the feed, and the feed behavior.
- Under **Feed name**, enter a name for the feed you are creating. It should be descriptive and easy to remember.
- Under **Organization**, enter a name for the organization that serves as the intel provider for the incoming feed.
- Use **Source reliability** to flag the incoming feed with a value from the drop-down list to help oher users assess how trustworthy the feed source is deemed to be.
  This value has the same meaning as the first character in the **two-character Admiralty System code** `(https://en.wikipedia.org/wiki/admiralty_code)`.

- **Require valid signature**: select this checkbox to enforce PGP signature validation of the incoming feed data.

  To work correctly, the source data provider needs to offer a PGP validation service based on a public and a private key.
  Moreover, you need to obtain a valid PGP public key from the source data provider, which you need to add to the platform trusted keys.

  To add a trusted key to the platform, do the following::

  - On the left-hand navigation sidebar click ⚙ **> System settings > Trusted keys > Edit settings** .
  - On the **Edit public trusted keys settings** page, click ✚ **Add** or ✚ **More** to add a new PGP public key:
    - **Key**: copy-paste into this field the public PGP key you want to add as a trusted key to verify the PGP signature of incoming data packages, including the `-----BEGIN PGP PUBLIC KEY BLOCK-----`and `-----END PGP PUBLIC KEY BLOCK-----` lines.
    - **Description**: add a short description to help users understand the purpose of the PGP public key.
    - Click **Save** to store your changes, or **Cancel** to discard them.
- **Skip extraction of observables from unstructured text**: select this checkbox to exclude from ingestion observable data detected in unstructured text sections.

  If you select the checkbox, the platform filters out any observable data detected inside titles, headers, descriptions, summaries, and other free-form, free text fields.
  Observable data inside unstructured text fields is usually not as relevant, and not as valuable in terms of intelligence, as observable data extracted from, for example, CybOX objects and fields.
- From the **Transport type** and **Content type** drop-down menus select the appropriate options to configure transport and content for the specified incoming feed.

Set a schedule

Under **Execution schedule** you can define how often you want to run the feed task:

- **None**: scheduled feed execution is disabled. You need to manually trigger the task to ingest or to publish data through an incoming or an outgoing feed, respectively.
- **Every [n] minutes**: the feed task runs automatically once every *[n]* minutes, where *[n]* defines the selected time interval in minutes.
  You define the execution interval in 5-minute increments from the corresponding drop-down menu.
- **Every hour, [n] minutes past the hour**: the feed task runs automatically once an hour every hour at the specified minute offset from the hour.
  You define how long in minutes after the beginning of an hour the task should run from the corresponding drop-down menu.
- **Every [n] hours**: the feed task runs automatically once every *[n]* hours, where *[n]* defines the time interval in hours between two consecutive feed task runs.
  You define how long the time interval between feed executions should be by selecting the number of hours from the corresponding drop-down menu.
- **Every day at [time]**: the feed task runs automatically once a day at the specified time.
  You define the time of the day when the task should run from the corresponding drop-down menus.
- **Every [n] days**: the feed task runs automatically once every *[n]* days, where *[n]* defines the time interval in days between two consecutive feed task runs.
  You define how long the time interval between feed executions should be by selecting the number of days from the corresponding drop-down menu.

- **Every week on [day of the week] at [time]** : the feed task runs automatically once a week on the designated day, at the specified time.
  You define the day of the week and time of the day when the task should run from the corresponding drop-down menus.
- **Every month on [day of the month] at [time]** : the feed task runs automatically once a month on the designated day of the month, at the specified time.
  You define the day of the week and time of the day when the task should run from the corresponding drop-down menus.
  Keep in mind that not all months of the year have 30 or 31 days.

Set a TLP override

Override TLP overwrites the **TLP** `(https://www.us-cert.gov/tlp)` color code associated with the feed entities with the one you set here. The selected TLP value is assigned to all the entities in the feed.

You can override the original or the current TLP color code of an entity, an incoming feed, or an outgoing feed.
When working as a filter, TLP colors select a decreasing range: if you set a TLP color as a filter the enricher, the feed, or the returned filtered results include all the entities flagged with the selected TLP color code, as well as all the entities whose TLP color indicates that they are progressively lower risk, less sensitive, and suitable for disclosure to broader audiences.
For example, if you select green the filtered results include entities with a TLP color set to green, as well as entities with a TLP color set to white, and entities with no TLP color code flag.

Set half-life values

It represents the amount of time it takes an entity to lose half its intelligence value.
It corresponds to the number of days it takes the intelligence value of a malicious entity to decay by 50%.

When configuring an incoming or an outgoing feed, you can set a half-life value in days for the following entity properties:

- **Campaign**
- **Course of action**
- **Exploit target**
- **Incident**
- **Indicator**
- **TTP**
- **Threat actor**
- **Report**

To set a half-life for one or more of these properties, do the following::

- Enter a numerical value in the entity property input field(s) you want to flag with a half-life value in days.
- Click **Save** to store your changes, or **Cancel** to discard them.

Save options

Besides committing current data by clicking **Save**, you can also click the downward-pointing arrow on the **Save** button to display a context menu with additional save options:

- **Save and new**: saves the current data for the active item, and it allows you to start creating a new item of the same type right away. For example, a dataset, a feed, a rule, a workspace, or a task.
- **Save and duplicate**: saves the current data for the active item, and it creates a pre-populated copy of the same item, which you can use as a template to speed up manual creation work.

Configure transport and content for specific incoming feeds

- Configure Anubis Cyberfeed transport and content
- Configure BFK transport and content
- Configure Crowdstrike Falcon Intelligence Indicator transport and content
- Configure Crowdstrike Falcon Intelligence Reports transport and content
- Configure CVE Search API transport and content
- Configure IMAP email fetcher transport and content
- Configure IMAP email attachment fetcher transport and content
- Configure FireEye iSIGHT Intelligence Report API transport and content
- Configure FTP download transport and content
- Configure Group-IB transport and content
- Configure HTTP download transport and content
- Configure Intel 471 transport and content types
- Configure mount point download transport and content
- Configure PhishMe Intelligence transport and content
- Configure SpyCloud Breach API transport and content
- Configure TAXII inbox transport and content
- Configure TAXII poll transport and content
- Configure Cisco Threat Grid Samples API transport and content
- Configure Cisco Threat Grid Curated Feed transport and content
- Configure Threat Recon transport and content

Start and stop feeds

Enable and disable feeds, as well as manually trigger a feed task run or stop a running feed.

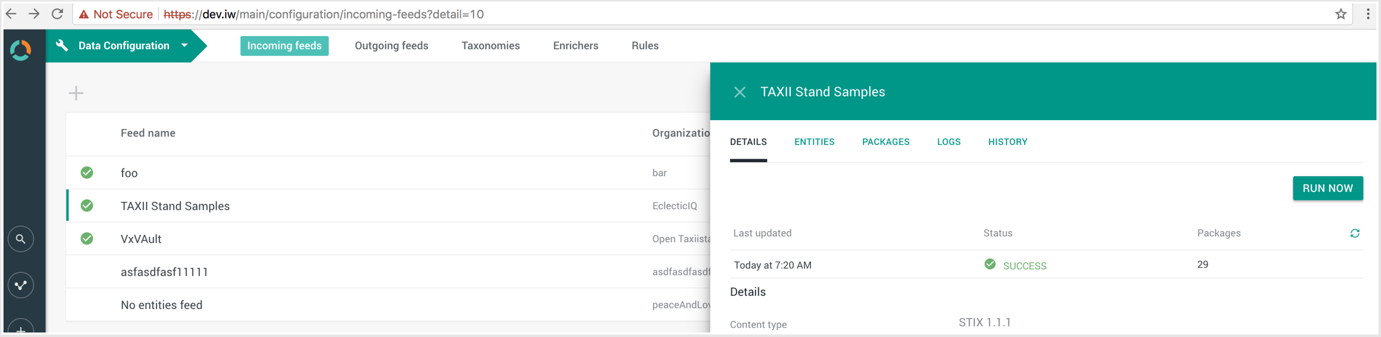
After configuring a feed, you can set a schedule to automate feed execution over time. If you do not set an execution schedule, the feed does not run, that is, it does not fetch or publish any data.

Manually start a feed

You can manually start an incoming or an outgoing feed run in one of the following ways:

On the incoming feed overview page

- Go to **Data configuration > Incoming feeds** or to **Data configuration > Outgoing feeds**, depending on whether you want to run an incoming or an outgoing feed.
- On the feed overview page, click the ⋮ icon corresponding to the feed you want to run.
- From the drop-down menu select **Run now**.



	Feed name	Organizatio
✓	foo	bar
✓	TAXII Stand Samples	EclecticIQ
✓	VxVAult	Open Taxiista
	asfasdfasf11111	asdfasdfasdf
	No entities feed	peaceAndLov

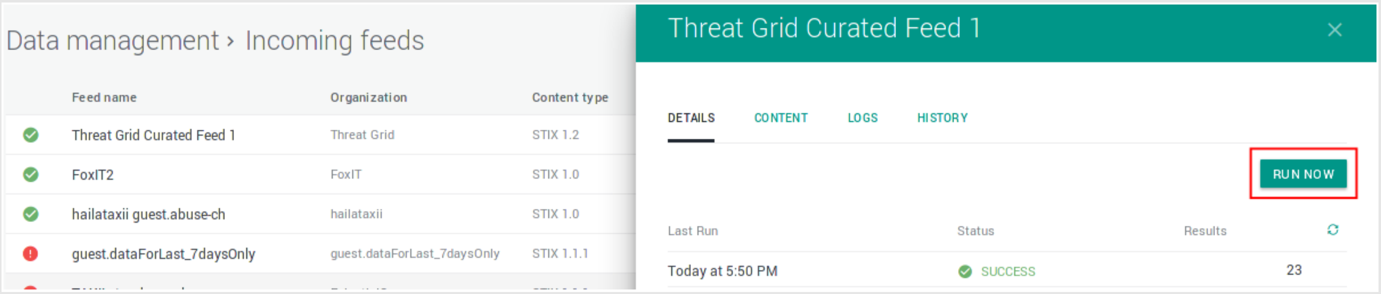
Last updated	Status	Packages
Today at 7:20 AM	✓ SUCCESS	29

Details

Content type STIX 1.1.1

On the incoming feed entity detail pane

- Go to **Data configuration > Incoming feeds** or to **Data configuration > Outgoing feeds**, depending on whether you want to run an incoming or an outgoing feed.
- On the feed overview page, click anywhere on the row corresponding to the feed you want to run.
- On the **Details** tab on feed detail pane click **Run now**.



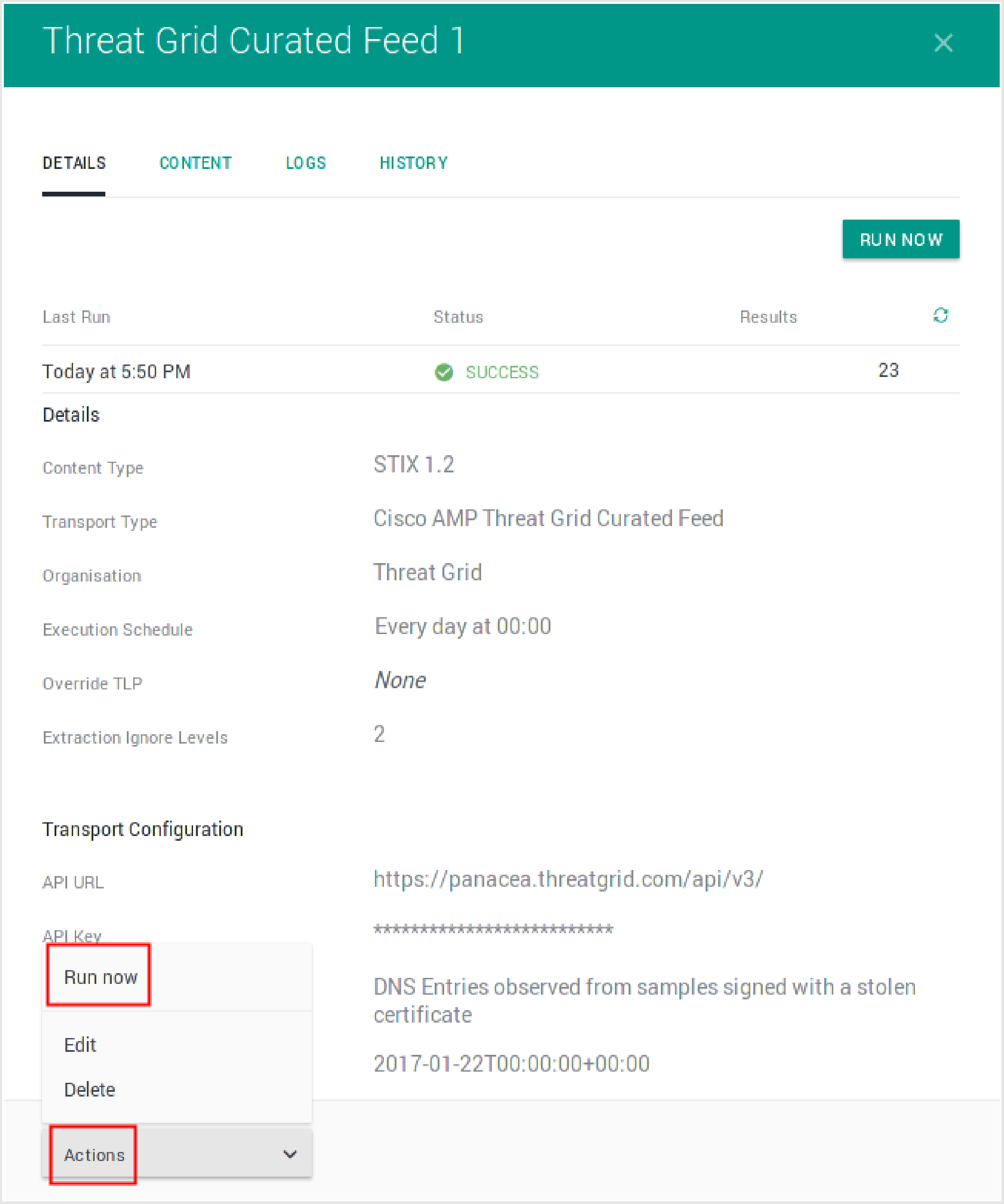
	Feed name	Organization	Content type
✓	Threat Grid Curated Feed 1	Threat Grid	STIX 1.2
✓	FoxIT2	FoxIT	STIX 1.0
✓	hailataxii guest.abuse-ch	hailataxii	STIX 1.0
!	guest.dataForLast_7daysOnly	guest.dataForLast_7daysOnly	STIX 1.1.1

Last Run	Status	Results
Today at 5:50 PM	✓ SUCCESS	23

Through the Actions menu

- Go to **Data configuration > Incoming feeds** or to **Data configuration > Outgoing feeds**, depending on whether you want to run an incoming or an outgoing feed.
- On the feed overview page, click anywhere on the row corresponding to the feed you want to run.

- On the **Details** tab on feed detail pane, scroll to the bottom of the pane, and then click **Actions**.
- From the pop-up menu select **Run now**.



Threat Grid Curated Feed 1

Last Run	Status	Results
Today at 5:50 PM	✓ SUCCESS	23

Details

Content Type STIX 1.2

Transport Type Cisco AMP Threat Grid Curated Feed

Organisation Threat Grid

Execution Schedule Every day at 00:00

Override TLP *None*

Extraction Ignore Levels 2

Transport Configuration

API URL https://panacea.threatgrid.com/api/v3/

API Key ***************************

DNS Entries observed from samples signed with a stolen certificate

2017-01-22T00:00:00+00:00

Manually stop a feed

You can either manually suspend/disable or stop/terminate a running incoming or outgoing feed.

Suspend and disable a running feed

To disable an active feed, do the following:

- Go to **Data configuration > Incoming feeds** or to **Data configuration > Outgoing feeds**, depending on whether you want to run an incoming or an outgoing feed.
- On the feed overview page, click anywhere on the row corresponding to the feed you want to run.
- On the **Details** tab on feed detail pane click **Disable**.
  You can enable the disabled feed at any time by clicking **Enable**.

You can disable feed execution also by setting the feed execution schedule to **None**:

- Go to **Data configuration > Incoming feeds** or to **Data configuration > Outgoing feeds**, depending on whether you want to suspend an incoming or an outgoing feed.
- On the feed overview page, click the ⋮ icon corresponding to the feed you want to run.
- From the drop-down menu select **Edit**.
- On the feed configuration page, go to the **Schedule** section, and then set **Execution schedule** to **None**.
- Click **Save** to store your changes, or **Cancel** to discard them.

Alternatively:

- Go to **Data configuration > Incoming feeds** or to **Data configuration > Outgoing feeds**, depending on whether you want to run an incoming or an outgoing feed.
- On the feed overview page, click anywhere on the row corresponding to the feed you want to run.
- On the feed detail pane, scroll to the bottom of the pane, and then click **Actions**.
- From the drop-down menu select **Edit**.
- On the feed configuration page, go to the **Schedule** section, and then set **Execution schedule** to **None**.
- Click **Save** to store your changes, or **Cancel** to discard them.

Stop and terminate a running feed

To stop the execution of a running feed and kill the task, do the following:

- On the left-hand navigation sidebar, click ⚙ **> System jobs > Running**.
- On the **System jobs > Running** overview page, browse to the running task(s) you want to terminate, and then click the corresponding ■ **Terminate** button to instantly stop executing the selected task(s).

Delete and purge feeds

If you do not need to keep an incoming feeds any longer, you can decide whether you want to delete the feed while keeping entities ingested through it, or purge the feed to delete it along with any linked data such as entities and relationships.

You may want to delete an incoming feed to avoid polluting the platform with unnecessary clutter or dirty data. For example, the content an incoming feed delivers may be no longer necessary, or an error in the feed configuration may have produced information you do not need.

In these cases, you can either delete or purge an incoming feed:

- *Delete the feed* to remove the incoming feed configuration.
    - The platform stops ingesting and processing data from the designated data source for the feed.
    - Existing data linked to the feed are preserved, such as previously ingested and processed entities and relationships.
- *Purge the feed* to remove the incoming feed configuration, and any data ingested through or linked to the feed.
    - The platform stops ingesting and processing data from the designated data source for the feed.
    - Data linked to the feed are completely removed as well, such as entities previously ingested through the feed.

Delete an incoming feed

Choose this option to:

- Stop ingesting data through the selected incoming feed.
- Delete the feed and its configuration from the platform.
- Keep the data ingested through the feed, such as entities and relationships.

To delete an incoming feed, do the following:

- On the top navigation bar click **Data configuration > Incoming feeds**.
- On the feed overview page click anywhere on the row corresponding to the feed you want to delete.
- On the feed detail pane, click **Actions > Delete**.

Alternatively:

- On the feed overview page click the ⋮ icon on the row corresponding to the feed you want to delete.
- From the drop-down menu select **Delete**.
- On the confirmation dialog, click **Delete** to confirm the action.

Purge an incoming feed

Choose this option to:

- Stop ingesting data through the selected incoming feed.

- Delete the feed and its configuration from the platform.
- Delete also any data ingested through the feed, such as entities and relationships.

To purge an incoming feed, do the following:

- On the top navigation bar click **Data configuration > Incoming feeds**.
- On the feed overview page click anywhere on the row corresponding to the feed you want to purge.
- On the feed detail pane, click **Actions > Delete**.

Alternatively:

- On the feed overview page click the ⋮ icon on the row corresponding to the feed you want to purge.
- From the drop-down menu select **Delete**.
- On the confirmation dialog, click **Purge** to confirm the action.

Configure IMAP email fetcher transport and content

Set up and configure transport and content types for IMAP email fetcher incoming feeds to retrieve and process information from email body contents.

To configure the general options for the IMAP email fetcher incoming feed, see Configure incoming feeds.

About IMAP email fetcher

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
Email message	Email message body content.	Structured, STIX-compliant report entities.

Configure the transport type

Transport type	Allowed content types
IMAP email fetcher	Email message

The IMAP email fetcher transport type for incoming feeds fetches and ingests email body content in the supported content types.
Any existing email attachments are not processed, and they are deleted after processing and ingesting the body content of the corresponding email(s).

- From the **Transport type** drop-down menu, select **IMAP email fetcher**.
- From the **Content type** drop-down menu, select **Email message**.
  The **IMAP email fetcher** transport type supports only the **Email message** content type.
- **Accept password protected archives**: select this checkbox to specify a global password to open any archives acquired through the incoming feed.
  If the archives are password-protected, enter it in the **Archive password** field. The specified password acts as a master password, and it is used to unlock all the archives retrieved with the feed.

Under **Transport configuration**, configure the following settings:

- **Host**: the address of the email server acting as a data source for the incoming feed.
  Example: *imap.call-cthulhu.com*
- **Username**: a valid user name to authenticate and be granted the necessary authorization to access the data source and to download/ingest data.
- **Password**: a valid password to authenticate and be granted the necessary authorization to access the data source and to download/ingest data.
- **Use SSL**: select this checkbox if the data source email server supports SSL, and if you want to enable the security layer for the feed.

- **To keyword**: you can enter a keyword to filter and include in the feed only content found in emails whose *To* recipient field includes the specified keyword search string.
- **From keyword**: you can enter a keyword to filter and include in the feed only content found in emails whose *From* sender field includes the specified keyword search string.
- **Subject keyword**: you can enter a keyword to filter and include in the feed only content found in emails whose *Subject* field includes the specified keyword search string.

Warning:

Email body or attachment download known issues

Currently, the **IMAP email fetcher** and the **IMAP email attachment fetcher** incoming feed transport types do not support retrieving email body content or email attachments, respectively, from the following web-based email service providers:

- Microsoft Office 365 Outlook
- Google Gmail

These providers will be supported in the coming future.

Configure IMAP email attachment fetcher transport and content

Set up and configure transport and content types for IMAP email attachment fetcher incoming feeds to retrieve email attachment content in multiple formats. You can then use the resulting reports for analysis or automation purposes.

To configure the general options for the IMAP email attachment fetcher incoming feed, see Configure incoming feeds.

About IMAP email attachment fetcher

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
IMAP email attachment fetcher	Structured and unstructured data in JSON, PDF, STIX, and plain text format.	Structured, STIX-compliant report entities.

Configure the transport type

Transport type	Allowed content types
IMAP email attachment fetcher	EclecticIQ JSON
	PDF
	STIX 1.0
	STIX 1.1
	STIX 1.1.1
	STIX 1.2
	Text

The IMAP email attachment fetcher transport type for incoming feeds fetches and ingests email attachments in the supported content types.
The email body is not processed, and the email message is deleted after processing and ingesting the corresponding attachment(s).

- From the **Transport type** drop-down menu, select **IMAP email attachment fetcher**.

- From the **Content type** drop-down menu, select the appropriate content type for the data you want to ingest through the incoming feed.
  The selected content type for the feed should match the data source format.
  This can vary, depending on the intel sources you retrieve the data from.
  The **IMAP email attachment fetcher** transport type enables fetching data in the following formats:
  - EclecticIQ JSON
  - PDF
  - STIX 1.0
  - STIX 1.1
  - STIX 1.1.1
  - STIX 1.2
  - Text
- **Accept password protected archives**: select this checkbox to specify a global password to open any archives acquired through the incoming feed.
  If the archives are password-protected, enter it in the **Archive password** field. The specified password acts as a master password, and it is used to unlock all the archives retrieved with the feed.

Under **Transport configuration**, configure the following settings:

- **Host**: the address of the email server acting as a data source for the incoming feed.
  Example: *imap.call-cthulhu.com*
- **Username**: a valid user name to authenticate and be granted the necessary authorization to access the data source and to download/ingest data.
- **Password**: a valid password to authenticate and be granted the necessary authorization to access the data source and to download/ingest data.
- **Use SSL**: select this checkbox if the data source email server supports SSL, and if you want to enable the security layer for the feed.
- **To keyword**: you can enter a keyword to filter and include in the feed only content found in emails whose *To* recipient field includes the specified keyword search string.
- **From keyword**: you can enter a keyword to filter and include in the feed only content found in emails whose *From* sender field includes the specified keyword search string.
- **Subject keyword**: you can enter a keyword to filter and include in the feed only content found in emails whose *Subject* field includes the specified keyword search string.

Warning:

Email body or attachment download known issues

Currently, the **IMAP email fetcher** and the **IMAP email attachment fetcher** incoming feed transport types do not support retrieving email body content or email attachments, respectively, from the following web-based email service providers:

- Microsoft Office 365 Outlook
- Google Gmail

These providers will be supported in the coming future.

Configure FTP download transport and content

Set up and configure transport and content types for FTP download incoming feeds to retrieve and process information from specific data sources supporting the FTP protocol.

To configure the general options for the FTP download incoming feed, see Configure incoming feeds.

About FTP download

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
FTP download	Structured and unstructured data in JSON, PDF, STIX, and plain text format.	Structured, STIX-compliant entities.

Configure the transport type

Transport type	Allowed content types
FTP download	EclecticIQ JSON
	PDF
	STIX 1.0
	STIX 1.1
	STIX 1.1.1
	STIX 1.2
	Text

- From the **Transport type** drop-down menu, select **FTP download**.

- From the **Content type** drop-down menu, select the appropriate content type for the data you want to ingest through the incoming feed.
  The selected content type for the feed should match the data source format.
  This can vary, depending on the intel sources you retrieve the data from.
  The **FTP download** transport type enables fetching data in the following formats:
  - EclecticIQ JSON
  - PDF
  - STIX 1.0
  - STIX 1.1
  - STIX 1.1.1
  - STIX 1.2
  - Text
- **Accept password protected archives**: select this checkbox to specify a global password to open any archives acquired through the incoming feed.
  If the archives are password-protected, enter it in the **Archive password** field. The specified password acts as a master password, and it is used to unlock all the archives retrieved with the feed.

Under **Transport configuration**, configure the following settings:

- **URL**: the location on the FTP server where the data source for the feed is made available.
- **Username**: a valid user name to authenticate and be granted the necessary authorization to access the data source and to download/ingest data.
- **Password**: a valid password to authenticate and be granted the necessary authorization to access the data source and to download/ingest data.
- **TLS**: select this checkbox if the data source FTP server supports TLS, and if you want to enable the security layer for the feed.

Configure HTTP download transport and content

Set up and configure transport and content types for HTTP download incoming feeds to retrieve and process information from specific data sources supporting the HTTP protocol.

To configure the general options for the HTTP download incoming feed, see Configure incoming feeds.

About HTTP download

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
HTTP download	Structured and unstructured data in JSON, PDF, STIX, and plain text format.	Structured, STIX-compliant entities.

Configure the transport type

Transport type	Allowed content types
HTTP download	EclecticIQ JSON
	PDF
	STIX 1.0
	STIX 1.1
	STIX 1.1.1
	STIX 1.2
	Text

- From the **Transport type** drop-down menu, select **HTTP download**.

- From the **Content type** drop-down menu, select the appropriate content type for the data you want to ingest through the incoming feed.
  The selected content type for the feed should match the data source format.
  This can vary, depending on the intel sources you retrieve the data from.
  The **HTTP download** transport type enables fetching data in the following formats:
  - EclecticIQ JSON
  - PDF
  - STIX 1.0
  - STIX 1.1
  - STIX 1.1.1
  - STIX 1.2
  - Text
- **Accept password protected archives**: select this checkbox to specify a global password to open any archives acquired through the incoming feed.
  If the archives are password-protected, enter it in the **Archive password** field. The specified password acts as a master password, and it is used to unlock all the archives retrieved with the feed.

Under **Transport configuration**, configure the following settings:

- **URL**: the location/directory/folder on the server or network unit hosting the data source for the feed.
- **Regex pattern**: you can define a regex to include in the incoming feed only the content delivered through the links that match the specified regex pattern.
  - If you do not enter any regex pattern, the feed fetches the base URL response body, and it assumes that any retrieved content is in the format specified under **Content type**.
  - If you enter a regex pattern, the feed assumes that any retrieved content is in HTML format, regardless of the format specified under **Content type**.
    Retrieved content is scanned for any links matching the regex pattern.
    Matches are downloaded, assuming that their format corresponds to the one specified under **Content type**.
- **Basic auth username**: a valid user name to authenticate and be granted the necessary authorization to access the data source and to download/ingest data.
  HTTP download uses basic HTTP authentication.
- **Basic auth password**: a valid password to authenticate and be granted the necessary authorization to access the data source and to download/ingest data.
- **Verify connection**: select this checkbox if you want to test the connection to verify that it works as expected.

- **Extra headers**: the HTTP server hosting the data source for the feed may require passing additional HTTP headers in the request.
  Click **+ Add** or **+ More** to include in the request one or more additional HTTP headers, along with the corresponding values:
  - *Accept-Charset*
  - *Accept-Encoding*
  - *Accept-Language*
  - *Accept-Datetime*
  - *Authorization*
  - *Connection*
  - *Cookie*
  - *Expect*
  - *From*
  - *Host*
  - *If-Match*
  - *If-Modified-Since*
  - *If-None-Match*
  - *If-Range*
  - *If-Unmodified-Since*
  - *Max-Forwards*
  - *Origin*
  - *Proxy-Authorization*
  - *User-Agent*
  - *Via*
  - *Warning*

Configure mount point download transport and content

Set up and configure transport and content types for mount point download incoming feeds to retrieve and process information from specific data sources available on the (local) network.

To configure the general options for the Mount point download incoming feed, see Configure incoming feeds.

About Mount point download

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
Mount point download	Structured and unstructured data in JSON, PDF, STIX, and plain text format.	Structured, STIX-compliant entities.

Configure the transport type

Transport type	Allowed content types
Mount point download	EclecticIQ JSON
	PDF
	STIX 1.0
	STIX 1.1
	STIX 1.1.1
	STIX 1.2
	Text

- From the **Transport type** drop-down menu, select **Mount point download**.

- From the **Content type** drop-down menu, select the appropriate content type for the data you want to ingest through the incoming feed.
  The selected content type for the feed should match the data source format.
  This can vary, depending on the intel sources you retrieve the data from.
  The **Mount point download** transport type enables fetching data in the following formats:
  - EclecticIQ JSON
  - PDF
  - STIX 1.0
  - STIX 1.1
  - STIX 1.1.1
  - STIX 1.2
  - Text
- **Accept password protected archives**: select this checkbox to specify a global password to open any archives acquired through the incoming feed.
  If the archives are password-protected, enter it in the **Archive password** field. The specified password acts as a master password, and it is used to unlock all the archives retrieved with the feed.

Under **Transport configuration**, configure the following settings:

- **Path**: the location/directory/folder on the server or network unit hosting the data source for the feed.
  The incoming feed first looks for the specified directory, on a server or a network unit, as the origin to poll data from.
- **Regex pattern**: you can define a regex to include in the incoming feed any file names matching the specified regex pattern.
  If you include a regex pattern, the incoming feed looks in that directory specified under **Path** for any files whose name matches the regex pattern.
  Matching files are added to the feed, so that the platform can ingest their content.
  If you do not enter any regex pattern, the feed fetches the base path response body, and it tries to open the base path.

Configure TAXII inbox transport and content

Set up and configure transport and content types for TAXII inbox incoming feeds to retrieve and process information from specific data sources supporting the TAXII transport type.

To configure the general options for the TAXII inbox incoming feed, see Configure incoming feeds.

About TAXII inbox

Assign unique names to TAXII feeds: TAXII inbox and TAXII poll incoming and outgoing feeds should all have unique names within the platform.

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
TAXII inbox	Structured STIX blobs and packages.	Structured, STIX-compliant entities.

Configure the transport type

Transport type	Allowed content types
TAXII inbox	EclecticIQ JSON
	PDF
	STIX 1.0
	STIX 1.1
	STIX 1.1.1
	STIX 1.2
	Text

- From the **Transport type** drop-down menu, select **TAXII inbox**.

- From the **Content type** drop-down menu, select the appropriate content type for the data you want to ingest through the incoming feed.
  The selected content type for the feed should match the data source format.
  This can vary, depending on the intel sources you retrieve the data from.
  The **TAXII inbox** transport type enables fetching data in the following formats:
  - EclecticIQ JSON
  - PDF
  - STIX 1.0
  - STIX 1.1
  - STIX 1.1.1
  - STIX 1.2
  - Text
- **Accept password protected archives**: select this checkbox to specify a global password to open any archives acquired through the incoming feed.
  If the archives are password-protected, enter it in the **Archive password** field. The specified password acts as a master password, and it is used to unlock all the archives retrieved with the feed.

Under **Transport configuration**, configure the following settings:

- **Public**: select this checkbox to make the incoming feed publicly accessible to all platform users.
  By default, this option is deselected.
- **Authorized groups**: from the drop-down menu select one or more platform user groups to authorize them and their members to access to incoming feed.
  This option is available when **Public** is deselected.

To set up a TAXII inbox incoming feed, the platform needs a configured and working TAXII inbox service, as well as configured user groups.

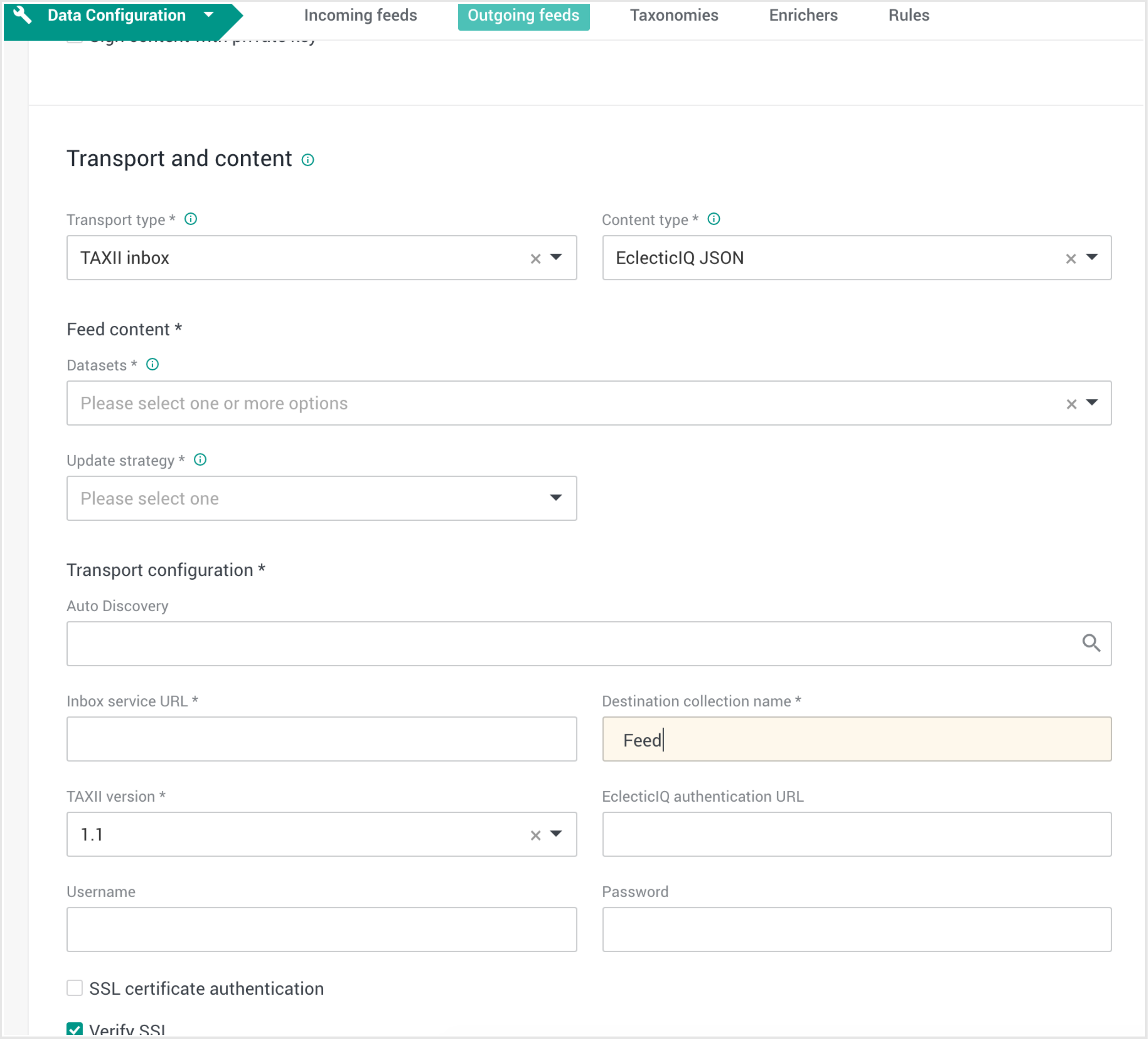
Push data to a TAXII inbox

An easy way to push content to a TAXII inbox that acts as a data source for an incoming feed is by feeding it data through a TAXII inbox outgoing feed.

To link the TAXII inbox outgoing feed data source to the TAXII inbox incoming feed data recipient, make sure you correctly set the following options on the outgoing feed configuration page:

- Under **Transport and content > Transport type** , set the type to **TAXII inbox**.
- Under **Transport and content > Content type** , set the same content type the corresponding receiving TAXII inbox incoming feed is expecting to acquire.
- Under **Transport and content > Transport configuration > Destination collection name** , enter the exact name of the corresponding receiving TAXII inbox incoming feed, as specified in the **Feed name** field.

The highlighted options need to match on the feed configuration pages:



Transport and content

Transport type *: TAXII inbox

Content type *: EclecticIQ JSON

Feed content *

Datasets *: Please select one or more options

Update strategy *: Please select one

Transport configuration *

Auto Discovery:

Inbox service URL *:

Destination collection name *: Feed

TAXII version *: 1.1

EclecticIQ authentication URL:

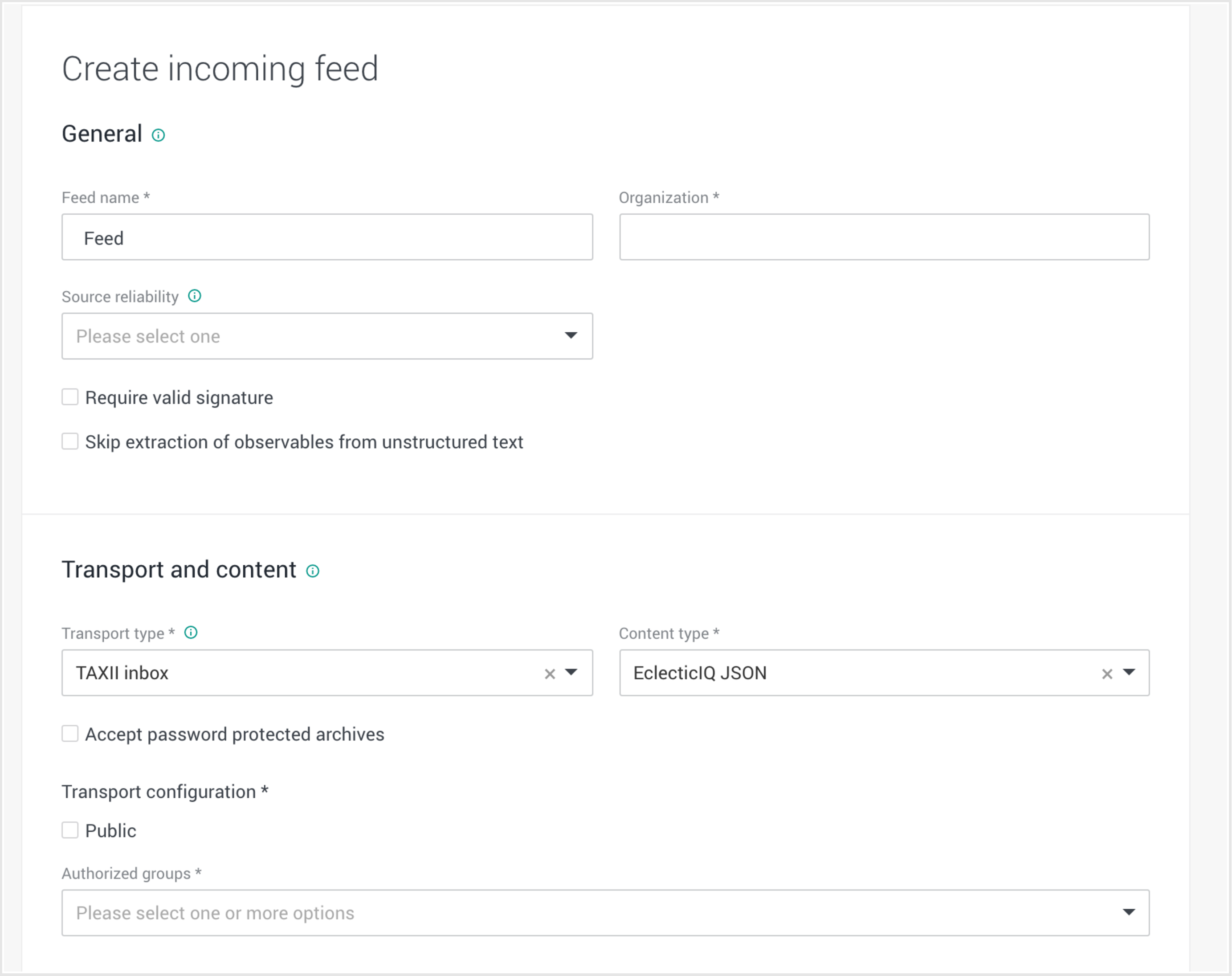
Username:

Password:

☐ SSL certificate authentication

☑ Verify SSL

TAXII inbox outgoing feed. This is the data source pushing content to the TAXII inbox incoming feed.



Create incoming feed

General

Feed name * Feed

Organization *

Source reliability Please select one

☐ Require valid signature

☐ Skip extraction of observables from unstructured text

Transport and content

Transport type * TAXII inbox

Content type * EclecticIQ JSON

☐ Accept password protected archives

Transport configuration *

☐ Public

Authorized groups * Please select one or more options

TAXII inbox incoming feed. This is the data recipient obtaining content from the TAXII inbox outgoing feed.

Configure TAXII poll transport and content

Set up and configure transport and content types for TAXII poll incoming feeds to poll discovery services toretrieve and process information from specific data sources supporting the TAXII transport type.

To configure the general options for the TAXII poll incoming feed, see Configure incoming feeds.

About TAXII poll

ℹ Assign unique names to TAXII feeds: TAXII inbox and TAXII poll incoming and outgoing feeds should all have unique names within the platform.

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
TAXII poll	Structured STIX blobs and packages.	Structured, STIX-compliant entities.

Configure the transport type

Transport type	Allowed content types
TAXII poll	EclecticIQ JSON
	PDF
	STIX 1.0
	STIX 1.1
	STIX 1.1.1
	STIX 1.2
	Text

- From the **Transport type** drop-down menu, select **TAXII poll**.

- From the **Content type** drop-down menu, select the appropriate content type for the data you want to ingest through the incoming feed.
  The selected content type for the feed should match the data source format.
  This can vary, depending on the intel sources you retrieve the data from.
  The **TAXII poll** transport type enables fetching data in the following formats:
  - EclecticIQ JSON
  - PDF
  - STIX 1.0
  - STIX 1.1
  - STIX 1.1.1
  - STIX 1.2
  - Text
- **Accept password protected archives**: select this checkbox to specify a global password to open any archives acquired through the incoming feed.
  If the archives are password-protected, enter it in the **Archive password** field. The specified password acts as a master password, and it is used to unlock all the archives retrieved with the feed.

Under **Transport configuration**, configure the following settings:

- **Auto Discovery**: enter the URL of a TAXII discovery service the feed can send a request to in order to determine the available services and poll them for updates.
  Example: *http://hailataxii.com/taxii-discovery-service*
- **Polling service URL**: enter the URL of a TAXII polling service the feed can poll for updates.
  Example: *http://hailataxii.com/taxii-polling-service*
- **Collection name**: enter the name of the data collection you want to poll and use as a source for the feed.
  The collection you define here needs to be available through the auto-discovery or the polling services.
  Example: *guest.Abuse_ch*
- **TAXII version**: specify the TAXII version the service uses to publish data.
  Example: *1.1*
- **Extra headers**: the TAXII server hosting the data source for the feed may require passing additional HTTP headers in the request.
  Click **+ Add** or **+ More** to include in the request one or more additional HTTP headers, along with the corresponding values.
- **Subscription ID**: enter the name, label or ID identifying the subscription session that makes the incoming feed content available.
  Usually, you need to sign up to a service to initiate such a session.
- **Ingest messages starting from**: select an initial date if you want to fetch content from the intel provider/data source starting from a specific date in the past.
- **Days per poll**: if you select a start date to poll data from, you can enter an integer to specify the maximum number of days to poll at a time.
  This enables polling in multiple smaller batches, instead of a single batch, starting from the selected initial date.
  This option works only if you select an ingestion start date.
- **Verify SSL**: if the TAXII server requires an SSL certificate to authenticate and to access the corresponding TAXII services, you can select this checkbox to test the SSL connection and to verify that it works correctly.
  - **SSL CA bundle file path**: enter the path to the CA bundle file containing the root, intermediate, and public certificates for the SSL authentication.

- **Basic authentication**: if the TAXII server requires basic authentication to access the corresponding TAXII services, select this checkbox to fill out the required information.
  - **Username**: a valid user name to authenticate and be granted the necessary authorization to access the data source and to download/ingest data.
  - **Password**: a valid password to authenticate and be granted the necessary authorization to access the data source and to download/ingest data.
  - **EclecticIQ authentication URL**: the EclecticIQ Platform URL including the endpoint that takes the user name and password inputs to send them to the authentication mechanism.
    Example: *http://<platform_host>/api/auth*
- **SSL certificate authentication**: if the TAXII server requires an SSL certificate to authenticate and to access the corresponding TAXII services, select this checkbox to fill out the required information.
  - **SSL certificate**: copy-paste the content of a valid SSL certificate to authenticate.
    Example:

```
-----BEGIN CERTIFICATE REQUEST-----
MIICvDCCAaQCAQAwdzELMAkGA1UEBhMCVVMxDTALBgNVBAgMBFV0YWgxDzANBgNV
BAcMBkxpbmRvbjEWMBQGA1UECgwNRGlnaUNlcnQgSW5jLjERMA8GA1UECwwIRGln
aUNlcnQxHTAbBgNVBAMMFGV4YW1wbGUuZGlnaWNlcnQuY29tMIIBIjANBgkqhkiG
9w0BAQEFAAOCAQ8AMIIBCgKCAQEA8+To7d+2kPWeBv/orU3LVbJwDrSQbeKamCmo
wp5bqDxIwV20zqRb7APUOKYoVEFFOEQs6T6gImnIolhbiH6m4zgZ/CPvWBOkZc+c
1Po2EmvBz+AD5sBdT5kzGQA6NbWyZGldxRthNLOs1efOhdnWFuhI162qmcflgpiI
WDuwq4C9f+YkeJhNn9dF5+owm8cOQmDrV8NNdiTqin8q3qYAHHJRW28glJUCZkTZ
wIaSR6crBQ8TbYNE0dc+Caa3DOIkz1EOsHWzTx+n0zKfqcbgXi4DJx+C1bjptYPR
BPZL8DAeWuA8ebudVT44yEp82G96/Ggcf7F33xMxe0yc+Xa6owIDAQABoAAwDQYJ
KoZIhvcNAQEFBQADggEBAB0kcrFccSmFDmxox0Ne01UIqSsDqHgL+XmHTXJwre6D
hJSZwbvEtOK0G3+dr4Fs11WuUNt5qcLsx5a8uk4G6AKHMzuhLsJ7XZjgmQXGECpY
Q4mC3yT3ZoCGpIXbw+iP3lmEEXgaQL0Tx5LFl/okKbKYwIqNiyKWOMj7ZR/wxWg/
ZDGRs55xuoeLDJ/ZRFf9bI+IaCUd1YrfYcHIl3G87Av+r49YVwqRDT0VDV7uLgqn
29XI1PpVUNCPQGn9p/eX6Qo7vpDaPybRtA2R7XLKjQaF9oXWeCUqy1hvJac9QFO2
97Ob1alpHPoZ7mWiEuJwjBPii6a9M9G30nUo39lBi1w=
-----END CERTIFICATE REQUEST-----
```

  - **SSL key**: copy-paste the content of a valid SSL key to authenticate
    Example:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEA3Tz2mr7SZiAMfQyuvBjM9Oi..Z1BjP5CE/Wm/Rr500P
RK+Lh9x5eJPo5CAZ3/ANBE0sTK0ZsDGMak2m1g7..3VHqIxFTz0Ta1d+NAj
wnLe4nOb7/eEJbDPkk05ShhBrJGBKKxb8n104o/..PdzbFMIyNjJzBM2o5y
5A13wiLitEO7nco2WfyYkQzaxCw0AwzlkVHiIyC..71pSzkv6sv+4IDMbT/
XpCo8L6wTarzrywnQsh+etLD6FtTjYbbrvZ8RQM..Hg2qxraAV++HNBYmNW
kbJ+q+rsJxQlaipn2M4lGuQJEfIxELFDyd3XpxP..Un/82NZNXlPmRIopXs
2T91jiLZEUKQw+n73j26adTbteuEaPGSrTZxBLR..yssO0wWomUyILqVeti
+PK+aXKwguI6bxLGZ3of0UH+mGsSl0mkp7kYZCm..OTQtfeRqP8rDSC7DgA
kHc5ajYqh04AzNFaxjRo+M3IGICUaOdKnXd0Fda..QwfoaX4QlRTgLqb7AN
ZTzM9WbmnYoXrx17kZlT3lsCgYEAm757XI3WJVj..WoLj1+v48WyoxZpcai
uv9bT4Cj+lXRS+gdKHK+SH7J3x2CRHVS+WH/SVC..DxuybvebDoT0TkKiCj
BWQaGzCaJqZa+POHK0klvS+9ln0/6k539p95tfX..X4TCzbVG6+gJiX0ysz
Yfehn5MCgYEAkMiKuWHCsVyCab3RUf6XA9gd3qY..fCTIGtS1tR5PgFIV+G
engiVoWc/hkj8SBHZz1n1xLN7KDf8ySU06MDggB..hJ+gXJKy+gf3mF5Kmj
DtkpjGHQzPF6vOe907y5NQLvVFGXUq/FIJZxB8k..fJdHEm2M4=
-----END RSA PRIVATE KEY-----
```

  - **SSL key password**: enter the SSL password or passphrase for the SSL key. This field is masked.

Configure Anubis Cyberfeed transport and content

Set up and configure transport and content types for AnubisNetworks Cyberfeed incoming feeds to retrieve and process information on compromised machines and DNS servers, as well as compromised web sites and malware files.

To configure the general options for the Anubis Cyberfeed incoming feed, see Configure incoming feeds.

About Anubis Cyberfeed

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
Infection detection Bank Trojans	Metadata from communication involving Trojan-infected machines.	An ID reference to the TTP containing information on the identified Trojan family.
		An indicator related with ID reference to the TTP, containing the Command and Control system address as server name or server address, as well as the first and last time the threat was sighted.
		A sighting related to the indicator, containing the IP address or the domain name of the compromised machine, HTTP request details like request method, cookies, HTTP user agent, the client IP address passed on to the server via a XFF HTTP header, any additionl arguments, as well as the first and last time the threat was sighted.
Infection detection DNS malware	Metadata from communication involving Trojan-infected DNS servers.	An ID reference to the TTP containing information on the identified Trojan family.
		An indicator related with ID reference to the TTP, containing the Command and Control system address as server name or server address.
		A sighting related to the indicator, containing the IP address of the compromised machine, and the **DNS query type** `(https://en.wikipedia.org/wiki/list_of_dns_record_types).`
Compromised systems website analysis	Indicators of compromise concerning malware-infected web sites.	An indicator with the malware-targeted URL, a description, any available behavior signatures, as well as the first and last time the threat was sighted.
		Observables for any found IP addresses, domain names, or hashes.

Feed	Ingested data	Processed data
Compromised systems malware analysis	Indicators of compromise concerning analyzed malware samples.	An indicator with details about the malware file, a description, any available behavior signatures, as well as the first and last time the threat was sighted.
		Observables for any found file sizes, file types, or file hashes.

By default, the entities the platform creates after processing the ingested data from Anubis Cyberfeed incoming feeds include the following properties:

Property	Default value	Entities
Confidence	*High*	TTPs, indicators, sightings
Likely impact	*High*	Indicators
Impact	*High*	Sightings

Configure the transport type

Transport type	Allowed content types
Anubis Cyberfeed	Anubis Cyberfeed JSON

- From the **Transport type** drop-down menu, select **Anubis Cyberfeed**.
- From the **Content type** drop-down menu, select **Anubis Cyberfeed JSON**.
  The **Anubis Cyberfeed** transport type supports only the **Anubis Cyberfeed JSON** content type.

The source organization providing data for the incoming feed is AnubisNetworks.
Under **Transport configuration**, configure the following settings:

- **API URL**: the URL pointing to the API endpoint exposing the service that makes the data available for retrieval through the feed. Contact the intel provider of the incoming feed to obtain this information.
- **API key**: contact AnubisNetworks to receive an API key, and then enter it in the corresponding input field.
- **Infection detection bank trojans**: select this checkbox to fetch metadata from data flows between machines compromised with Trojans and the AnubisNetworks sinkhole platform.
  This channel provides information such as IP addresses, Trojan family, request metadata, request payload, and pattern verification.
- **Infection detection DNS malware**: select this checkbox to fetch metadata from DNS servers on Trojans trying to contact the AnubisNetworks sinkhole platform. This information allows to detect potential compromises when Command and Control communication is intercepted before it reaches the recipient Command and Control center.
  This channel provides information such as DNS request details, origin IP addresses, and Trojan family.
- **Compromised systems website analysis**: select this checkbox to fetch metadata to detect compromised web sites and servers on the specififed networks, to detect and profile compromised hosts, and to support mitigation by providing indicators and observables.

- **Compromised systems malware analysis**: select this checkbox to fetch metadata to detect malicious file infections on the specififed networks, and to support mitigation by providing indicators and observables.

Configure BFK transport and content

Set up and configure transport and content types for BFK incoming feeds to retrieve and process reports on cyber threats and activities, as well as information on NIDs (Network Intrusion Detections).

To configure the general options for the BFK API incoming feed, see Configure incoming feeds.

About BFK API

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
BFK API	Reports and NIDs (Network Intrusion Detections).	Ingested reports are saved as report entities in the platform, whereas an ingested NID results in an entity with a linked TTP.

Configure the transport type

Transport type	Allowed content types
BFK API	BFK Threat Intelligence JSON

- From the **Transport type** drop-down menu, select **BFK API**.
- From the **Content type** drop-down menu, select **BFK Threat Intelligence JSON**.
  The **BFK API** transport type supports only the **BFK Threat Intelligence JSON** content type.

The source organization providing data for the incoming feed is BFK edv-consulting.
Under **Transport configuration**, configure the following settings:

- **Username**: a valid user name to authenticate and be granted the necessary authorization to access the data source and to download/ingest data.
- **Password**: a valid password to authenticate and be granted the necessary authorization to access the data source and to download/ingest data.
  Contact the intelligence provider to subscribe to the service and to obtain the required authentication and authorization credentials.
- **Polling start time**: select an initial date if you want to fetch content from the intel provider/data source starting from a specific date in the past.
  By default, the max. amount of days in the past per each query/request is set to 60 days.

Configure Cisco Threat Grid Curated Feed transport and content

Set up and configure transport and content types for Cisco Threat Grid Curated Feed incoming feeds to retrieve and process information on Trojan-infected machines and servers, on malware-infected DNS servers and web sites, on parked domains, DNS sinkholes, and stolen certificates.

To configure the general options for the Cisco Threat Grid Curated Feed incoming feed, see Configure incoming feeds.

About Cisco Threat Grid Curated Feed

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data
banking-dns	Banking Trojan network communications: (meta)data from communication involving Trojan-infected machines.
dll-hijacking-dns	Data obtained from the analysis of samples leveraging DLL sideloading and/or hijacking techniques
doc-net-com-dns	Document (PDF, Office) network communications: (meta)data from communication involving infected document files.
downloaded-pe-dns	Data obtained from the analysis of samples downloading executables over the network.
dynamic-dns	Data obtained from the analysis of samples leveraging dynamic DNS providers.
irc-dns	Data obtained from Internet Relay Chat (IRC) network communications.
modified-hosts-dns	Information about modified Windows hosts files.
parked-dns	Information about parked domains resolving to **RFC1918** `(https://tools.ietf.org/html/rfc1918)`, localhost and broadcast addresses.
public-ip-check-dns	Check For Public IP Address Network Communications.
ransomware-dns	Data obtained from the analysis of samples communicating with ransomware servers.
rat-dns	Information about remote access Trojans (RAT), and any communications with their Command and Control systems.
sinkholed-ip-dns	DNS entries obtained from the analysis of samples communicating with known DNS sinkholes.
stolen-cert-dns	DNS entries observed in samples signed with a stolen certificate.

The ingested data produce indicators with embedded observables, where each observable represents an indicator of compromise (IOC).

Configure the transport type

Transport type	**Allowed content types**
Cisco Threat Grid Curated Feed	STIX 1.2

- From the **Transport type** drop-down menu, select **Cisco Threat Grid Curated Feed**.
- From the **Content type** drop-down menu, select **STIX 1.2**.
  Currently, the **Cisco Threat Grid Curated Feed** transport type supports only the **STIX 1.2** content type.
- **API URL**: the URL pointing to the API endpoint exposing the service that makes the data available for retrieval through the feed. Contact the intel provider of the incoming feed to obtain this information.
  This field should be pre-populated with *https://panacea.threatgrid.com/api/v2/*.
- **API key**: contact Cisco to receive an API key, and then enter it in the corresponding input field.
- **Feed type**: from the drop-down menu select the data source you want the incoming feed to retrieve data from. The available channels are:
  - **Banking Trojan Network Communications**
  - **Feed contains Domains communicated to by samples leveraging DLL Sideloading and/or hijacking techniques**
  - **Document (PDF, Office) Network Communications**
  - **Samples Downloading Executables Network Communications**
  - **Samples Leveraging Dynamic DNS Providers**
  - **Internet Relay Chat (IRC) Network Communications**
  - **Modified Windows Hosts File Network Communications**
  - **Parked Domains resolving to RFC1918, Localhost and Broadcast Addresses**
  - **Check For Public IP Address Network Communications**
  - **Samples Communicating with Ransomware Servers**
  - **Remote Access Trojan (RAT) Network Communications**
  - **DNS entries for samples communicating with a known dns sinkhole**
  - **DNS Entries observed from samples signed with a stolen certificate**
- **Ingest feed starting from**: select an initial date if you want to fetch content from the intel provider/data source starting from a specific date in the past.
  If you do not specify any start date, the default start date is 6 months in the past. This means that if you leave this field empty, the incoming feed will fetch data as old as 6 months until the present time.

Configure Cisco Threat Grid Samples API transport and content

Set up and configure transport and content types for Cisco Threat Grid Samples API incoming feeds to retrieve and process information on compromised IP addresses, domains, hashes, registry keys, and network streams.

To configure the general options for the Cisco Threat Grid Samples API incoming feed, see Configure incoming feeds.

About Cisco Threat Grid Samples API

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
Samples API	Data obtained from the analysis of submitted samples. Samples can be files, file types, running processes, URLs, IP domains, hashes, and registry keys.	Observables, where each observable represents an indicator of compromise (IOC).

Configure the transport type

Transport type	Allowed content types
Cisco Threat Grid Samples API	Cisco Threat Grid JSON

- From the **Transport type** and **Content type** drop-down lists, choose the appropriate transport and content types for the data, based on the available options.
  The selected content type for the feed shoud match the data source format.
  This can vary, depending on the intel source(s) you retrieve the data from.

The intel provider for the feed is Cisco Threat Grid Samples API.
Under **Transport configuration**, configure the following settings:

- **API URL**: the URL pointing to the API endpoint exposing the service that makes the data available for retrieval through the feed. Contact the intel provider of the incoming feed to obtain this information.
- **API key**: contact Cisco to receive an API key, and then enter it in the corresponding input field.
- **Confidence threshold**: integer. Defines a minimum value to filter out and exclude from the feed any threats whose reliability level, that is, a score representing the trustworthiness of the information, is lower than the specified value.
  - Allowed range: *0-100*
  - Default value: *80*

- **Severity threshold**: integer. Defines a minimum value to filter out and exclude from the feed any threats whose severity level, that is, a score representing the potential risk they may pose, is lower than the specified value.
    - Allowed range: *0-100*
    - Default value: *80*
- **Threat score threshold**: integer. Defines a minimum value to filter out and exclude from the feed any threats ranking lower than the specified value. It represents a threat potential aggressiveness and maliciousness.
    - Allowed range: *0-100*
    - Default value: *80*

Setting thresholds enables conditional execution. For example, if an entity in the feed holds a value exceeding the predefined threshold, it can trigger a process to execute a follow-up action.

- **Ingest samples starting from**: select an initial date if you want to fetch content from the intel provider/data source starting from a specific date in the past.
  If you do not specify any start date, the default start date is 6 months in the past. This means that if you leave this field empty, the incoming feed will fetch data as old as 6 months until the present time.
- By default, the following checkboxes are preselected to include these observables in the feed:
    - **Organization only**: select this checkbox to enable the enricher to check and display only submitted samples created by the organization the current user belongs to. That is, the organization needs to be the author of the submitted samples. When selected, this field is validated against the API key value granting access to the service.
    - **Fetch IPs**: includes samples about compromised IP addresses.
    - **Fetch URLs**: includes samples about compromised URLs.
    - **Fetch domains**: includes samples about compromised domain names.
    - **Fetch artifacts**: includes samples about compromised artifacts; for example, file hash values.
    - **Fetch registry keys**: includes samples about compromised (Windows) registry key data.
    - **Fetch network streams**: includes samples about network stream information exchanged among compromised machines.

Configure Crowdstrike Falcon Intelligence Indicator transport and content

Set up and configure transport and content types for Crowdstrike Falcon Intelligence Indicator incoming feeds to retrieve and process information on indicators, such as compromised devices, malicious domains, hashes, and more.

To configure the general options for the Crowdstrike Falcon Intelligence Indicator incoming feed, see Configure incoming feeds.

About Crowdstrike Falcon Intelligence Indicator

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
Crowdstrike Falcon Intelligence Indicator	Indicators found in their environment, related to entities such as threat actors or other indicators.	Indicators retrieved from the Falcon Intelligence platform such as compromised devices, malicious domains, hashes, and so on starting from the specified polling date.

Configure the transport type

Transport type	Allowed content types
Crowdstrike Falcon Intelligence Indicator Feed	Crowdstrike Indicator JSON

- From the **Transport type** drop-down menu, select **Crowdstrike Falcon Intelligence Indicator Feed**.
- From the **Content type** drop-down menu, select **Crowdstrike Indicator JSON**.
  The **Crowdstrike Falcon Intelligence Indicator** transport type supports only the **Crowdstrike Indicator JSON** content type.

The source organization providing data for the incoming feed is Crowdstrike.
Under **Transport configuration**, configure the following settings:

- **API ID**: contact Crowdstrike to receive an API ID, and then enter it in the corresponding input field.
  You need a valid API ID and a corresponding API key as authentication credentials to access the Crowdstrike Falcon Intel API and to consume it.
- **API key**: contact Crowdstrike to receive an API key, and then enter it in the corresponding input field.
- **Polling start time**: select an initial date if you want to fetch content from the intel provider/data source starting from a specific date in the past.
  By default, the max. amount of days in the past per each query/request is set to 60 days.

Configure Crowdstrike Falcon Intelligence Reports transport and content

Set up and configure transport and content types for Crowdstrike Falcon Intelligence Report incoming feeds to retrieve and process Crowdstrike intelligence reports, periodic reports, and tippers.

To configure the general options for the Crowdstrike Falcon Intelligence Reports incoming feed, see Configure incoming feeds.

About Crowdstrike Falcon Intelligence Reports

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
Crowdstrike Falcon Intelligence Reports	Crowdstrike Intelligence Reports (CSIR), Periodic Reports (CSMR), Tippers (CSIT)	Reports retrieved from the Falcon Intelligence platform in JSON format.

Configure the transport type

Transport type	Allowed content types
Crowdstrike Falcon Intelligence Reports Feed	Crowdstrike report JSON

- From the **Transport type** drop-down menu, select **Crowdstrike Falcon Intelligence Reports Feed**.
- From the **Content type** drop-down menu, select **Crowdstrike report JSON**.
  The **Crowdstrike Falcon Intelligence Reports** transport type supports only the **Crowdstrike report JSON** content type.

The source organization providing data for the incoming feed is Crowdstrike.
Under **Transport configuration**, configure the following settings:

- **API ID**: contact Crowdstrike to receive an API ID, and then enter it in the corresponding input field.
  You need a valid API ID and a corresponding API key as authentication credentials to access the Crowdstrike Falcon Intel API and to consume it.
- **API key**: contact Crowdstrike to receive an API key, and then enter it in the corresponding input field.
- **Polling start time**: select an initial date if you want to fetch content from the intel provider/data source starting from a specific date in the past.
  By default, the max. amount of days in the past per each query/request is set to 60 days.

- Select one or more checkboxes to include one or more report types to ingest through the incoming feed:
    - **Include Intelligence reports**: reports about specific topics, cases, and scenarios.
    - **Include Periodic reports**: a periodic report published on a regular basis and offering up-to-date information on common, recurring threat scenarios.
    - **Include Tippers**: tips and trickes to improve cyber security defenses.

You need a valid subscription to Crowdstrike Falcon Premium Intel to access CrowdStrike intelligence publications.

Configure CVE Search API transport and content

Set up and configure transport and content types for CVE Search API incoming feeds to retrieve and process information on common software and hardware vulnerabilities, along with the corresponding exposures.

To configure the general options for the CVE Search API incoming feed, see Configure incoming feeds.

About CVE Search API

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
CVE Search API	CVE information about common software and hardware vulnerabilities, and exposures.	Exploit target entities. The entity ID is derived from the CVE ID.

Configure the transport type

Transport type	Allowed content types
CVE Search API	CVE Search JSON

- From the **Transport type** drop-down menu, select **CVE Search API**.
- From the **Content type** drop-down menu, select **CVE Search JSON**.
  The **CVE Search API** transport type supports only the **CVE report JSON** content type.

The source organization providing data for the incoming feed is the Computer Incident Response Center Luxembourg (CIRCL).
Under **Transport configuration**, configure the following settings:

- **CVE API URL** : the URL pointing to the API endpoint exposing the service that grants access to the enricher data.
  Contact the intel provider to subscribe to the service and to obtain this information.
  For this incoming feed it corresponds to *https://cve.circl.lu/api/last*.

By default, every time you run the feed, it ingests the most recent 30 CVE entries.
Run this feed on a regular basis to keep up to date with the latest CVE definitions.

Configure FireEye iSIGHT Intelligence Report API transport and content

Set up and configure transport and content types for FireEye iSIGHT Intelligence Report API incoming feeds to retrieve and process STIX reports on vulnerabilities, malware, and other threats such as threat actors, stategies, tactics, and techniques.

To configure the general options for the FireEye iSIGHT Intelligence Report API incoming feed, see Configure incoming feeds.

About FireEye iSIGHT Intelligence Report API

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
FireEye iSIGHT Intelligence Report API	FireEye iSIGHT intelligence reports published since a specific start date.	STIX reports on vulnerabilities, malware, and threats such as threat actors, stategies, tactics, and techniques.

Configure the transport type

Transport type	Allowed content types
FireEye iSIGHT Intelligence Report API	STIX 1.1.1

- From the **Transport type** drop-down menu, select **FireEye iSIGHT Intelligence Report API**.
- From the **Content type** drop-down menu, select **STIX 1.1.1**.

The intel provider for the feed is iSIGHT API.
Under **Transport configuration**, configure the following settings:

- **FireEye API URL**: the URL pointing to the API endpoint exposing the service that grants access to the incoming feed data. Contact the intel provider to subscribe to the service and to obtain this information.
- **API public key**: enter the API public key to authorize your account. To access the iSIGHT API you need to authenticate using a public and a private key.
  Make sure both keys are securely stored.
- **API private key**: enter the API private key to authorize your account. To access the iSIGHT API you need to authenticate using a public and a private key.
  Make sure both keys are securely stored.

- **Ingest feed starting from**: select an initial date if you want to fetch content from the intel provider/data source starting from a specific date in the past.
  By default, the max. amount of days in the past per each query/request is set to 89 days.
  If you set an ingestion start date at a point in time further back in the past, the feed sends multiple requests to retrieve the data.
- **Include Malware intelligence type**: select this checkbox to include in the feed intelligence reports about malware.
- **Include Threat intelligence type** : select this checkbox to include in the feed intelligence reports about threats such as threat actors, stategies, tactics, techniques, campaigns, and so on.
- **Include Vulnerability intelligence type**: select this checkbox to include in the feed intelligence reports about vulnerabilities.

Configure Group-IB transport and content

Set up and configure transport and content types for Group-IB incoming feeds to retrieve and process information on compromised logins and (email) accounts, payment cards, mobile devices, money mules, and so on.

To configure the general options for the Group-IB incoming feed, see Configure incoming feeds.

About Group-IB

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data
accs	Information on compromised login and password credentials.
cards	Information on compromised payment bank cards.
imei	Signatures of compromised or infected mobile devices.
mules	Information on fraudulent bank accounts (money mules).
files	Information on compromised or infected files.
phishing	Information on phishing resources.
ddos	Information on on DDoS attacks.
hacktivism	Information on hacktivist groups and hacktivism.
sample	Information on malicious programs and applications.
malware	Information on malware.
cnc	Indicators to identify Command and Control servers.
threats	Description of threats, including indicators of compromise.
indicator_tor_nodes	Indicators to identify Tor nodes.
indicator_open_proxy	Indicators to identify open proxies.
indicator_socks_proxy	Indicators to identify SOCKS proxies.

Configure the transport type

Transport type	Allowed content types
TAXII poll	STIX 1.1.1

- From the **Transport type** drop-down menu, select **TAXII poll**.
- From the **Content type** drop-down menu, select **STIX 1.1.1**.
- **Accept password protected archives**: select this checkbox to specify a global password to open any archives acquired through the incoming feed.
  If the archives are password-protected, enter it in the **Archive password** field. The specified password acts as a master password, and it is used to unlock all the archives retrieved with the feed.

Under **Transport configuration**, configure the following settings:

- **Auto Discovery**: if applicable, enter the URL of a Group-IB TAXII discovery service the feed can send a request to in order to determine the available services and poll them for updates.
  The current TAXII discovery URL to retrieve the available Group-IB servicess is *https://bt.group-ib.com/taxii/services/discovery*.
- **Polling service URL**: enter the URL of the Group-IB TAXII polling service the feed can check for updates.
  The current TAXII polling URL to retrieve Group-IB content is *https://bt.group-ib.com/taxii/services/poll*.
- **Collection name**: enter the name of the data collection you want to poll and use as a source for the feed.
  The collection you specify here needs to be available through the auto-discovery or the polling services.

  Group-IB offers a range of thematic intel collections focusing on cyber threats and targets such as money mules, compromised banking accounts and payment cards, Tor nodes, and so on.
  Collection availability and access to collections may depend on the terms and conditions of your subscription to Group-IB services.
- **TAXII version**: specify the TAXII version the service uses to publish data.
  Example: *1.1*
- **Extra headers**: click **+ Add** or **+ More** to include in the request additional HTTP headers, along with the corresponding values.
  Group-IB requires the following extra headers:
  - `X-Auth-Key`: use this header to pass the key or user name you can obtain when subscribing to Group-IB services.
  - `X-Auth-Login`: use this header to pass the password you can obtain when subscribing to Group-IB services.
- **Subscription ID**: enter the name, label or ID identifying the Group-IB subscription session that authorizes accessing the source content available for retrieval through the feed.
- **Ingest messages starting from**: select an initial date if you want to fetch content from the intel provider/data source starting from a specific date in the past.
- **Days per poll**: if you select a start date to poll data from, you can enter an integer to specify the maximum number of days to poll at a time.
  This enables polling in multiple smaller batches, instead of a single batch, starting from the selected initial date.
  This option works only if you select an ingestion start date.
- **Verify SSL**: if the Group-IB TAXII server requires an SSL certificate to authenticate and to access the corresponding TAXII services, you can select this checkbox to test the SSL connection and to verify that it works correctly.
  - **SSL CA bundle file path**: enter the path to the CA bundle file containing the root, intermediate, and public certificates for the SSL authentication.
- **Basic authentication**: leave the checkbox deselected. You do not need to configure basic authentication for this STIX/TAXII incoming feed setup, since Group-IB requires passing authentication credentials through extra HTTP headers.

- **SSL certificate authentication**: if the TAXII server requires an SSL certificate to authenticate and to access the corresponding TAXII services, select this checkbox to fill out the required information.
    - **SSL certificate**: copy-paste the content of a valid SSL certificate to authenticate.
      Example:

```
-----BEGIN CERTIFICATE REQUEST-----
MIICvDCCAaQCAQAwdzELMAkGA1UEBhMCVVMxDTALBgNVBAgMBFV0YWgxDzANBgNV
BAcMBkxpbmRvbjEWMBQGA1UECgwNRGlnaUNlcnQgSW5jLjERMA8GA1UECwwIRGln
aUNlcnQxHTAbBgNVBAMMFGV4YW1wbGUuZGlnaWNlcnQuY29tMIIBIjANBgkqhkiG
9w0BAQEFAAOCAQ8AMIIBCgKCAQEA8+To7d+2kPWeBv/orU3LVbJwDrSQbeKamCmo
wp5bqDxIwV20zqRb7APUOKYoVEFFOEQs6T6gImnIolhbiH6m4zgZ/CPvWBOkZc+c
1Po2EmvBz+AD5sBdT5kzGQA6NbWyZGldxRthNLOs1efOhdnWFuhI162qmcflgpiI
WDuwq4C9f+YkeJhNn9dF5+owm8cOQmDrV8NNdiTqin8q3qYAHHJRW28glJUCZkTZ
wIaSR6crBQ8TbYNE0dc+Caa3DOIkz1EOsHWzTx+n0zKfqcbgXi4DJx+C1bjptYPR
BPZL8DAeWuA8ebudVT44yEp82G96/Ggcf7F33xMxe0yc+Xa6owIDAQABoAAwDQYJ
KoZIhvcNAQEFBQADggEBAB0kcrFccSmFDmxox0Ne01UIqSsDqHgL+XmHTXJwre6D
hJSZwbvEtOK0G3+dr4Fs11WuUNt5qcLsx5a8uk4G6AKHMzuhLsJ7XZjgmQXGECpY
Q4mC3yT3ZoCGpIXbw+iP3lmEEXgaQL0Tx5LFl/okKbKYwIqNiyKWOMj7ZR/wxWg/
ZDGRs55xuoeLDJ/ZRFf9bI+IaCUd1YrfYcHIl3G87Av+r49YVwqRDT0VDV7uLgqn
29XI1PpVUNCPQGn9p/eX6Qo7vpDaPybRtA2R7XLKjQaF9oXWeCUqy1hvJac9QFO2
97Ob1alpHPoZ7mWiEuJwjBPii6a9M9G30nUo39lBi1w=
-----END CERTIFICATE REQUEST-----
```

- **SSL key**: copy-paste the content of a valid SSL key to authenticate
  Example:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEA3Tz2mr7SZiAMfQyuvBjM9Oi..Z1BjP5CE/Wm/Rr500P
RK+Lh9x5eJPo5CAZ3/ANBE0sTK0ZsDGMak2m1g7..3VHqIxFTz0Ta1d+NAj
wnLe4nOb7/eEJbDPkk05ShhBrJGBKKxb8n104o/..PdzbFMIyNjJzBM2o5y
5A13wiLitEO7nco2WfyYkQzaxCw0AwzlkVHiIyC..71pSzkv6sv+4IDMbT/
XpCo8L6wTarzrywnQsh+etLD6FtTjYbbrvZ8RQM..Hg2qxraAV++HNBYmNW
kbJ+q+rsJxQlaipn2M4lGuQJEfIxELFDyd3XpxP..Un/82NZNXlPmRIopXs
2T91jiLZEUKQw+n73j26adTbteuEaPGSrTZxBLR..yssO0wWomUyILqVeti
+PK+aXKwguI6bxLGZ3of0UH+mGsSl0mkp7kYZCm..OTQtfeRqP8rDSC7DgA
kHc5ajYqh04AzNFaxjRo+M3IGICUaOdKnXd0Fda..QwfoaX4QlRTgLqb7AN
ZTzM9WbmnYoXrx17kZlT3lsCgYEAm757XI3WJVj..WoLj1+v48WyoxZpcai
uv9bT4Cj+lXRS+gdKHK+SH7J3x2CRHVS+WH/SVC..DxuybvebDoT0TkKiCj
BWQaGzCaJqZa+POHK0klvS+9ln0/6k539p95tfX..X4TCzbVG6+gJiX0ysz
Yfehn5MCgYEAkMiKuWHCsVyCab3RUf6XA9gd3qY..fCTIGtS1tR5PgFIV+G
engiVoWc/hkj8SBHZz1n1xLN7KDf8ySU06MDggB..hJ+gXJKy+gf3mF5Kmj
DtkpjGHQzPF6vOe907y5NQLvVFGXUq/FIJZxB8k..fJdHEm2M4=
-----END RSA PRIVATE KEY-----
```

- **SSL key password**: enter the SSL password or passphrase for the SSL key. This field is masked.

Configure Intel 471 transport and content types

Set up and configure transport and content types for Intel 471 incoming feeds to retrieve and process information on compromised logins and (email) accounts, payment cards, and mobile devices.

To configure the general options for the Intel 471 API incoming feed, see Configure incoming feeds.

About Intel 471 API

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
Intel 471 API	The channel provides data on compromised IP addresses, domains, URLs emails, and actors.	Reports containing references to the information source, any additional comments by researchers or threat analysts, and any raw text content. Compromised domains, URLs, actors and so on are included in the report as observables.

⚠ **Warning:** You should set the **Override TLP** option to **Amber** for Intel 471 feed content, since this information should not be redistributed outside the organization.

Configure the transport type

Transport type	Allowed content types
Intel 471 API	Intel 471

The intel provider for the feed is Intel 471.
Under **Transport configuration**, configure the following settings:

- **API URL**: the URL pointing to the API endpoint exposing the service that makes the data available for retrieval through the feed. Contact the intel provider of the incoming feed to obtain this information.
- **API key**: contact Intel 471 to receive an API key, and then enter it in the corresponding input field.
- **Email**: a valid email address to be granted access to the Intel 471 API endpoint to fetch the incoming feed from.
- **Ingest reports starting from**: select an initial date if you want to fetch content from the intel provider/data source starting from a specific date in the past.

Configure PhishMe Intelligence transport and content

Set up and configure transport and content types for PhishMe Intelligence incoming feeds to retrieve and process PhishMe intelligence reports on malware and phishing campaigns.

To configure the general options for the PhishMe Intelligence incoming feed, see Configure incoming feeds.

About PhishMe Intelligence

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
PhishMe Intelligence	PhishMe Intelligence reports published since a specific start date.	STIX reports focusing on malware and phishing campaigns.

Configure the transport type

Transport type	Allowed content types
PhishMe Intelligence API	STIX 1.1.1

- From the **Transport type** drop-down menu, select **PhishMe Intelligence API**.
- From the **Content type** drop-down menu, select **STIX 1.1.1**.
  The **PhishMe Intelligence API** transport type supports only the **STIX 1.1.1** content type, which is essentially standard STIX 1.1.1.

The source organization providing data for the incoming feed is PhishMe.
Under **Transport configuration**, configure the following settings:

- **PhishMe Intelligence API URL**: the URL pointing to the API endpoint exposing the service that grants access to the incoming feed data. Contact the intel provider to subscribe to the service and to obtain this information.
  This field should be pre-populated with *https://www.threathq.com*.
- **API Username**: a valid user name to authenticate and be granted the necessary authorization to access the data source and to download/ingest data.
- **API Password**: a valid password to authenticate and be granted the necessary authorization to access the data source and to download/ingest data.
  Contact the intelligence provider to subscribe to the service and to obtain the required authentication and authorization credentials.
- **Ingest feed starting from**: select an initial date if you want to fetch content from the intel provider/data source starting from a specific date in the past.
  By default, the max. amount of days in the past per each query/request is set to 365 days/one year.

Configure SpyCloud Breach API transport and content

Set up and configure transport and content types for SpyCloud Breach API incoming feeds to retrieve and process information on incidents, security breaches, and account takeovers (ATO).

To configure the general options for the SpyCloud Breach API incoming feed, see Configure incoming feeds.

About SpyCloud Breach API

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
SpyCloud Breach API	Incident and breach data, along with relevant context.	Incident entities focusing on security breaches and account takeovers, CIQ entities, CybOX observables, related observables. Where available, context data include threat actor, targeted victim, affected assets, and geolocation details.

Configure the transport type

Transport type	Allowed content types
SpyCloud Breach API	SpyCloud JSON

- From the **Transport type** drop-down menu, select **SpyCloud Breach API**.
- From the **Content type** drop-down menu, select **SpyCloud JSON**.
  The **SpyCloud Breach API** transport type supports only the **SpyCloud JSON** content type.

Configure Threat Recon transport and content

Set up and configure transport and content types for Threat Recon incoming feeds to retrieve and process indicators with information on compromised and malicious IP addresses, domains, aa well as whois information.

To configure the general options for the Threat Recon JSON API incoming feed, see Configure incoming feeds.

About Threat Recon JSON API

This intel provider/intel source enables intelligence ingestion through the following channels:

Feed	Ingested data	Processed data
Threat Recon JSON API	The channel provides data on compromised IP addresses, domains, as well as whois information.	Indicators with embedded observables, where each observable represents an indicator of compromise (IOC).

Configure the transport type

Transport type	Allowed content types
Threat Recon JSON API	Threat Recon

- From the **Transport type** drop-down menu, select **TAXII poll**.
- From the **Content type** drop-down menu, select **STIX 1.1.1**.

The intel provider for the feed is Threat Recon.
Under **Transport configuration**, configure the following settings:

- **API key**: contact Threat Recon to receive an API key, and then enter it in the corresponding input field.

Incoming feeds reference

Reference section with lookup information on supported incoming feeds, content types, and transport types.

Available incoming feeds

The overview lists and points to the articles on the available incoming feeds. Each article describes how to configure the specific options for each incoming feed.
Typically, incoming feeds use different *transport types* and *content types*. The other configuration options are identical across all incoming feeds.

Title	Excerpt
Configure Anubis Cyberfeed transport and content	Set up and configure transport and content types for AnubisNetworks Cyberfeed incoming feeds to retrieve and process information on compromised machines and DNS servers, as well as compromised web ...
Configure BFK transport and content	Set up and configure transport and content types for BFK incoming feeds to retrieve and process reports on cyber threats and activities, as well as information on NIDs (Network Intrusion Detections).
Configure Crowdstrike Falcon Intelligence Indicator transport and content	Set up and configure transport and content types for Crowdstrike Falcon Intelligence Indicator incoming feeds to retrieve and process information on indicators, such as compromised devices, malicio...
Configure Crowdstrike Falcon Intelligence Reports transport and content	Set up and configure transport and content types for Crowdstrike Falcon Intelligence Report incoming feeds to retrieve and process Crowdstrike intelligence reports, periodic reports, and tippers.
Configure CVE Search API transport and content	Set up and configure transport and content types for CVE Search API incoming feeds to retrieve and process information on common software and hardware vulnerabilities, along with the corresponding ...
Configure IMAP email fetcher transport and content	Set up and configure transport and content types for IMAP email fetcher incoming feeds to retrieve and process information from email body contents.
Configure IMAP email attachment fetcher transport and content	Set up and configure transport and content types for IMAP email attachment fetcher incoming feeds to retrieve email attachment content in multiple formats. You can then use the resulting reports fo...
Configure FireEye iSIGHT Intelligence Report API transport and content	Set up and configure transport and content types for FireEye iSIGHT Intelligence Report API incoming feeds to retrieve and process STIX reports on vulnerabilities, malware, and other threats such a...

Title	Excerpt
Configure FTP download transport and content	Set up and configure transport and content types for FTP download incoming feeds to retrieve and process information from specific data sources supporting the FTP protocol.
Configure Group-IB transport and content	Set up and configure transport and content types for Group-IB incoming feeds to retrieve and process information on compromised logins and (email) accounts, payment cards, mobile devices, money mul...
Configure HTTP download transport and content	Set up and configure transport and content types for HTTP download incoming feeds to retrieve and process information from specific data sources supporting the HTTP protocol.
Configure Intel 471 transport and content types	Set up and configure transport and content types for Intel 471 incoming feeds to retrieve and process information on compromised logins and (email) accounts, payment cards, and mobile devices.
Configure mount point download transport and content	Set up and configure transport and content types for mount point download incoming feeds to retrieve and process information from specific data sources available on the (local) network.
Configure PhishMe Intelligence transport and content	Set up and configure transport and content types for PhishMe Intelligence incoming feeds to retrieve and process PhishMe intelligence reports on malware and phishing campaigns.
Configure SpyCloud Breach API transport and content	Set up and configure transport and content types for SpyCloud Breach API incoming feeds to retrieve and process information on incidents, security breaches, and account takeovers (ATO).
Configure TAXII inbox transport and content	Set up and configure transport and content types for TAXII inbox incoming feeds to retrieve and process information from specific data sources supporting the TAXII transport type.
Configure TAXII poll transport and content	Set up and configure transport and content types for TAXII poll incoming feeds to poll discovery services toretrieve and process information from specific data sources supporting the TAXII transpor...
Configure Cisco Threat Grid Samples API transport and content	Set up and configure transport and content types for Cisco Threat Grid Samples API incoming feeds to retrieve and process information on compromised IP addresses, domains, hashes, registry keys, an...
Configure Cisco Threat Grid Curated Feed transport and content	Set up and configure transport and content types for Cisco Threat Grid Curated Feed incoming feeds to retrieve and process information on Trojan-infected machines and servers, on malware-infected D...
Configure Threat Recon transport and content	Set up and configure transport and content types for Threat Recon incoming feeds to retrieve and process indicators with information on compromised and malicious IP addresses, domains, aa well as w...

Content types

These are the data formats the platform can process through feeds.
Under *Feed type* **in** defines an input format that incoming feeds ingest; **out** defines an output format that outgoing feeds publish.

Content type	Feed type	Description
Anubis Cyberfeed JSON	in	JSON format representing entity data as JSON objects.
ArcSight CEF	out	The Common Event Format is a text-based standard for log records proposed by ArcSight. It allows sharing, consuming, and parsing event information across devices such as SIEM platforms and Syslog servers.
Cisco Threat Grid Samples JSON	in	JSON format representing entity data as JSON objects.
EclecticIQ Entities CSV	out	Comma separated CSV format for tabular data representation of entities.
EclecticIQ JSON	in, out	JSON format representing entity data as JSON objects.
EclecticIQ Observables CSV	out	Comma separated CSV format for tabular data representation of observables.
Group-IB accounts, Group-IB cards, Group-IB IMEIs	in	Group-IB proprietary data format to exchange information on compromised accounts, payment cards, and mobile devices.
Intel 471	in	Intel 471 proprietary data format.
PDF	in, out	Standard PDF format, preferably native (not scanned).
STIX 1.0	in, out	STIX data model **v. 1.0** `(http://stixproject.github.io/data-model/1.0/).`
STIX 1.1	in, out	STIX data model **v. 1.1** `(http://stixproject.github.io/data-model/1.1/).`
STIX 1.1.1	in, out	STIX data model **v. 1.1.1** `(http://stixproject.github.io/data-model/1.1.1/).`
STIX 1.2	in, out	STIX data model **v. 1.2** `(http://stixproject.github.io/data-model/1.2/).`
Text/Plain text value	in, out	Plain text format. This content type allows you to enter free text and literals, wildcards (where supported), as well as JSON paths to point to specific entity property fields, and regex patterns to filter data.
Threat Recon	in	Threat Recon JSON output returned by the **Threat Recon API** `(https://threatrecon.co/api)`. Threat Recon focuses on providiung information about indicators.
STIX 1.1.1	in	FireEye iSIGHT Intelligence Report API outputs reports in STIX 1.1.1 format. Reports concern threat topics such as vulnerabilities, malware, threat actors, stategies, tactics, and techniques.

Content type	Feed type	Description
BFK Threat Intelligence JSON	in	BFK reports and NIDs (Network Intrusion Detections) are saved as JSON report entities; they concern threat topics such as threat actors, targeted victims, tactics, and techniques.
Crowdstrike Indicator JSON	in	Indicators retrieved from the Falcon Intelligence platform are stored as JSON; they concern compromised devices, malicious domains, hashes, and so on starting from a specified polling date.

Transport types

These are the supported communications protocols the platform uses to ingest data through incoming feeds.

Transport type	Feed type	Description
Anubis Cyberfeed	in	Provides data on bank Trojans, compromised DNS servers, malware-infected web site and malware files.
Cisco Threat Grid Curated Feed	in	Provides data on compromised IP addresses, domains, hashes, registry keys, and network streams.
Cisco Threat Grid Samples API	in	Allows submitting malware samples for analysis, as well as investigating a domain, an IP, or a URL to obtain information about potential threats.
FTP download	in	Custom feed ingesting data through FTP.
TAXII poll for Group-IB	in	The Group-IB TAXII poll service offers a range of channels that provide information and comprehensive coverage on cyber threats and vulnerabilities such as compromised logins, passwords, corporate email accounts, compromised bank card numbers and online banking keys, compromised mobile devices, money mules, Tor nodes, and so on.
HTTP download	in	Custom feed ingesting data through HTTP.
IMAP email fetcher	in	Custom feed using the IMAP email protocol to ingest data included in emails as attachments.
Intel 471 API	in	Provides data on compromised IP addresses, domains, URLs emails, with a focus on threat actors.
Mount point download	in	Custom feed using a local or a network drive as a data source.
TAXII inbox	in	Custom feed ingesting data through the TAXII inbox service.
TAXII poll	in	Custom feed ingesting data through the TAXII poll service.
Threat Recon JSON API	in	Provides data on compromised IP addresses, domains, as well as whois information.

Transport type	Feed type	Description
FireEye iSIGHT Intelligence Report API	in	Provides intelligence reports on vulnerabilities, malware, and other threats such as threat actors, stategies, tactics, and techniques.
IMAP email attachment fetcher	in	Ingests email attachments that are then transformed into reports. The reports can be used for analysis (human consumption) or instrumentation (machine consumption/automation).
BFK API	in	Ingested BFK reports are saved as report entities in the platform, whereas ingested NIDs (Network Intrusion Detections) result in entities with a linked TTP.
Crowdstrike Falcon Intelligence Indicator Feed	in	Ingests information about indicators by polling the Falcon Intel Indicator API. It is possible to set a start date to poll data from.
Crowdstrike Falcon Intelligence Reports Feed	in	Ingests Crowdstrike Intelligence Reports (CSIR), Threat Assessments (CSTA), Alerts (CSA), Periodic Reports (CSMR), Tippers (CSIT) in JSON format.
PhishMe Intelligence API	in	Ingests PhishMe reports focusing on malware and phishing campaigns.
CVE Search API	in	Ingests CVE information about common software and hardware vulnerabilities, along with the corresponding exposures. The enricher contacts the Computer Incident Response Center Luxembourg (CIRCL) cve-search API to retrieve all the available details associated with the input CVE IDs. Ingested data is stored in the platform as exploit target entities. The exploit target ID is derived from the corresponding source CVE.
SpyCloud Breach API	in	Ingests incident and security breach information from the SpyCloud breach catalog, along with context-relevant metadata such as affected sites, number of records ingested from the breach, if and when the stolen records were put for sale online, and so on. Searches for breach records matching predefined inputs such as domain name, IP address, or e-mail address. Ingested data is stored in the platform as incident entities, **CIQ 3.0** `(http://docs.oasis-open.org/ciq/v3.0/specs/ciq-specs-v3.html)` entities, CybOX observables, and related observables.

Configure enrichers

Configure enrichers to augment intelligence value with additional context obtained from selected intel providers and data sources.

Enrichers poll external data sources to provide additional context and detail to augment — hence, enrich — the intelligence value of entities and observables stored in the platform.

The platform ships with a broad selection of built-in, ready-to-use enrichers to obtain geolocation IP and whois details, DNS domain and malware information, as well as other relevant data to help analysts draw a sharper and more comprehensive picture of the cyber threat relationships and the cyber threat scenarios under investigation.

This article describes how to configure the general options for all enrichers. These options are identical for all standard platform enrichers.

Besides the general options, each enricher has specific configuration settings. For example, API endpoints, API keys, login credentials to access a subscription service or a data source, and so on.

The configuration options that are specific to each enricher are grouped together under the **Parameters** section on the enricher configuration page.
For the configuration details, follow the links under Configure specific enricher parameters.

Configure the general options

To configure or to edit an enricher task, do the following:

- On the top navigation bar click **Data configuration > Enrichers**.
- On the enricher overview page click the tile corresponding to the enricher you want to configure or modify.
- On the enricher detail page click the **Edit** button.

✓ Input fields marked with an asterisk are required.

- **Name**: the name used to identify the enricher. It should be descriptive and easy to remember.
- **Description**: additional textual details. If you want, you can add a short description to provide more information and context.
- **Cache validity (sec)**: defines for how long enrichment data remains stored in the cache. The value is expressed in seconds.
- **Rate limit (per sec)**: sets the maximum allowed number of requests/executions per second.
- **Monthly execution cap (executions)**: sets a maximum allowed number of requests/executions per month.
  Together with rate limiting, execution cap helps control data traffic for the enricher; for example, when the API or the service you are connecting to enforces usage limits.
- **Source reliability**: from the drop-down menu select an option to flag the feed or enricher content with a predefined reliability value to help other users assess how trustworthy the data source is.
  Values in this menu have the same meaning as the first character in the **two-character Admiralty System code** `(https://en.wikipedia.org/wiki/admiralty_code)`.
  Example: *B - Usually reliable*

- **Enabled**: checkbox. Select the **Enabled** checkbox to enable the enricher task immediately after editing and saving it.
  If you select the checkbox, the enricher or the rule is executed automatically.
  If you deselect it, you need to manually run the enricher or the rule.

Under **Parameters**, define the specific configuration parameters for the selected enricher.

Edit enricher tasks

To configure or to edit an enricher task, do the following:

- On the top navigation bar click **Data configuration > Enrichers**.
- On the enricher overview page click the tile corresponding to the enricher you want to configure or modify.
- On the enricher detail page click the **Edit** button.

✓ Input fields marked with an asterisk are required.

- **Name**: the name used to identify the enricher. It should be descriptive and easy to remember.
- **Description**: additional textual details. If you want, you can add a short description to provide more information and context.
- **Cache validity (sec)**: defines for how long enrichment data remains stored in the cache. The value is expressed in seconds.
- **Rate limit (per sec)**: sets the maximum allowed number of requests/executions per second.
- **Monthly execution cap (executions)**: sets a maximum allowed number of requests/executions per month.
  Together with rate limiting, execution cap helps control data traffic for the enricher; for example, when the API or the service you are connecting to enforces usage limits.
- **Source reliability**: from the drop-down menu select an option to flag the feed or enricher content with a predefined reliability value to help other users assess how trustworthy the data source is.
  Values in this menu have the same meaning as the first character in the **two-character Admiralty System code** `(https://en.wikipedia.org/wiki/admiralty_code).`
  Example: *B - Usually reliable*
- **Enabled**: checkbox. Select the **Enabled** checkbox to enable the enricher task immediately after editing and saving it.
  If you select the checkbox, the enricher or the rule is executed automatically.
  If you deselect it, you need to manually run the enricher or the rule.

Under **Parameters**, define the specific configuration parameters for the selected enricher.

ℹ Some enrichers may require a paid subscription to the data service provider to be granted access to the source data.
Contact the intel service provider whose data you want to use as a source for the enricher to request the necessary details, such as API endpoint or API key, as well as any required authentication and authorization credentials.

Configure enricher rules

Enrichment rules define what to do with the retrieved enrichment data.
Rules act like filters, and they set the logical constraints defining:

- The platform data sources to augment with enrichment information.
  Data sources you can enrich are incoming feeds, other enrichers, and groups.
- Within the selected platform data sources, the entity type(s) to augment with enrichment information.
- The enrichers to use to fetch enrichment data.

Add enricher rules

To add a new enricher rule, do the following:

- On the top navigation bar click **Data configuration > Rules > Enrichment** .
- The **Rules > Enrichment** page shows an overview of the configured enricher rules.
  You can sort the items on the view by column header. To do so, click the column header you want to base the data sorting on. An upward-pointing ▲ or a downward-pointing ▼ arrow in the header indicates ascending and descending sort order, respectively.
- Click the ✚ icon.

> ✔ Input fields marked with an asterisk are required.

On the **Create enrichment rule** page, fill out the fields to create the new enricher rule:

- **Name**: define a name to identify the rule. It should be descriptive and easy to remember.
- **Description**: additional textual details. If you want, you can add a short description to provide more information and context.
- Click ✚ **Add** or ✚ **More** to add a filtering option.
- **Source**: from the drop-down menu select the incoming feed, enricher, or group whose entities and observables you want to augment with additional information.
- **Entity types**: from the drop-down menu select the entity types you want to enrich with additional information.
- **TLP**: from the drop-down menu select the TLP color code you want to use to filter enrichment data.
  TLP `(https://www.us-cert.gov/tlp)` provides an intuitive reference to assess how sensitive information is, focusing in particular on how serious it is, and whom it should or should not be shared with.
- Click ✚ **Add** or ✚ **More** to add a new filtering option. For example, to include another incoming feed or a different entity type. A filter can take only one source and one entity type at a time, but you can set up rules with as many filters as you need.
- **Enrichers**: from the drop-down menu select one or more enrichers to apply the rule to.
  When a rule is applied to one or more enrichers, it filters the enrichment data polled from the enricher source, based on the specified rule filters and criteria.
- Select the **Enabled** checkbox to enable the rule immediately after creating it.

- Click **Save** to store your changes, or **Cancel** to discard them.

Save options

Besides committing current data by clicking **Save**, you can also click the downward-pointing arrow on the **Save** button to display a context menu with additional save options:

- **Save and new**: saves the current data for the active item, and it allows you to start creating a new item of the same type right away. For example, a dataset, a feed, a rule, a workspace, or a task.
- **Save and duplicate**: saves the current data for the active item, and it creates a pre-populated copy of the same item, which you can use as a template to speed up manual creation work.

Edit enricher rules

To edit enricher rules, do the following:

- On the top navigation bar click **Data configuration > Rules > Enrichment**.
- The **Rules > Enrichment** page shows an overview of the configured enricher rules.
  You can sort the items on the view by column header. To do so, click the column header you want to base the data sorting on. An upward-pointing ▲ or a downward-pointing ▼ arrow in the header indicates ascending and descending sort order, respectively.

To edit the details of a specific rule, do the following:

- Click an area on the row corresponding to the rule you want to examine. An overlay slides in from the side of the screen to display the rule detail pane.
- On the rule detail pane click **Actions > Edit**.

Alternatively:

- Click the ⋮ icon on the row corresponding to the enricher you want to configure or modify.
- From the drop-down menu select **Edit**.

> ✔ Input fields marked with an asterisk are required.

- **Name**: define a name to identify the rule. It should be descriptive and easy to remember.
- **Description**: additional textual details. If you want, you can add a short description to provide more information and context.
- **Source**: from the drop-down menu select the incoming feed, enricher, or group whose entities and observables you want to augment with additional information.
- **Entity types**: from the drop-down menu select the entity types you want to enrich with additional information.
- **TLP**: from the drop-down menu select the TLP color code you want to use to filter enrichment data.
  TLP `(https://www.us-cert.gov/tlp)` provides an intuitive reference to assess how sensitive information is, focusing in particular on how serious it is, and whom it should or should not be shared with.
- Click ✚ **Add** or ✚ **More** to add a new filtering option. For example, to include another incoming feed or a different entity type.

- **Enrichers**: from the drop-down menu select one or more enrichers to apply the rule to. They are external data providers that are polled to obtain relevant enricher raw data; for example, whois lookup, reverse DNS, or GeoIP information.
- Select the **Enabled** checkbox to enable the rule immediately after creating it.
- Click **Save** to store your changes, or **Cancel** to discard them.

Delete enricher rules

To delete an enricher rule, do the following:

- On the top navigation bar click **Data configuration > Rules > Enrichment**.
- The **Rules > Enrichment** page shows an overview of the configured enricher rules.
  You can sort the items on the view by column header. To do so, click the column header you want to base the data sorting on. An upward-pointing ▲ or a downward-pointing ▼ arrow in the header indicates ascending and descending sort order, respectively.

To delete a specific rule, do the following:

- Click an area on the row corresponding to the rule you want to delete. An overlay slides in from the side of the screen to display the rule detail pane.
- On the rule detail pane click **Actions > Delete**.

Alternatively:

- Click the ⋮ icon on the row corresponding to the rule you want to delete.
- From the drop-down menu select **Delete**.
- On the confirmation pop-up dialog, click **Delete** to confirm the action.
- The rule is deleted.

Configure specific enricher parameters

- Censys enricher
- CIRCL IPs related to SSL certificate enricher
- CIRCL SSL Certificate Fetcher enricher
- Cisco Threat Grid enricher
- Crowdstrike Falcon Intelligence Indicator enricher
- CVE Search enricher
- DomainTools Hosted Domains enricher
- DomainTools Malicious Server Domains enricher
- DomainTools Parsed Whois enricher
- DomainTools Reputation enricher
- DomainTools Reverse Whois enricher

- DomainTools Suspicious Domains enricher
- Elasticsearch sightings enricher
- Create custom enrichers
- Farsight DNSDB enricher
- FireEye iSIGHT enricher
- Flashpoint AggregINT enricher
- Flashpoint Blueprint enricher
- Flashpoint Thresher enricher
- Fox-IT InTELL Portal enricher
- Intel 471 enricher
- OpenDNS OpenResolve enricher
- PassiveTotal Passive DNS enricher
- PassiveTotal IP/Domain enricher
- PassiveTotal Malware enricher
- PassiveTotal Whois enricher
- PyDat enricher
- Recorded Future enricher
- RIPEstat GeoIP enricher
- RIPEstat Whois enricher
- Shodan enricher
- Splunk sightings enricher
- SpyCloud Breach Data enricher
- ThreatCrowd enricher
- Unshorten-URL enricher
- VirusTotal enricher

Enable and disable enrichers

Enable and disable enrichers to start and stop enrichment tasks as needed.

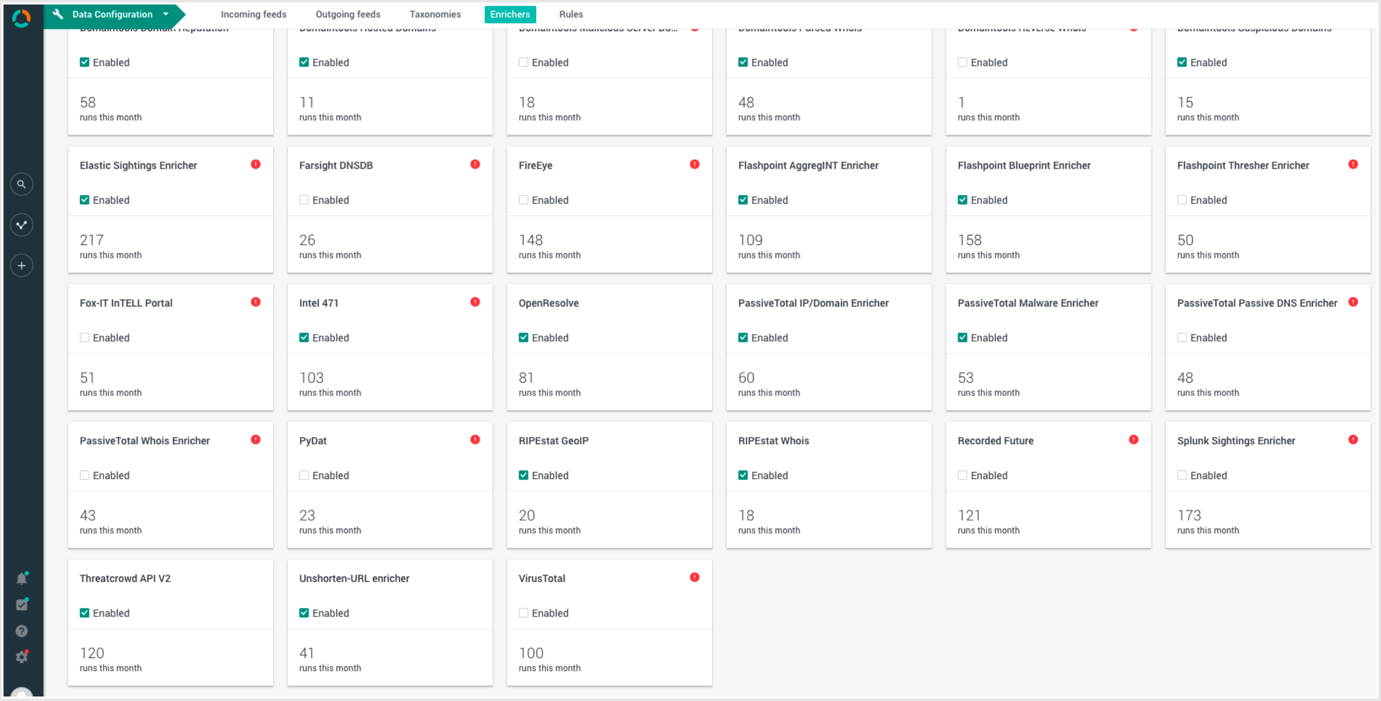
View enricher tasks

To view enricher tasks, do the following:

- On the top navigation bar click **Data configuration > Enrichers** .
- On the enricher overview page click the enricher you want to examine.
- On the enricher detail page, you can view all the details about the selected enricher, including the rules driving the enricher behavior, recently executed enriching tasks, and the state.
- You can click the state value or an enrichment rule to display additional information.

> When the state value returns **FAILURE**, click the link to view the task execution traceback and to begin troubleshooting.

The **Data configuration > Enrichers** view shows all configured enrichers polling third-party and/or external services to acquire additional information to integrate observables with, so that they can provide more context to the cyber threat entities they belong to.

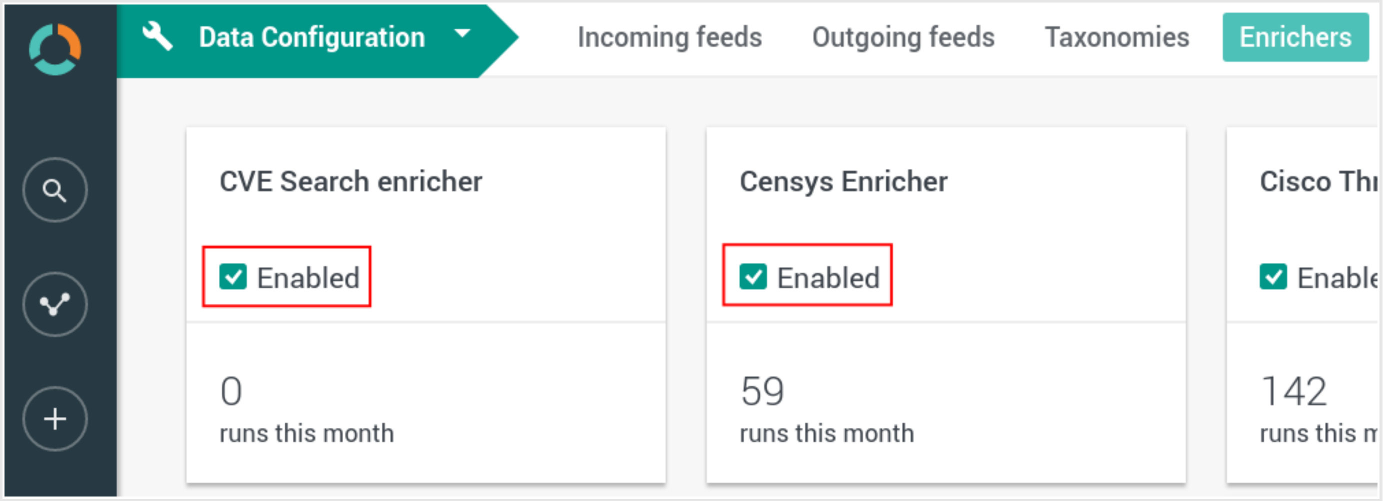


☑ Enabled 58 runs this month	☑ Enabled 11 runs this month	☐ Enabled 18 runs this month	☑ Enabled 48 runs this month	☐ Enabled 1 runs this month	☑ Enabled 15 runs this month
Elastic Sightings Enricher ☑ Enabled 217 runs this month	Farsight DNSDB ☐ Enabled 26 runs this month	FireEye ☐ Enabled 148 runs this month	Flashpoint AggregINT Enricher ☑ Enabled 109 runs this month	Flashpoint Blueprint Enricher ☑ Enabled 158 runs this month	Flashpoint Thresher Enricher ☐ Enabled 50 runs this month
Fox-IT InTELL Portal ☐ Enabled 51 runs this month	Intel 471 ☑ Enabled 103 runs this month	OpenResolve ☑ Enabled 81 runs this month	PassiveTotal IP/Domain Enricher ☑ Enabled 60 runs this month	PassiveTotal Malware Enricher ☑ Enabled 53 runs this month	PassiveTotal Passive DNS Enricher ☐ Enabled 48 runs this month
PassiveTotal Whois Enricher ☐ Enabled 43 runs this month	PyDat ☐ Enabled 23 runs this month	RIPEstat GeoIP ☑ Enabled 20 runs this month	RIPEstat Whois ☑ Enabled 18 runs this month	Recorded Future ☐ Enabled 121 runs this month	Splunk Sightings Enricher ☐ Enabled 173 runs this month
Threatcrowd API V2 ☑ Enabled 120 runs this month	Unshorten-URL enricher ☑ Enabled 41 runs this month	VirusTotal ☐ Enabled 100 runs this month			

Enable and disable enrichers

To enable an enricher so that it runs automatically, do the following:

- On the top navigation bar click **Data configuration > Enrichers** .
- On the enricher overview page browse to the enricher you want to enable.
- Select the **Enabled** checkbox to enable the enricher so that it runs automatically.



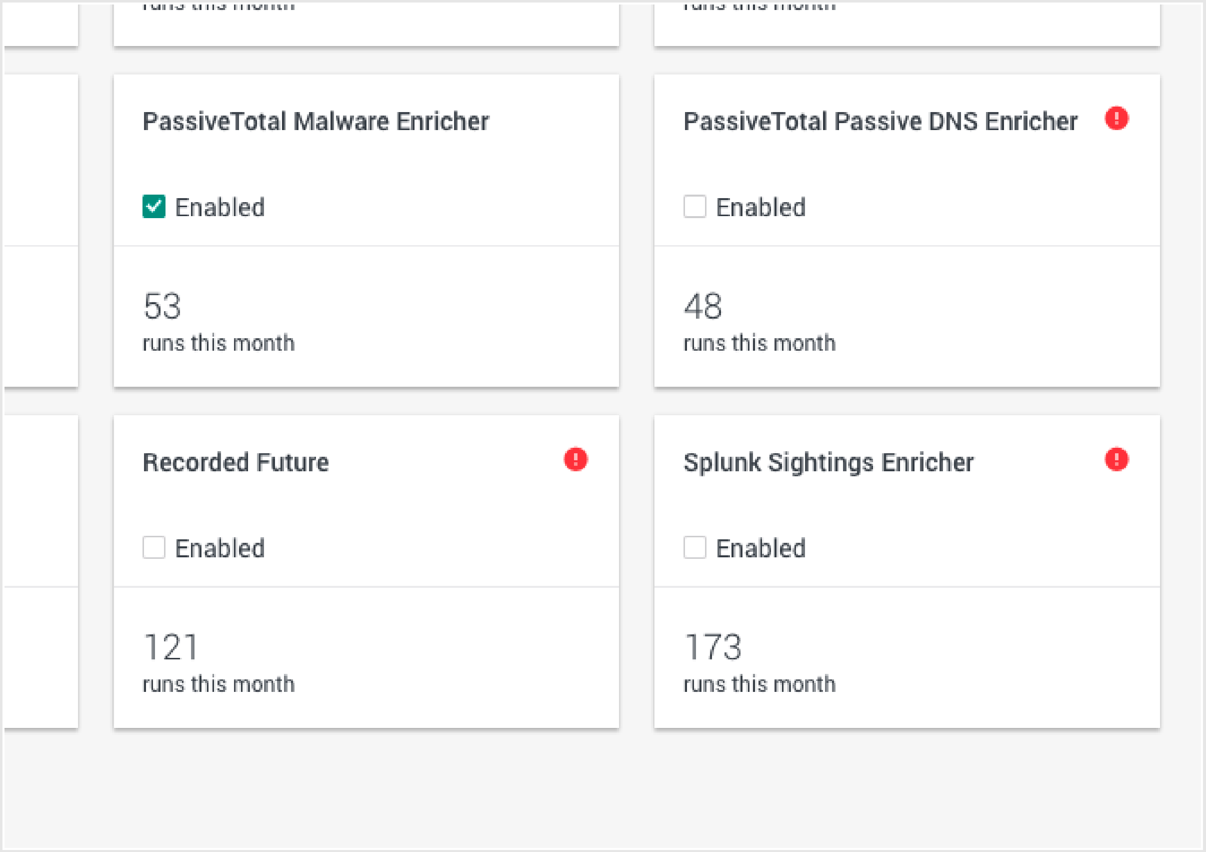
CVE Search enricher
☑ Enabled
0
runs this month

Censys Enricher
☑ Enabled
59
runs this month

Cisco Thr
☑ Enable
142
runs this m

The enricher task is active, and the enricher runs automatically.

To disable an enricher and prevent it from running, deselect the corresponding **Enabled** checkbox.



PassiveTotal Malware Enricher
☑ Enabled
53
runs this month

PassiveTotal Passive DNS Enricher
☐ Enabled
48
runs this month

Recorded Future
☐ Enabled
121
runs this month

Splunk Sightings Enricher
☐ Enabled
173
runs this month

The enricher task is inactive, and the task run returned one or more errors.

Run enrichers

Run enrichers, either automatically or manually, to augment existing information with additional context from a range of cyber threat data like IP addresses, domains, email addresses, network streams, hashes, and forums.

Enrich entities

You can enrich entities in the following ways:

- Automatically, or
- Manually.

Enrichment rules and enrichment tasks drive the enrichment process to:

- Poll selected and trustworthy intelligence data sources;
- Retrieve relevant, accurate, and reliable data to augment platform entities with additional bits of information that provide additional context.

Rules

Enrichment rules define what to do with the retrieved enrichment data.
Rules act like filters, and they set the logical constraints defining:

- The platform data sources to augment with enrichment information.
  Data sources you can enrich are incoming feeds, other enrichers, and groups.
- Within the selected platform data sources, the entity type(s) to augment with enrichment information.
- The enrichers to use to fetch enrichment data.

Tasks

Enrichment tasks define process execution by setting the following options:

- The data fetching mechanism — for example, an API endpoint exposing the enrichment data service.
- Specific data sources — for example, datasets targeting threat actors like hackers and terrorist groups.
- Data rate limit and monthly execution cap values to control the amount of polled data.
- A source reliability flag for the incoming enrichment data to simplify assessing the quality of the retrieved data.

Enrich automatically

To automatically enrich entities, make sure enricher tasks are active, and the necessary enrichment rules are configured.

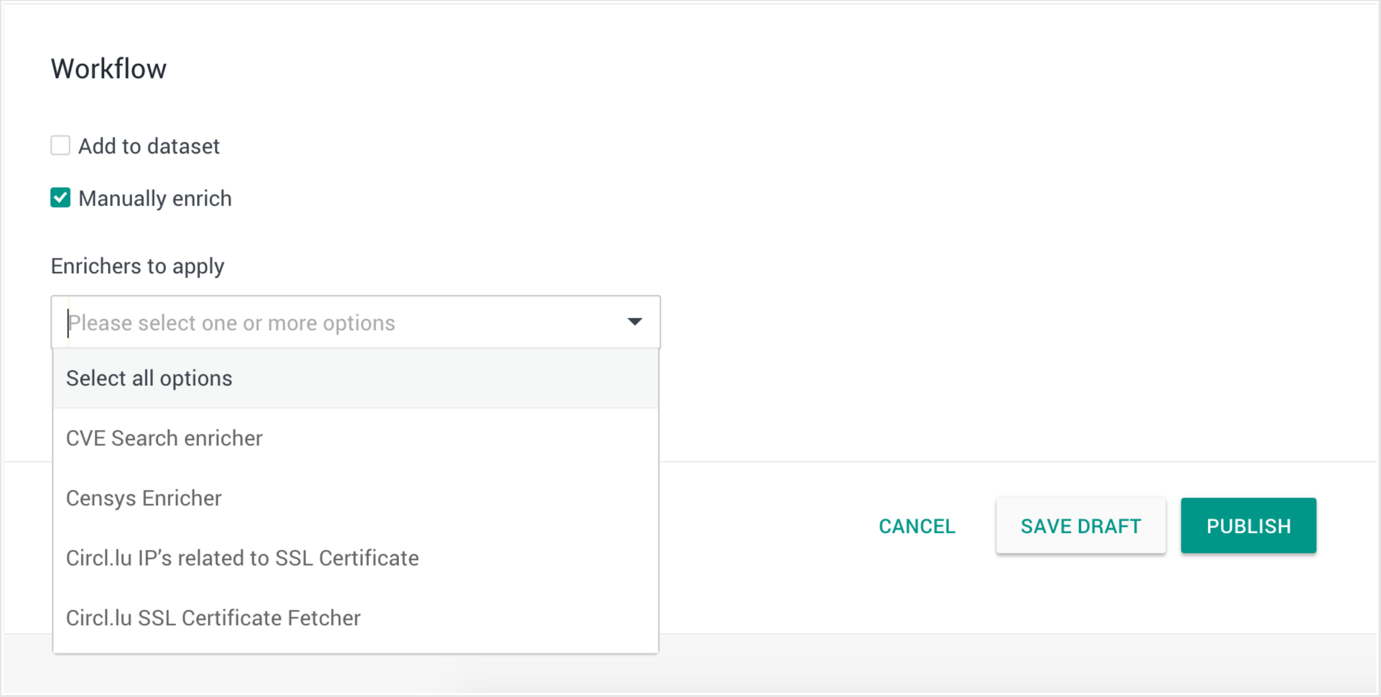
Rules give you control over the type of information you want to retrieve or exclude, and what you want to do with it. You can assign one or more enricher sources to specific observable types. You can set multiple filters to cover usage scenarios as needed. You can then examine the returned enrichment observable data, as well as route it to other devices that enforce cyber threat detection or prevention.

To run the enricher automatically, go to the enricher edit mode, and make sure the **Enabled** checkbox on the edit form is selected.
If it is deselected, check it, and then click **Save**.

Enrich manually

To adjust enrichment behavior to manually apply it to the entities you want to enrich, do the following:

- Open an entity in edit mode.
  For example, on the top navigation bar click **Browse > Published** to display an overview of the published entities available in the platform.
- On the row corresponding to the entity you want to manually enrich, click the ⋮ icon to display the context menu.
- From the drop-down menu select **Edit**.
- At the bottom of the entity editor page click the **Manually enrich** checkbox.
  A new input field with a drop-down menu becomes available.
- From the drop-down menu select one or more enrichers you want to apply to the entity.



Workflow

☐ Add to dataset

☑ Manually enrich

Enrichers to apply

Please select one or more options
Select all options
CVE Search enricher
Censys Enricher
Circl.lu IP's related to SSL Certificate
Circl.lu SSL Certificate Fetcher
CANCEL
SAVE DRAFT
PUBLISH

- Click **Save draft** to store your changes without publishing the entity, **Publish** to release the new version of the entity including your changes, or **Cancel** to discard the changes.

Alternatively, you can manually enrich an entity by selecting it; for example, from a dataset, from **Browse** or from **Discovery**.
An overlay slides in from the side of the screen to display the entity detail pane.

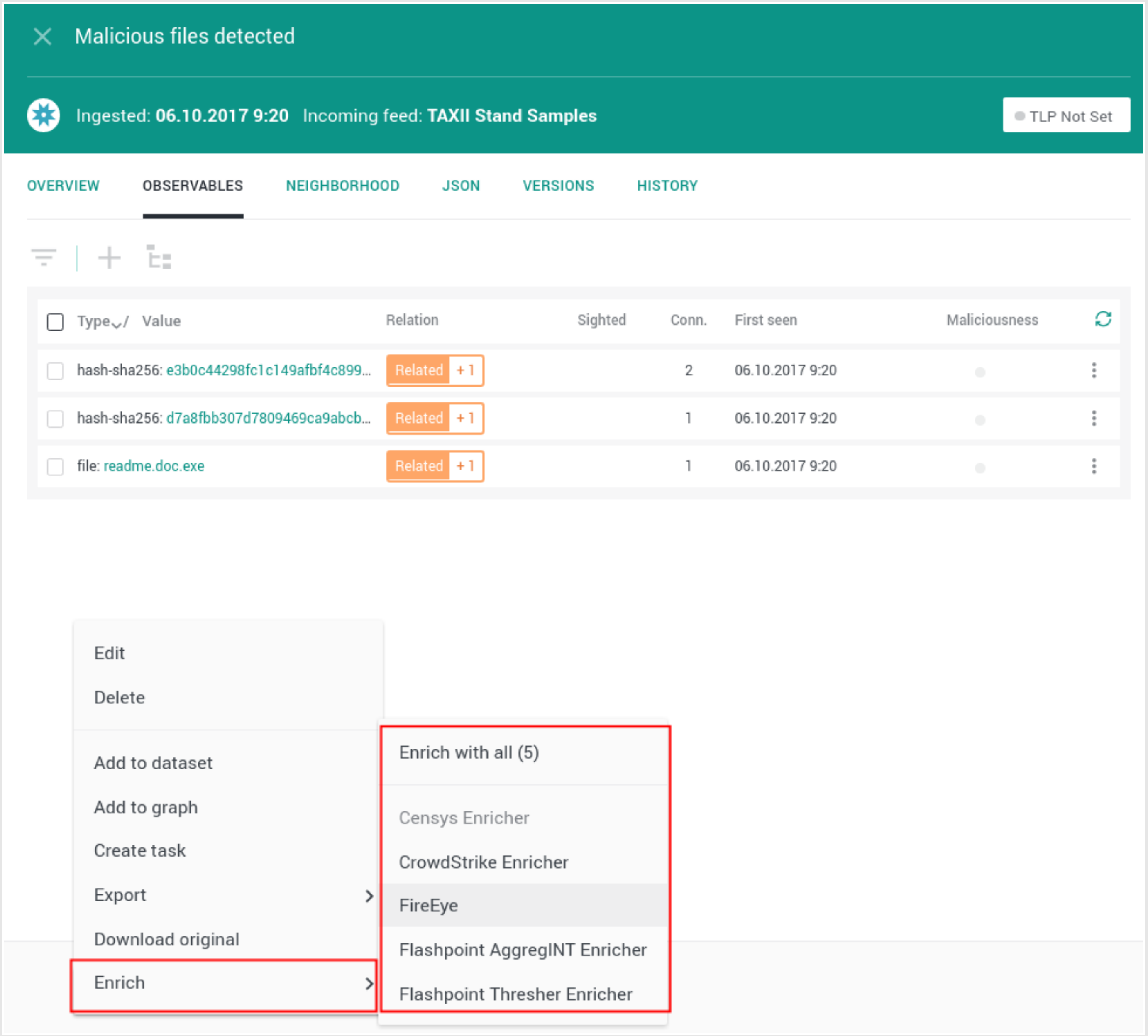
- On the entity detail pane, click **Observables**.
- The **Observables** tab shows an overview of the enrichment observables the entity has been augmented with.

To manually enrich the entity observables:

- Click the ⟳ refresh icon to trigger a task run that polls all the enrichers configured for the entity.

Alternatively:

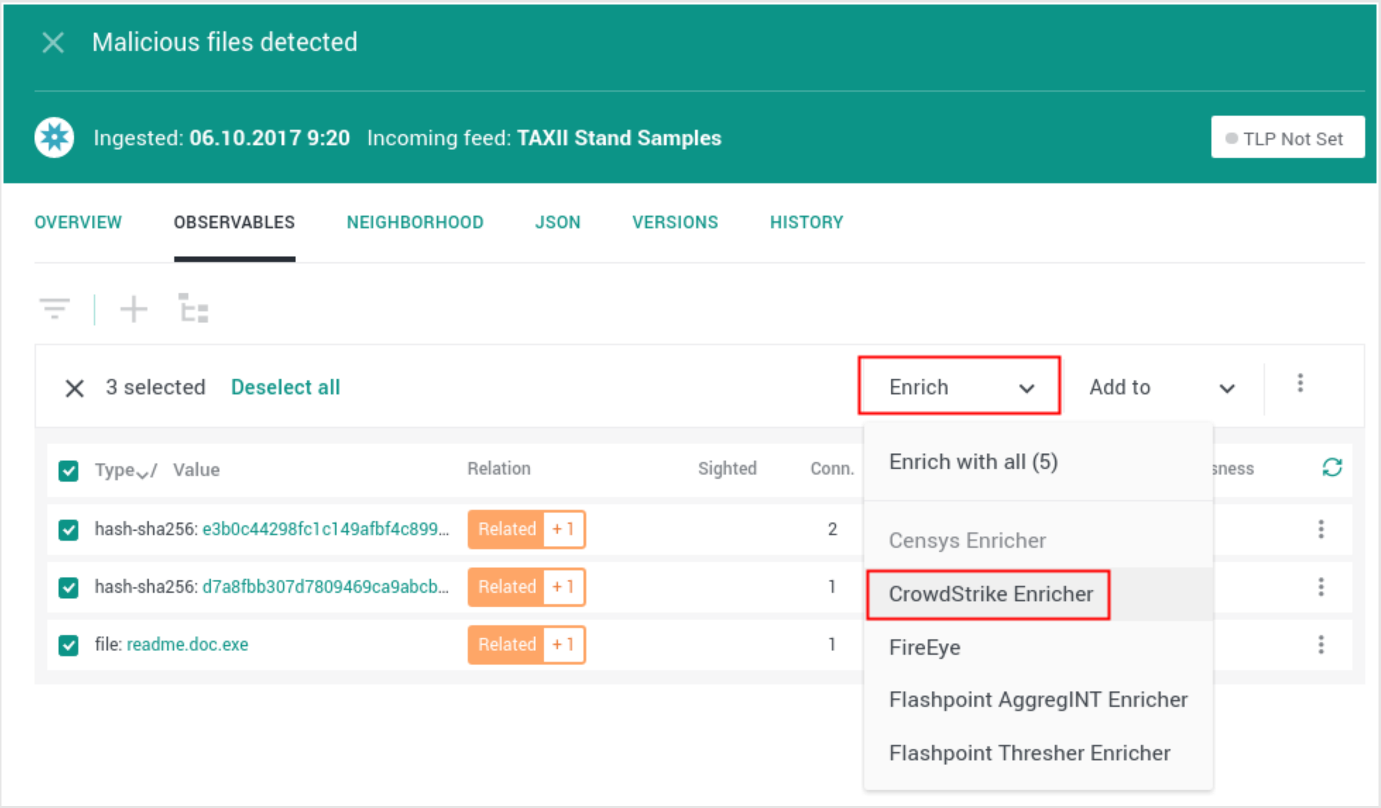
- From the **Actions** pop-up menu, select **Enrich > Enrich with all**.
- The platform polls all applicable enrichers for the entity, and it enriches all the entity observables with the retrieved data.



☐ Type / Value	Relation	Sighted	Conn.	First seen	Maliciousness
☐ hash-sha256: e3b0c44298fc1c149afbf4c899...	Related +1		2	06.10.2017 9:20	
☐ hash-sha256: d7a8fbb307d7809469ca9abcb...	Related +1		1	06.10.2017 9:20	
☐ file: readme.doc.exe	Related +1		1	06.10.2017 9:20	

To poll a specific enricher:

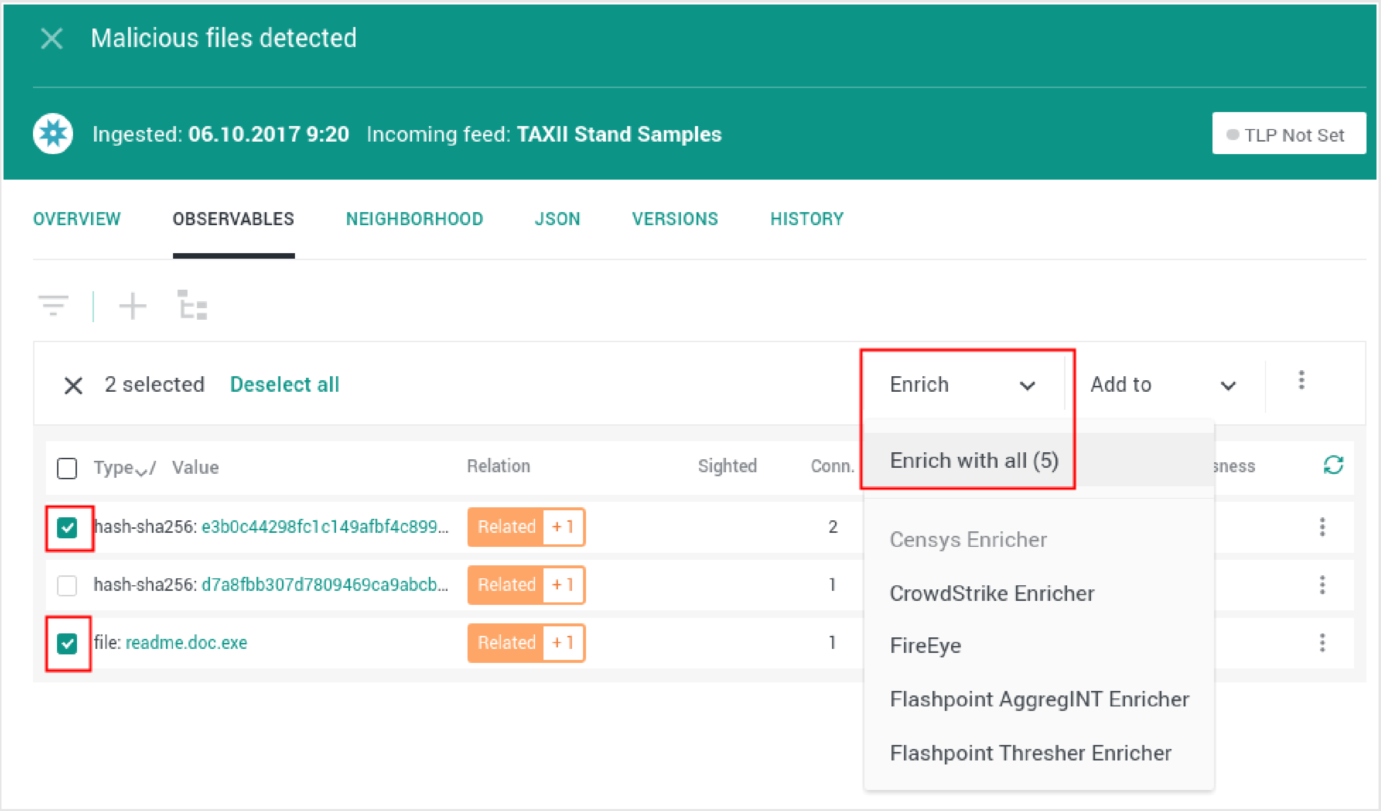
- From the **Actions** pop-up menu, select **Enrich**, and then click the specific enricher whose task run you want to trigger.
- The platform polls the specified enricher for the entity, and it enriches all supported entity observables with the retrieved data.



	Type / Value	Relation	Sighted	Conn.
☑	hash-sha256: e3b0c44298fc1c149afbf4c899...	Related +1		2
☑	hash-sha256: d7a8fbb307d7809469ca9abcb...	Related +1		1
☑	file: readme.doc.exe	Related +1		1

To enrich only specific observables:

- On the **Observables** tab, select the checkboxes corresponding to the observables you want to enrich.
- From the **Enrich** drop-down menu, select **Enrich with all**.
- The platform polls all applicable enrichers for the entity, and it enriches the selected entity observables with the retrieved data.



	Type / Value	Relation	Sighted	Conn.
☑	hash-sha256: e3b0c44298fc1c149afbf4c899...	Related +1		2
☐	hash-sha256: d7a8fbb307d7809469ca9abcb...	Related +1		1
☑	file: readme.doc.exe	Related +1		1

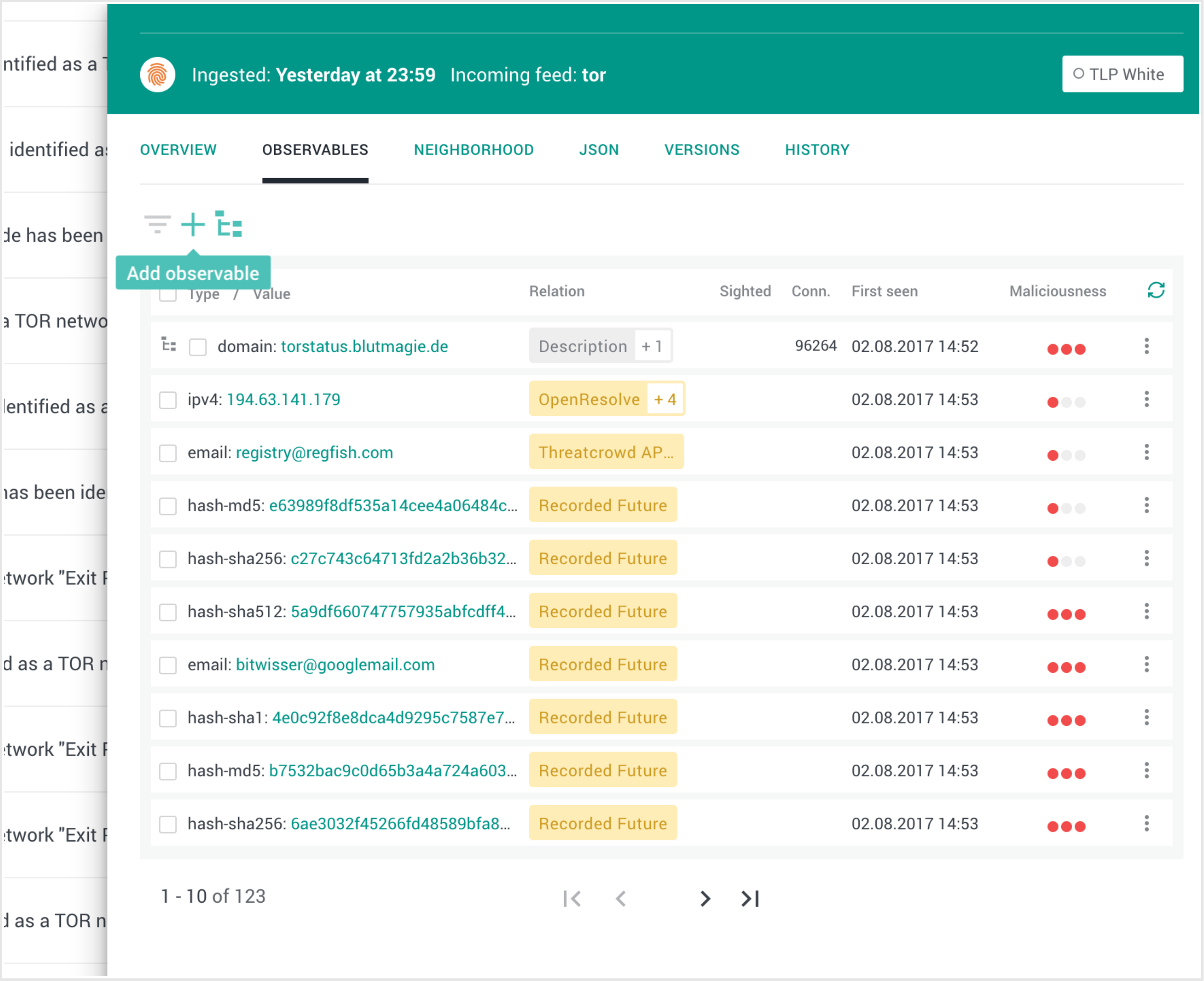
The available enricher tasks in the drop-down menu are automatically filtered to show only the applicable enrichers for the entity.

Enrichers automatically augment all the entities that accept the enricher's content type as an observable. In other words, the observable types an entity supports define the applicable enrichers an entity can use.

Review enrichment observables

To view enrichment information on the entity detail pane, do the following:

- Select an entity; for example, from a dataset, from **Browse** or from **Discovery**.
  An overlay slides in from the side of the screen to display the entity detail pane.
- On the entity detail pane, click **Observables**.
- The **Observables** tab shows an overview of the enrichment observables the entity has been augmented with.



Ingested: **Yesterday at 23:59** Incoming feed: **tor**

○ TLP White

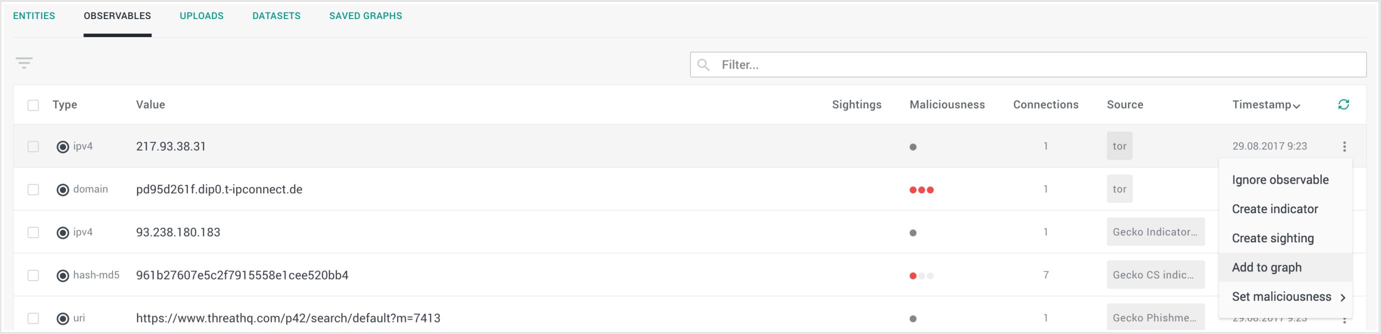
☐ Type / Value	Relation	Sighted	Conn.	First seen	Maliciousness
☐ domain: torstatus.blutmagie.de	Description + 1		96264	02.08.2017 14:52	●●●
☐ ipv4: 194.63.141.179	OpenResolve + 4			02.08.2017 14:53	●○○
☐ email: registry@regfish.com	Threatcrowd AP...			02.08.2017 14:53	●○○
☐ hash-md5: e63989f8df535a14cee4a06484c...	Recorded Future			02.08.2017 14:53	●○○
☐ hash-sha256: c27c743c64713fd2a2b36b32...	Recorded Future			02.08.2017 14:53	●○○
☐ hash-sha512: 5a9df660747757935abfcdff4...	Recorded Future			02.08.2017 14:53	●●●
☐ email: bitwisser@googlemail.com	Recorded Future			02.08.2017 14:53	●●●
☐ hash-sha1: 4e0c92f8e8dca4d9295c7587e7...	Recorded Future			02.08.2017 14:53	●●●
☐ hash-md5: b7532bac9c0d65b3a4a724a603...	Recorded Future			02.08.2017 14:53	●●●
☐ hash-sha256: 6ae3032f45266fd48589bfa8...	Recorded Future			02.08.2017 14:53	●●●

1 - 10 of 123

Review enrichment observables on the graph

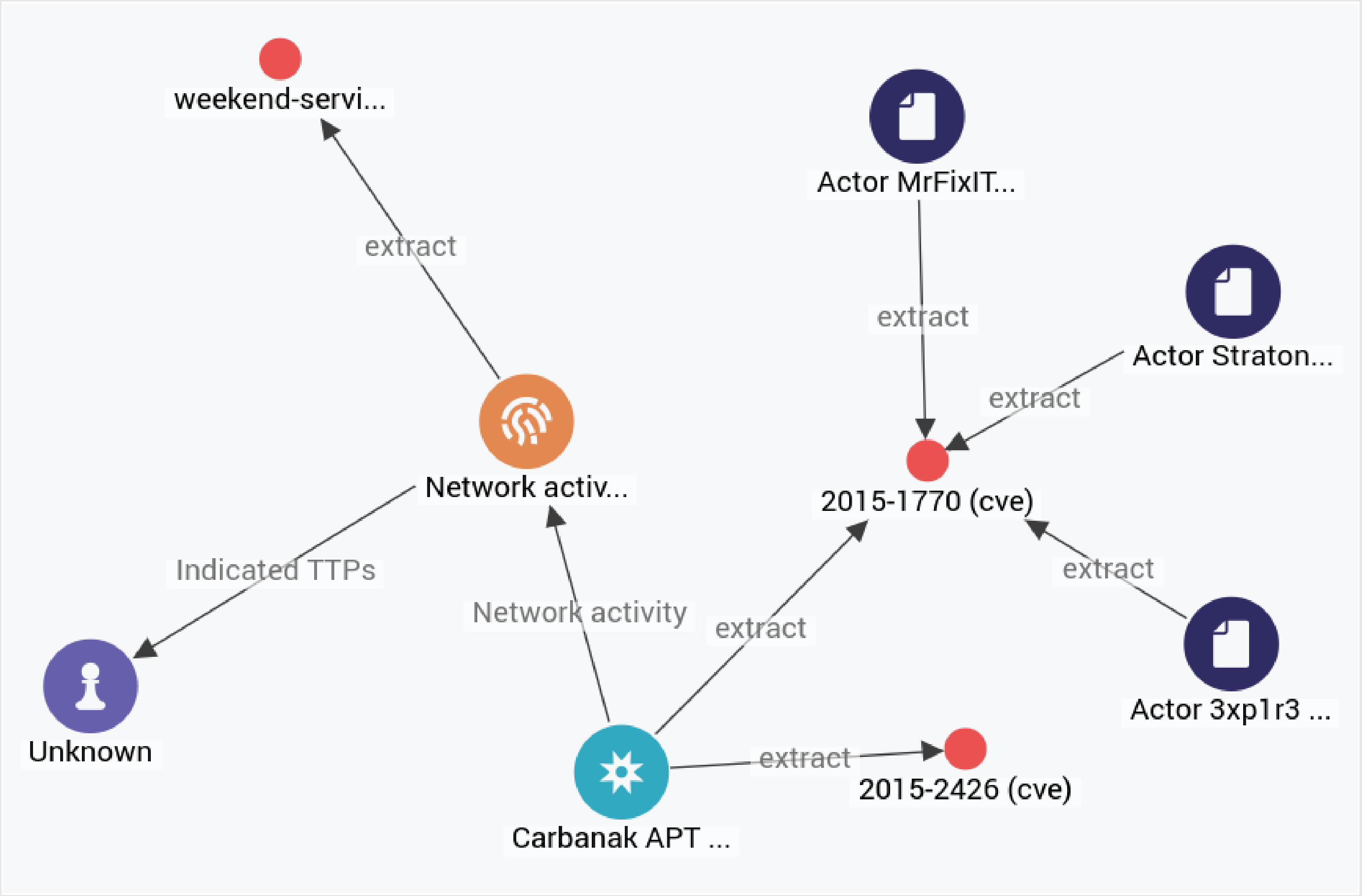
To view enrichment data and their connections with other entities and observables on the graph, do the following:

- On the row corresponding to the observable you want to load on the graph, click the ⋮ icon, and then select **Add to graph**.

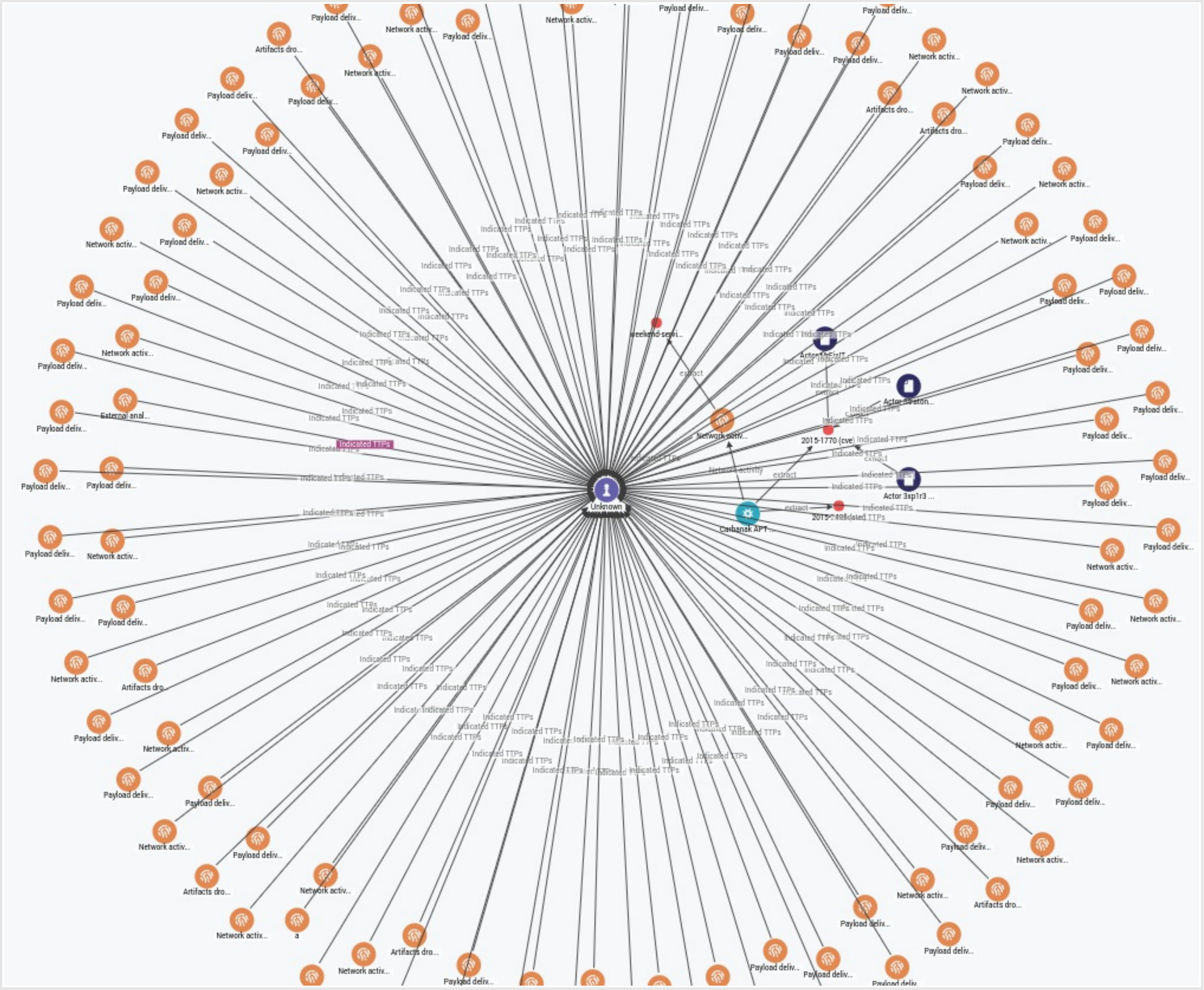


☐	Type	Value	Sightings	Maliciousness	Connections	Source	Timestamp
☐	ipv4	217.93.38.31		●	1	tor	29.08.2017 9:23
☐	domain	pd95d261f.dip0.t-ipconnect.de		●●●	1	tor	
☐	ipv4	93.238.180.183		●	1	Gecko Indicator...	
☐	hash-md5	961b27607e5c2f7915558e1cee520bb4		●○○	7	Gecko CS indic...	
☐	uri	https://www.threathq.com/p42/search/default?m=7413		●	1	Gecko Phishme...	

- To load the parent entity whose detail pane you are viewing, instead of its observables, from the pop-up **Actions** menu at the bottom of the pane select **Add to graph**.
- Click the graph thumbnail on the lower side of the screen to expand it.
- On the graph, right-click the entity you want to inspect, and from the context menu select **Load entities > All**, **Load observables > All** or **Load entities by extract > All** .

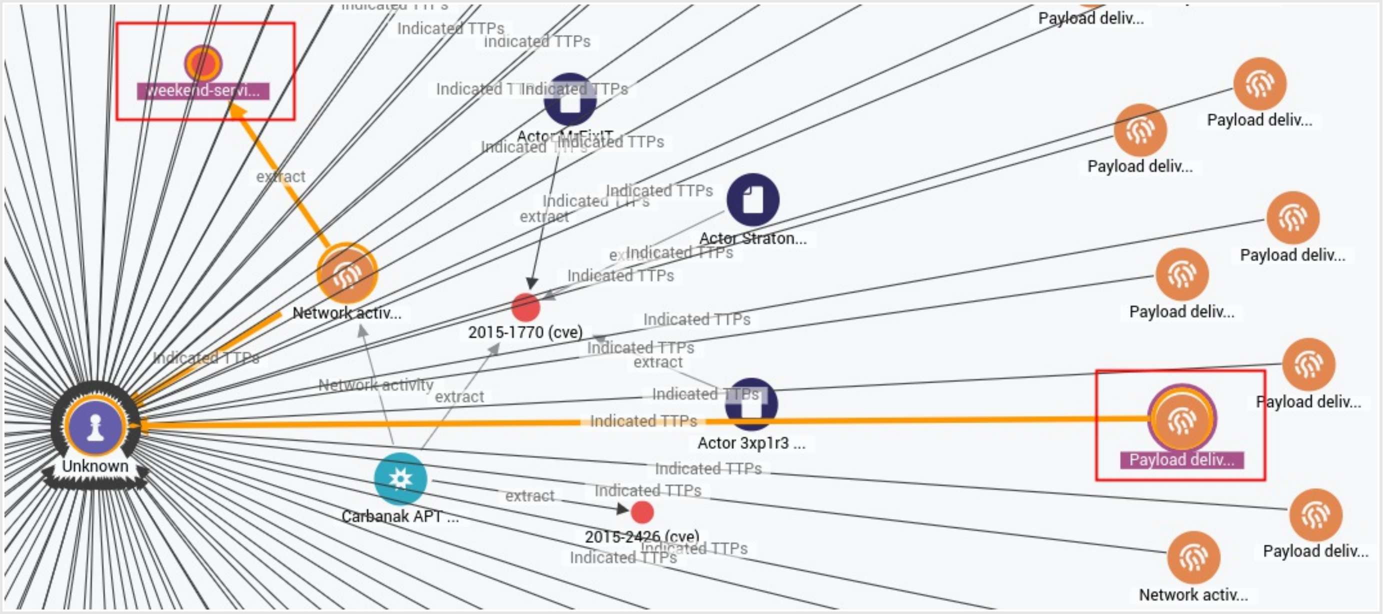


- Right-click an extract or an entity for further inspection and from the context menu select **Load entities > All**, **Load observables > All** or **Load entities by extract > All** .



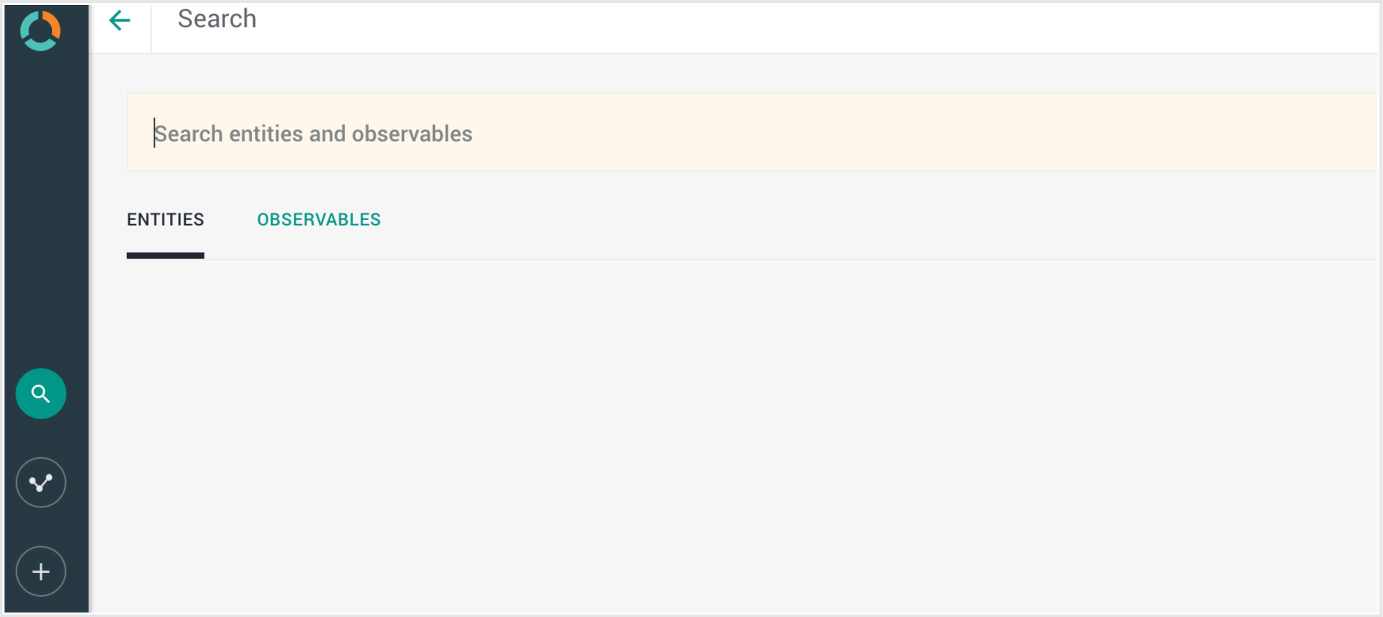
To see how entities, observables and enrichment observables are connected, and to inspect relationships between distant items, do the following:

- **CTRL** + click two nodes on the graph to select them.
- Right-click either selected node, and from the context menu select **Find path** to query the graph database about the existence of a path between the nodes, or **Show path** to highlight asn existing path on the graph.
- If a path does exist, the selected nodes and all the intermediate ones are highlighted on the graph to show the path that links them.



Search for enrichment observables

You can use the search box to look for enrichment observables. You can find the search box on the top bar:



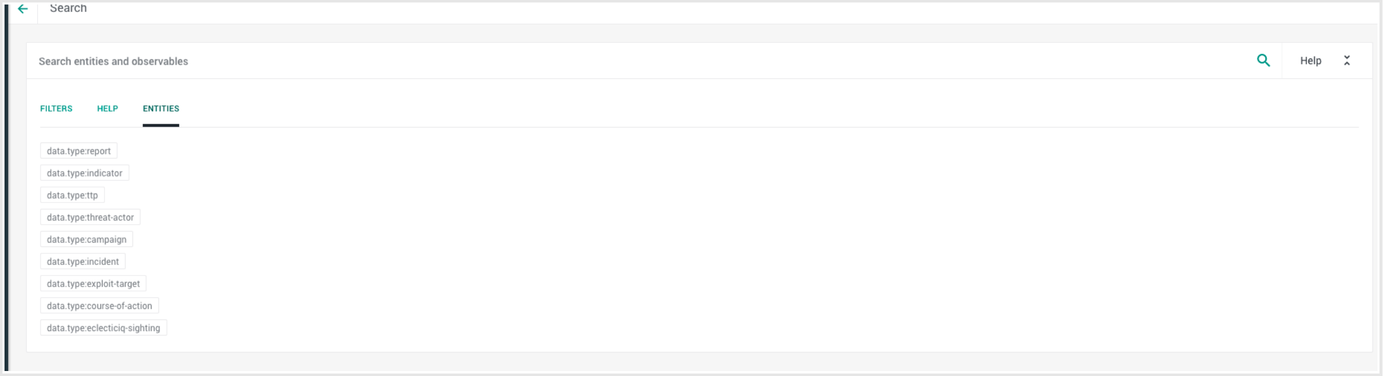
Enter search terms and search queries, and then press **ENTER** or click the search icon to run the search. Searches you run through this search box are executed platform-wide.

The search functionality uses **Elasticsearch query syntax** `(https://www.elastic.co/guide/en/elasticsearch/reference/current/full-text-queries.html).`

To access a cheatsheet with search examples using entity types, filters, and for help with the search syntax, click **Help** to display thematic drop-down lists with common search queries:

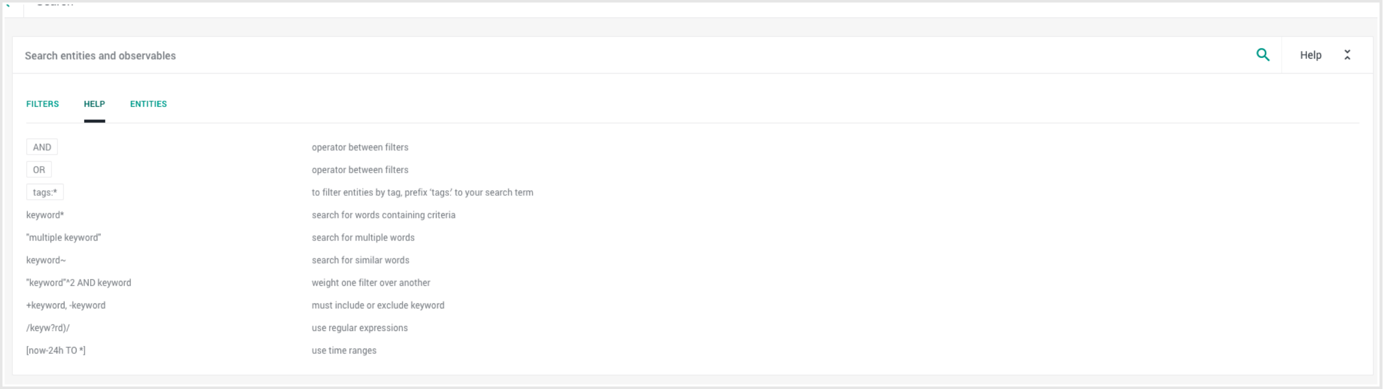
- **Filters**: examples of quick search filters.

- **Help**: examples of regex, Boolean, wildcards, and tag search usage.
- **Entities**: examples of searchable entity types.



data.type:report
data.type:indicator
data.type:ttp
data.type:threat-actor
data.type:campaign
data.type:incident
data.type:exploit-target
data.type:course-of-action
data.type:eclecticiq-sighting

Besides full text search, you can use Boolean operators, wildcards, regex, and you can combine these filtering options to create more refined searches.



AND	operator between filters
OR	operator between filters
tags:*	to filter entities by tag, prefix ‘tags:’ to your search term
keyword*	search for words containing criteria
"multiple keyword"	search for multiple words
keyword~	search for similar words
"keyword"^2 AND keyword	weight one filter over another
+keyword, -keyword	must include or exclude keyword
/keyw?rd)/	use regular expressions
[now-24h TO *]	use time ranges

Use operators to combine multiple quick filters and create a more complex search query.

Search example:

```
enrichment_extracts.meta.blacklisted:true AND enrichment_extracts.kind:hash-md5
```

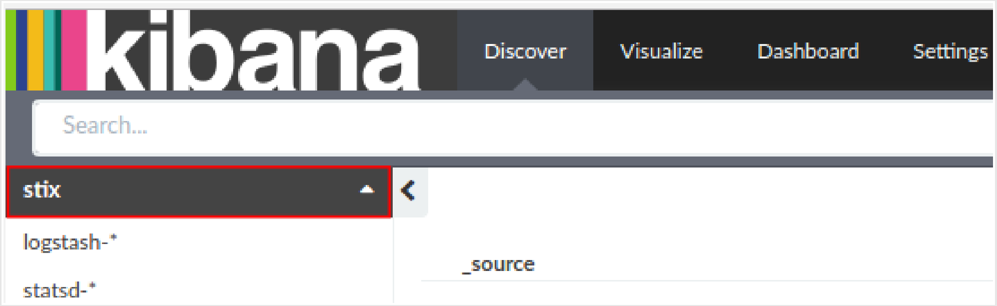
Search result example:

```
{
  "instance_meta": {
    "classification": "bad",
    "confidence": "low",
    "blacklisted": true
  },

  "id": 123456,
  "value": "7cd7ekhc5f5742fdccf655767b15h7g78",

  "meta": {
    "classification": "bad",
    "confidence": "low",
    "blacklisted": true
  },

  "kind": "hash-md5",
  "instance_id": 123456
}
```

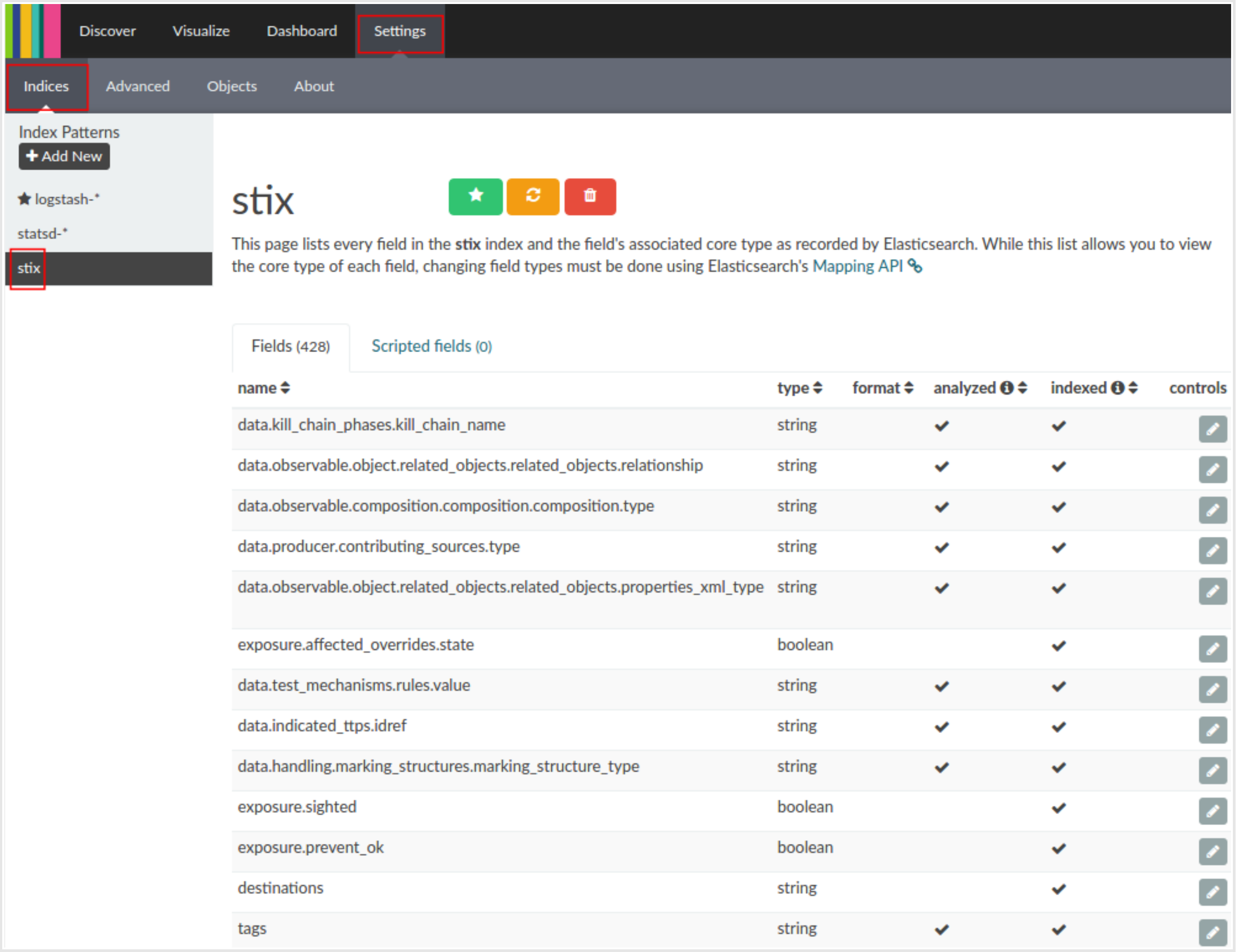
Field	Description	Example
enrichment_extracts.id	string — The alphanumeric ID string that uniquely identifies the enrichment observable.	`01h12x45-01q2-1234-od01-123456h78h90`
enrichment_extracts.kind	string — The enrichment observable data type.	`domain`
enrichment_extracts.meta.blacklisted	Boolean — An observable is blacklisted when it is included in the results returned by an *ignore* extraction rule. Allowed values: `true`, `false`.	`true`
enrichment_extracts.meta.classification	string — This value is defined in **Rules** by selecting appropriate options under **Action** and **Confidence**. Allowed classification metadata values are `good`, `bad`, and `unknown`.	`good`
enrichment_extracts.meta.confidence	string — This value is defined in **Rules** by selecting the appropriate option under **Action** and **Confidence**. The selected action must be **Mark as malicious** for the **Confidence** drop-down list to become available. Allowed confidence metadata values are `low`, `medium`, and `high`.	`high`
enrichment_extracts.value	string — The actual value of the enrichment observable, based on the enrichment observable data type.	`doom.dismay.biz`

For reference, you can look up a complete list of all available search query fields in Kibana:

- To access Kibana, in the web browser address bar enter a URL with the following format:
  https://<platform_host>/api/kibana/app/kibana#/
  Keep the trailing */*
  Example: *https://platform.host.com/api/kibana/app/kibana#/*
- Select the **stix** index field:



- On the main menu bar, select **Settings**:



stix

This page lists every field in the **stix** index and the field's associated core type as recorded by Elasticsearch. While this list allows you to view the core type of each field, changing field types must be done using Elasticsearch's Mapping API

name	type	format	analyzed	indexed	controls
data.kill_chain_phases.kill_chain_name	string		✓	✓	✎
data.observable.object.related_objects.related_objects.relationship	string		✓	✓	✎
data.observable.composition.composition.composition.type	string		✓	✓	✎
data.producer.contributing_sources.type	string		✓	✓	✎
data.observable.object.related_objects.related_objects.properties_xml_type	string		✓	✓	✎
exposure.affected_overrides.state	boolean			✓	✎
data.test_mechanisms.rules.value	string		✓	✓	✎
data.indicated_ttps.idref	string		✓	✓	✎
data.handling.marking_structures.marking_structure_type	string		✓	✓	✎
exposure.sighted	boolean			✓	✎
exposure.prevent_ok	boolean			✓	✎
destinations	string			✓	✎
tags	string		✓	✓	✎

Censys enricher

The Censys enricher returns a wealth of information about IP addreses, from a network ASN to their geographic location, so that you can explore relationships between events, actors, and targets.

This article describes how to configure the Censys enricher parameters.
To configure the general options for the Censys enricher, see Configure enrichers.

Censys	**enricher**
Enricher name	Censys
API endpoint	`https://censys.io/api/v1/search/ipv4`
Input	asn, city, company, country, country_code, geo-lat, geo-long, hash-md5, hash-sha1, hash-sha256, ipv4, postcode
Output	Enriches supported observable types by providing additional context such as geolocation, country and city information, as well as **ASN** `(https://en.wikipedia.org/wiki/autonomous_system_(internet))` details.
Description	Returns relevant contextual information about the submitted observable types to augment their intelligence value with geographic and geolocation details, hashes, and **ASN** `(https://en.wikipedia.org/wiki/autonomous_system_(internet))` details. It makes it easier to discover relationships between events, actors, and targets.

Configure the Censys enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the Censys enricher.
  Supported observable types:
  - *asn*
  - *city*
  - *company*
  - *country*
  - *country_code*
  - *geo-lat*
  - *geo-long*
  - *hash-md5*
  - *hash-sha1*
  - *hash-sha256*
  - *ipv4*
  - *postcode*

Under **Parameters**, define the specific configuration options for the Censys enricher:

- **API URL**: the URL pointing to the API endpoint exposing the service that grants access to the enricher data source. Contact the intel provider to subscribe to the service and to obtain this information, as well as any required authentication and authorization credentials.
  The API URL for this enricher exposes the Censys **Search API** `(https://censys.io/api/v1/docs/search)`.
  The *ipv4* URL parameter allows sending requests to the IP address search index.
- **API ID**: **create an account** `(https://censys.io/register)` to receive the login credentials you need to authenticate and access the API service.
  Enter here your API user ID.
- **API secret**: enter the secret key associated with your API user profile, so that you can log in and consume the API service.

- **Observable queries**: from the drop-down menu select the observable type and the corresponding observable value the rule should look for.
    - In the first input field, from the drop-down menu select the *observable type* the rule should look for.
      Supported observable types:
        - *asn*
        - *city*
        - *company*
        - *country*
        - *country_code*
        - *geo-lat*
        - *geo-long*
        - *hash-md5*
        - *hash-sha1*
        - *hash-sha256*
        - *ipv4*
        - *postcode*
    - In the second input field, specify the *observable value* associated with the observable type that the rule should look for.

      You can use free text, wildcards, **Elasticsearch query syntax** `(https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html)`, as well as the *{kind}* and *{value}* placeholders to reference an observable type and value, respectively.
      When the query executes, the placeholders take the values from the input observable key (*{kind}*) and value (*{value}*) pairs, respectively.

      Example:
      The **@{value}* query searches for observable values matching the input observable values it is fed at runtime.

      Censys allows using **specific data fields** `(https://censys.io/overview)` to search for data related to IP hosts.
      You can combine these data definitions with the *{kind}* and *{value}* placeholders.
- Click **✚ Add** or **✚ More** to add a new filtering option. For example, to include in the search additional key/value pairs like IP addresses, hashes, or domains.
- Click **Save** to store your changes, or **Cancel** to discard them.

Query the Censys API

Observable queries accept key/value pairs:

- In the first input field you define the type of observable you want to search for.
- In the second input field you specify the data pattern to use in the search to look for the desired values.

The Censys **Search API** `(https://censys.io/api/v1/docs/search)` allows simple searches for words or phrases, as well as complex searches using query syntax and query parameters:

- For the search syntax, refer to **Censys Overview** `(https://censys.io/overview)`, under **Search Syntax**.

- For the query parameters you can pass, refer to **Censys Overview** `(https://censys.io/overview)`, under **Data Definitions > IPv4 Hosts**.

When you chain multiple parameters in the same query without specifying any Boolean operator, the default Boolean operator between parameters is *or*.

The table overview shows query examples you can enter under **Observable queries** to search for specific observable types and values.

First query field	**Second query field**
asn	*autonomous_system.asn: {value}*
city	*location.city: "{value}"*
company	*metadata.manufacturer: "{value}"*
country	*location.country: "{value}"*
country-code	*location.country_code: "{value}"*
geo-lat	*location.latitude: {value}*
geo-long	*location.longitude: {value}*
ipv4	*ip: {value}*
postcode	*location.postal_code: "{value}"*
hash-md5	*25.smtp.starttls.tls.certificate.parsed.fingerprint_md5: "{value}" or 25.smtp.starttls.tls.chain.parsed.fingerprint_md5: "{value}" or 80.http.get.headers.content_md5: "{value}" or 80.http.open_proxy.connect.headers.content_md5: "{value}" or 80.http.open_proxy.get.headers.content_md5: "{value}" or 143.imap.starttls.tls.certificate.parsed.fingerprint_md5: "{value}" or 143.imap.starttls.tls.chain.parsed.fingerprint_md5: "{value}" or 443.https.tls.certificate.parsed.fingerprint_md5: "{value}" or 443.https.tls.chain.parsed.fingerprint_md5: "{value}" or 110.pop3.starttls.tls.certificate.parsed.fingerprint_md5: "{value}" or 110.pop3.starttls.tls.chain.parsed.fingerprint_md5: "{value}" or 993.imaps.tls.tls.certificate.parsed.fingerprint_md5: "{value}" or 993.imaps.tls.tls.chain.parsed.fingerprint_md5: "{value}" or 995.pop3s.tls.tls.certificate.parsed.fingerprint_md5: "{value}" or 995.pop3s.tls.tls.chain.parsed.fingerprint_md5: "{value}"*
hash-sha1	*25.smtp.starttls.tls.certificate.parsed.fingerprint_sha1: "{value}" or 25.smtp.starttls.tls.chain.parsed.fingerprint_sha1: "{value}" or 143.imap.starttls.tls.certificate.parsed.fingerprint_sha1: "{value}" or 143.imap.starttls.tls.chain.parsed.fingerprint_sha1: "{value}" or 443.https.tls.certificate.parsed.fingerprint_sha1: "{value}" or 443.https.tls.chain.parsed.fingerprint_sha1: "{value}" or 110.pop3.starttls.tls.certificate.parsed.fingerprint_sha1: "{value}" or 110.pop3.starttls.tls.chain.parsed.fingerprint_sha1: "{value}" or 993.imaps.tls.tls.certificate.parsed.fingerprint_sha1: "{value}" or 993.imaps.tls.tls.chain.parsed.fingerprint_sha1: "{value}" or 995.pop3s.tls.tls.certificate.parsed.fingerprint_sha1: "{value}" or 995.pop3s.tls.tls.chain.parsed.fingerprint_sha1: "{value}"*

First query field	Second query field
hash-sha256	*25.smtp.starttls.tls.certificate.parsed.fingerprint_sha256: “{value}” or 25.smtp.starttls.tls.chain.parsed.fingerprint_sha256: “{value}” or 80.http.get.body_sha256: “{value}” or 80.http.open_proxy.get.body_sha256: “{value}” or 143.imap.starttls.tls.certificate.parsed.fingerprint_sha256: “{value}” or 143.imap.starttls.tls.chain.parsed.fingerprint_sha256: “{value}” or 443.https.tls.certificate.parsed.fingerprint_sha256: “{value}” or 443.https.tls.chain.parsed.fingerprint_sha256: “{value}” or 110.pop3.starttls.tls.certificate.parsed.fingerprint_sha256: “{value}” or 110.pop3.starttls.tls.chain.parsed.fingerprint_sha256: “{value}” or 993.imaps.tls.tls.certificate.parsed.fingerprint_sha256: “{value}” or 993.imaps.tls.tls.chain.parsed.fingerprint_sha256: “{value}” or 995.pop3s.tls.tls.certificate.parsed.fingerprint_sha256: “{value}” or 995.pop3s.tls.tls.chain.parsed.fingerprint_sha256: “{value}”*

By default, the Censys enricher timeout value is set to *1 minute*.

CIRCL IPs related to SSL certificate enricher

The CIRCL IPs related to SSL certificate enricher uses the CIRCL API to poll the CIRCL Passive SSL database and to obtain all IP addreses associated with the input SSL SHA-1 hash fingerprints.

This article describes how to configure the CIRCL IPs related to SSL certificate enricher parameters.
To configure the general options for the CIRCL IPs related to SSL certificate enricher, see Configure enrichers.

CIRCL IPs related to SSL certificate	**enricher**
Enricher name	CIRCL IPs related to SSL certificate
API endpoint	`https://www.circl.lu/v2pssl/cquery/{}`
Input	hash-sha1
Output	Enriches SSL fingerprint hash observables with all associated IP addresses.
Description	The enricher contacts the Computer Incident Response Center Luxembourg (CIRCL) API and the CIRCL Passive SSL database to retrieve all observed IP addresses associated with the input SSL fingerprint *hash-sha1* observables. The CIRCL Passive SSL database holds historical data matching SSL certificates and IP addresses.

Configure the CIRCL IPs related to SSL certificate enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the CIRCL IPs related to SSL certificate enricher.
  Supported observable types:
  - *hash-sha1*

Under **Parameters**, define the specific configuration options for the CIRCL IPs related to SSL certificate enricher:

- **API user name**: sign up and subscribe to the service to obtain the required API user name and API key credentials to access the API endpoint exposing the service.
- **API key**: **contact CIRCL** `(https://www.circl.lu/contact/)` to receive an API key, and then enter it in the corresponding input field.
- Click **Save** to store your changes, or **Cancel** to discard them.

ℹ By default, the CIRCL IPs related to SSL certificate enricher timeout value is set to *1 minute*.

CIRCL SSL Certificate Fetcher enricher

The CIRCL SSL Certificate Fetcher enricher uses the CIRCL API to poll the CIRCL Passive SSL database and to obtain parsed SSL certificates and all domain names associated with the input SSL SHA-1 hash fingerprints.

This article describes how to configure the CIRCL SSL Certificate Fetcher enricher parameters.
To configure the general options for the CIRCL SSL Certificate Fetcher enricher, see Configure enrichers.

CIRCL SSL Certificate Fetcher	**enricher**
Enricher name	CIRCL SSL Certificate Fetcher
API endpoint	`https://www.circl.lu/v2pssl/cfetch/{}`
Input	hash-sha1
Output	Enriches SSL fingerprint hash observables with the parsed certificate and all associated domain names.
Description	The enricher contacts the Computer Incident Response Center Luxembourg (CIRCL) API and the CIRCL Passive SSL database to retrieve the parsed certificate and all observed domain names associated with the input SSL fingerprint *hash-sha1* observables. The CIRCL Passive SSL database holds historical data matching SSL certificates fingerprints with the corresponding parsed certificates and the related domain names, that is, the certificate `subject CN` and `subjectAltName`/*Subject Alternative Name (SAN)*, if available.

Configure the CIRCL SSL Certificate Fetcher enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the CIRCL SSL Certificate Fetcher enricher.
  Supported observable types:
  - *hash-sha1*

Under **Parameters**, define the specific configuration options for the CIRCL SSL Certificate Fetcher enricher:

- **API user name**: sign up and subscribe to the service to obtain the required API user name and API key credentials to access the API endpoint exposing the service.
- **API key**: **contact CIRCL** `(https://www.circl.lu/contact/)` to receive an API key, and then enter it in the corresponding input field.
- Click **Save** to store your changes, or **Cancel** to discard them.

ℹ By default, the CIRCL SSL Certificate Fetcher enricher timeout value is set to *1 minute*.

Cisco Threat Grid enricher

The Cisco Threat Grid enricher provides information on a range of cyber threat data like IP addresses, domains, registry keys, network streams, and hash files.

This article describes how to configure the Cisco Threat Grid enricher parameters.
To configure the general options for the Cisco Threat Grid enricher, see Configure enrichers.

Cisco Threat Grid	**enricher**
Enricher name	Cisco Threat Grid
API endpoint	`https://panacea.threatgrid.com/api/v2/`
Input	domain, hash-md5, hash-sha1, hash-sha256, hash-sha512, host, ipv4, ipv6, uri, winregistry
Output	Enriches supported observable types, as well as all found observables based on the enricher configuration, with information such as IP addresses, domains, host names, hashes, and Windows registry keys.
Description	Polls data from the Cisco Threat Grid API. It provides information on a range of cyber threat data like IP addresses, domains, registry keys, network streams, and hash files.

Configure the Cisco Threat Grid enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the Cisco Threat Grid enricher.
  Supported observable types:
  - *domain*
  - *hash-md5*
  - *hash-sha1*
  - *hash-sha256*
  - *hash-sha512*
  - *host*
  - *ipv4*
  - *ipv6*
  - *uri*
  - *winregistry*

Under **Parameters**, define the specific configuration options for the Cisco Threat Grid enricher:

- **API URL**: the URL pointing to the API endpoint exposing the service that grants access to the enricher data source. Contact the intel provider to subscribe to the service and to obtain this information, as well as any required authentication and authorization credentials.
- **API key**: contact Cisco to receive an API key, and then enter it in the corresponding input field.
- **Organization only**: select this checkbox to enable the enricher to check and display only submitted samples created by the organization the current user belongs to. That is, the organization needs to be the author of the submitted samples. When selected, this field is validated against the API key value granting access to the service.
- **Max low confidence threat score**: you can set an *upper threshold* to automatically flag enriched observables with a *low confidence* value.
  After completing the sample analysis, enriched observables with a *lower* threat score than the specified value are flagged as **Malicious - Low confidence**.
  - Enter an integer value between *0* and *100*.
  - Default value: *85*.
- **Min high confidence threat score**: you can set a *bottom threshold* to automatically flag enriched observables with *high confidence* value.
  After completing the sample analysis, enriched observables with a *higher* threat score than the specified value are flagged as **Malicious - High confidence**.
  - Enter an integer value between *0* and *100*.
  - Default value: *95*.
- Enriched observables with a threat score falling in the range defined by **Max low confidence threat score** (range lower limit) and **Min high confidence threat score** (range upper limit) are flagged as **Malicious - Medium confidence**.
- Click **Save** to store your changes, or **Cancel** to discard them.

Automatic flagging with high/low confidence maliciousness through Cisco Threat Grid supports only the following observable types:

- *hash-sha1*
- *hash-sha256*
- *hash-md5*
- *uri*

By default, the Cisco Threat Grid enricher timeout value is set to *5 minutes*.

Crowdstrike Falcon Intelligence Indicator enricher

The Crowdstrike Falcon Intelligence Indicator enricher returns observables extracted from indicators to provide additional context to existing platform intelligence.

This article describes how to configure the Crowdstrike Falcon Intelligence Indicator enricher parameters.
To configure the general options for the Crowdstrike Falcon Intelligence Indicator enricher, see Configure enrichers.

Crowdstrike Falcon Intelligence Indicator	**enricher**
Enricher name	Crowdstrike Falcon Intelligence Indicator
API endpoint	`https://intelapi.crowdstrike.com/indicator/v1/search/{}`
Input	domain, email, email-subject, file, hash-md5, hash-sha1, hash-sha256, ipv4, ipv6, mutex, name, persona, port, uri
Output	Enriches supported observable types with information extracted from indicators.
Description	Enriches platform entities and observables with additional context such as IP addresses, domain names, email addresses, hashes, and more.

Configure the Crowdstrike Falcon Intelligence Indicator enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the Crowdstrike Falcon Intelligence Indicator enricher.
  Supported observable types:
  - *domain*
  - *email*
  - *email-subject*
  - *file*
  - *hash-md5*
  - *hash-sha1*
  - *hash-sha256*
  - *ipv4*
  - *ipv6*
  - *mutex*
  - *name*
  - *persona*
  - *port*
  - *uri*

Under **Parameters**, define the specific configuration options for the Crowdstrike Falcon Intelligence Indicator enricher:

- **API ID**: contact Crowdstrike to receive an API ID, and then enter it in the corresponding input field.
  You need a valid API ID and a corresponding API key as authentication credentials to access the Crowdstrike Falcon Intel API and to consume it.
- **API key**: contact Crowdstrike to receive an API key, and then enter it in the corresponding input field.
- Click **Save** to store your changes, or **Cancel** to discard them.

By default, the Crowdstrike Falcon Intelligence Indicator enricher timeout value is set to *1 minute*.

CVE Search enricher

The CVE Search enricher contacts CIRCL cve-search API to poll information about common software and hardware vulnerabilities, along with the corresponding exposures.

This article describes how to configure the CVE Search enricher parameters.
To configure the general options for the CVE Search enricher, see Configure enrichers.

CVE Search	**enricher**
Enricher name	CVE Search
API endpoint	`https://cve.circl.lu/api/cve/`
Input	cve
Output	Enriches supported observable types with CVE details.
Description	Enriches supported observable types with information about common software and hardware vulnerabilities, along with the corresponding exposures. The enricher contacts the Computer Incident Response Center Luxembourg (CIRCL) cve-search API to retrieve all the available details associated with the input CVE IDs. CVE information about common software and hardware vulnerabilities, along with the corresponding exposures, is stored in the platform as enrichment observables.

Configure the CVE Search enricher parameters

✔ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the CVE Search enricher.
  Supported observable types:
  - *cve*

Under **Parameters**, define the specific configuration options for the CVE Search enricher:

- **CVE API URL** : the URL pointing to the API endpoint exposing the service that grants access to the enricher data.
  Contact the intel provider to subscribe to the service and to obtain this information.
  For this enricher it corresponds to *https://cve.circl.lu/api/cve/*.
- Click **Save** to store your changes, or **Cancel** to discard them.

ℹ By default, the CVE Search enricher timeout value is set to *1 minute*.

DomainTools Hosted Domains enricher

The Domaintools Hosted Domains enricher returns all domain names related to the specified input IP addresses.

This article describes how to configure the DomainTools Hosted Domains enricher parameters.
To configure the general options for the DomainTools Hosted Domains enricher, see Configure enrichers.

DomainTools Hosted Domains	enricher
Enricher name	DomainTools Hosted Domains
API endpoint	`http://api.domaintools.com/v1/{}/host-domains`
Input	ipv4
Output	Enriches supported observable types with domain and host name information.
Description	Enriches IPv4 observables by returning all the domain names hosted on, and therefore related to, the input IP addresses.

Configure the DomainTools Hosted Domains enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the DomainTools Hosted Domains enricher.
  Supported observable types:
  - *ipv4*

Under **Parameters**, define the specific configuration options for the DomainTools Hosted Domains enricher:

- **Username**: sign up and subscribe to the service to obtain the required API user name and API key credentials to access the API endpoint exposing the service.
- **API key**: contact DomainTools to receive an API key, and then enter it in the corresponding input field.
- Click **Save** to store your changes, or **Cancel** to discard them.

ℹ By default, the DomainTools Hosted Domains enricher timeout value is set to *1 minute*.

DomainTools Malicious Server Domains enricher

The DomainTools Malicious Server Domains enricher returns malicious domain names related to the same primary and/or secondary name servers, along with their risk scores to automatically flag server domains with an appropriate maliciousness confidence level.

This article describes how to configure the DomainTools Malicious Server Domains enricher parameters.
To configure the general options for the DomainTools Malicious Server Domains enricher, see Configure enrichers.

DomainTools Malicious Server Domains	**enricher**
Enricher name	DomainTools Malicious Server Domains
API endpoint	`http://api.domaintools.com/v1/{}/name-server-domains/`
Input	domain, host
Output	Enriches supported observable types with malicious domain names that are served from the same name server.
Description	Enriches domain and host observable types with a list of malicious domain names related to the same primary or secondary name server. It includes configurable thresholds to assign maliciousness confidence levels to the processed domains and hosts, and to ignore non-malicious domains/hosts.

Configure the DomainTools Malicious Server Domains enricher parameters

✔ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the DomainTools Malicious Server Domains enricher.
  Supported observable types:
  - *domain*
  - *host*

Under **Parameters**, define the specific configuration options for the DomainTools Malicious Server Domains enricher:

- **Username**: sign up and subscribe to the service to obtain the required API user name and API key credentials to access the API endpoint exposing the service.
- **API key**: contact DomainTools to receive an API key, and then enter it in the corresponding input field.

- **Low maliciousness threshold**: domain and host names with a higher DomainTools risk score than the value defined here are flagged as **Malicious - Low confidence**.
  After completing the analysis, enriched domain and host names with a *higher* risk score than the *low maliciousness threshold* and lower than the medium and high maliciousness thresholds are flagged as **Malicious - Low confidence**.
  - Enter a value between *0* and *99.99*.
  - Default value: *10*.
- **Medium maliciousness threshold**: domain and host names with a higher DomainTools risk score than the value defined here are flagged as **Malicious - Medium confidence**.
  After completing the analysis, enriched domain and host names with a *higher* risk score than the *medium maliciousness threshold* and lower than the high maliciousness threshold are flagged as **Malicious - Medium confidence**.
  - Enter a value between *0* and *99.99*.
  - Default value: *40*.
- **High maliciousness threshold**: domain and host names with a higher DomainTools risk score than the value defined here are flagged as **Malicious - High confidence**.
  After completing the analysis, enriched domain and host names with a *higher* risk score than the *high maliciousness threshold* are flagged as **Malicious - High confidence**.
  - Enter a value between *0* and *99.99*.
  - Default value: *80*.
- **Ignore non-malicious domains**: select this checkbox to to exclude from ingestion any domains and hosts whose reputation/risk score value is lower than the specified *low maliciousness threshold*.
- Click **Save** to store your changes, or **Cancel** to discard them.

By default, the DomainTools Malicious Server Domains enricher timeout value is set to *1 minute*.

DomainTools Reputation enricher

The Domaintools Reputation enricher returns risk scores to assess the reputation of the specified input domain and host names.

This article describes how to configure the DomainTools Reputation enricher parameters.
To configure the general options for the DomainTools Reputation enricher, see Configure enrichers.

DomainTools Reputation	enricher
Enricher name	DomainTools Reputation
API endpoint	`http://api.domaintools.com/v1/reputation`
Input	domain, host
Output	Enriches supported observable types with reputation information.
Description	Enriches domain and host name observables with reputation/risk score information to assess maliciousness confidence levels, based on user-defined threshold values.

Configure the DomainTools Reputation enricher parameters

✔ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the DomainTools Reputation enricher.
  Supported observable types:
  - *domain*
  - *host*

Under **Parameters**, define the specific configuration options for the DomainTools Reputation enricher:

- **Username**: sign up and subscribe to the service to obtain the required API user name and API key credentials to access the API endpoint exposing the service.
- **API key**: contact DomainTools to receive an API key, and then enter it in the corresponding input field.
- **Low maliciousness threshold**: domain and host names with a higher DomainTools risk score than the value defined here are flagged as **Malicious - Low confidence**.
  After completing the analysis, enriched domain and host names with a *higher* risk score than the *low maliciousness threshold* and lower than the medium and high maliciousness thresholds are flagged as **Malicious - Low confidence**.
  - Enter a value between *0* and *99.99*.
  - Default value: *10*.

- **Medium maliciousness threshold**: domain and host names with a higher DomainTools risk score than the value defined here are flagged as **Malicious - Medium confidence**.
  After completing the analysis, enriched domain and host names with a *higher* risk score than the *medium maliciousness threshold* and lower than the high maliciousness threshold are flagged as **Malicious - Medium confidence**.
  - Enter a value between *0* and *99.99*.
  - Default value: *40*.
- **High maliciousness threshold**: domain and host names with a higher DomainTools risk score than the value defined here are flagged as **Malicious - High confidence**.
  After completing the analysis, enriched domain and host names with a *higher* risk score than the *high maliciousness threshold* are flagged as **Malicious - High confidence**.
  - Enter a value between *0* and *99.99*.
  - Default value: *80*.
- Click **Save** to store your changes, or **Cancel** to discard them.

By default, the DomainTools Reputation enricher timeout value is set to *1 minute*.

DomainTools Parsed Whois enricher

The DomainTools Parsed Whois enricher returns malicious domain names related to the same primary and/or secondary name servers.

This article describes how to configure the DomainTools Parsed Whois enricher parameters.
To configure the general options for the DomainTools Parsed Whois enricher, see Configure enrichers.

DomainTools Parsed Whois	**enricher**
Enricher name	DomainTools Parsed Whois
API endpoint	`http://api.domaintools.com/v1/{}/whois/parsed`
Input	domain, host, ipv4
Output	Enriches supported observable types with structured Whois information.
Description	Enriches domains, hosts, and IP addresses with Whois information. The JSON output includes the most recent Whois record for the requested domain or IP range, as well as parsed, structured data such as registrant, registrar, contacts, and so on. It helps searching for, indexing, and cross-referencing data in a set of Whois records.

Configure the DomainTools Parsed Whois enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the DomainTools Parsed Whois enricher.
  Supported observable types:
  - *domain*
  - *host*
  - *ipv4*

Under **Parameters**, define the specific configuration options for the DomainTools Parsed Whois enricher:

- **Username**: sign up and subscribe to the service to obtain the required API user name and API key credentials to access the API endpoint exposing the service.
- **API key**: contact DomainTools to receive an API key, and then enter it in the corresponding input field.
- Click **Save** to store your changes, or **Cancel** to discard them.

ℹ By default, the DomainTools Parsed Whois enricher timeout value is set to *1 minute*.

DomainTools Suspicious Domains enricher

The DomainTools Suspicious Domains enricher returns suspicious and potentially malicious domains related to the input IP addesses, along with their risk scores to automatically flag domains with an appropriate maliciousness confidence level.

This article describes how to configure the DomainTools Suspicious Domains enricher parameters.
To configure the general options for the DomainTools Suspicious Domains enricher, see Configure enrichers.

DomainTools Suspicious Domains	**enricher**
Enricher name	DomainTools Suspicious Domains
API endpoint	`https://api.domaintools.com/v1/{}/host-domains`
Input	ipv4
Output	Enriches supported observable types with suspicious domain and host name information.
Description	Enriches IPv4 observables with suspicious domains related to the input IP addresses. It includes configurable thresholds to assign maliciousness confidence levels to the processed IP addresses, and to ignore non-malicious IPs.

Configure the DomainTools Suspicious Domains enricher parameters

✔ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the DomainTools Suspicious Domains enricher.
  Supported observable types:
  - *ipv4*

Under **Parameters**, define the specific configuration options for the DomainTools Suspicious Domains enricher:

- **Username**: sign up and subscribe to the service to obtain the required API user name and API key credentials to access the API endpoint exposing the service.
- **API key**: contact DomainTools to receive an API key, and then enter it in the corresponding input field.
- **Low maliciousness threshold**: domain and host names with a higher DomainTools risk score than the value defined here are flagged as **Malicious - Low confidence**.
  After completing the analysis, enriched domain and host names with a *higher* risk score than the *low maliciousness threshold* and lower than the medium and high maliciousness thresholds are flagged as **Malicious - Low confidence**.
  - Enter a value between *0* and *99.99*.
  - Default value: *10*.

- **Medium maliciousness threshold**: domain and host names with a higher DomainTools risk score than the value defined here are flagged as **Malicious - Medium confidence**.
  After completing the analysis, enriched domain and host names with a *higher* risk score than the *medium maliciousness threshold* and lower than the high maliciousness threshold are flagged as **Malicious - Medium confidence**.
  - Enter a value between *0* and *99.99*.
  - Default value: *40*.
- **High maliciousness threshold**: domain and host names with a higher DomainTools risk score than the value defined here are flagged as **Malicious - High confidence**.
  After completing the analysis, enriched domain and host names with a *higher* risk score than the *high maliciousness threshold* are flagged as **Malicious - High confidence**.
  - Enter a value between *0* and *99.99*.
  - Default value: *80*.
- **Ignore non-malicious domains**: select this checkbox to to exclude from ingestion any domains whose reputation/risk score value is lower than the specified *low maliciousness threshold*.
- Click **Save** to store your changes, or **Cancel** to discard them.

By default, the DomainTools Suspicious Domains enricher timeout value is set to *1 minute*.

Elasticsearch sightings enricher

The Elasticsearch sightings enricher creates sightings from matching results returned from a search in an external Elasticsearch instance.

This article describes how to configure the Elasticsearch sightings enricher parameters.
To configure the general options for the Elasticsearch sightings enricher, see Configure enrichers.

Elasticsearch sightings	**enricher**
Enricher name	Elasticsearch sightings
API endpoint	`http://<elasticsearch_instance_url>:9200/<schema_resource>`
Input	domain, hash-md5, hash-sha1, hash-sha256, hash-sha512, host, ipv4, ipv6, uri
Output	Creates sightings from matching results returned from a search in an external Elasticsearch instance.
Description	Searches an external Elasticsearch instance. Any hits matching the search criteria are processed to automatically generate corresponding sightings.

Configure the Elasticsearch sightings enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the Elasticsearch sightings enricher.
  Supported observable types:
  - *domain*
  - *hash-md5*
  - *hash-sha1*
  - *hash-sha256*
  - *hash-sha512*
  - *host*
  - *ipv4*
  - *ipv6*
  - *uri*

Under **Parameters**, define the specific configuration options for the Elasticsearch sightings enricher:

- **ElasticSearch URL**: enter the URL pointing to the external Elasticsearch instance you want to use as a source for the enricher, including the sub-resource pointing to the **data mapping schema** `(https://www.elastic.co/guide/en/elasticsearch/guide/current/mapping-intro.html)`.
  Example: *http://localhost:9200/default*

  In a usage scenario, you may want to obtain data from an external Elasticsearch instance that acts as a centralized log aggregator to check for correlations with the platform observables, indicators, and other entities. If it is possible to establish a relationship between Elasticsearch data and a platform entity, a sighting is automatically created.
- **Username**: enter valid user name credentials to authenticate and to receive authorization to access the resource(s).
  Example: *nigeltufnel*.
- **Password**: enter valid password credentials to authenticate and to receive authorization to access the resource(s).
  Example: *thesegoto11*.
- **Observable queries**: from the drop-down menu select the observable type and the corresponding observable value the rule should look for.
  - In the first input field, from the drop-down menu select the *observable type* the rule should look for.
    Supported observable types:
    - *domain*
    - *hash-md5*
    - *hash-sha1*
    - *hash-sha256*
    - *hash-sha512*
    - *host*
    - *ipv4*
    - *ipv6*
    - *uri*
  - In the second input field, specify the *observable value* associated with the observable type that the rule should look for.

    You can use free text, wildcards, **Elasticsearch query syntax** `(https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html)`, as well as the *{kind}* and *{value}* placeholders to reference an observable type and value, respectively.

    When the query executes, the placeholders take the values from the input observable key (*{kind}*) and value (*{value}*) pairs, respectively.

    Example:
    The **@{value}* query searches for observable values matching the input observable values it is fed at runtime.
- Click ✚ **Add** or ✚ **More** to add a new filtering option. For example, to include in the search additional key/value pairs like IP addresses, hashes, or domains.
- **Search results limit**: if you want to limit the returned search results, so that the search result entries do not exceed a predefined amount, you can set a cap here.
  For example: *10*.
- Click **Save** to store your changes, or **Cancel** to discard them.

ℹ By default, the Elasticsearch sightings enricher timeout value is set to *2 minutes*.

Farsight DNSDB enricher

The Farsight DNSDB enricher provides historical passive DNS information to relate domain names to the IP addreses they point to, or IPs pointing to different domains over time.

This article describes how to configure the Farsight DNSDB enricher parameters.
To configure the general options for the Farsight DNSDB enricher, see Configure enrichers.

Farsight DNSDB	**enricher**
Enricher name	Farsight DNSDB
API endpoint	`https://api.dnsdb.info/{}`
Input	domain, host, ipv4, ipv6
Output	Enriches supported observable types with passive DNS lookup information such as the name of the domain or host name owner, or the IP address a domain or host name points to.
Description	Historical passive DNS lookup enricher. It can retrieve previous domains pointing to a specified IP address in the past, domain names hosted by a nameserver, domain names pointing to an IP network, and subdomains existing below a parent domain name.

Configure the Farsight DNSDB enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the Farsight DNSDB enricher.
  Supported observable types:
  - *domain*
  - *host*
  - *ipv4*
  - *ipv6*

Under **Parameters**, define the specific configuration options for the Farsight DNSDB enricher:

- **API URL**: the URL pointing to the API endpoint exposing the service that grants access to the enricher data source.
  Contact the intel provider to subscribe to the service and to obtain this information, as well as any required authentication and authorization credentials.
  The API URL to reach the Farsight DNSDB service is *https://api.dnsdb.info/{}*.
- **API key**: contact Farsight to receive an API key for the DNSDB service, and then enter it in the corresponding input field.
- **Search results limit**: enter an integer to limit the maximum amount of returned results.
  Default value: each time the enricher runs, it can return max. *1000* matches.

- **Time last seen**: enter an integer to set a starting point in the past to retrieve matches from. The number indicates the number of days in the past from the current time.
  Default value: *365* (each time the enricher runs, it looks for matches up to one year old)
- Click **Save** to store your changes, or **Cancel** to discard them.

FireEye iSIGHT enricher

The FireEye iSIGHT enricher retrieves threat intelligence reports about threats and malware related to areas such as critical infrastructure, cyber crime and espionage, hacktivism, frauds, and vulnerability and exploitation.

This article describes how to configure the FireEye iSIGHT enricher parameters.
To configure the general options for the FireEye iSIGHT enricher, see Configure enrichers.

FireEye iSIGHT	enricher
Enricher name	FireEye iSIGHT
API endpoint	`https://api.isightpartners.com/search/{}`
Input	asn, domain, email, email-subject, file, hash-md5, hash-sha1, hash-sha256, host, ipv4, ipv6, uri
Output	Enriches supported observable types related to the matching input observable types.
Description	Enriches platform observables with data about threatwww.threatcrowd.org/s and malware related to areas such as critical infrastructure, cyber crime and espionage, hacktivism, frauds, and vulnerability and exploitation.

Configure the FireEye iSIGHT enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the FireEye iSIGHT enricher.
  Supported observable types:
  - *asn*
  - *domain*
  - *email*
  - *email-subject*
  - *file*
  - *hash-md5*
  - *hash-sha1*
  - *hash-sha256*
  - *host*
  - *ipv4*
  - *ipv6*
  - *uri*

Under **Parameters**, define the specific configuration options for the FireEye iSIGHT enricher:

- **FireEye API URL**: the URL pointing to the API endpoint exposing the service that grants access to the incoming feed data. Contact the intel provider to subscribe to the service and to obtain this information.
  The API URL to reach the FireEye iSIGHT service is *https://api.isightpartners.com/search/{}*.
- **API public key**: enter the API public key to authorize your account. To access the FireEye iSIGHT service you need to authenticate using a public and a private key.
  Make sure both keys are securely stored.
- **API private key**: enter the API private key to authorize your account. To access the FireEye iSIGHT service you need to authenticate using a public and a private key.
  Make sure both keys are securely stored.
- **Search results limit**: enter an integer to limit the maximum amount of returned results.
  Default value: each time the enricher runs, it can return max. *1000* matches.
- **Day offset**: enter an integer to set a starting point in the past to retrieve matches from. The number indicates the number of days in the past from the current time.
  Default value: *365* (each time the enricher runs, it looks for matches up to one year old)
- **ThreatScape products**: from the drop-down menu select which ThreatScape products you want to associate with the enricher, so that it can use them as data sources:
  - *Hacktivism*: data source focusing on hacktivists, black-hat hackers, and terrorist groups.
  - *Enterprise*: data source focusing on corporate threats.
  - *Cyber Espionage*: data source focusing on cyber and industrial espionage.
  - *Critical Infrastructure*: data source focusing on threats to **critical infrastructure** `(https://en.wikipedia.org/wiki/critical_infrastructure)` such as electricity generators, water suppliers, telecom, and so on.
  - *Cyber Crime*: data source focusing on criminal activities carried out through computers and Internet.
  - *High Value Auction Fraud*: data source focusing on illicit activities targeting auctions.
  - *Vulnerability and Exploitation*: data source focusing on vulnerabilities and exploits.
- **Intelligence types**: from the drop-down menu select the intelligence report types you want to associate with the enricher, so that it can use them as data sources:
  - *Malware*: the enricher searches intelligence reports about malware.
  - *Threat*: the enricher searches intelligence reports about threats such as threat actors, stategies, tactics, techniques, campaigns, and so on.
- Click **Save** to store your changes, or **Cancel** to discard them.

Flashpoint AggregINT enricher

The Flashpoint AggregINT enricher provides information on a range of cyber threat data like actor information, email and IP addresses, domains, and hash files. It focuses on threat actors and criminal groups.

This article describes how to configure the Flashpoint AggregINT enricher parameters.
To configure the general options for the Flashpoint AggregINT enricher, see Configure enrichers.

Flashpoint AggregINT	**enricher**
Enricher name	Flashpoint AggregINT
API endpoint	`https://endlesstunnel.info/v3`
Input	actor-id, domain, email, hash-md5, hash-sha1, hash-sha256, hash-sha512, host, ipv4, ipv6, uri
Output	Enriches supported observable types with information such as IP addresses, domains, host names, and hash files.
Description	Polls data from the Flashpoint API. It provides information on malware, hosts, domains, IP addresses, and hashed files. The enricher can search thematic datasets focusing on hackers, terrorist and white supremacist groups, communities in conflict, state actors involved in cyberwarfare, and **CBRN** `(https://en.wikipedia.org/wiki/cbrn_defense)` threats. It produces enrichment observables like forum name, forum room name, user name of the author of a post (as actor-id), post content, thread title, UTC date and time of a post in **ISO 8601** `(https://en.wikipedia.org/wiki/iso_8601)` **(RFC 3339)** `(https://tools.ietf.org/html/rfc3339)` format.

Configure the Flashpoint AggregINT enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the Flashpoint AggregINT enricher.
  Supported observable types:
  - *actor-id*
  - *domain*
  - *email*
  - *hash-md5*
  - *hash-sha1*
  - *hash-sha256*
  - *hash-sha512*
  - *host*
  - *ipv4*
  - *ipv6*
  - *uri*

Under **Parameters**, define the specific configuration options for the Flashpoint AggregINT enricher:

- **API URL**: the URL pointing to the API endpoint exposing the service that grants access to the enricher data source. Contact the intel provider to subscribe to the service and to obtain this information, as well as any required authentication and authorization credentials.
- **Username**: enter the user name associated with the Flashpoint AggregINT account to access and consume the Flashpoint AggregINT service.
- **Password**: enter the password associated with the Flashpoint AggregINT account to access and consume the Flashpoint AggregINT service.
- **Hacker dataset**: select this checkbox to search data on hacker groups and activities.
- **Terrorist dataset**: select this checkbox to search data on terrorist groups and activities.
- **White supermacist dataset**: select this checkbox to search data on white supremacist groups and activities.
- **CBRN dataset**: select this checkbox to search data on **CBRN** `(https://en.wikipedia.org/wiki/cbrn_defense)` threats.
- **State actor dataset**: select this checkbox to search data on state actors, that is, individuals who act on behalf of a governmental body, and their activities.
- **Communities in conflict dataset**: select this checkbox to search data on groups and communities currently in conflict with each other.
- Click **Save** to store your changes, or **Cancel** to discard them.

> By default, the Flashpoint AggregINT enricher timeout value is set to *2 minutes*.

Flashpoint Blueprint enricher

The Flashpoint Blueprint enricher provides information on a range of cyber threat data like actor information, email and IP addresses, domains, and hash files. It focuses on threat actors and criminal groups.

This article describes how to configure the Flashpoint Blueprint enricher parameters.
To configure the general options for the Flashpoint Blueprint enricher, see Configure enrichers.

Flashpoint Blueprint	enricher
Enricher name	Flashpoint Blueprint
API endpoint	`https://endlesstunnel.info/v3`
Input	actor-id, domain, email, hash-md5, hash-sha1, hash-sha256, hash-sha512, host, ipv4, ipv6, uri
Output	Enriches supported observable types with information such as IP addresses, domains, host names, and URLs.
Description	Polls data from the Flashpoint API. It provides information based on geolocation and IP ranges, as well as on country scope. The enricher can search thematic datasets focusing on hackers, terrorist and white supremacist groups, state actors involved in cyberwarfare, and **CBRN** `(https://en.wikipedia.org/wiki/cbrn_defense)` threats. It produces enrichment observables like city/country name or IP address hit, latitude/longitude or IP address hit, forum name and thread title related to a hit, user name uniquely matched to an IP address hit.

Configure the Flashpoint Blueprint enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the Flashpoint Blueprint enricher.
  Supported observable types:
  - *actor-id*
  - *domain*
  - *email*
  - *hash-md5*
  - *hash-sha1*
  - *hash-sha256*
  - *hash-sha512*
  - *host*
  - *ipv4*
  - *ipv6*
  - *uri*

Under **Parameters**, define the specific configuration options for the Flashpoint Blueprint enricher:

- **API URL**: the URL pointing to the API endpoint exposing the service that grants access to the enricher data source. Contact the intel provider to subscribe to the service and to obtain this information, as well as any required authentication and authorization credentials.
- **Username**: enter the user name associated with the Flashpoint Blueprint account to access and consume the Flashpoint Blueprint service.
- **Password**: enter the password associated with the Flashpoint Blueprint account to access and consume the Flashpoint Blueprint service.
- **Hacker dataset**: select this checkbox to search data on hacker groups and activities.
- **Terrorist dataset**: select this checkbox to search data on terrorist groups and activities.
- **White supermacist dataset**: select this checkbox to search data on white supremacist groups and activities.
- **CBRN dataset**: select this checkbox to search data on **CBRN** `(https://en.wikipedia.org/wiki/cbrn_defense)` threats.
- **State actor dataset**: select this checkbox to search data on state actors, that is, individuals who act on behalf of a governmental body, and related activities.
- Click **Save** to store your changes, or **Cancel** to discard them.

> By default, the Flashpoint Blueprint enricher timeout value is set to *2 minutes*.

Flashpoint Thresher enricher

The Flashpoint Thresher enricher provides information on a range of cyber threat data like actor information, email and IP addresses, domains, and hash files. It focuses on threat actors and criminal groups.

This article describes how to configure the Flashpoint Thresher enricher parameters.
To configure the general options for the Flashpoint Thresher enricher, see Configure enrichers.

Flashpoint Thresher	**enricher**
Enricher name	Flashpoint Thresher
API endpoint	`https://endlesstunnel.info/v3`
Input	domain, file, hash-sha1, host, ipv4, uri
Output	Enriches supported observable types with information such as IP addresses, domains, URLs, hashes, and files.
Description	Polls data from the Flashpoint API. The enricher can search thematic datasets focusing on hackers, terrorist and white supremacist groups, and **CBRN** `(https://en.wikipedia.org/wiki/cbrn_defense)` threats. It produces enrichment observables with Flashpoint torrent thresher data.

Configure the Flashpoint Thresher enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the Flashpoint Thresher enricher.
  Supported observable types:
  - *domain*
  - *file*
  - *hash-sha1*
  - *host*
  - *ipv4*
  - *uri*

Under **Parameters**, define the specific configuration options for the Flashpoint Thresher enricher:

- **API URL**: the URL pointing to the API endpoint exposing the service that grants access to the enricher data source. Contact the intel provider to subscribe to the service and to obtain this information, as well as any required authentication and authorization credentials.

- **Username**: enter the user name associated with the Flashpoint Thresher account to access and consume the corresponding Flashpoint Thresher service.
- **Password**: enter the password associated with the Flashpoint Thresher account to access and consume the corresponding Flashpoint Thresher service.
- **Hacker dataset**: select this checkbox to search data on hacker groups and related activities.
- **Terrorist dataset**: select this checkbox to search data on terrorist groups and related activities.
- **White supermacist dataset**: select this checkbox to search data on white supremacist groups and related activities.
- **CBRN dataset**: select this checkbox to search data on **CBRN** `(https://en.wikipedia.org/wiki/cbrn_defense)` threats.
- Click **Save** to store your changes, or **Cancel** to discard them.

> ℹ By default, the Flashpoint Thresher enricher timeout value is set to *2 minutes*.

Fox-IT InTELL Portal enricher

The Fox-IT InTELL Portal enricher provides information from a range of sources, such as forums and sites that have registered potentially suspicious activity.

This article describes how to configure the Fox-IT InTELL Portal enricher parameters.
To configure the general options for the Fox-IT InTELL Portal enricher, see Configure enrichers.

Fox-IT InTELL Portal	**enricher**
Enricher name	Fox-IT InTELL Portal
API endpoint	`https://cybercrime-portal.fox-it.com/`
Input	domain, hash-md5, hash-sha1, hash-sha256, host, ipv4, ipv6, uri
Output	Enriches supported observable types with relevant contextual information from forums, chats, and IRC channels.
Description	Based on Fox-IT InTELL, the portal gathers cyber threat intelligence from a range of sources, such as forums and sites that have registered potentially suspicious activity.

Configure the Fox-IT InTELL Portal enricher parameters

> ✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the Fox-IT InTELL Portal enricher.
  Supported observable types:
  - *domain*
  - *hash-md5*
  - *hash-sha1*
  - *hash-sha256*
  - *host*
  - *ipv4*
  - *ipv6*
  - *uri*

Under **Parameters**, define the specific configuration options for the Fox-IT InTELL Portal enricher:

- **Fox-IT InTELL portal URL** : the URL pointing to the API endpoint exposing the service that grants access to the enricher data. Contact the intel provider to subscribe to the service and to obtain this information.

- **SSL certificate file path**: enter the path to the locally stored *.pem* or *.crt* SSL certificate you obtain from Fox-IT after subscribing to InTELL.
- **SSL key file path**: enter the path to the locally stored *.pem* or *.key* SSL private key related to the SSL certificate.
- **Username**: enter the user name associated with the Fox-IT InTELL Portal account to access and consume the InTELL service.
- **Password**: enter the password associated with the Fox-IT InTELL Portal account to access and consume the InTELL service.
- Click **Save** to store your changes, or **Cancel** to discard them.

ℹ By default, the Fox-IT InTELL Portal enricher timeout value is set to *2 minutes*.

Intel 471 enricher

The Intel 471 enricher provides information on compromised IP addresses, domains, URLs, and emails, as well as first-hand information about threat actors and groups.

This article describes how to configure the Intel 471 enricher parameters.
To configure the general options for the Intel 471 enricher, see Configure enrichers.

Intel 471	**enricher**
Enricher name	Intel 471
API endpoint	`https://api.intel471.com/v1/`
Input	actor-id, domain, email, hash-md5, hash-sha256, host, ipv4, ipv6, uri
Output	Enriches supported observable types with data focusing on threat actor information.
Description	Besides data on compromised IP addresses, domains, URLs, and emails, Intel 471 focuses on providing first-hand information about threat actors and groups.

Configure the Intel 471 enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the Intel 471 enricher.
  Supported observable types:
  - *actor-id*
  - *domain*
  - *email*
  - *hash-md5*
  - *hash-sha256*
  - *host*
  - *ipv4*
  - *ipv6*
  - *uri*

Under **Parameters**, define the specific configuration options for the Intel 471 enricher:

- **API URL**: the URL pointing to the API endpoint exposing the service that grants access to the enricher data source. Contact the intel provider to subscribe to the service and to obtain this information, as well as any required authentication and authorization credentials.
- **API key**: contact Intel 471 to receive an API key, and then enter it in the corresponding input field.

- **Email**: enter the email address associated with the Intel 471 account to access and consume the Intel 471 API service.
- Click **Save** to store your changes, or **Cancel** to discard them.

By default, the Intel 471 enricher timeout value is set to *2 minutes*.

OpenDNS OpenResolve enricher

The Cisco OpenDNS OpenResolve enricher provides reverse-DNS lookup information.

This article describes how to configure the OpenDNS OpenResolve enricher parameters.
To configure the general options for the OpenDNS OpenResolve enricher, see Configure enrichers.

Cisco OpenDNS OpenResolve	**enricher**
Enricher name	OpenDNS OpenResolve
API endpoint	`http://api.openresolve.com/{}/{}`
Input	domain, host, ipv4, ipv6
Output	Enriches supported observable types with reverse-DNS lookup information.
Description	OpenResolve by OpenDNS offers a REST API to use DNS resolvers and to retrieve reverse-DNS lookup information.

Configure the OpenDNS OpenResolve enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the OpenDNS OpenResolve enricher.
  Supported observable types:
  - *domain*
  - *host*
  - *ipv4*
  - *ipv6*

The OpenDNS OpenResolve enricher has no specific parameters to configure.

- To modify the general options for the enricher, click **Edit**.
- Click **Save** to store your changes, or **Cancel** to discard them.

By default, the OpenDNS OpenResolve enricher timeout value is set to *2 minutes*.

PassiveTotal IP/Domain enricher

The PassiveTotal IP/Domain enricher provides additional information for the queried IP address or domain name.

This article describes how to configure the PassiveTotal IP/Domain enricher parameters.
To configure the general options for the PassiveTotal IP/Domain enricher, see Configure enrichers.

PassiveTotal IP/Domain	**enricher**
Enricher name	PassiveTotal IP/Domain
API endpoint	`https://api.passivetotal.org/v2`
Input	domain, host, ipv4, ipv6
Output	Enriches supported observable types with **enrichment** `(https://passivetotal.readthedocs.io/en/latest/api.html#enrichment-request)` information.
Description	Polls data from the **PassiveTotal API** `(https://api.passivetotal.org/api/docs/#api-enrichment-getv2enrichmentquery)`. It provides additional context related to the queried IP address or domain name. For example, it returns domain name, any sub-domains, inet details, **autonomous systen number (ASN)** `(https://en.wikipedia.org/wiki/autonomous_system_(internet))`, as well as geolocation information. Analysts can query the returned data to look for further connections that may be relevant during an investigation.

Configure the PassiveTotal IP/Domain enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the PassiveTotal IP/Domain enricher.
  Supported observable types:
  - *domain*
  - *host*
  - *ipv4*
  - *ipv6*

Under **Parameters**, define the specific configuration options for the PassiveTotal IP/Domain enricher:

- **API URL**: the URL pointing to the API endpoint exposing the service that grants access to the enricher data source. Contact the intel provider to subscribe to the service and to obtain this information, as well as any required authentication and authorization credentials.

- **API key**: contact PassiveTotal to receive an API key, and then enter it in the corresponding input field.
- **Email**: enter the email address associated with the PassiveTotal IP/Domain account to access and consume the PassiveTotal IP/Domain API service.
- Click **Save** to store your changes, or **Cancel** to discard them.

ℹ By default, the PassiveTotal IP/Domain enricher timeout value is set to *2 minutes*.

PassiveTotal Malware enricher

The PassiveTotal Malware enricher provides malware information related to the queried host or domain, such as malware hash and collection date.

This article describes how to configure the PassiveTotal Malware enricher parameters.
To configure the general options for the PassiveTotal Malware enricher, see Configure enrichers.

PassiveTotal Malware	**enricher**
Enricher name	PassiveTotal Malware
API endpoint	`https://api.passivetotal.org/v2`
Input	domain, host
Output	Enriches supported observable types with **malware enrichment** `(https://passivetotal.readthedocs.io/en/latest/api.html?highlight=malware#enrichment-request)` information.
Description	Polls data from the **PassiveTotal API** `(https://api.passivetotal.org/api/docs/#api-enrichment-getv2enrichmentmalwarequery)`. It returns malware information related to the queried host or domain, such as malware hash — hash-md5, hash-sha1, hash-sha256, hash-sha512 — and collection date. The returned malware entries are also tagged with the following metadata: `enrichment_extracts.meta.classification: bad` — it corresponds to the value you set under **Data configuration > Rules > Observable > ✚ > Action > Mark as malicious**; `enrichment_extracts.meta.confidence: low` — it corresponds to the value you set under **Data configuration > Rules > Observable > ✚ > Confidence > Malicious - Low confidence**.

Configure the PassiveTotal Malware enricher parameters

✔ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the PassiveTotal Malware enricher.
  Supported observable types:
  - *domain*
  - *host*

Under **Parameters**, define the specific configuration options for the PassiveTotal Malware enricher:

- **API URL**: the URL pointing to the API endpoint exposing the service that grants access to the enricher data source. Contact the intel provider to subscribe to the service and to obtain this information, as well as any required authentication and authorization credentials.
- **API key**: contact PassiveTotal to receive an API key, and then enter it in the corresponding input field.

- **Email**: enter the email address associated with the PassiveTotal Malware account to access and consume the PassiveTotal Malware API service.
- Click **Save** to store your changes, or **Cancel** to discard them.

By default, the PassiveTotal Malware enricher timeout value is set to *2 minutes*.

PassiveTotal Passive DNS enricher

The PassiveTotal Passive DNS enricher provides historical information to enable cross-referencing IP addresses to the corresponding DNS domain names over time.

This article describes how to configure the PassiveTotal Passive DNS enricher parameters.
To configure the general options for the PassiveTotal Passive DNS enricher, see Configure enrichers.

PassiveTotal Passive DNS	enricher
Enricher name	PassiveTotal Passive DNS
API endpoint	`https://api.passivetotal.org/v2`
Input	domain, host, ipv4, ipv6
Output	Enriches supported observable types with **passive DNS** `(https://www.riskiq.com/products/learn-threat-research-and-analysis/)` information.
Description	Polls data from the **PassiveTotal API** `(https://api.passivetotal.org/api/docs/#api-dns-getv2dnspassivequery)`. It provides historical information that allows cross-referencing IP addresses to the corresponding DNS domain names over time. Analysts can examine how a domain name resolves to different IP addresses over time. They can then query the IP addresses to retrieve more domain names that may be relevant in an investigation.

Configure the PassiveTotal Passive DNS enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the PassiveTotal Passive DNS enricher.
  Supported observable types:
  - *domain*
  - *host*
  - *ipv4*
  - *ipv6*

Under **Parameters**, define the specific configuration options for the PassiveTotal Passive DNS enricher:

- **API URL**: the URL pointing to the API endpoint exposing the service that grants access to the enricher data source. Contact the intel provider to subscribe to the service and to obtain this information, as well as any required authentication and authorization credentials.
- **API key**: contact PassiveTotal to receive an API key, and then enter it in the corresponding input field.

- **Email**: enter the email address associated with the PassiveTotal Passive DNS account to access and consume the PassiveTotal Passive DNS API service.
- Click **Save** to store your changes, or **Cancel** to discard them.

> By default, the PassiveTotal Passive DNS enricher timeout value is set to *2 minutes*.

PassiveTotal Whois enricher

The PassiveTotal Whois enricher provides information about individuals or entities associated with an IP address or a domain name, as well as geolocation details.

This article describes how to configure the PassiveTotal Whois enricher parameters.
To configure the general options for the PassiveTotal Whois enricher, see Configure enrichers.

PassiveTotal Whois	**enricher**
Enricher name	PassiveTotal Whois
API endpoint	`https://api.passivetotal.org/v2`
Input	domain, host, ipv4, ipv6
Output	Enriches supported observable types with **whois** `(https://www.riskiq.com/products/learn-threat-research-and-analysis/)` information.
Description	Polls data from the **PassiveTotal API** `(https://api.passivetotal.org/api/docs/#api-whois-getv2whoisquery)`. It provides information about individuals or entities associated with an IP address or a domain name, as well as geolocation details. Analysts can retrieve registrar, organization, country, city, street, telephone, and email details. They can then use these details to run further queries to obtain, for example, more domain names associated with the same individual or the same company.

Configure the PassiveTotal Whois enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the PassiveTotal Whois enricher.
  Supported observable types:
  - *domain*
  - *host*
  - *ipv4*
  - *ipv6*

Under **Parameters**, define the specific configuration options for the PassiveTotal Whois enricher:

- **API URL**: the URL pointing to the API endpoint exposing the service that grants access to the enricher data source. Contact the intel provider to subscribe to the service and to obtain this information, as well as any required authentication and authorization credentials.
- **API key**: contact PassiveTotal to receive an API key, and then enter it in the corresponding input field.

- **Email**: enter the email address associated with the PassiveTotal Whois account to access and consume the PassiveTotal Whois API service.
- Click **Save** to store your changes, or **Cancel** to discard them.

> By default, the PassiveTotal Whois enricher timeout value is set to *2 minutes*.

PyDat enricher

The PyDat enricher provides whois, including historical whois, and passive DNS lookup information.

This article describes how to configure the PyDat enricher parameters.
To configure the general options for the PyDat enricher, see Configure enrichers.

PyDat	**enricher**
Enricher name	PyDat
API endpoint	`http://&lt;pydat_instance_url&gt;:8000/` (example)
Input	domain, ipv4, ipv6
Output	Enriches supported observable types with whois data, current IP resolution and passive DNS information.
Description	**PyDat** `(https://github.com/mitrecnd/whodat#pydat)` is installed locally, and it can work together with an **Elasticsearch instance** `(https://github.com/mitrecnd/whodat/tree/master/pydat#pydat-with-elasticsearch)` to provide whois, including historical whois, and passive DNS lookup information. Analysts can retrieve name, organization, country, city, street, ZIP code, telephone, and email details.

Configure the PyDat enricher parameters

✓ Input fields marked with an asterisk are required.

You need to install and set up PyDat locally. The product does not work outside a local network.
You need to configure the host before you can access PyDat features through the API endpoint.
See also:

- **Mitre blog on PyDat** `(http://www.mitre.org/capabilities/cybersecurity/overview/cybersecurity-blog/using-whois-and-passive-dns-for-intelligence)`
- **PyDat GitHub repo** `(https://github.com/mitrecnd/whodat)`

Under **Parameters**, define the specific configuration options for the PyDat enricher:

- **API URL**: the URL allowing access to the local **PyDat** `(https://github.com/mitrecnd/whodat#pydat-api)` instance.
  Example: *http://<pydat_instance_url>:8000/*
- Click **Save** to store your changes, or **Cancel** to discard them.

ℹ By default, the PyDat enricher timeout value is set to *2 minutes*.

Recorded Future enricher

The Recorded Future enricher enables you to tap into the data stream generated by the Recorded Future Temporal Analytics Engine to retrieve search results potentially malicious IPs, domains, email addresses, and hashes related to the input observable types, along with their risk scores to automatically flag domains with an appropriate maliciousness confidence level.

This article describes how to configure the Recorded Future enricher parameters.
To configure the general options for the Recorded Future enricher, see Configure enrichers.

Recorded Future	enricher
Enricher name	Recorded Future
API endpoint	`https://app.recordedfuture.com/live/sc/entity/{}`
Input	domain, hash-md5, hash-sha1, hash-sha256, hash-sha256, ipv4, ipv6
Output	Enriches supported observable types with pattern matching search results produced by the Recorded Future Temporal Analytics Engine.
Description	The enricher returns additional data such as IPs, domains, email addresses, and hashes related to the submitted observables in the specified types, as well as maliciousness confidence levels based on the retrieved risk scores.

Configure the Recorded Future enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the Recorded Future enricher.
  Supported observable types:
  - *domain*
  - *hash-md5*
  - *hash-sha1*
  - *hash-sha256*
  - *hash-sha256*
  - *ipv4*
  - *ipv6*

Under **Parameters**, define the specific configuration options for the Recorded Future enricher:

- **API key**: contact Recorded Future to receive an API key, and then enter it in the corresponding input field.

Maliciousness confidence rating is based on the Recorded Future risk scoring, where *0* means *no current evidence of risk*, whereas *99* means *very malicious*:

- **Low maliciousness threshold**: analyzed supported observables with a higher Recorded Future risk score than the value defined here are flagged as **Malicious - Low confidence**.
  After completing the analysis, enriched observables with a *higher* risk score than the *low maliciousness threshold* and lower than the medium and high maliciousness thresholds are flagged as **Malicious - Low confidence.**
    - Enter a value between *0* and *99*.
    - Default value: *5*.
- **Medium maliciousness threshold**: analyzed supported observables with a higher Recorded Future risk score than the value defined here are flagged as **Malicious - Medium confidence**.
  After completing the analysis, enriched observables with a *higher* risk score than the *medium maliciousness threshold* and lower than the high maliciousness threshold are flagged as **Malicious - Medium confidence**.
    - Enter a value between *0* and *99*.
    - Default value: *24*.
- **High maliciousness threshold**: analyzed supported observables with a higher Recorded Future risk score than the value defined here are flagged as **Malicious - High confidence.**
  After completing the analysis, enriched observables with a *higher* risk score than the *high maliciousness threshold* are flagged as **Malicious - High confidence**.
    - Enter a value between *0* and *99*.
    - Default value: *65*.
- Click **Save** to store your changes, or **Cancel** to discard them.

RIPEstat GeoIP enricher

The RIPEstat GeoIP enricher provides geolocation information related to the queried IP addresses.

This article describes how to configure the RIPEstat GeoIP enricher parameters.
To configure the general options for the RIPEstat GeoIP enricher, see Configure enrichers.

RIPEstat GeoIP	**enricher**
Enricher name	RIPEstat GeoIP
API endpoint	`https://stat.ripe.net/data/geoloc/{}` (**Geoloc** `(https://stat.ripe.net/docs/data_api#geoloc)`)
Input	ipv4, ipv6
Output	Enriches supported observable types with geolocation information related to IP addresses: coordinates, country, and city.
Description	Geolocation IP information from the RIPEstat web-based interface (**Data API** `(https://stat.ripe.net/docs/data_api)`), including latitude, longitude, country, and city.

Configure the RIPEstat GeoIP enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the RIPEstat GeoIP enricher.
  Supported observable types:
  - *ipv4*
  - *ipv6*

Under **Parameters**, define the specific configuration options for the RIPEstat GeoIP enricher:

- **API URL**: the basic URL allowing access to the **RIPEstat Data API** `(https://stat.ripe.net/docs/data_api)`.
  The value is: *https://stat.ripe.net/data*.
- Click **Save** to store your changes, or **Cancel** to discard them.

ℹ By default, the RIPEstat GeoIP enricher timeout value is set to *2 minutes*.

RIPEstat Whois enricher

The RIPEstat Whois enricher provides whois information related to the queried IP addresses.

This article describes how to configure the RIPEstat Whois enricher parameters.
To configure the general options for the RIPEstat Whois enricher, see Configure enrichers.

RIPEstat Whois	**enricher**
Enricher name	RIPEstat Whois
API endpoint	`https://stat.ripe.net/data/whois/{}` (**Whois** `(https://stat.ripe.net/docs/data_api#whois)`)
Input	ipv4, ipv6
Output	Enriches supported observable types with whois information related to IP addresses.
Description	Whois information from the RIPEstat web-based interface (**Whois REST API** `(https://github.com/ripe-ncc/whois/wiki/whois-rest-api)`), including inet number, name, organization, country, city, street, and telephone.

Configure the RIPEstat Whois enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the RIPEstat Whois enricher.
  Supported observable types:
  - *ipv4*
  - *ipv6*

Under **Parameters**, define the specific configuration options for the RIPEstat Whois enricher:

- **API URL**: the basic URL allowing access to the **RIPEstat Data API** `(https://stat.ripe.net/docs/data_api)`.
  The value is: *https://stat.ripe.net/data*.
- Click **Save** to store your changes, or **Cancel** to discard them.

By default, the RIPEstat Whois enricher timeout value is set to *2 minutes*.

Shodan enricher

The Shodan enricher uses input data such as country and city names, organization and personal names, ZIP codes, email addresses, and so on to return a list of matching IP addresses corresponding to your Internet-connected devices, along with location and user details.

This article describes how to configure the Shodan enricher parameters.
To configure the general options for the Shodan enricher, see Configure enrichers.

Shodan	**enricher**
Enricher name	Shodan
API endpoint	`https://api.shodan.io/shodan/`
Input	asn, city, company, country, country-code, domain, email, geo-lat, geo-long, hash-md5, hash-sha1, hash-sha256, hash-sha512, host, ipv4, ipv6, organization, person, port, postcode, uri
Output	Enriches supported observable types with the following information, when available: country name, city name, ZIP code, longitute, latitude, organization name, host name, IP address, open ports and services related to input IP addresses.
Description	Shodan matches input data such as email addresses, physical locations, company or personal names, to related IP addresses. It enables discovering which devices in your system or environment are connected to the Internet, where they are located, and who uses them.

Configure the Shodan enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the Shodan enricher.
  Supported observable types:
  - *asn*
  - *city*
  - *company*
  - *country*
  - *country-code*
  - *domain*
  - *email*
  - *geo-lat*
  - *geo-long*
  - *hash-md5*
  - *hash-sha1*
  - *hash-sha256*
  - *hash-sha512*
  - *host*
  - *ipv4*
  - *ipv6*
  - *organization*
  - *person*
  - *port*
  - *postcode*
  - *uri*

Under **Parameters**, define the specific configuration options for the Shodan enricher:

- **API user name**: sign up and subscribe to the service to obtain the required API user name and API key credentials to access the API endpoint exposing the service.
- **API key**: contact Shodan and create a user account to receive an API key; then enter it in the corresponding input field.
- Click **Save** to store your changes, or **Cancel** to discard them.

Although the Shodan enricher supports the `ipv6` observable type as a valid input, at the moment the Shodan API does not return any results for `ipv6` searches.

Splunk sightings enricher

The Splunk sightings enricher searches the indexes in the specified Splunk instance. Matching data is extracted and saved to the platform as sightings.

This article describes how to configure the Splunk sightings enricher parameters.
To configure the general options for the Splunk sightings enricher, see Configure enrichers.

Splunk sightings	**enricher**
Enricher name	Splunk sightings
API endpoint	`http://&lt;splunk_instance_url&gt;:8089/` (example)
Input	domain, email, hash-md5, hash-sha1, hash-sha256, hash-sha512, host, ipv4, ipv6, uri
Output	Creates sightings for matching input observables, based on the search result items retrieved in the specified Splunk instance.
Description	Based on the search queries defined in the enricher, the enricher looks for matching data in the specified Splunk instance. Matching data is extracted and saved to the platform as sightings.

Configure the Splunk sightings enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the Splunk sightings enricher.
  Supported observable types:
  - *domain*
  - *email*
  - *hash-md5*
  - *hash-sha1*
  - *hash-sha256*
  - *hash-sha512*
  - *host*
  - *ipv4*
  - *ipv6*
  - *uri*

Under **Parameters**, define the specific configuration options for the Splunk sightings enricher:

- **Splunk URL**: enter the URL pointing to the Splunk instance you want the enricher to search for matching data.
  The default management port listening to the *splunkd* daemon is port *8089*: make sure this **port is open** (`https://answers.splunk.com/answers/58888/what-are-the-ports-that-i-need-to-open.html`).
  Example: *https://splunk.instance.com:8089*
- **Username**: enter the user name associated with a valid Splunk account to access the Splunk instance you want to use as a data source for the enricher.
- **Password**: enter the password associated with a valid Splunk account to access the Splunk instance you want to use as a data source for the enricher.
- **Observable queries**: you define observable queries as key/value pairs. Each query targets a specific observable data type. You can add as many queries as you need.
  - The first field holds the query key: from the drop-down menu select the observable data type you want to look for in the specified Splunk instance.
    Supported observable types:
    - *domain*
    - *email*
    - *hash-md5*
    - *hash-sha1*
    - *hash-sha256*
    - *hash-sha512*
    - *host*
    - *ipv4*
    - *ipv6*
    - *uri*
  - The second field holds the query value: define the value of the observable type you want the enricher to look for in the specified Splunk instance.

    Input values need to be in a valid format for the selected observable type.
    Allowed formats:
    - Literal values; for example a domain name, an IP address, or an email address;
    - **Wildcards** (`https://docs.splunk.com/documentation/splunk/latest/search/wildcards`);
    - **Boolean expressions** (`https://docs.splunk.com/documentation/splunk/latest/search/booleanexpressions`);
    - **Field expressions** (`https://docs.splunk.com/documentation/splunk/latest/search/fieldexpressions`);
    - **SPL** (`https://docs.splunk.com/splexicon:spl`) and **regex** (`https://docs.splunk.com/documentation/splunk/latest/search/splandregularexpressions`).
    - *{kind}* and *{value}* placeholders to reference an observable type and value, respectively.

      When the query executes, the placeholders take the values from the input observable key (*{kind}*) and value (*{value}*) pairs, respectively.

      Example:
      The **@{value}* query searches for observable values matching the input observable values it is fed at runtime.
- Click ✚ **Add** or ✚ **More** to add new rows as needed, where you can enter additional queries targeting other observable data types.

- **Search results limit**: enter an integer if you want to limit the number of results the search returns for each query.
- Click **Save** to store your changes, or **Cancel** to discard them.

Search result matches generate sightings that are saved to the platform.
Each sighting includes the following information:

- A unique ID;
- A URL pointing to the Splunk instance data source;
- A URL with the query that retrieved the data;
- Details about the sighted observable.

For example, a Splunk index reference, the source log the data was found in, a timestamp, and any raw response data, if available.

ℹ By default, the Splunk sightings enricher timeout value is set to *2 minutes*.

SpyCloud Breach Data enricher

The SpyCloud Breach Data enricher uses input data such as IP addresses, domain names, email addresses and user access credentials to return incident and security breach information that is processed as incident entities, related observables, and enrichment observables.

This article describes how to configure the SpyCloud Breach Data enricher parameters.
To configure the general options for the SpyCloud Breach Data enricher, see Configure enrichers.

SpyCloud Breach Data	**enricher**
Enricher name	SpyCloud Breach Data
API endpoint	`https://api.spycloud.io/sp-v1/breach`
Input	domain, email, handle, ipv4, ipv6
Output	Enriches supported observable types and incident entities with account takeover (ATO) and security breach details. Generates Cybox observables, related observables, and CIQ objects. Supported output observables: *city*, *country-code*, *domain*, *email*, *geo-lat*, *geo-long*, *handle*, *host*, *ipv4*, *ipv6*, *organization*, *person*, *postcode*, *telephone*.
Description	Enriches supported observable types, as well as incident entities, with information about account takeover and security breaches, such as a description of the incident, targeted organization(s), type(s) and number of stolen records — for example, leaked or stolen assets or user access credentials — as well as when the records were made public or put for sale on the Internet. Each entry in a breach description — for example, an entry of a breached database containing login credentials (user name, password, and email address) — is a breach record.

Configure the SpyCloud Breach Data enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the SpyCloud Breach Data enricher.
  Supported observable types:
  - *domain*
  - *email*
  - *handle*
  - *ipv4*
  - *ipv6*

Under **Parameters**, define the specific configuration options for the SpyCloud Breach Data enricher:

- **SpyCloud API URL** : the URL pointing to the API endpoint exposing the service that grants access to the enricher data. Contact the intel provider to subscribe to the service and to obtain this information.

  The current API endpoint exposing the SpyCloud Breach Data service is *https://api.spycloud.io/sp-v1/breach*.

- **SpyCloud API key** : contact SpyCloud and create a user account to receive an API key; then enter it in the corresponding input field.

- **Enrich incidents starting from** : click the 📅 icon and select a date in the past. The enricher searches the SpyCloud breach catalog for relevant data starting from the specified date until the current time.

  If no date in the past is specified, the enricher defaults to searching for relevant data up to 60 days/2 months back from the current time.

  Format: *dd.MM.yyyy hh:mm:ss*
  Example: *07.02.2017 23:00:00*

- Click **Save** to store your changes, or **Cancel** to discard them.

The SpyCloud Breach Data enricher augments existing incident entities; if no matching incidents exist in the platform, it creates new incidents from the retrieved enrichment data.

When it retrieves personal data related to an actor or a victim, it checks if such information already exists in the platform; otherwise, it creates **CIQ** `(https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ciq)` **3.0** `(http://docs.oasis-open.org/ciq/v3.0/specs/ciq-specs-v3.html)`-compliant *identity* type objects.

ThreatCrowd enricher

The ThreatCrowd enricher returns suspicious and potentially malicious domains, IP addreses, email addresses, file hashes, and antivirus detections, so that you can explore relationships between events, actors, and targets.

This article describes how to configure the ThreatCrowd enricher parameters.
To configure the general options for the ThreatCrowd enricher, see Configure enrichers.

ThreatCrowd	**enricher**
Enricher name	ThreatCrowd
API endpoint	`https://www.threatcrowd.org/{}`
Input	domain, email, hash-md5, hash-sha1, hash-sha256, hash-sha512, host, ipv4, ipv6, malware
Output	Enriches supported observable types with suspicious and potentially malicious domains, IP addreses, email addresses, file hashes, and antivirus detections.
Description	Returns suspicious and potentially malicious domains, IP addreses, email addresses, file hashes, and antivirus detections, so that you can explore relationships between events, actors, and targets.

Configure the ThreatCrowd enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the ThreatCrowd enricher.
  Supported observable types:
  - *domain*
  - *email*
  - *hash-md5*
  - *hash-sha1*
  - *hash-sha256*
  - *hash-sha512*
  - *host*
  - *ipv4*
  - *ipv6*
  - *malware*

Under **Parameters**, define the specific configuration options for the ThreatCrowd enricher:

- **Time last seen**: enter an integer to set a starting point in the past to retrieve matches from. The number indicates the number of days in the past from the current time.
  Default value: *365* (each time the enricher runs, it looks for matches up to one year old)
- Click **Save** to store your changes, or **Cancel** to discard them.

> ℹ By default, the ThreatCrowd enricher timeout value is set to *1 minute*.

Unshorten-URL enricher

The Unshorten-URL polls the specified URL shortener services to return the resolved original URLs corresponding to the submitted shortened ones.

This article describes how to configure the Unshorten-URL enricher parameters.
To configure the general options for the Unshorten-URL enricher, see Configure enrichers.

RIPEstat GeoIP	**enricher**
Enricher name	Unshorten-URL
API endpoint	`https://unshorten.me/s/{}`
Input	uri
Output	Original URL corresponding to the submitted shortened one.
Description	It takes shortened URL as an input, and it returns the corresponding resolved original URLs, which can then be analyzed in the platform to discover relationships with other entities.

Configure the Unshorten-URL enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select the observable type representing the shortened URLs that the enricher submits to the specified services.
  Supported observable types:
  - *uri*

Under **Parameters**, define the specific configuration options for the Unshorten-URL enricher:

- **Providers**: enter one or more URL shortener services to use with the enricher.

  Separate multiple URL shortener services with either a comma or a white space.
  Example: *bit.ly,goo.gl,tinyurl.com*, or *bit.ly goo.gl tinyurl.com*

  You do not need to prefix the domains with the transmission protocol. If included, *http://* or *https://* is stripped at runtime.
- Click **Save** to store your changes, or **Cancel** to discard them.

VirusTotal enricher

The VirusTotal enricher provides information on malware, domains (passive DNS) and IP addresses.

This article describes how to configure the VirusTotal enricher parameters.
To configure the general options for the VirusTotal enricher, see Configure enrichers.

VirusTotal	**enricher**
Enricher name	VirusTotal
API endpoint	`https://www.virustotal.com/vtapi/v2/{}`
Input	domain, hash-md5, hash-sha1, hash-sha256, ipv4, ipv6, uri
Output	Enriches supported observable types with maliciousness confidence level information.
Description	Polls data from the VirusTotal API. It provides information on malware, domains (passive DNS) and IP addresses. Submitted data is checked against 60+ antimalware products, resulting in a detection ratio output and additional metadata information, when available.

Configure the VirusTotal enricher parameters

✓ Input fields marked with an asterisk are required.

- **Observable types**: select one or more observable types you want to enrich with data retrieved through the VirusTotal enricher.
  Supported observable types:
  - *domain*
  - *hash-md5*
  - *hash-sha1*
  - *hash-sha256*
  - *ipv4*
  - *ipv6*
  - *uri*

Under **Parameters**, define the specific configuration options for the VirusTotal enricher:

- **API key**: **sign up** `(https://www.virustotal.com/en/documentation/public-api/#getting-started)` to the VirusTotal community to automatically be assigned a personal API key to access the VirusTotal public API, and then enter it in this field.
- **Scan URLs**: select this checkbox to to **submit URLs** `(https://www.virustotal.com/en/documentation/public-api/#scanning-urls)` to VirusTotal.

- **Scan files**: select this checkbox to to **submit files/file hashes** `(https://www.virustotal.com/en/documentation/public-api/#scanning-files)` to VirusTotal.
  File hashes are embedded inside entities as raw artifacts.
- **Max low confidence infection rate**: you can set an *upper threshold* to automatically flag enriched observables with a *low confidence* value.
  After completing the sample analysis, enriched observables with a *lower* detection ratio than the specified value are flagged as **Malicious - Low confidence**.
  - Enter a numeric value between *0.1* and *0.9* — that is, *0 < value < 1* .
  - Default value: *0.3*.
- **Min high confidence infection rate**: you can set a *bottom threshold* to automatically flag enriched observables with *high confidence* value.
  After completing the sample analysis, enriched observables with a *higher* detection ratio than the specified value are flagged as **Malicious - High confidence**.
  - Enter a numeric value between *0.1* and *0.9* — that is, *0 < value < 1* .
  - Default value: *0.6*.
- Enriched observables with a detection ratio falling in the range defined by **Max low confidence infection rate** (range lower limit) and **Min high confidence infection rate** (range upper limit) are flagged as **Malicious - Medium confidence**.
- Click **Save** to store your changes, or **Cancel** to discard them.

About the confidence infection rate

VirusTotal positives / VirusTotal engines = confidence infection rate

To calculate the confidence infection rate value, the platform divides the number of positives — that is, infected or malicious results — the VirusTotal sample analysis returns by the total number of engines VirusTotal uses to perform the analysis.

Automatic flagging with high/low confidence maliciousness through VirusTotal supports only the following observable types:

- *hash-sha1*
- *hash-sha256*
- *hash-md5*
- *uri*

By default, the VirusTotal enricher timeout value is set to *2 minutes*.

Create custom enrichers

Implement custom extensions to integrate EclecticIQ Platform with external intel providers and data sources through incoming feeds and enrichers, as well as to publish platform intel downstream in your prevention and detection toolchain.

About extensions

EclecticIQ Platform integrates with many external prevention/detection solutions and intel providers. It can exchange information through feeds, retrieve data through enrichers, and it can communicate with third-party systems through its API and ad-hoc apps that implement interoperability with specific products such as Splunk and IBM QRadar.

EclecticIQ Platform ships with out-of-the-box, ready-to-use enrichers to augment cyber threat intel with observables providing additional context. It also includes a web-based UI to create incoming and outgoing feeds, as needed.

Besides the default feeds and enrichers, you can create and implement your own. Custom feeds and enrichers implemented by third-parties other than EclecticIQ are called extensions, since they extend the platform native feature set. You can create extensions to implement additional transport types or content types for incoming or outgoing feeds, as well as new enrichers to poll data from specific intel providers.

Create enricher extensions

Before getting your hands dirty, have a look at the main steps to create an enricher extension from a boilerplate:

- Download, clone, or copy the ***eclecticiq-extension-example*** `(https://github.com/eclecticiq/platform-extensions/tree/master/eclecticiq-extension-example)` extension.
- Import dependencies.
- Create a JSON schema for the UI, if your enricher extension features a UI.
- Set a schema definition for validation.
- Define the enricher type and its instantiation through decorator functions.
- Include the necessary logic to configure the enricher behavior:
  - Define the tasks the enricher should execute;
  - Define the extract types you want the enricher to look for and retrieve.
  - Configure appropriate transport and content types to handle data formats and retrieval.
- Restart Supervisor, so that all managed processes can configure the newly added extension in **Data configuration > Enrichers**.
- Enable the extension.
- Initialize the extension by running the fixtures (applies only to enricher extensions).

Prepare the boilerplate

To make it easier to create custom enricher extensions, you can use our boilerplate enricher: *eclecticiq-extension-example*.
It is a sample enricher that augments entities with social URI observables polled from Twitter and/or Facebook.
Use it as a scaffold you can rework and customize into the desired enricher extension.

- Download, clone or copy the ***eclecticiq-extension-example*** `(https://github.com/eclecticiq/platform-extensions/tree/master/eclecticiq-extension-example)` extension, save it locally, and decompress it, if necessary.
- Rename the directories as needed.
- In the root directory, open ***setup.py*** `(https://github.com/eclecticiq/platform-extensions/blob/master/eclecticiq-extension-example/setup.py)`:

```
from setuptools import setup, find_packages

setup(
    name='eclecticiq-extension-example',
    version="1.0",
    description="Example extension for EclecticIQ Platform",

    # Look for packages to build the extension
    packages=find_packages(),

    # List dependencies the extension requires
    install_requires=[
        'eiq-platform'
    ],

    # Look for and include data files, if applicable
    include_package_data=True,

    # Set an entry point for the extension
    # so that it can initialize
    entry_points={
        'eiq.extensions': [
            'example = eiq.extensions.example:prepare_extension'
        ],
    }
)
```

Edit the setup file

These are the ***setup.py*** `(https://github.com/eclecticiq/platform-extensions/blob/master/eclecticiq-extension-example/setup.py)` parts you can, and should, edit as applicable:

`name`

In the extension `name`, change `example` to a more meaningful name for your enricher extension, but leave the `eclecticiq-extension-` prefix as is.
Example:

```
name='eclecticiq-extension-fraud-ip-observables',
```

`version`

Change the `version` number as appropriate.

Make sure you implement versioning for your enrichers.
For example, increment the version number after adding or removing arguments/parameters.
The platform checks `version`, and it won't load the enricher if it detects parameter changes without a corresponding version number increment.
Example:

```
version="1.1",
```

`description`

Change the `description` value, so that it provides basic details about the enricher extension.
Example:

```
description="Custom extension to retrieve and save fraudulent IPs as observables",
```

Include dependencies

Add to the `install_requires` list the Python libraries and modules your enricher extension needs to access for it to work as expected.
`eiq-platform` is a mandatory dependency, it needs to always be included in the list.
Other Python libraries and modules you need to import and include in this list vary, depending on the specific enricher extension you are building.

Example:

```
install_requires=[
    'eiq-platform',
    'requests',
    'cabby',
    'furl'
],
```

Set the entry point

`eiq.extensions` is the designated entry point referring to the extension definitions.
The platform needs this pointer to recognize, load, and register extensions. Do not remove it.
Change `example` to a more meaningful name for your enricher extension, but leave the `eclecticiq.extensions.` prefix as is.
Example: `eclecticiq.extensions.fraud-ip-observables`

Example:

```
'eiq.extensions': [
    'fraud-ip-observables = eclecticiq.extensions.fraud-ip-observables:prepare_extension'
],
```

Define the enricher initialization

This part of the procedure customizes the fixtures you will need to run later to initialize the enricher, after enabling it.

`prepare_extension` defines how to initialize the enricher extension you are building.
In `prepare_extension` you specify what the extension should get ready before you execute it for the first time.

Change the `Extension` return values as applicable. Use the actual, correct names, descriptions, and values you define, set, and plan to use in your enricher extension.

Open ***init.py*** `(https://github.com/eclecticiq/platform-extensions/blob/master/eclecticiq-extension-example/eclecticiq/extensions/example/__init__.py).`

Boilerplate:

```
from eiq.extension_api import Extension

from .enrichers import enrich_from_social_network
from .transformers import transform_malware_domainlist_csv
from .provider import oauth_http_download

def prepare_extension():
    return Extension(
        name=__name__,
        description='Example extension for EclecticIQ Platform',
        enrichers=[enrich_from_social_network],
        transformers=[transform_malware_domainlist_csv],
        transports=[oauth_http_download]
    )
```

Example:

```
# Import the 'Extension' class from the extension API
from eiq.extension_api import Extension

# Import your custom enricher
from .enrichers import enrich_fraud_ip_observables
# Import the data formats the enricher needs to handle
from .transformers import <content_type>
# Import the data carrier the enricher uses to exchange data
from .provider import <transport_type>

# Define how to initialize the custom enricher
def prepare_extension():
    return Extension(
        name=__name__,
        description='Custom extension to retrieve and save fraudulent IPs as observables',
        enrichers=[enrich_fraud_ip_observables],
        transformers=[<content_type>],
        transports=[<transport_type>]
    )
```

Start by importing the platform class defining extensions, as well as your custom extension, so that we can initialize it. You import the standard extension class for the platform through the `extension_api`. This is the dedicated API to use when developing custom extensions.

Import the enricher from ***enrichers.py*** `(https://github.com/eclecticiq/platform-extensions/blob/master/eclecticiq-extension-example/eclecticiq/extensions/example/enrichers.py).`
This module defines, among others, enricher extension instance and type, as well as any functions containing the logic driving the enricher behavior, and the UI schema, if applicable.

Import one or more content types from ***transformers.py*** `(https://github.com/eclecticiq/platform-extensions/blob/master/eclecticiq-extension-example/eclecticiq/extensions/example/transformers.py),` so that the enricher can recognize and handle the expected data formats.
This module defines the available content types for the data the enricher extension handles: this is where you import the allowed data formats for the enricher from.

Import a transport type from ***provider.py*** `(https://github.com/eclecticiq/platform-extensions/blob/master/eclecticiq-extension-example/eclecticiq/extensions/example/provider.py)`, so that the enricher can send and receive the expected data.
This module defines the available transport types for the data the enricher extension handles.

Change the `description`, `enrichers`, `transformers`, and `transports` metadata values `Extension` returns, so that they reflect the actual name, description, enricher type, data format, and data carrier to use in the enricher extension.

Build the enricher

Enricher extension logic and any necessary functions that drive the enricher behavior live in the ***enrichers.py*** `(https://github.com/eclecticiq/platform-extensions/blob/master/eclecticiq-extension-example/eclecticiq/extensions/example/enrichers.py)` module.

Boilerplate:

```
import json

import requests
from marshmallow import Schema, fields

# Import the enricher extension dependencies
from eiq.extensions_api import (
    EnrichmentResult,
    enricher_instance,
    enricher_type,
)

# Define the Marshmallow schema for the enricher params
# The UI schema is validated against the params defined here
class SocialURIEnricherSchema(Schema):
    check_twitter = fields.Boolean(required=True)
    check_facebook = fields.Boolean(required=True)

# Define the UI schema
ui_form_schema = [
    {
        "label": "Check Twitter",
        "name": "check_twitter",
        "required": True,
        "type": "checkbox",
        "format": "bool",
        "hint": "Should the enricher query Twitter"
    },

    {
        "label": "Check Facebook",
        "name": "check_facebook",
        "required": True,
        "type": "checkbox",
        "format": "bool",
        "hint": "Should the enricher query Facebook"
    },
]

@enricher_instance(
    # Increment the version number if defaults change after
    # deploying the enricher.
    # For example, if it gets new/additional parameters
    version=1,

    # Enricher name and description displayed to users
    name='Example Social URI Enricher',
    description=('Query for registered Twitter/Facebook '
                 'accounts with provided handle/name'),

    # Boolean switch to enable or disable the enricher by default
    is_active=True,

    # If the enricher creates entities, this is the default reliability
    # value assigned to them
    source_reliability='C',

    # Additional enricher parameters.
    # They must match the name and type configured in
    # 'ui_form_schema' and in the Marshmallow schema
    check_twitter=True,
    check_facebook=True,
)

@enricher_type(
    # Define the observable types the enricher supports
    input_extract_types=['handle', 'name', 'person'],

    # Assign a UI schema to the enricher editor page in the web UI
    parameter_ui_form_schema=ui_form_schema,

    # Define the validation and deserialization UI schema
    parameter_serialization_schema=SocialURIEnricherSchema,

    # URL templates link users to the enricher source
    source_urls={
        'handle': 'https://twitter.com/${input}',
        'name': 'https://twitter.com/${input}',
        'person': 'https://twitter.com/${input}',
    },
)

# Define the logic driving the enricher behavior
def enrich_from_social_network(
        # Request arguments
        extract_type,
        extract_value,
        check_twitter,
        check_facebook):
    # The response should return:
    # - A list with matching observables
    # - A list with the raw responses, for reference
    enrichment_extracts = []
    raw_responses = []  # Keep raw response headers for user reference

    if check_twitter:
        # Send an HTTP HEAD request to Twitter
        # to see if it takes the handle
        twitter_url = 'https://twitter.com/{}'.format(extract_value)
        response = requests.head(twitter_url)

        if response.ok:
            # Return and append the raw response(s)
            raw_responses.append(dict(response.headers))
            # Return and append supported observable types and values
            enrichment_extracts.append({
                'kind': 'uri',
                'value': twitter_url
            })

    if check_facebook:
        # Send an HTTP HEAD request to Facebook
        # to see if it takes the handle
        facebook_url = 'https://facebook.com/{}'.format(extract_value)
        response = requests.head(facebook_url)

        if response.ok:
            raw_responses.append(dict(response.headers))
            enrichment_extracts.append({
                'kind': 'uri',
                'value': facebook_url
            })

    return EnrichmentResult(
        # Return the raw responses as UTF-8 JSON
        raw_data=json.dumps(raw_responses).encode('utf-8'),
        # Return a key/value pair JSON list with the retrieved observables
        extracts=enrichment_extracts,
        # Return a list with entities, if applicable
        entities=[])
```

Let’s break it down.

Import dependencies

Make sure you include the necessary Python libraries and modules in ***enrichers.py*** `(https://github.com/eclecticiq/platform-extensions/blob/master/eclecticiq-extension-example/eclecticiq/extensions/example/enrichers.py)`, so that the custom enricher extension can access the functionality required to work as expected.

The Python libraries and modules you may need to make available to your custom extension vary, depending on the extension design, scope, and purpose.

For example:

Dependency	Description
`import requests`	Adds handy automation to **HTTP requests** `(http://docs.python-requests.org/en/master/)`.
`from marshmallow import Schema, fields`	Marshmallow schemas validate UI schemas and form input.

Dependency	Description
`from eiq.extensions_api import (EnrichmentResult, enricher_instance, enricher_type,)`	The foundations to build your enricher on.

```
import json

import requests
from marshmallow import Schema, fields

# Import the enricher extension dependencies
from eiq.extensions_api import (
    EnrichmentResult,
    enricher_instance,
    enricher_type,
)
```

Set the UI schema definition

If your enricher extension features a UI, you need to include a UI JSON schema, which you need to validate.
The schema definition to validate UI schema and form input is based on a Marshmallow schema definition.

The **Marshmallow schema** `(https://marshmallow.readthedocs.io/)` defines the behavior of the controls and the components on the UI form, and the `ui_form_schema` JSON schema needs to match it to pass validation.

First, import the following classes from Marshmallow:

```
from marshmallow import Schema, fields
```

Then, set the Marshmallow schema definition to validate the `ui_form_schema` JSON schema against.
You can customize the Marshmallow schema definition as needed.

Example:

```
# Define the Marshmallow schema for the enricher params
# The UI schema is validated against the params defined here
class SocialURIEnricherSchema(Schema):
    check_twitter = fields.Boolean(required=True)
    check_facebook = fields.Boolean(required=True)
```

Lastly, include `parameter_serialization_schema` in the `enricher_type` decorator in ***enrichers.py*** `(https://github.com/eclecticiq/platform-extensions/blob/master/eclecticiq-extension-example/eclecticiq/extensions/example/enrichers.py)`, and set it so that it points to the appropriate schema definition name for the UI schema validation.

Example:

```
# Define the validation and deserialization UI schema
parameter_serialization_schema=SocialURIEnricherSchema,
```

Define the UI schema

If your enricher extension requires a UI frontend where users can make selections and set specific options, you need to include a UI schema in JSON format.
Each JSON field in the schema defines a UI component to implement in the extension. For example, an input field, or a checkbox.

Include the UI schema as a JSON array inside ***enrichers.py*** `(https://github.com/eclecticiq/platform-extensions/blob/master/eclecticiq-extension-example/eclecticiq/extensions/example/enrichers.py)`.
You can customize the UI schema as needed.

Example:

```
    # Definition of the UI form rendered in the platform UI
    # Data configuration > Enrichers > <enricher_name> > Edit enricher task
    # 'ui_form_schema' is the UI schema name; do not change it.
    ui_form_schema = [
        {
            "label": "Check Twitter",
            "name": "check_twitter",
            "required": True,
            "type": "checkbox",
            "format": "bool",
            "hint": "Should the enricher query Twitter"
        },

        {
            "label": "Check Facebook",
            "name": "check_facebook",
            "required": True,
            "type": "checkbox",
            "format": "bool",
            "hint": "Should the enricher query Facebook"
        },
    ]
```

Then, include `parameter_ui_form_schema` in the `enricher_type` decorator in ***enrichers.py*** `(https://github.com/eclecticiq/platform-extensions/blob/master/eclecticiq-extension-example/eclecticiq/extensions/example/enrichers.py)`, and set it so that it points to `ui_form_schema`.

Example:

```
# Assign the UI schema to the enricher editor page in the web UI
parameter_ui_form_schema=ui_form_schema,
```

You can define any UI schema that satisfies your requirements, provided it complies with the following guidelines:

- The UI schema format must be valid JSON.
- A UI schema for a form is a user-defined list of fields.

- Define each field with key/value pairs.
- Each key/value pair describes an attribute of the field.

Field attributes

You are free to define the naming convention and the terminology for the field names. However, field attributes are constrained and predefined. Each field takes at least two or more attributes.

`name` and `type` are required attributes, and you always need to include them in a field description.
All other attributes are optional.

`name`

The name identifying the field in the JSON object.
This name is usually not displayed to users. It is included in the JSON object containing the field, the UI schema, and the extension schema that is returned when sending an API request to the `/api/extensions/` endpoint.
Example: *includeWhois*

`label`

The name of the field as displayed as a label on the resulting object in the UI form.
Example: *Include whois information*

`type`

It defines the type of field, that is, the object it represents on the UI form:

- `text`: a one-row text input field.
  It can take the following sub-attributes:
  - `format`: it defines and restricts the allowed input format for the field, which needs to be in the specified format value.
    Allowed values:
    - `datetime`
    - `host`
    - `url`
    - `email`
    - `regex`
    - `path`
    - `text`
    - `int`
    - `float`
    - `bool`
- `textarea`: a multiple row alphanumeric text input field.
- `password`: an alphanumeric input field that accepts a user password.
- `select`: a list with multiple options. Users can select one or more options.
  It can take the following sub-attributes:
  - `options`: a JSON array with key/value pairs. Each key/value pair defines one option.
    Format: `[{"name": "...", "value" :"..."},...]`
  - `multiple`: Boolean, either `true` or `false`. It defines whether users are allowed to make multiple selections.

- `radio`: a control element that allows users to select one option in a set of options.
  It can take the following sub-attributes:
  - `options`: a JSON array with key/value pairs. Each key/value pair defines one option.
    Format: `[{"name": "...", "value" :"..."},...]`
- `checkbox`: a control element that allows users to select/deselect, enable/disable an item or a feature.
  It can take the following sub-attributes:
  - `options`: a JSON array with key/value pairs. Each key/value pair defines one option.
    Format: `[{"name": "...", "value" :"..."},...]`
  - If you do not include the `options` sub-attribute, the `checkbox` type defaults to a single component accepting Boolean values, either `true` or `false`.
- `extra`: include this type if you want to include in your UI form any additional free-form parameters, for example HTTP headers.
  It can take the following sub-attributes:
  - `names`: a JSON array holding the name values of the extra free-form parameters. For example, the specific HTTP header names you want to add.
    Format: `["name1", "name2", ...]`
  - `allow_new`: Boolean, either `true` or `false`. It allows/denies adding new keys to the extra parameter list.

`required`

Boolean, either `true` or `false`.
It flags the field as either mandatory, that is, users must specify a value for the field, or optional.

`default`

Any value you specify for this attribute corresponds to the default value the field is pre-populated with (autofill).

`hint`

A tooltip text to give a short explanation of the field and the action the user should carry out.
Example: *Enter a numeric value between 1 and 10.*

`when`

It defines a conditional flow to show or hide the component when the specified criteria are met or not met.
Format: `{"component_x": "value_y"}`, that is, when `component_x` is set to `value_y`, the component the fields belongs to is displayed on the UI.

Define the enricher behavior

The foundations of your enricher extension are the `enricher_instance` and `enricher_type` decorators.

`enricher_instance`

This decorator defines the default values the enricher takes when it is instantiated in the platform.

The following arguments are mandatory:

- `version`: make sure you implement versioning for your enrichers.
  For example, increment the version number after adding or removing arguments/parameters.
  The platform checks `version`, and it won’t load the enricher if it detects parameter changes without a corresponding version number increment.
- `name`: a human-readable, user-friendly name that helps user understand what the enricher does.
  The `name` value is displayed on the UI.

- `description`: a human-readable, user-friendly, short free-text description that helps user understand what the enricher does.
  The `description` value is displayed on the UI.
- `is_active`: the `True` or `False` Boolean value controls the state of the enricher **Enabled** checkbox on the UI: either selected or deselected, respectively.

The following arguments are optional:

- `source_reliability`: if your enricher extension is designed to create entities as an output, this parameter sets a default reliability value for the entity data source.
- Any custom parameters defining the Marshmallow schema definition to validate the UI schema.

Example:

```
@enricher_instance(
    # Bounce the version number if you
    # add, change or remove parameters
    version=1,

    name='Example Social URI Enricher',
    description=('Query for registered Twitter/Facebook '
                 'accounts with provided handle/name'),

    is_active=True,
    # Optional, if your enricher returns entities
    source_reliability='C',

    # Custom Marshmallow UI schema definition values
    # to validate the enricher UI, if applicable
    check_twitter=True,
    check_facebook=True,
)
```

`enricher_type`

This decorator defines the enricher extension behavior. It contains the specific logic and the functionality powering your enricher.

The following arguments are mandatory:

- `input_extract_types`: a list of observable types.
  The observable types in this list are the input data the enricher takes.
  When it runs, the enricher searches for enrichment data to augment the observable types in the list.
- `parameter_ui_form_schema`: it needs to refer to `ui_form_schema`. Do not change the parameter name.
- `parameter_serialization_schema`: it needs to refer to the Marshmallow schema definition. Do not change the parameter name.
- `source_urls`: a key/value pair dictionary that produces clickable links to external data sources related to the retrieved observables. Do not change the parameter name.
  - The key name needs to be an observable type included in `input_extract_types`.
  - The value is a URL with the following format: `https://<data_source.com>/${input}`.
    The `${input}` URL variable takes the `extract_value` value.

    The resulting URL includes the original query as query URL params.
    The resulting clickable link points to the original web site the enrichment data was obtained from.

Example:

```
@enricher_type(
    input_extract_types=['handle', 'name', 'person'],

    parameter_ui_form_schema=ui_form_schema,
    parameter_serialization_schema=SocialURIEnricherSchema,

    source_urls={
        'handle': 'https://twitter.com/${input}',
        'name': 'https://twitter.com/${input}',
        'person': 'https://twitter.com/${input}',
    },
)
```

`def magic`

Now it’s time for the magic to happen. Define a workhorse function to do the grunt work.

The function you pass with the decorator should contain the necessary logic to search for and retrieve the desired input data, any conditional flow and error handling, and to output the input data as valid observables for the platform.

The following arguments are mandatory:

- `extract_type`: include this parameter, so that your function can return observable types matching `input_extract_types`.
- `extract_value`: include this parameter, so that your function can return values associated with the observable types defined in `input_extract_types`.
- Any custom-defined Marshmallow schema definition variables, if your enricher features a UI component.

The function should also create two variables to hold the returned data:

- `enrichment_extracts`: an empty list that will be populated with any matching observables included in the response.
- `raw_responses`: an empty list that will be populated with the raw responses, for reference.

Example:

```
def my_custom_enricher(
        # Mandatory arguments:
        extract_type,
        extract_value,
        ...,
        # Pass as arguments also the Marshmallow schema definition vars:
        check_twitter,
        check_facebook,
        ...
        ):

    # The function should return 2 lists:
    # - Any retrieved enrichment observables,
    # - The raw responses, for reference
    enrichment_extracts = []
    raw_responses = []  # Keep raw response headers for user reference

    ...
    # Your black magic happens here
    ...
```

`return EnrichmentResult`

Complete the function by defining the objects it should return in the response.

The `EnrichmentResult` class helps store and handle output data. It returns the following output:

- Raw response, that is, enrichment raw data (as UTF-8 JSON),
- Observables (as a list),
- Entities (as a list of entity IDs).

When building an enricher extension, it is a good idea to always use these data types to return, even when no data may be returned for observables or entities. The raw data response should always be included in the return arguments.

Example:

```
return EnrichmentResult(
        raw_data=json.dumps(raw_responses).encode('utf-8'),
        extracts=enrichment_extracts,
        entities=[])
```

Define transport and content handlers

Your enricher extension needs to include two more components:

- ***provider.py*** `(https://github.com/eclecticiq/platform-extensions/blob/master/eclecticiq-extension-example/eclecticiq/extensions/example/provider.py)`: defines a mechanism to transport data.
  Typically, a provider implements a data transport protocol, such as HTTP, FTP, TAXII, and so on.
  Enrichers require a provider that fetches data from a source, and then passes it on to the platform for further processing.
- ***transformers.py*** `(https://github.com/eclecticiq/platform-extensions/blob/master/eclecticiq-extension-example/eclecticiq/extensions/example/transformers.py)`: defines one or more mechanisms to convert retrieved raw data to the JSON format the platform can accept.

Define the data provider

To define the data provider for the enricher, you can use the ***provider.py*** `(https://github.com/eclecticiq/platform-extensions/blob/master/eclecticiq-extension-example/eclecticiq/extensions/example/provider.py)` as a scaffold you can rework and customize into the desired data transport provider for your enricher extension.

- Begin by importing the necessary Python modules and dependencies to handle the desired source data content type.
  In the example, the transport type is HTTP using OAuth authentication.
- `extension_api` is the dedicated API to use when developing custom extensions.
  From `extension_api`, import the following functions:
  - `options`: the `provider_options` passes it as an argument to control and validate UI options.
    Marshmallow uses it to set and validate dynamic form options in the UI.
  - `provider`: decorator function that marks a function as a provider type.
  - `provider_options`: decorator function to define a set of configuration options for a data provider.

```
from requests_oauthlib import OAuth1Session

from eiq.extensions_api import options, provider, provider_options
```

- The `provider` decorator flags a function as a data transport type/data provider in the platform.
  The decorator takes the following arguments:
  - `name`: an alphanumeric string holding the internal name identifying the data provider.
  - `title`: an alphanumeric string an alphanumeric string displayed in the UI. It corresponds to the transport type name for an enricher or an incoming feed.
  - `description`: an alphanumeric string displayed in the UI. It corresponds to a short description about the data provider/transport type purpose.

```
# The ``provider`` decorator will mark a certain function as a type
# of provider, along with some metadata primarily to inform users.
@provider(
    name='eiq.providers.oauth_http_download',
    title='OAuth HTTP Download',
    description=('Download content the configured URL using OAuth 1 '
                 'authentication')
)
```

- The `provider_options` decorator enables you to define any custom or additional data provider settings that platform users can select and configure in the UI through the enricher form.

  The decorator can take any custom argument key names, depending on how the source data provider interface works. Argument keys can take the following values:
  - `options.boolean`: it works like a standard Python `bool` object; it takes either `True` or `False`.
    On the UI it displays a selection checkbox form element.
  - `options.datetime`: it works like a standard Python `datetime.datetime` object.
    On the UI it displays a date-time calendar picker form element.
  - `options.integer`: it works like a standard Python `int` object.
    On the UI it displays a text input form element.
  - `options.password`: it works like a standard Python `str` object.
    On the UI it displays a password input form element.
  - `options.string`: it works like a standard Python `str` object.
    On the UI it displays a text input form element.
  - `options.url`: it works like a standard Python `str` object.
    On the UI it displays a text input form element that validates the input as a valid URL.

Each of these parameters can take additional *kwargs* to include extra information displayed to users on the UI:

- `label`: an alphanumeric string displayed in the UI. It corresponds to a UI option caption.
- `hint`: an alphanumeric string displayed in the UI. It corresponds to a pop-up tooltip triggered by a mouseover event.
- `required`: flags a UI option as required or optional. It takes either `True` or `False`, respectively.
- `default`: depending on the option type, it pre-populates the UI element with a default value.
- `validate`: it takes a function to validate user input with.
  Design the validation function so that it takes *one* value, and it returns `False` if the value is not valid.

```
# With ``provider_options`` additional configuration for a provider
# can be specified. These options will show up to users in the UI in
# an HTML form.
@provider_options(
    url=options.url(label="URL", hint="URL to download content from"),
    client_key=options.string(label="Client key"),
    client_secret=options.password(label="Client secret"),
    resource_owner_key=options.string(label="Resource owner key"),
    resource_owner_secret=options.password(label="Resource owner secret"),
)
```

Time for more magic: define a data provider to do the grunt work. The function you pass with the decorator should contain the necessary logic to handle data transport for the designated data format, and it should return/submit the data for further processing in the platform.

The function logic varies, based on your specific needs, and on the source data provider transport and content types. Make sure the function can `ctx.submit` or `return` the data for further processing, depending on how you design it.

The data provider function takes the following arguments:

- `ctx`: this must always be the first argument.
- Any arguments specified in `provider_options`.

```
# The provider function must accept a ``ctx`` object as its first
# argument. Other optional arguments are:
# :param content_type: The name of the related content_type for the incoming
#                      feed this function runs for.
# :param log: A logger instance
def oauth_http_download(
        ctx,
        url,
        client_key,
        client_secret,
        resource_owner_key,
        resource_owner_secret,
        log):

    oauth1_session = OAuth1Session(
        client_key,
        client_secret=client_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret)

    rs = oauth1_session.get(url)
    rs.raise_for_status()
    log.info("Fetched {} bytes of data".format(len(rs.content)))
    ctx.submit(rs.content)
```

Define the data transformer

To define the data format conversion mechanism for the enricher, you can use the ***transformers.py*** `(https://github.com/eclecticiq/platform-extensions/blob/master/eclecticiq-extension-example/eclecticiq/extensions/example/transformers.py)` as a scaffold you can rework and customize into the desired input-data-format-to-JSON converter for your enricher extension.

- Begin by importing the necessary Python modules and dependencies to handle the desired source data content type. In the example, the input data is in *.csv* format.

```
import csv
import io
```

- `extension_api` is the dedicated API to use when developing custom extensions.
  From `extension_api`, import the following functions:
  - `declare_content_type`: it enables you to set the input data format your enricher expects to retrieve.
  - `transformer`: it enables you to configure the mechanism to convert the expected data input format to platform-friendly JSON.
    It returns a dictionary containing `"type": "package"` similar to a STIX package, or a dictionary containing `"type": "entities"`, and `"entities": [<dicts with valid entity data>]`.

```
from eiq.extension_api import (
    declare_content_type,
    transformer,
)
```

- Define the input data format your enricher expects to retrieve by assigning it `declare_content_type` as a value.
  In the example, the input data is in *.csv* format.
  Use `declare_content_type` to define the data format by setting the following parameters:
  - `id`: the identifier value is a **URN** `(https://tools.ietf.org/html/rfc2141).`
    Format: `urn:<namespace_identifier>:<namespace_specific_string>:<version>`
    Example: `urn:malwaredomainlist:csv:1.0`
  - `name`: an alphanumeric string displayed in the UI. It corresponds to the enricher name.
  - `description`: an alphanumeric string displayed in the UI. It corresponds to a short description about the enricher purpose.
  - `file_extension`: sets the expected file extension type for the input data.
    It needs to correspond to the MIME type.
    Example: `csv`
  - `mime_type`: sets the expected **MIME type** `(https://www.iana.org/assignments/media-types/media-types.xhtml)` for the input data.
    It needs to correspond to the file extension.
    Example: `text/csv`

```
malware_domainlist_csv = declare_content_type(
    id="urn:malwaredomainlist:csv:1.0",
    name="Malware Domain List CSV",
    description="",
    file_extension="csv",
    mime_type="text/csv",
)
```

- The `transformer` decorator passes the function you define to handle data conversion from the configured input data format to platform-friendly JSON.
  It takes an `incoming_content_types` list type argument, whose values are the defined input data format names.

  Your custom function should always pass `blob` as an argument.
  The function logic varies, based on your specific needs, and on the source data provider transport and content types.
  The generic steps you should consider when building such a function are:
  - Set the expected data format and data encoding.
  - Define the logic to iterate through the input data, and to extract the desired data.
  - Create a variable to act as a container to hold the input raw data in a list.
  - Return a JSON dictionary with the desired data.
- You may want to delegate entity or observable object creation to a separate function. In the example, it is the `make_entity` function.
  - Since the input data format in the example is *.csv*, the function takes `row` as a parameter to extract the desired data, which is then stored in two JSON objects:
    - `data`: a JSON array containing the extracted entity or observable object data.
    - `meta`: a JSON array containing the extracted entity or observable object metadata.
  - Return the `data` and `meta` JSON objects.
- The function you pass with the `transformer` decorator uses the entity or observable object creation function to return a JSON array with the extracted (linked) entities.

```
@transformer(incoming_content_types=[malware_domainlist_csv])
def transform_malware_domainlist_csv(blob):
    # The blob object is of type bytes.
    # The expected encoding from malwaredomainlist.com
    # is utf-8
    csv_data = blob.decode('utf-8')

    # DictReader expects to iterate over lines of text, so we wrap the data:
    csv_data = io.StringIO(csv_data)
    csv_reader = csv.DictReader(
        csv_data,
        fieldnames=[
            'date_utc',
            'domain',
            'ip',
            'reverse_lookup',
            'description',
            'registrant',
            'asn'
        ]
    )

    # List of entities generated form each extracted csv row
    entities = [make_entity(row) for row in csv_reader]

    # Return (linked) entities
    return {
        'linked-entities': {
            'entities': entities,
        },
    }

def make_entity(row):
    data = {
        'type': 'indicator'
        'type': 'indicator',
        'title': row['domain'] + ' MalwareDomainList',
        'description': (
            'domain:         {}\n'
            'ip:             {}\n'
            'reverse lookup: {}\n'
            'description:    {}\n'
            'registrant:     {}\n'
            'asn:            {}'.format(
                row['domain'],
                row['ip'],
                row['reverse_lookup'],
                row['description'],
                row['registrant'],
                row['asn'])),
        'timestamp': row['date_utc'],
        'producer': {
            'description': 'Malware Domain List',
            'type': 'information-source',
        },
    }

    meta = {
        'manual_extracts': [
            {'kind': 'uri', 'value': row['domain'], 'level': 1},
            {'kind': 'ipv4', 'value': row['ip'], 'level': 1},
            {'kind': 'domain', 'value': row['reverse_lookup'], 'level': 1},
            {'kind': 'registrar', 'value': row['registrant'], 'level': 1},
            {'kind': 'asn', 'value': row['asn'], 'level': 1},
        ],
    }

    return {'data': data, 'meta': meta}
```

Package and deploy the enricher

Create a Python package for the extension you just built, and then use `pip install` to install it on the target system where the platform is running.

- Pack the extension to create a source distribution by running the following command(s):

```
$ python setup.py sdist
```

- Copy the packaged extension to the target location where you want to deploy it.
- Launch the platform Python virtual environment. To enable `venv`, run the following command(s):

```
$ source /opt/eclecticiq/platform/api/bin/activate
```

- In the virtual environment, install the extension by running the following command(s):

```
$ pip install /tmp/<your-custom-enricher-extension>.tar.gz
```

The `pip` example installs from `/tmp/` to avoid dealing with file access rights and permissions.

Restart processes

After completing the extension installation restart all *Supervisor* processes, so that all managed processes can configure the newly added extension in **Data configuration > Enrichers**.

- Reload Supervisor configurations and restart all Supervisor-managed processes by running the following command(s):

```
$ supervisorctl reload
```

Check that the enricher is registered

Make an API call with HTTPie

Verify that the extension is picked up and registered correctly.
To do so, save the following script to a *.sh* file, and then make it executable:

```
#!/bin/bash

#
# Helper for interacting with the platform API from the command line. This tool
# is a wrapper around httpie. It will take care of authentication and some
# other things. It takes at least three arguments:
#
# - host name
# - http method
# - relative url (without the /api/ part)
#
# Any additional arguments are propagated to httpie.
#
# Examples:
#
#   platform-api-http https://some.host/ GET /users/
#
#   platform-api-http localhost:8000 POST /some/endpoint/ @request-stored-in-a-file.json
#

set -e

readonly HTTPIE=http
readonly HTTPIE_ARGS="--check-status --verify=no"
readonly USERNAME=<valid_platform_signin_user_name>
readonly PASSWORD=<valid_platform_signin_password>

usage() {
    echo "Usage: $(basename $0) host method path [http-args]" > /dev/stderr
    exit 1
}

main () {
    local HOST="$1"
    local METHOD="$2"
    local API_PATH="$3"
    shift 3 || usage
    local TOKEN=$(${HTTPIE} ${HTTPIE_ARGS} POST "${HOST}/api/auth" username=${USERNAME}
password=${PASSWORD} | jq --raw-output '.token')
    local URL="${HOST}/api${API_PATH}"
    ${HTTPIE} ${HTTPIE_ARGS} ${METHOD} ${URL} Authorization:''"Bearer ${TOKEN}"'' "$@"
}

main "$@"
```

To make the script executable, run the following command(s):

```
$ chmod +x ~/<filename>.sh
```

The script takes the following input parameters:

Parameter	Description
`https://<platform_host>/`	*Required* — The name of the host used to reach the API endpoint and to communicate with the API service.
`POST`, `GET`, `PUT`, `DELETE`	*Required* — A valid **HTTP method** (`http://www.restapitutorial.com/lessons/httpmethods.html`) to create, read, update, or delete a resource.

Parameter	Description
`/<API_endpoint>/`	*Required* — A relative URL pointing to the API endpoint that exposes the service you want to consume.
`?url=true&query=search-or-filter&params=4`	*Optional* — URL query parameters to send any additional search parameters and/or to filter the results returned in the response.

Besides appending URL query parameters, you can also send your request parameters as a JSON file. Example:

```
$ platform-api-http https://platform.host/ get /entities/ @request-parameters.json
```

To make a **HTTPie** `(https://httpie.org/)` call using the script, use the following format:

```
$ platform-api-http https://<host> <method> <api_path>
```

To check if the newly created enricher extension is correctly registered in the platform, make an API call to the `/extensions/` API endpoint:

```
$ platform-api-http https://platform.host.com get /extensions/
```

API call response

The call returns a JSON object containing all registered extensions.
Search for your enricher extension by name, description, or creation date.
If your enricher extension is included in the returned list, it is registered correctly.

Enable the enricher

- In the returned JSON object listing all registered extensions, search for your extension.
- In the extension JSON object, look for the following fields:
  - `id`: its value is a progressive integer that uniquely identifies the extension.
  - `is_active`: Boolean, either `True` or `False`. This flags the extension as either enabled or disabled, respectively.
- If `is_active` is set to `False`, the extension is currently disabled, and you need to enable it before you can use it.
  To enable the extension, make the following API call:

```
$ platform-api-http https://{platform_host} put /extensions/{id_number} data:='{ "data" : {
"is_active" : true } }'
```

When you pass a JSON object with entity data in the body of your API request, you always need to wrap it in a data wrapper: `{ "data" : { ... } }`

- Enabled extension names should *not* be included in the `DISABLED_EXTENSIONS` list in the */opt/eclecticiq/etc/eclecticiq/platform_settings.py* configuration file.
  Any extensions on this list are automatically disabled.
  If an extension is on this list and you want to enable it, remove it from the list.

Initialize the enricher

The enricher extension is enabled, but not yet initialized. Platform enrichers though need to be initialized through fixtures before they become available.

Create and run the fixtures

- Log in to the system hosting the platform with either a user profile with admin rights, or with the `eclecticiq` user.
  You may need to grant the `eclecticiq` user admin privileges. If so, run the following command(s):
- Explicitly set the platform environment variable in the platform configuration file:

```
$ export EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
```

- Launch the platform Python virtual environment. To enable `venv`, run the following command(s):

```
$ source /opt/eclecticiq/platform/api/bin/activate
```

- Start a Python shell:

```
$ /opt/eclecticiq/platform/api/bin/eiq-platform shell
```

- In the Python shell, create the fixtures for the extensions by running the following command(s):

```
>>> from eclecticiq.extensions.boilerplate import create_fixtures
>>> create_fixtures()
```

Test the enricher

“Nah, my code doesn’t need testing.”
(anonymous, booted)

Run extension validation

The `eiq-platform extensions validate` command validates the state of all registered and enabled extensions.

- If the validation completes successfully, the command returns `Extensions look OK`.
- If the validation fails for one or more extensions, the command returns `Validation for {extension_name} failed:`, and it includes exception information for troubleshooting.

This command validates extensions based on the following criteria:

- The custom enricher extension name needs to match the `name` value defined in *setup.py*
- The extension name needs to be included in the platform `extension_registry` list, so that the platform can correctly recognize it and load it.
- The extension name should *not* be included in the `DISABLED_EXTENSIONS` list in the */opt/eclecticiq/etc/eclecticiq/platform_settings.py* configuration file.
- The extension needs to support the configuration parameters of any enricher tasks associated with it.

To run `eiq-platform extension validate`, do the following:

- Explicitly set the platform environment variable in the platform configuration file;
- Run the command:

```
$ export EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
$ /opt/eclecticiq/platform/api/bin/eiq-platform extensions validate
```

Error handling

If the validation detects issues that can prevent extensions from working correctly, it returns error messages with a short description of the problem.

Error message	**Description**
`Validation for {extension_name} failed:`	The enricher extension did not pass validation. Review the exception information included in the error message to start troubleshooting.
`Extension '{extension_name}' is in the list of disabled extensions.`	The extension name is included in the `DISABLED_EXTENSIONS` list in the */opt/eclecticiq/etc/eclecticiq/platform_settings.py* configuration file. The extension enricher is disabled. Enable it.
`Unknown extension '{extension_name}'`	The extension name is not included in `DISABLED_EXTENSIONS` list, or in the `extension_registry` list. Start troubleshooting by checking packaging and deployment , and that the enricher extension is registered.

Error message	Description
`Invalid parameters for enricher ({extension_name}). Errors: {returned_error_list}`	One or more parameters are invalid. For example, they may be assigned a wrong format, or the enricher extension parameter schema is not validated correctly.
`Task has parameters but enricher does not support any.`	The enricher extension does not support one or more parameters defined in the enricher extension parameter schema.

Test the enricher with a test file

You can test your code programmatically by creating a test file that provides a valid sample request and a valid sample response for the enricher extension you built. The ***test_socialurienricher.py*** `(https://github.com/eclecticiq/platform-extensions/blob/master/eclecticiq-extension-example/tests/test_socialurienricher.py)` file provides a boilerplate to build your customized enricher extension test file.

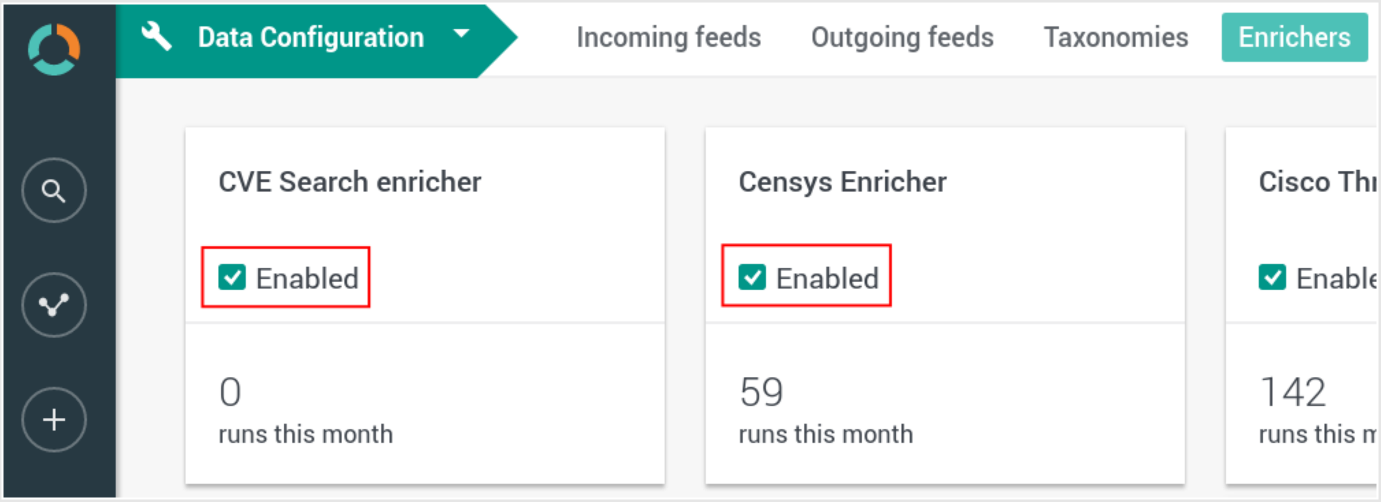
The test file uses **HTTPretty** `(https://httpretty.readthedocs.io/en/latest/index.html)` to mock HTTP responses, and it makes REST API testing easy and transparent.
Check the **HTTPretty GitHub repository** `(https://github.com/gabrielfalcao/httpretty)` for more details and usage examples.

Example:

```
# Import the libraries, modules and classes you need to test your extension
import httpretty

from <path.to.your.custom.enricher.extension> import (<CustomEnricherExtensionName>)

@httpretty.activate
# This example uses dummy names and values.
# Replace them with the appropriate ones for your extension.
# This function mocks a HTTP 200 response.
def test_socialurienricher_found():

    # Mock the API endpoint and any additional URL params
    # Mock the HTTP status code the response should return
    httpretty.register_uri(
        httpretty.HEAD, 'https://twitter.com/jhonny', status=200)

    httpretty.register_uri(
        httpretty.HEAD, 'https://facebook.com/jhonny', status=200)

    # Mock the observable types and values the response should return
    result = enrich_from_social_network(
        extract_type='name',
        extract_value='jhonny',

        # Mock any UI-configurable settings
        check_twitter=True,
        check_facebook=True,
    )

    # Verify that the response returns
    # the expected amount of observables
    # generated from the retrieved data
    assert len(result.extracts) == 2

    assert all(e['kind'] == 'uri' for e in result.extracts)

    values = [e['value'] for e in result.extracts]

    assert 'https://facebook.com/jhonny' in values
    assert 'https://twitter.com/jhonny' in values


@httpretty.activate
# This function mocks a HTTP 404 response.
def test_socialurienricher_not_found():
    httpretty.register_uri(
        httpretty.HEAD, 'https://twitter.com/jhonny', status=404)
    httpretty.register_uri(
        httpretty.HEAD, 'https://facebook.com/jhonny', status=404)

    result = enrich_from_social_network(
        extract_type='name',
        extract_value='jhonny',
        check_twitter=True,
        check_facebook=True,
    )

    assert len(result.extracts) == 0
```

Test the enricher through the platfom UI

- To check if your enricher extension is available in the platform UI, go to **Data configuration > Enrichers**.
- Your enricher extension should be displayed in the tiled overview, and the corresponding **Enabled** checkbox should be selected to notify that it is enabled.



CVE Search enricher

☑ Enabled

0
runs this month

Censys Enricher

☑ Enabled

59
runs this month

Cisco Thr

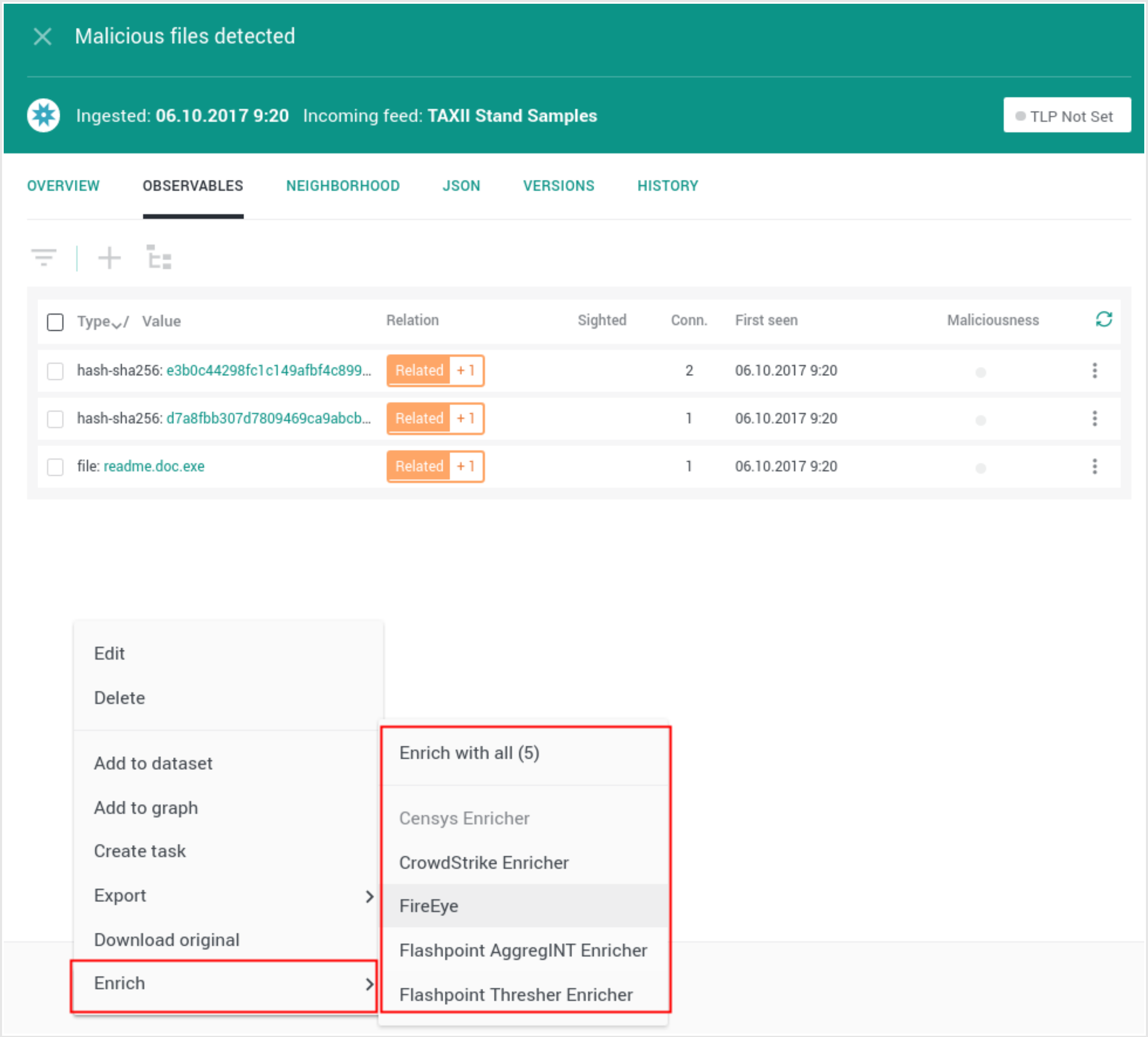
☑ Enable

142
runs this m

- To test if your enricher extension works as expected, look for an entity with observables that your enricher extension supports.
- Trigger a manual enrichment:
    - On the entity detail pane, click **Observables**.
    - The **Observables** tab shows an overview of the enrichment observables the entity has been augmented with.
    - To manually enrich the entity observables:
- Click the ↻ refresh icon to trigger a task run that polls all the enrichers configured for the entity.

Alternatively:

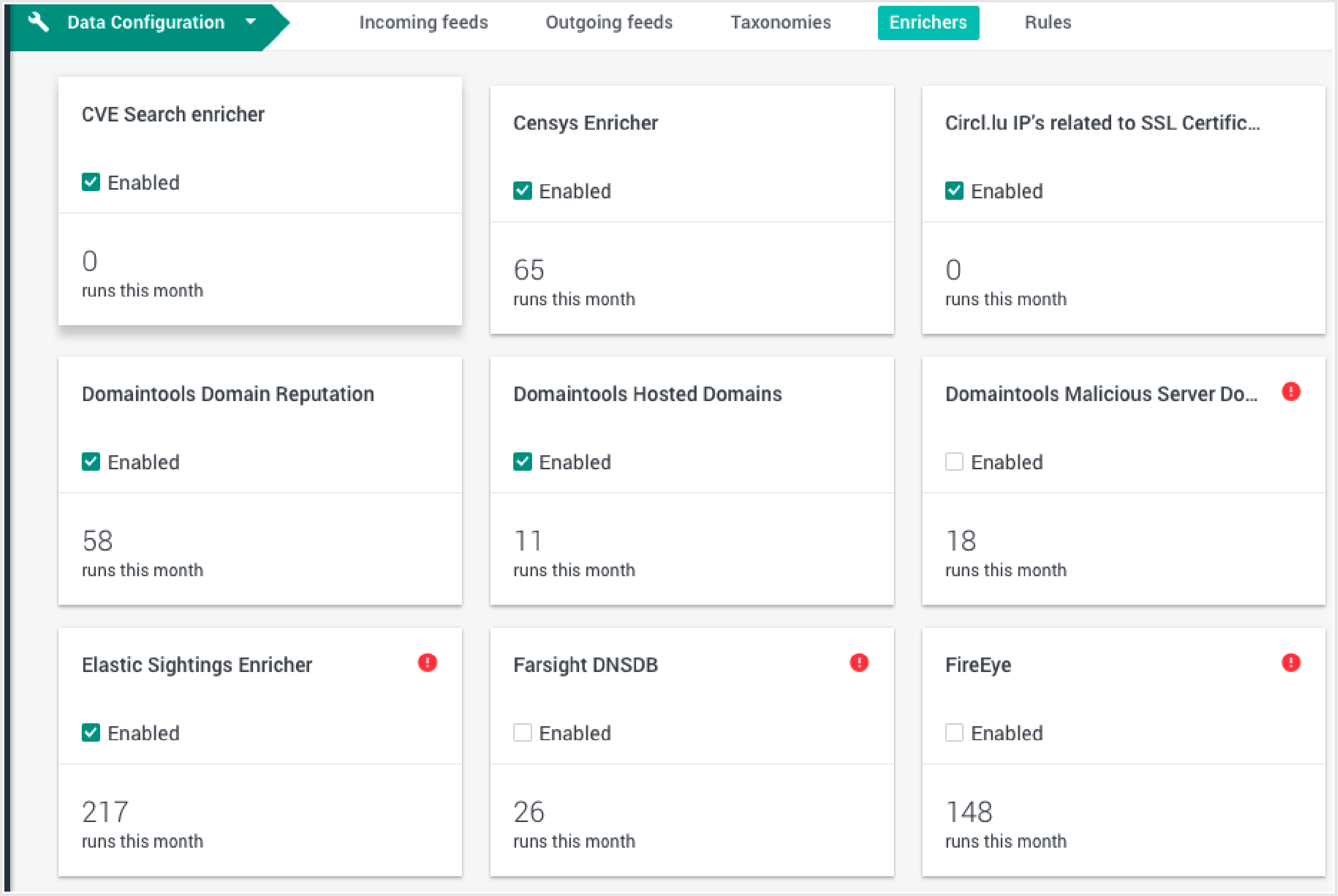
- From the **Actions** pop-up menu, select **Enrich > Enrich with all**.
- The platform polls all applicable enrichers for the entity, and it enriches all the entity observables with the retrieved data.
    - To poll a specific enricher:
- From the **Actions** pop-up menu, select **Enrich**, and then click the specific enricher whose task run you want to trigger.
- The platform polls the specified enricher for the entity, and it enriches all supported entity observables with the retrieved data.



☐	Type / Value	Relation	Sighted	Conn.	First seen	Maliciousness
☐	hash-sha256: e3b0c44298fc1c149afbf4c899...	Related +1		2	06.10.2017 9:20	
☐	hash-sha256: d7a8fbb307d7809469ca9abcb...	Related +1		1	06.10.2017 9:20	
☐	file: readme.doc.exe	Related +1		1	06.10.2017 9:20	

If you do not see any new observables after polling your enricher extension, check if the enricher crashed, and start investigating possible causes for the malfunction.

- In the platform UI, go to **Data configuration > Enrichers** .
  On the enricher catalog page you can see a tile overview of the configured enrichers for the platform.
- Look for your enricher extension. If a **(!)** icon is displayed, the enricher task failed to run correctly.



CVE Search enricher
☑ Enabled
0
runs this month

Censys Enricher
☑ Enabled
65
runs this month

Circl.lu IP's related to SSL Certific...
☑ Enabled
0
runs this month

Domaintools Domain Reputation
☑ Enabled
58
runs this month

Domaintools Hosted Domains
☑ Enabled
11
runs this month

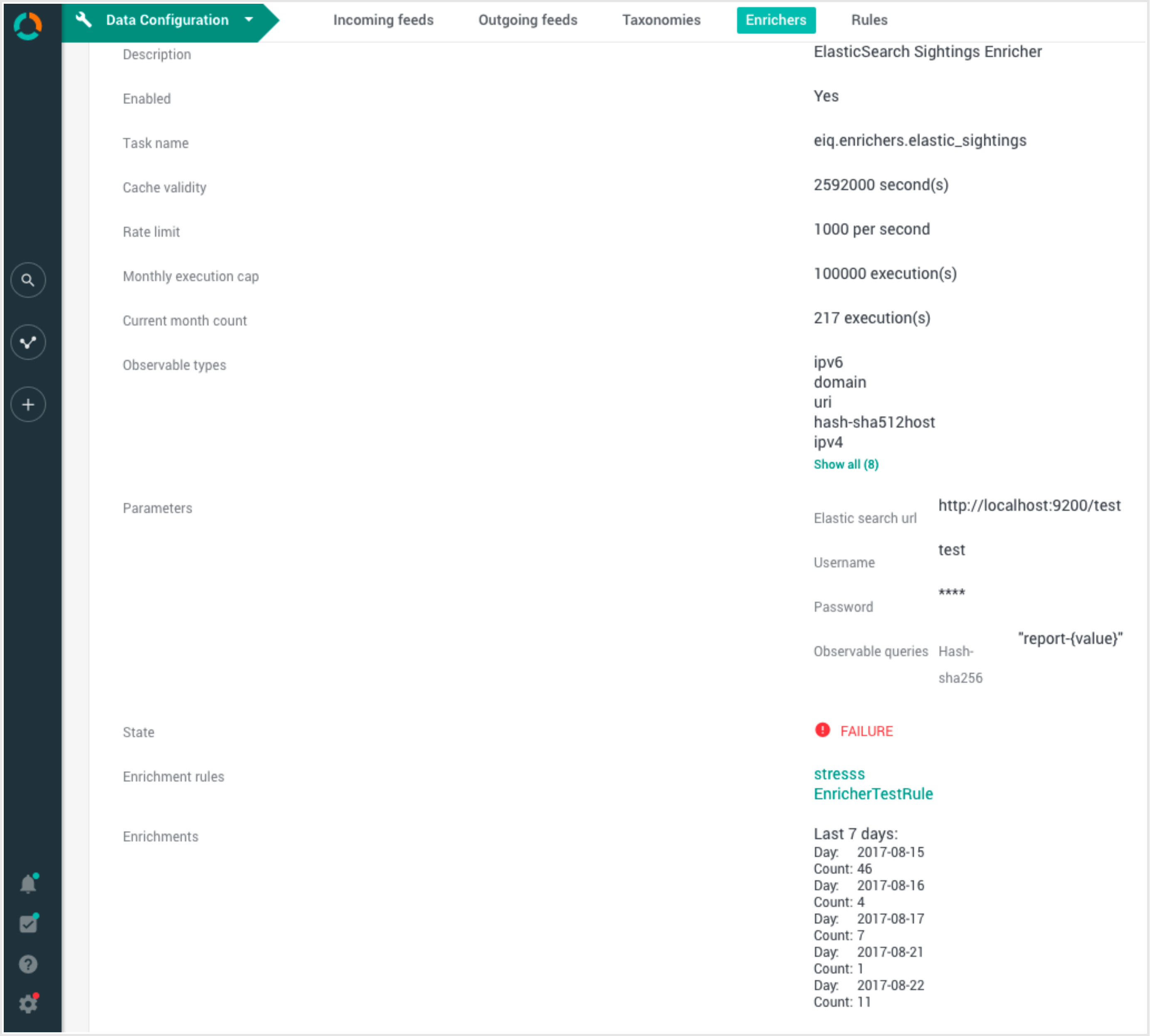
Domaintools Malicious Server Do...
☐ Enabled
18
runs this month

Elastic Sightings Enricher
☑ Enabled
217
runs this month

Farsight DNSDB
☐ Enabled
26
runs this month

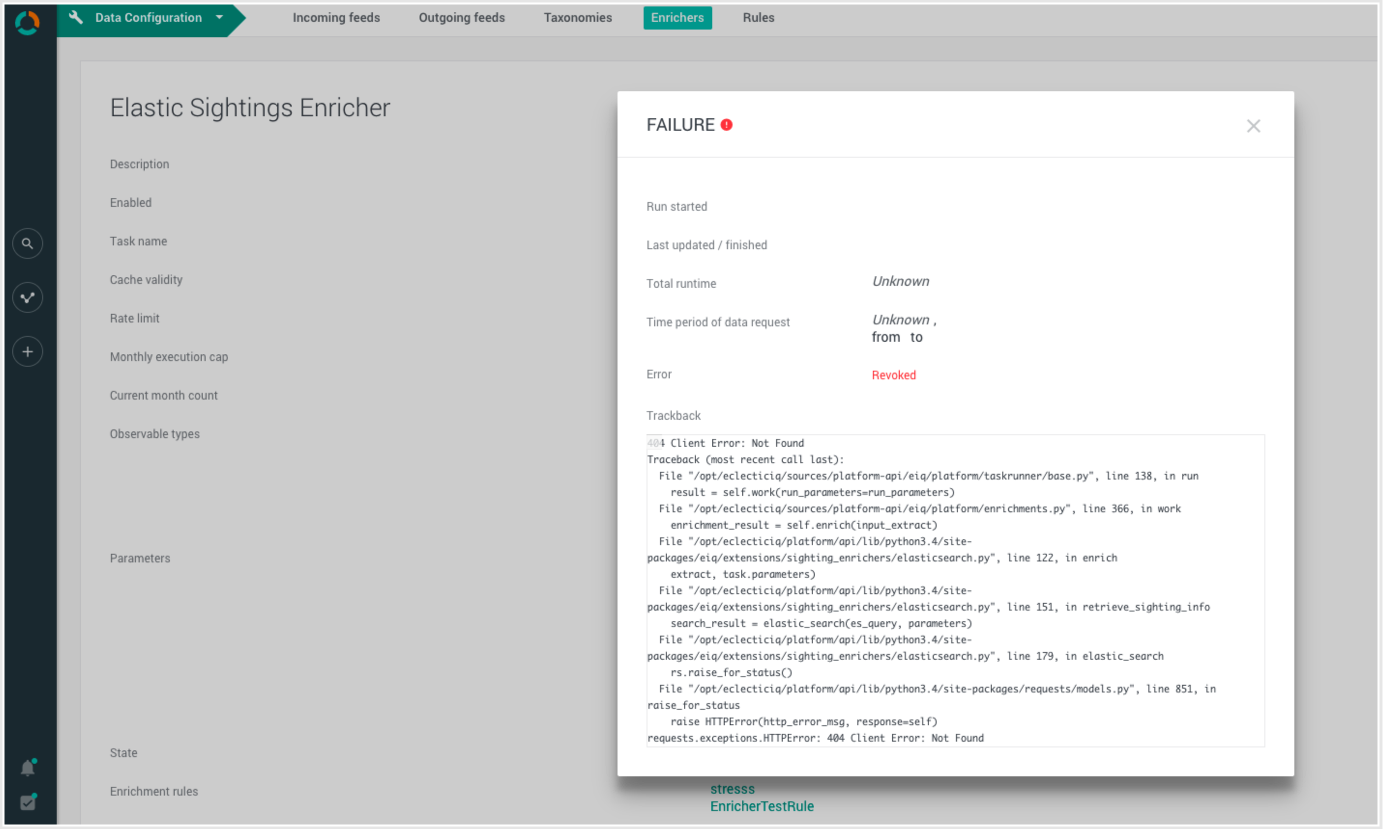
FireEye
☐ Enabled
148
runs this month

- Click the enricher tile.
- On the enricher detail page, click **(!) Failure**.



Description	ElasticSearch Sightings Enricher
Enabled	Yes
Task name	eiq.enrichers.elastic_sightings
Cache validity	2592000 second(s)
Rate limit	1000 per second
Monthly execution cap	100000 execution(s)
Current month count	217 execution(s)
Observable types	ipv6 domain uri hash-sha512host ipv4 Show all (8)
Parameters	Elastic search url: http://localhost:9200/test Username: test Password: **** Observable queries: Hash-sha256 "report-{value}"
State	FAILURE
Enrichment rules	stresss EnricherTestRule
Enrichments	Last 7 days: Day: 2017-08-15 Count: 46 Day: 2017-08-16 Count: 4 Day: 2017-08-17 Count: 7 Day: 2017-08-21 Count: 1 Day: 2017-08-22 Count: 11

- An error dialog is displayed. The dialog title notifies the type of error, whereas the traceback area gives a detailed stack trace in reverse chronological order. The stack trace should give you at least some hints about the possible causes of the failure.



Total runtime: Unknown

Time period of data request: Unknown ,
from to

Error: Revoked

Trackback

```
404 Client Error: Not Found
Traceback (most recent call last):
  File "/opt/eclecticiq/sources/platform-api/eiq/platform/taskrunner/base.py", line 138, in run
    result = self.work(run_parameters=run_parameters)
  File "/opt/eclecticiq/sources/platform-api/eiq/platform/enrichments.py", line 366, in work
    enrichment_result = self.enrich(input_extract)
  File "/opt/eclecticiq/platform/api/lib/python3.4/site-
packages/eiq/extensions/sighting_enrichers/elasticsearch.py", line 122, in enrich
    extract, task.parameters)
  File "/opt/eclecticiq/platform/api/lib/python3.4/site-
packages/eiq/extensions/sighting_enrichers/elasticsearch.py", line 151, in retrieve_sighting_info
    search_result = elastic_search(es_query, parameters)
  File "/opt/eclecticiq/platform/api/lib/python3.4/site-
packages/eiq/extensions/sighting_enrichers/elasticsearch.py", line 179, in elastic_search
    rs.raise_for_status()
  File "/opt/eclecticiq/platform/api/lib/python3.4/site-packages/requests/models.py", line 851, in
raise_for_status
    raise HTTPError(http_error_msg, response=self)
requests.exceptions.HTTPError: 404 Client Error: Not Found
```

Disable extensions

To disable an enricher extension and prevent the platform from loading it, do the following:

- Open */opt/eclecticiq/etc/eclecticiq/platform_settings.py* in read-write mode.
- Browse to the `DISABLED_EXTENSIONS` parameter.
  If it is not there, add it to the settings file.
  This parameter is a list holding the `name` values of all disabled extensions, as defined in the setup file.
- Add your enricher extension `name` value to the list.
  Example:

```
DISABLED_EXTENSIONS =  [
    'eclecticiq-extension-fraud-ip-observables',
    ...,
]
```

- Save the settings file and exit.

Enrichers reference

Reference section with lookup information on the default enrichers, enricher inputs and outputs, and a list with the supported enrichment data types. Enrichers augment existing cyber threat intelligence value by adding contextual information.

Enrichers

Enrichers poll external data sources to provide additional context and detail to augment — hence, enrich — the intelligence value of entities and observables stored in the platform.

The platform ships with a broad selection of built-in, ready-to-use enrichers to obtain geolocation IP and whois details, DNS domain and malware information, as well as other relevant data to help analysts draw a sharper and more comprehensive picture of the cyber threat relationships and the cyber threat scenarios under investigation.

Enrichers automatically augment all the entities that accept the enricher's content type as an observable. In other words, the observable types an entity supports define the applicable enrichers an entity can use.

Default enrichers

Title	Excerpt
Censys enricher	The Censys enricher returns a wealth of information about IP addreses, from a network ASN to their geographic location, so that you can explore relationships between events, actors, and targets.
CIRCL IPs related to SSL certificate enricher	The CIRCL IPs related to SSL certificate enricher uses the CIRCL API to poll the CIRCL Passive SSL database and to obtain all IP addreses associated with the input SSL SHA-1 hash fingerprints.
CIRCL SSL Certificate Fetcher enricher	The CIRCL SSL Certificate Fetcher enricher uses the CIRCL API to poll the CIRCL Passive SSL database and to obtain parsed SSL certificates and all domain names associated with the input SSL SHA-1 hash fingerprints.
Cisco Threat Grid enricher	The Cisco Threat Grid enricher provides information on a range of cyber threat data like IP addresses, domains, registry keys, network streams, and hash files.
Crowdstrike Falcon Intelligence Indicator enricher	The Crowdstrike Falcon Intelligence Indicator enricher returns observables extracted from indicators to provide additional context to existing platform intelligence.
CVE Search enricher	The CVE Search enricher contacts CIRCL cve-search API to poll information about common software and hardware vulnerabilities, along with the corresponding exposures.
DomainTools Hosted Domains enricher	The Domaintools Hosted Domains enricher returns all domain names related to the specified input IP addresses.

Title	Excerpt
DomainTools Malicious Server Domains enricher	The DomainTools Malicious Server Domains enricher returns malicious domain names related to the same primary and/or secondary name servers, along with their risk scores to automatically flag server domains with an appropriate maliciousness confidence level.
DomainTools Parsed Whois enricher	The DomainTools Parsed Whois enricher returns malicious domain names related to the same primary and/or secondary name servers.
DomainTools Reputation enricher	The Domaintools Reputation enricher returns risk scores to assess the reputation of the specified input domain and host names.
DomainTools Reverse Whois enricher	The DomainTools Reverse Whois enricher returns malicious domain names related to the name, address, telephone number, email address or physical address of the registrant listed in current or historical whois records.
DomainTools Suspicious Domains enricher	The DomainTools Suspicious Domains enricher returns suspicious and potentially malicious domains related to the input IP addesses, along with their risk scores to automatically flag domains with an appropriate maliciousness confidence level.
Elasticsearch sightings enricher	The Elasticsearch sightings enricher creates sightings from matching results returned from a search in an external Elasticsearch instance.
Create custom enrichers	Implement custom extensions to integrate EclecticIQ Platform with external intel providers and data sources through incoming feeds and enrichers, as well as to publish platform intel downstream in your prevention and detection toolchain.
Farsight DNSDB enricher	The Farsight DNSDB enricher provides historical passive DNS information to relate domain names to the IP addreses they point to, or IPs pointing to different domains over time.
FireEye iSIGHT enricher	The FireEye iSIGHT enricher retrieves threat intelligence reports about threats and malware related to areas such as critical infrastructure, cyber crime and espionage, hacktivism, frauds, and vulnerability and exploitation.
Flashpoint AggregINT enricher	The Flashpoint AggregINT enricher provides information on a range of cyber threat data like actor information, email and IP addresses, domains, and hash files. It focuses on threat actors and criminal groups.
Flashpoint Blueprint enricher	The Flashpoint Blueprint enricher provides information on a range of cyber threat data like actor information, email and IP addresses, domains, and hash files. It focuses on threat actors and criminal groups.
Flashpoint Thresher enricher	The Flashpoint Thresher enricher provides information on a range of cyber threat data like actor information, email and IP addresses, domains, and hash files. It focuses on threat actors and criminal groups.
Fox-IT InTELL Portal enricher	The Fox-IT InTELL Portal enricher provides information from a range of sources, such as forums and sites that have registered potentially suspicious activity.
Intel 471 enricher	The Intel 471 enricher provides information on compromised IP addresses, domains, URLs, and emails, as well as first-hand information about threat actors and groups.
OpenDNS OpenResolve enricher	The Cisco OpenDNS OpenResolve enricher provides reverse-DNS lookup information.

Title	Excerpt
PassiveTotal Passive DNS enricher	The PassiveTotal Passive DNS enricher provides historical information to enable cross-referencing IP addresses to the corresponding DNS domain names over time.
PassiveTotal IP/Domain enricher	The PassiveTotal IP/Domain enricher provides additional information for the queried IP address or domain name.
PassiveTotal Malware enricher	The PassiveTotal Malware enricher provides malware information related to the queried host or domain, such as malware hash and collection date.
PassiveTotal Whois enricher	The PassiveTotal Whois enricher provides information about individuals or entities associated with an IP address or a domain name, as well as geolocation details.
PyDat enricher	The PyDat enricher provides whois, including historical whois, and passive DNS lookup information.
Recorded Future enricher	The Recorded Future enricher enables you to tap into the data stream generated by the Recorded Future Temporal Analytics Engine to retrieve search results potentially malicious IPs, domains, email addresses, and hashes related to the input observable types, along with their risk scores to automatically flag domains ...
RIPEstat GeoIP enricher	The RIPEstat GeoIP enricher provides geolocation information related to the queried IP addresses.
RIPEstat Whois enricher	The RIPEstat Whois enricher provides whois information related to the queried IP addresses.
Shodan enricher	The Shodan enricher uses input data such as country and city names, organization and personal names, ZIP codes, email addresses, and so on to return a list of matching IP addresses corresponding to your Internet-connected devices, along with location and user details.
Splunk sightings enricher	The Splunk sightings enricher searches the indexes in the specified Splunk instance. Matching data is extracted and saved to the platform as sightings.
SpyCloud Breach Data enricher	The SpyCloud Breach Data enricher uses input data such as IP addresses, domain names, email addresses and user access credentials to return incident and security breach information that is processed as incident entities, related observables, and enrichment observables.
ThreatCrowd enricher	The ThreatCrowd enricher returns suspicious and potentially malicious domains, IP addreses, email addresses, file hashes, and antivirus detections, so that you can explore relationships between events, actors, and targets.
Unshorten-URL enricher	The Unshorten-URL polls the specified URL shortener services to return the resolved original URLs corresponding to the submitted shortened ones.
VirusTotal enricher	The VirusTotal enricher provides information on malware, domains (passive DNS) and IP addresses.

Enricher types

Enricher	API endpoint	De
Elasticsearch sightings	`http://<elasticsearch_instance_url>:9200/<schema_resource>`	Searches an external Elasticsear search criteria are processed to a sightings.
Fox-IT InTELL Portal	`https://cybercrime-portal.fox-it.com/`	Based on Fox-IT InTELL, the port a range of sources, such as forum potentially suspicious activity.
Intel 471	`https://api.intel471.com/v1/`	Besides data on compromised IP Intel 471 focuses on providing firs and groups.
OpenDNS OpenResolve	`http://api.openresolve.com/{}/{}`	OpenResolve by OpenDNS offers to retrieve reverse-DNS lookup in
PyDat	`http://<pydat_instance_url>:8000/`	**PyDat** `(https://github.com/mi` locally, and it can work together w `(https://github.com/mitrecnc` `with-elasticsearch)` to provide passive DNS lookup information. organization, country, city, street,
RIPEstat GeoIP	`https://stat.ripe.net/data/geoloc/{}`	Geolocation IP information from th **API** `(https://stat.ripe.net/` longitude, country, and city.
RIPEstat Whois	`https://stat.ripe.net/data/whois/{}`	Whois information from the RIPEs **API** `(https://github.com/ripe` including inet number, name, orga telephone.
Cisco Threat Grid	`https://panacea.threatgrid.com/api/v2/`	Polls data from the Cisco Threat range of cyber threat data like IP network streams, and hash files.
VirusTotal	`https://www.virustotal.com/vtapi/v2/{}`	Polls data from the VirusTotal API domains (passive DNS) and IP ad against 60+ antimalware products additional metadata information,
Flashpoint AggregINT	`https://endlesstunnel.info/v3`	Polls data from the Flashpoint AP hosts, domains, IP addresses, an thematic datasets focusing on ha groups, communities in conflict, st **CBRN** `(https://en.wikipedia` produces enrichment observables user name of the author of a post UTC date and time of a post in **IS** `(https://en.wikipedia.org/wi` `(https://tools.ietf.org/html`

Enricher	API endpoint	De
Flashpoint Blueprint	`https://endlesstunnel.info/v3`	Polls data from the Flashpoint AP geolocation and IP ranges, as wel search thematic datasets focusing supremacist groups, state actors i (`https://en.wikipedia.org/wi` enrichment observables like city/c latitude/longitude or IP address hi a hit, user name uniquely matched
Flashpoint Thresher	`https://endlesstunnel.info/v3`	Polls data from the Flashpoint AP datasets focusing on hackers, terr and **CBRN** (`https://en.wikipe` threats. It produces enrichment ob thresher data.
PassiveTotal Whois	`https://api.passivetotal.org/v2`	Polls data from the **PassiveTotal** (`https://api.passivetotal.or` `getv2whoisquery`). It provides in associated with an IP address or a details. Analysts can retrieve regis telephone, and email details. They further queries to obtain, for exam with the same individual or the sa
PassiveTotal Passive DNS	`https://api.passivetotal.org/v2`	Polls data from the **PassiveTotal** (`https://api.passivetotal.or` `getv2dnspassivequery`). It provi cross-referencing IP addresses to over time. Analysts can examine h different IP addresses over time. T retrieve more domain names that
PassiveTotal IP/Domain	`https://api.passivetotal.org/v2`	Polls data from the **PassiveTotal** (`https://api.passivetotal.or` `getv2enrichmentquery`). It provi queried IP address or domain nan name, any sub-domains, inet deta **(ASN)** (`https://en.wikipedia.org/wi` as well as geolocation information to look for further connections tha investigation.

Enricher	API endpoint	De
PassiveTotal Malware	`https://api.passivetotal.org/v2`	Polls data from the **PassiveTotal** `(https://api.passivetotal.or` `getv2enrichmentmalwarequery)` to the queried host or domain, suc sha1, hash-sha256, hash-sha512 malware entries are also tagged w `enrichment_extracts.meta.cla` to the value you set under **Data c** **✚ > Action > Mark as malicious** `enrichment_extracts.meta.cor` value you set under **Data configu** **Confidence > Malicious - Low c**
Splunk sightings	`http://<splunk_instance_url>:8089/`	Based on the search queries defir for matching data in the specified extracted and saved to the platfor
DomainTools Hosted Domains	`http://api.domaintools.com/v1/{}/host-domains`	Enriches IPv4 observables by retu and therefore related to, the input
DomainTools Reputation	`http://api.domaintools.com/v1/reputation`	Enriches domain and host name c information to assess maliciousne defined threshold values.
DomainTools Suspicious Domains	`https://api.domaintools.com/v1/{}/host-domains`	Enriches IPv4 observables with su IP addresses. It includes configura confidence levels to the processe malicious IPs.
FireEye iSIGHT	`https://api.isightpartners.com/search/{}`	Enriches platform observables wit threatwww.threatcrowd.org/s and critical infrastructure, cyber crime vulnerability and exploitation.
Recorded Future	`https://app.recordedfuture.com/live/sc/entity/{}`	The enricher returns additional da addresses, and hashes related to specified types, as well as malicio retrieved risk scores.
Unshorten-URL	`https://unshorten.me/s/{}`	It takes shortened URL as an inpu resolved original URLs, which can discover relationships with other e
Farsight DNSDB	`https://api.dnsdb.info/{}`	Historical passive DNS lookup enr pointing to a specified IP address nameserver, domain names pointi existing below a parent domain na
ThreatCrowd	`https://www.threatcrowd.org/{}`	Returns suspicious and potentially addresses, file hashes, and antivir relationships between events, acto

Enricher	API endpoint	De
Censys	`https://censys.io/api/v1/search/ipv4`	Returns relevant contextual inform types to augment their intelligence details, hashes, and **ASN** `(https://en.wikipedia.org/wi` details. It makes it easier to disco actors, and targets.
DomainTools Malicious Server Domains	`http://api.domaintools.com/v1/{}/name-server-domains/`	Enriches domain and host observa domain names related to the sam includes configurable thresholds t levels to the processed domains a domains/hosts.
DomainTools Parsed Whois	`http://api.domaintools.com/v1/{}/whois/parsed`	Enriches domains, hosts, and IP a JSON output includes the most re domain or IP range, as well as pa registrar, contacts, and so on. It h referencing data in a set of Whois
Crowdstrike Falcon Intelligence Indicator	`https://intelapi.crowdstrike.com/indicator/v1/search/{}`	Enriches platform entities and obs IP addresses, domain names, em
DomainTools Reverse Whois	`http://api.domaintools.com/v1/reverse-whois/{}`	Enriches supported observable ty current or historical Whois records actors.
CVE Search	`https://cve.circl.lu/api/cve/`	Enriches supported observable ty software and hardware vulnerabili exposures. The enricher contacts Center Luxembourg (CIRCL) cve- details associated with the input C common software and hardware v corresponding exposures, is store observables.
CIRCL IPs related to SSL certificate	`https://www.circl.lu/v2pssl/cquery/{}`	The enricher contacts the Comput Luxembourg (CIRCL) API and the retrieve all observed IP addresses fingerprint *hash-sha1* observabl holds historical data matching SS
CIRCL SSL Certificate Fetcher	`https://www.circl.lu/v2pssl/cfetch/{}`	The enricher contacts the Comput Luxembourg (CIRCL) API and the retrieve the parsed certificate and with the input SSL fingerprint *has Passive SSL database holds histo fingerprints with the correspondin domain names, that is, the certific \`subjectAltName\`/*Subject Altern

Enricher	API endpoint	De
Shodan	`https://api.shodan.io/shodan/`	Shodan matches input data such company or personal names, to re discovering which devices in your to the Internet, where they are loc
SpyCloud Breach Data	`https://api.spycloud.io/sp-v1/breach`	Enriches supported observable ty information about account takeove description of the incident, targete of stolen records — for example, credentials — as well as when the sale on the Internet. Each entry in entry of a breached database con password, and email address) —

Enricher input

The overview shows the supported observable data types you can use as input for the enrichers. These are the value types the *enrichment_extracts.kind* search query field returns.

Enricher	Supported kinds (observable types)
Elasticsearch sightings	domain, hash-md5, hash-sha1, hash-sha256, hash-sha512, host, ipv4, ipv6, uri
Fox-IT InTELL Portal	domain, hash-md5, hash-sha1, hash-sha256, host, ipv4, ipv6, uri
Intel 471	actor-id, domain, email, hash-md5, hash-sha256, host, ipv4, ipv6, uri
OpenDNS OpenResolve	domain, host, ipv4, ipv6
PyDat	domain, ipv4, ipv6
RIPEstat GeoIP	ipv4, ipv6
RIPEstat Whois	ipv4, ipv6
Cisco Threat Grid	domain, hash-md5, hash-sha1, hash-sha256, hash-sha512, host, ipv4, ipv6, uri, winregistry
VirusTotal	domain, hash-md5, hash-sha1, hash-sha256, ipv4, ipv6, uri
Flashpoint AggregINT	actor-id, domain, email, hash-md5, hash-sha1, hash-sha256, hash-sha512, host, ipv4, ipv6, uri
Flashpoint Blueprint	actor-id, domain, email, hash-md5, hash-sha1, hash-sha256, hash-sha512, host, ipv4, ipv6, uri
Flashpoint Thresher	domain, file, hash-sha1, host, ipv4, uri
PassiveTotal Whois	domain, host, ipv4, ipv6
PassiveTotal Passive DNS	domain, host, ipv4, ipv6

Enricher	Supported kinds (observable types)
PassiveTotal IP/Domain	domain, host, ipv4, ipv6
PassiveTotal Malware	domain, host
Splunk sightings	domain, email, hash-md5, hash-sha1, hash-sha256, hash-sha512, host, ipv4, ipv6, uri
DomainTools Hosted Domains	ipv4
DomainTools Reputation	domain, host
DomainTools Suspicious Domains	ipv4
FireEye iSIGHT	asn, domain, email, email-subject, file, hash-md5, hash-sha1, hash-sha256, host, ipv4, ipv6, uri
Recorded Future	domain, hash-md5, hash-sha1, hash-sha256, hash-sha256, ipv4, ipv6
Unshorten-URL	uri
Farsight DNSDB	domain, host, ipv4, ipv6
ThreatCrowd	domain, email, hash-md5, hash-sha1, hash-sha256, hash-sha512, host, ipv4, ipv6, malware
Censys	asn, city, company, country, country_code, geo-lat, geo-long, hash-md5, hash-sha1, hash-sha256, ipv4, postcode
DomainTools Malicious Server Domains	domain, host
DomainTools Parsed Whois	domain, host, ipv4
Crowdstrike Falcon Intelligence Indicator	domain, email, email-subject, file, hash-md5, hash-sha1, hash-sha256, ipv4, ipv6, mutex, name, persona, port, uri
DomainTools Reverse Whois	address, asn, city, company, country, country-code, email, name, organization, person, postcode, street, telephone
CVE Search	cve
CIRCL IPs related to SSL certificate	hash-sha1
CIRCL SSL Certificate Fetcher	hash-sha1
Shodan	asn, city, company, country, country-code, domain, email, geo-lat, geo-long, hash-md5, hash-sha1, hash-sha256, hash-sha512, host, ipv4, ipv6, organization, person, port, postcode, uri
SpyCloud Breach Data	domain, email, handle, ipv4, ipv6

Enricher output

Enrichers automatically augment all the entities that accept the enricher’s content type as an observable. In other words, the observable types an entity supports define the applicable enrichers an entity can use.

The overview describes the output each enricher generates. The resulting enrichment observables are associated with the entities they bear relationships to.

Enricher	Generated output
Elasticsearch sightings	Creates sightings from matching results returned from a search in an external Elasticsearch instance.
Fox-IT InTELL Portal	Enriches supported observable types with relevant contextual information from forums, chats, and IRC channels.
Intel 471	Enriches supported observable types with data focusing on threat actor information.
OpenDNS OpenResolve	Enriches supported observable types with reverse-DNS lookup information.
PyDat	Enriches supported observable types with whois data, current IP resolution and passive DNS information.
RIPEstat GeoIP	Enriches supported observable types with geolocation information related to IP addresses: coordinates, country, and city.
RIPEstat Whois	Enriches supported observable types with whois information related to IP addresses.
Cisco Threat Grid	Enriches supported observable types, as well as all found observables based on the enricher configuration, with information such as IP addresses, domains, host names, hashes, and Windows registry keys.
VirusTotal	Enriches supported observable types with maliciousness confidence level information.
Flashpoint AggregINT	Enriches supported observable types with information such as IP addresses, domains, host names, and hash files.
Flashpoint Blueprint	Enriches supported observable types with information such as IP addresses, domains, host names, and URLs.
Flashpoint Thresher	Enriches supported observable types with information such as IP addresses, domains, URLs, hashes, and files.
PassiveTotal Whois	Enriches supported observable types with **whois** `(https://www.riskiq.com/products/learn-threat-research-and-analysis/)` information.
PassiveTotal Passive DNS	Enriches supported observable types with **passive DNS** `(https://www.riskiq.com/products/learn-threat-research-and-analysis/)` information.
PassiveTotal IP/Domain	Enriches supported observable types with **enrichment** `(https://passivetotal.readthedocs.io/en/latest/api.html#enrichment-request)` information.

Enricher	Generated output
PassiveTotal Malware	Enriches supported observable types with **malware enrichment** `(https://passivetotal.readthedocs.io/en/latest/api.html?highlight=malware#enrichment-request)` information.
Splunk sightings	Creates sightings for matching input observables, based on the search result items retrieved in the specified Splunk instance.
DomainTools Hosted Domains	Enriches supported observable types with domain and host name information.
DomainTools Reputation	Enriches supported observable types with reputation information.
DomainTools Suspicious Domains	Enriches supported observable types with suspicious domain and host name information.
FireEye iSIGHT	Enriches supported observable types related to the matching input observable types.
Recorded Future	Enriches supported observable types with pattern matching search results produced by the Recorded Future Temporal Analytics Engine.
Unshorten-URL	Original URL corresponding to the submitted shortened one.
Farsight DNSDB	Enriches supported observable types with passive DNS lookup information such as the name of the domain or host name owner, or the IP address a domain or host name points to.
ThreatCrowd	Enriches supported observable types with suspicious and potentially malicious domains, IP addreses, email addresses, file hashes, and antivirus detections.
Censys	Enriches supported observable types by providing additional context such as geolocation, country and city information, as well as **ASN** `(https://en.wikipedia.org/wiki/autonomous_system_(internet))` details.
DomainTools Malicious Server Domains	Enriches supported observable types with malicious domain names that are served from the same name server.
DomainTools Parsed Whois	Enriches supported observable types with structured Whois information.
Crowdstrike Falcon Intelligence Indicator	Enriches supported observable types with information extracted from indicators.
DomainTools Reverse Whois	Enriches supported observable types with reverse Whois information.
CVE Search	Enriches supported observable types with CVE details.
CIRCL IPs related to SSL certificate	Enriches SSL fingerprint hash observables with all associated IP addresses.
CIRCL SSL Certificate Fetcher	Enriches SSL fingerprint hash observables with the parsed certificate and all associated domain names.
Shodan	Enriches supported observable types with the following information, when available: country name, city name, ZIP code, longitute, latitude, organization name, host name, IP address, open ports and services related to input IP addresses.

Enricher	Generated output
SpyCloud Breach Data	Enriches supported observable types and incident entities with account takeover (ATO) and security breach details. Generates Cybox observables, related observables, and CIQ objects. Supported output observables: *city*, *country-code*, *domain*, *email*, *geo-lat*, *geo-long*, *handle*, *host*, *ipv4*, *ipv6*, *organization*, *person*, *postcode*, *telephone*.

Enrichment data types

actor-id
address
asn
bank-account
card
card-owner
cce
city
company
country
country-code
cve
cwe
domain
email
email-subject
eui-64
file
forum-name
forum-room
forum-thread
fox-it-portal-uri
geo
geo-lat

geo-long
handle
hash-md5
hash-sha1
hash-sha256
hash-sha512
host
industry
inetnum
ipv4
ipv6
mac-48
malware
mutex
name
nationality
netname
organization
person
port
postcode
process
product
registrar
rule
snort
street
telephone
uri
uri-hash-sha256
winregistry

yara