OpenCTI data sharing

medium.com/luatix/opencti-data-sharing-6da7dc045d14

Julien Richard October 29, 2021

Julien Richard

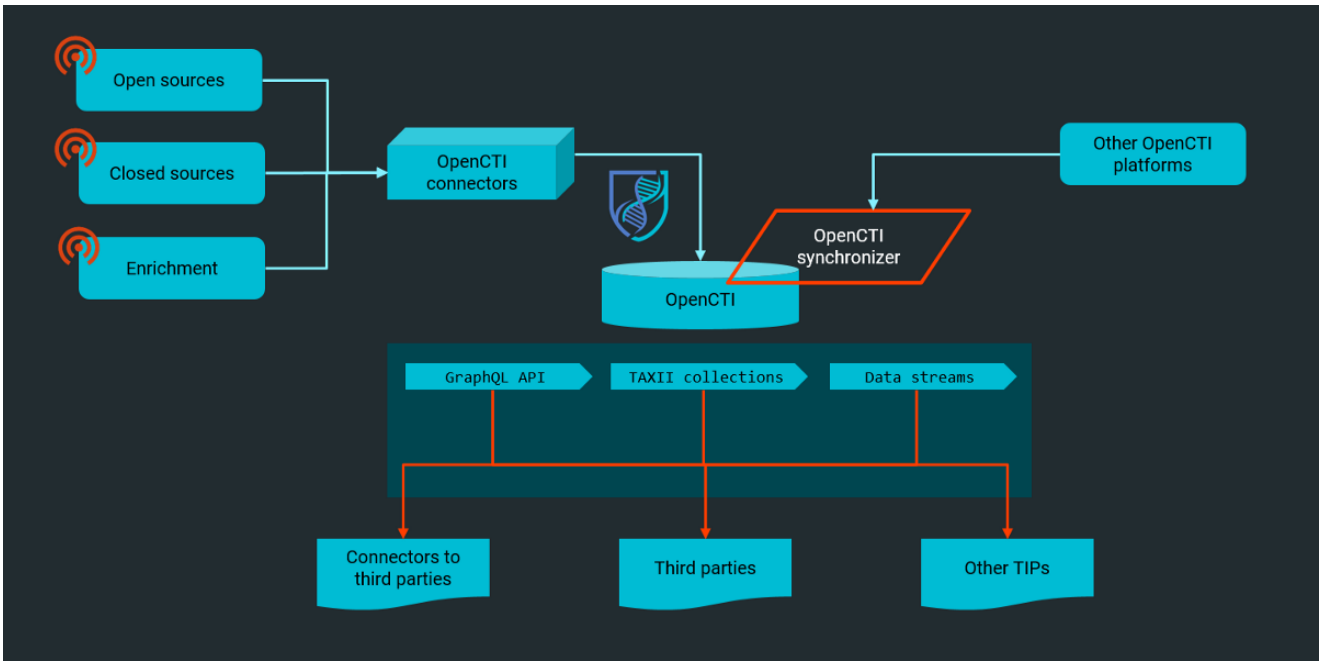
Oct 29, 2021

.

10 min read

In this blog post, we would like to explain a little bit how users and developers are able to extract data from OpenCTI whether using API, TAXII™ collections or data streams.

As you might know, data sharing in the cybersecurity field matters a lot (and this is not the only one 😀) and we think this is an area where OpenCTI should bring great added value to the community.



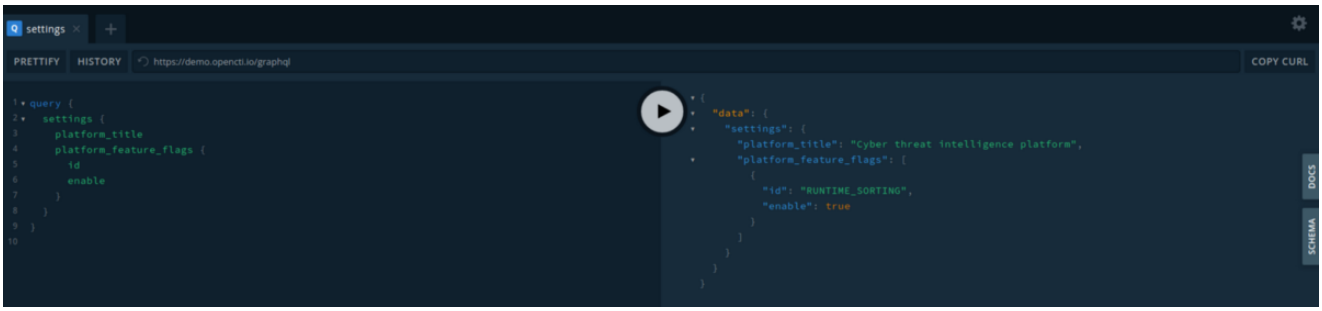
Major data sharing features in OpenCTI

OpenCTI has a lot of capabilities to extract the information from the database to feed other systems. We will give you more knowledge about the different ways to do it, with a specific focus on streams capabilities.

GraphQL API

The first way to extract data from OpenCTI is by using the GraphQL API (/graphql). If you choose to directly consume this API, you will have to manipulate filtering, ordering and pagination API arguments to get the latest data periodically. This can lead to some delay in refreshing your data.

If you would like to discover the API endpoints and capabilities, the best path is to explore the embedded GraphQL playground in the platform.

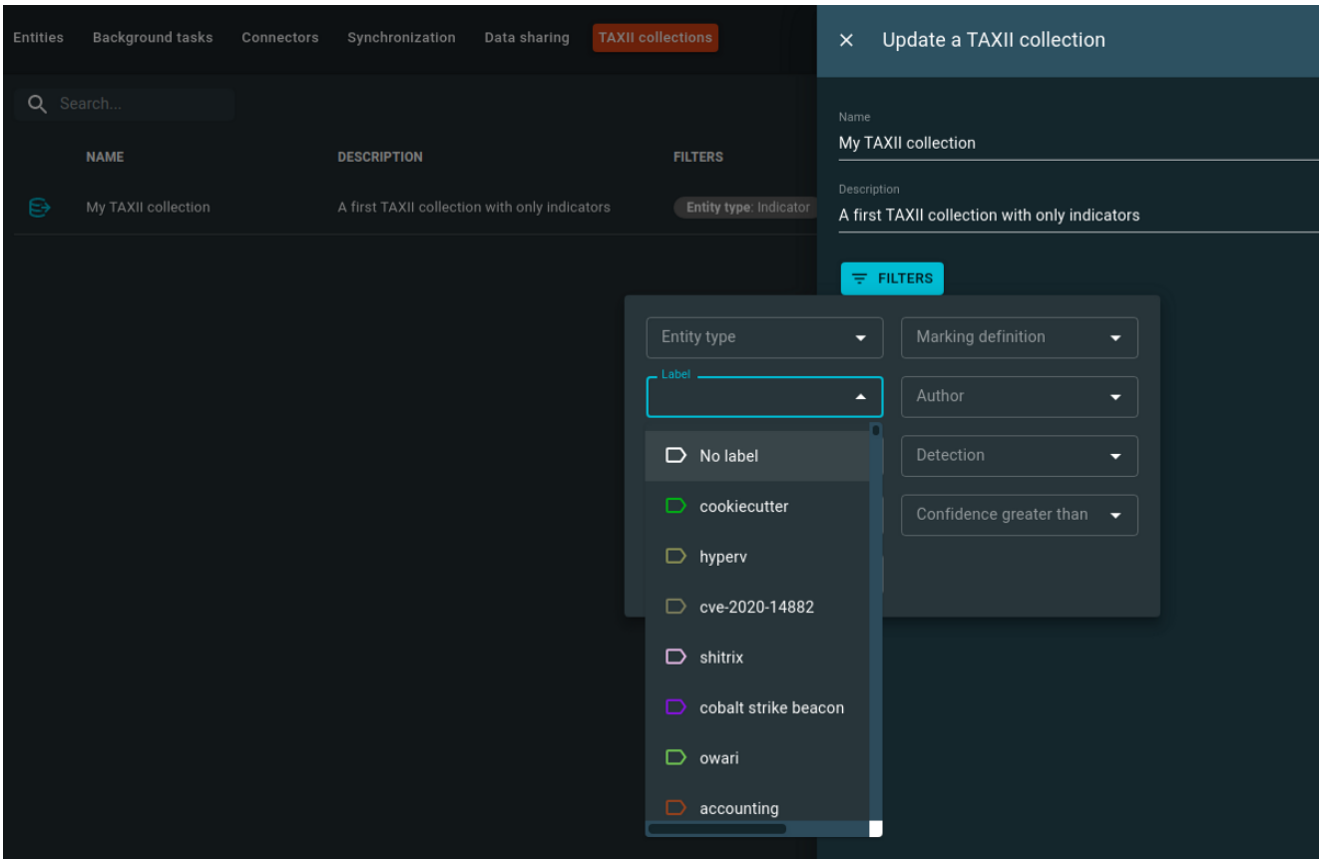


GraphQL playground

TAXII™ Collections

OpenCTI has an embedded TAXII™ API endpoint which provides valid STIX 2.1 bundles. If you wish to know more about the TAXII™ standard, please read the official introduction.

In OpenCTI you can create as many TAXII 2.1 collections as needed. Each of them can have specific filters to publish only a subset of the platform overall knowledge (specific types of entities, labels, marking definitions, etc.).



An example of OpenCTI TAXII 2.1 collection
After creating a new collection, every systems with a proper access token can consume the collection using different kind of authentication (basic, bearer, etc.)

As when using the GraphQL, TAXII 2.1 collections have a classic pagination system that should be handled by the consumer. Also, it's important to understand that element dependencies (nested IDs) inside the collection are not always contained/resolved in the bundle (this may lead to have a thousands of entities just for one), so consistency needs to be handled at the client level.

The lack of real time update and the dependencies issue are the main drivers that led us to create what we call in the platform the "streams (live or raw)".

Streams

OpenCTI provides 2 different kinds of stream which aim to provide various usages. These streams are both available through the API directly, using the Server-sent events protocol (to learn more about SSE, you can check the Wikipedia article).

Disclaimer: the streams contain only knowledge data (STIX 2.1). All internal configurations, users, groups, roles, etc. are not streamed. For all attached files, the streams provide metadata information as well as links to authenticated download.

The "raw" stream

> This stream is a low level stream directly written in our Redis database. Every time something is written in OpenCTI, the platform stores the event in this stream: **create**, **update**, **delete** and **merge**.

The “raw” stream is available on the URL: . To access it, the logged in user must have the capability “*Connect and consume the platform stream*”. The available content is also **implicitly filtered with the allowed marking definitions** (aka *data segregation*).

Each type of event has a specific format so consuming this stream needs an understanding of this ontology and developing dedicated methods to handle it.

Creation event
The creation event “data” field contains the full data in pure STIX 2.1 format.


```
id: 1635416661232-0event: createdata: {  "markings": [],  "origin": {    "ip":
"::ffff:172.16.2.1",    "user_id": "45626d43-f88c-4c98-8729-f33b7730e6ba",
"referer": "https://opencti/dashboard/threats/threat_actors?"  },  "data": {
"name": "Test Threat Actor",    "description": "Test Threat Actor",    "confidence":
15,    "spec_version": "2.1",    "created_at": "2021-10-28T10:24:21.178Z",
"updated_at": "2021-10-28T10:24:21.178Z",    "revoked": false,    "lang": "en",
"created": "2021-10-28T10:24:21.178Z",    "modified": "2021-10-28T10:24:21.178Z",
"id": "threat-actor--aeefb1fc-7418-53cc-8ffb-78ee20289ff7",    "x_opencti_id":
"c0243521-b62d-4881-9584-968c6ecc0bc8",    "type": "threat-actor",
"x_opencti_type": "Threat-Actor"  },  "message": "creates a Threat-Actor `Test Threat
Actor`",  "version": "3"}
```


Delete event
The delete event field “data” contains the *id* of the deleted element and also the associated data in pure STIX 2.1 format at the deletion time (elements deleted recursively will be available in x_opencti_context field).


```
id: 1635416953680-0event: deletedata: {  "markings": [],  "origin": {    "ip":
"::ffff:172.16.2.1",    "user_id": "45626d43-f88c-4c98-8729-f33b7730e6ba",
"referer": "https://opencti/dashboard/threats/threat_actors/c0243521-b62d-4881-9584-
968c6ecc0bc8"  },  "data": {    "id": "threat-actor--aeefb1fc-7418-53cc-8ffb-
78ee20289ff7",    "name": "Test Threat Actor",    "description": "Test Threat Actor",
"confidence": 15,    "spec_version": "2.1",    "created_at": "2021-10-
28T10:24:21.178Z",    "updated_at": "2021-10-28T10:24:21.178Z",    "revoked": false,
"lang": "en",    "created": "2021-10-28T10:24:21.178Z",    "modified": "2021-10-
28T10:24:21.178Z",    "x_opencti_id": "c0243521-b62d-4881-9584-968c6ecc0bc8",
"type": "threat-actor",    "x_opencti_type": "Threat-Actor",    "x_opencti_context":
{      "deletions": []    }  },  "message": "deletes a Threat-Actor `Test Threat
Actor`",  "version": "3"}
```


Update event The update event contains the *id* of the updated element along with a specific patch object element containing the *replace/add/remove* operations.

```
id: 1635417153690-0event: updatedata:{  "markings": [],  "origin": {    "ip":
"::ffff:172.16.2.1",    "user_id": "45626d43-f88c-4c98-8729-f33b7730e6ba",
"referer": "https://opencti/dashboard/threats/threat_actors/92903430-97e6-45f6-a976-
ff99ec9809db"  },  "data": {    "x_opencti_patch": {      "replace": {
"description": {          "current": "This is a test description",
"previous": "test"        }      }    },    "id": "threat-actor--b088c4b6-52d5-533f-
b5e0-1a255970ec6e",    "x_opencti_id": "92903430-97e6-45f6-a976-ff99ec9809db",
"type": "threat-actor",    "x_opencti_type": "Threat-Actor"  },  "message": "replaces
`This is a test description` in `description`",  "version": "3"}
```


Another example:

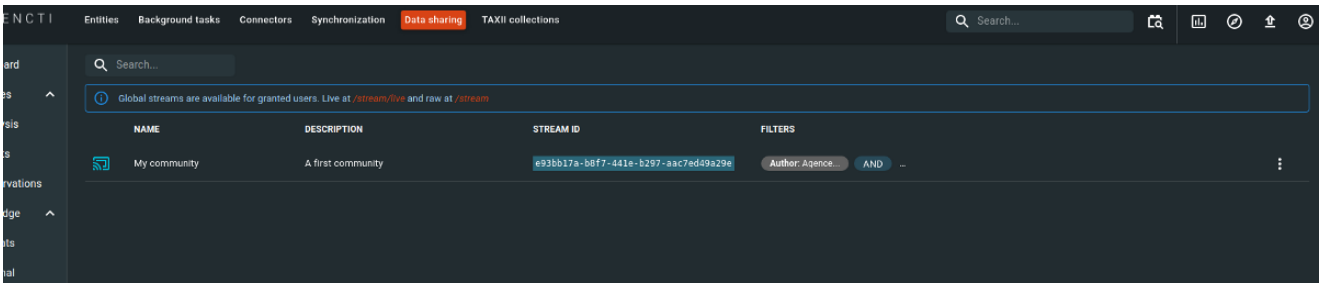

```
{  "markings": [],  "origin": {    "ip": "::ffff:172.16.2.1",    "user_id":
"45626d43-f88c-4c98-8729-f33b7730e6ba",    "referer":
"https://opencti/dashboard/threats/threat_actors/92903430-97e6-45f6-a976-
ff99ec9809db"  },  "data": {    "x_opencti_patch": {      "replace": {
"created_by_ref": {          "current": {            "value": "identity--01291fc4-
aa94-566f-9867-35217412a2f1",            "reference": "ISP Kievnet",
"x_opencti_id": "c0a3dca9-b07a-47c0-8c47-9baf831b9af5"          },
"previous": null        }      }    },    "id": "threat-actor--b088c4b6-52d5-533f-
b5e0-1a255970ec6e",    "x_opencti_id": "92903430-97e6-45f6-a976-ff99ec9809db",
"type": "threat-actor",    "x_opencti_type": "Threat-Actor"  },  "message": "replaces
`identity--01291fc4-aa94-566f-9867-35217412a2f1` in `created_by_ref`",  "version":
"3"}
```


Merge event

The merge event contains the list of the merged element along with the list of deleted elements due to the merge (in x_opencti_context field).

id: 1635417404792-0event: mergedata: { "markings": ["99518d4a-7cb2-4422-8b80-aec475882446"], "origin": {   "user_id": "45626d43-f88c-4c98-8729-f33b7730e6ba", "referer": "background_task" }, "data": {   "id": "malware--8bd2cac8-fbde-5b15-8d67-5dc91dfaa1f1",   "x_opencti_stix_ids": ["malware--47fe49ff-03f0-5033-96e3-9fe326a94369"   ],   "name": "Abaddon",   "description": "Uses Discord as C&C, has ransomware feature.",   "aliases": ["Abaddon RAT"   ],   "is_family": true,   "first_seen": "1970-01-01T00:00:00.000Z",   "last_seen": "5138-11-16T09:46:40.000Z",   "confidence": 15,   "revoked": false,   "lang": "en",   "created": "2021-09-13T04:42:44.664Z",   "modified": "2021-10-28T10:36:44.697Z",   "spec_version": "2.1",   "created_at": "2021-09-13T04:42:44.664Z",   "updated_at": "2021-10-28T10:36:44.697Z",   "created_by_ref": "identity--180d3ffd-a014-54ff-a817-211dddd29059",   "object_marking_refs": ["marking-definition--613f2e26-407d-48c7-9eca-b8e91df99dc9"   ], "external_references": [{   "source_name": "Bleepingcomputer RAT malware 2020",   "description": "Abrams, L. (2020, October 23). New RAT malware gets commands via Discord, has ransomware feature. Retrieved April 1, 2021.", "url": "https://www.bleepingcomputer.com/news/security/new-rat-malware-gets-commands-via-discord-has-ransomware-feature/",   "external_id": null   }   ], "x_opencti_id": "699db9c9-98eb-4d1b-9ab4-849611c7e084",   "type": "malware", "x_opencti_type": "Malware",   "x_opencti_patch": {   "add": { "x_opencti_stix_ids": ["malware--47fe49ff-03f0-5033-96e3-9fe326a94369"]   },   "replace": {   "aliases": {   "current": ["Abaddon RAT"   ],   "previous": null   },   "modified": { "current": "2021-10-28T10:36:44.697Z",   "previous": "2021-09-13T04:42:44.664Z"   },   "updated_at": {   "current": "2021-10-28T10:36:44.697Z",   "previous": "2021-09-13T04:42:45.350Z"   }   } },   "x_opencti_context": {   "sources": [{   "id": "malware--47fe49ff-03f0-5033-96e3-9fe326a94369",   "name": "Abaddon RAT", "is_family": false,   "first_seen": "1970-01-01T00:00:00.000Z", "last_seen": "5138-11-16T09:46:40.000Z",   "confidence": 15, "revoked": false,   "lang": "en",   "created": "2021-09-13T15:15:23.191Z",   "modified": "2021-09-13T15:15:23.191Z", "spec_version": "2.1",   "created_at": "2021-09-16T05:21:18.065Z", "updated_at": "2021-09-16T05:21:18.195Z",   "created_by_ref": "identity--e52b2fa3-2af0-5e53-ad38-17d54b3d61cb",   "object_marking_refs": ["marking-definition--613f2e26-407d-48c7-9eca-b8e91df99dc9"   ], "x_opencti_id": "c0a7affc-063e-497a-82cc-d31050f81e89",   "type": "malware", "x_opencti_type": "Malware"   }   ],   "deletions": [],   "shifts": [{   "id": "relationship--4219eeaa-369c-4f3d-97cb-0d7ceea990a4", "x_opencti_id": "dddcebec-8361-40f6-b34f-1cd9faa91663",   "type": "relationship",   "relationship_type": "uses",   "x_opencti_patch": { "replace": {   "source_ref": {   "current": { "x_opencti_id": "699db9c9-98eb-4d1b-9ab4-849611c7e084"   }, "previous": {   "value": "c0a7affc-063e-497a-82cc-d31050f81e89" }   }   }   }   },   [...]   }   ]   } }, "message": "merges Malware `Abaddon RAT` in `Abaddon`", "version": "3"}

Current internal usages of the raw stream



Raw stream URL is displayed in the data sharing section

Create the history

The history connector is a simple usage of the raw stream consumption that get each event in the stream to build the history index.

Rebuild a complete platform

A connector is available on demand to consume the raw stream of a platform and ingest data in a new one. Nevertheless, it's not released yet since this use case needs to maintain all the events all the events in the Redis stream indefinitely, so Redis sizing can be huge.

Rule engine real time behavior

The "rule engine" (released in 5.0.0) uses this stream internally to perform real time creation/update/deletion of the inferred elements created in the context of the enabled reasoning rules.

Example of inferred relationship

Share data to other systems

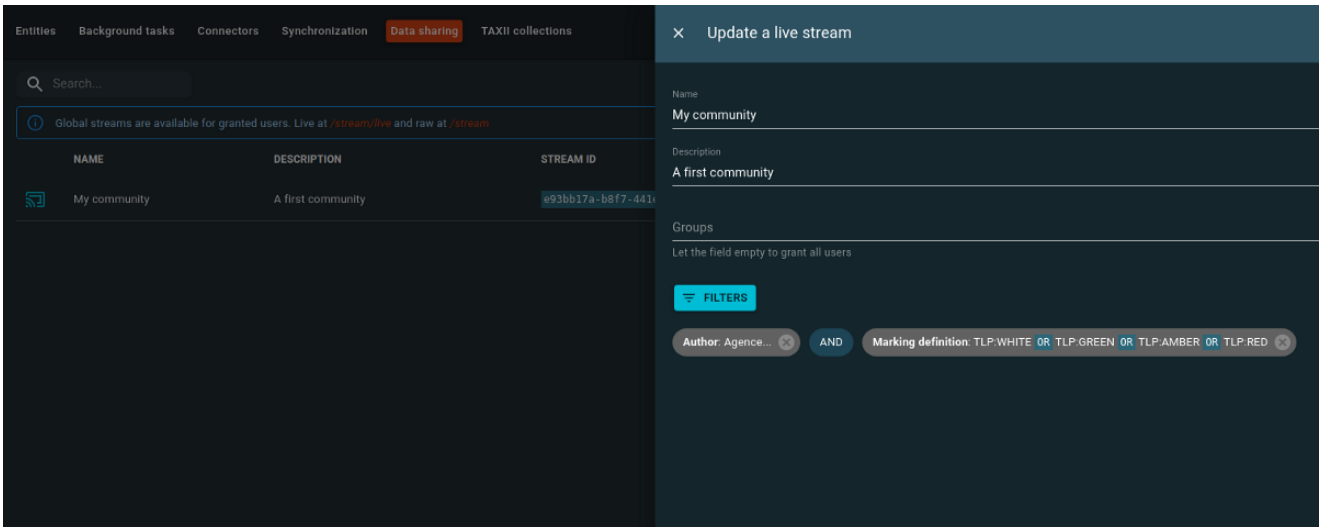
You can obviously use this stream to react on OpenCTI data in other systems. If you need help to understand the event types, formats, etc., you can join our Slack channel and get help.

Live/data sharing stream

> This stream is a high level stream (and the most used) where data are coming both from the ElasticSearch database and the raw stream. Each time something is updated in OpenCTI, the stream will publish the complete STIX 2.1 data based on the specified filters in the stream configuration.

An out-of-the-box live stream without any filter is available by default on the URL: . To access it, the logged in user must have the capability "*Connect and consume the platform stream*". The available content is also **implicitly filtered with the allowed marking definitions** (aka *data segregation*).

Beyond default raw and live streams, any user with the appropriate capabilities can also create his own live streams with filtering and group access restrictions directly in the OpenCTI user interface.



Example of user defined live stream

Why this is different from the raw stream?

*Data are always complete*This stream have only two types of event: *create* and *delete*. This allows to push only events containing the full data in STIX 2.1 format. There is no partial update/patching ontology as it is in the raw stream.

Data dependencies are implicitly injected
That's a huge difference between the raw stream and any other approach like the TAXII one. This stream takes care for consumer of the data dependencies. Basically, if you create a live stream filtered to only get the "**Malwares**", the stream will also published all data attached to this malware by transitivity.

Using this capability, the client does not need to process everything from the beginning to have consistent data but can just start consuming now and have all the required knowledge before going "live".

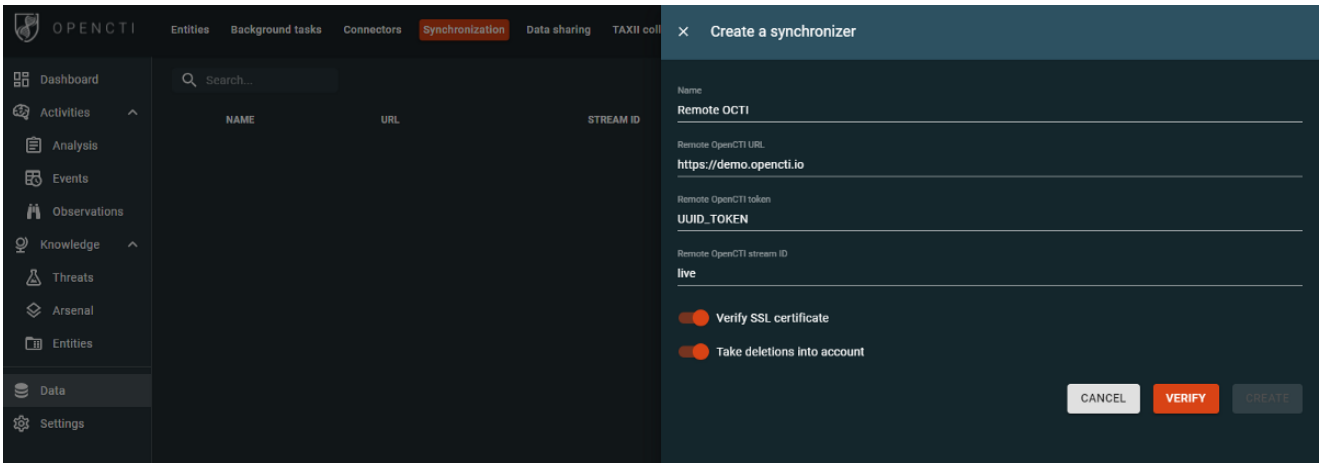
*Back pressure management*Not yet available in raw stream (this is the subject of on-going works), the live stream have the capability to handle some commands to manage the back pressure. I will explain a bit more the mechanism in the synchronization section. Please continue to read if you are interested 😀.

Current internal usages of the live streams

Backup and restore
Using the connector "backup-files", it's possible to use live streams to backup all OpenCTI data in STIX 2.1 JSON files (and raw files associated to element) in a directory. Also, a "restore-files" connector is available to restore the file from the a directory containing "backup-files" data.

Share data to other systems
Obviously this stream can be used to react on OpenCTI data in third party systems.

*Platform synchronization*Since OpenCTI 5.0.0, a built-in synchronization management feature is available in the user interface. To create a new synchronizer, an administrator should know the remote OpenCTI URL, token and live stream ID to consume. This, the embedded process will connect to the live stream of the remote OpenCTI, get the data as soon as its published in order to "create/update/delete" the elements.



Creation of a synchronizer

The back pressure problem

When we have developed the built-in synchronization management system within OpenCTI, we have encountered several issues and especially one concerning the back pressure problem.

Without any resilient queuing system between two different OpenCTI platforms, the observed ingestion speed on the consumer side was slower than the stream data production. After reviewing different options, it turned to be impossible to introduce this kind of middleware.

Therefore, we have decided to develop a system which controls the speed of the remote server to manage a limited queue size on the receiver.

How its currently works?

1. Synchronizer connects to the stream and starts getting data. Connect to and get a [connection_id] as a reference.Data is stored in a memory internal queue.
2. Synchronizer processes the data in order of the internal queue.
3. When queue size reach the memory limit (500 elements)
4. Synchronizer ask remote OpenCTI to slow downPost to [connection_id]specifying a delay.Data are now received at low speed.
5. Synchronizer continue to process the queue that now decrease.
6. When queue size reach the memory base acceptance (100 elements)Post to [connection_id]specifying the minimal delay.

This way the memory queue size on the receiver are always between 100 and 1000 elements depending of the receiver injection speed. Along with a state we maintain to be able to restart the processing if needed.

Conclusion

As you read in this blog post sharing data is a really important topic for OpenCTI and we expose a lot of different ways to do it. We hope this article will help you to use the right approach for you needs. And if you need other ways to doing it, please join our Slack channel or create an issue on the Github project.