BitSight Connector Implementation Guide

Version 1.1.0

Warning and Disclaimer

ThreatQuotient, Inc. provides this document “as is”, without representation or warranty of any kind, express or implied, including without limitation any warranty concerning the accuracy, adequacy, or completeness of such information contained herein. ThreatQuotient, Inc. does not assume responsibility for the use or inability to use the software product as a result of providing this information.

Last Updated: Tuesday, November 13, 2018

Contents

Versioning

- Current integration version `1.1.0`
- Supported on ThreatQ versions `4.11.1` and later

Introduction

This BitSght Connector ingests threat intelligence events from the BitSight vendor. The connector definition file maps how the threat intelligence data from the `alerts` endpoint is mapped to ThreatQ specific events and their related attributes.

A list of alerts can be retrieved from the endpoint `/ratings/v1/alerts`. Below is an example.

```
[
   {
       "alert_type": "PERCENT_CHANGE",
       "company_guid": "a5e23bf0-38d4-4cea-aa50-19ee75da481d",
       "href": "https://api.bitsighttech.com/ratings/v1/alerts/percent/6662353",
       "company_name": "Black Hills Technologies",
       "alert_date": "2018-08-04",
       "guid": 6662353,
       "folder_guid": "5d7bb4ba-bb2e-47ad-b6d9-a603f99fb950"
   }
]
```

A supplemental feed then pulls a specific individual alert from the list above. The endpoint for the above example is `/ratings/v1/alerts/percent/6662353`. The specific alert example is shown below.

```
{
        "guid": 6662353,
        "alert_date": "2018-08-04",
        "company_name": "Black Hills Technologies",
        "company_guid": "a5e23bf0-38d4-4cea-aa50-19ee75da481d",
        "start_date": "2018-07-31",
        "start_rating": 600,
        "end_rating": 560,
        "folder_guid": "5d7bb4ba-bb2e-47ad-b6d9-a603f99fb950",
        "company_url": "/company/14976400/",
        "rating_change_pct": -6,
        "alert_severity": "WARN",
        "alert_type": "PERCENT_CHANGE"
}
```

ThreatQ Mapping

ThreatQ provides the following default mapping for the feed. The mapping summarizes how the information from each field from the feed is converted to ThreatQ specific objects.

Feed Data Path	ThreatQ Entity	ThreatQ Object Type or Attribute Key	Normalization	Examples
alert_date	Event.happened_at		yyyy-MM-dd format	

Feed Data Path	ThreatQ Entity	ThreatQ Object Type or Attribute Key	Normalization	Examples
alert_type	Event.attribute	Alert Type	Title Case	Rating Threshold Nist Category Risk Category Portfolio Quality
company_name alert_type	Event.name		Title Case {company_name} - {alert_type}	Black Hills Technologies - Percent Change
start_rating	Event.attribute	Start Rating		
end_rating	Event.attribute	End Rating		
alert_severity	Event.attribute	Alert Severity	Title Case	Increase Warn Critical
company_name	Event.attribute	Company		Black Hills Technologies

Feed Data Path	ThreatQ Entity	ThreatQ Object Type or Attribute Key	Normalization	Examples
risk_category	Event.attribute	Risk Category		botnet_infections
nist_category	Event.attribute	NIST Category		PR.PT
nist_category_name	Event.attribute	NIST Category Name		Proactive Technology
start_grade	Event.attribute	Start Grade		
end_grade	Event.attribute	End Grade		
grade_threshold	Event.attribute	Grade Threshold		

Installation

The installation instructions for this integration differ based on the ThreatQ version you have installed.

ThreatQ Versions Before 4.7

It is assumed that the user has python installed with a minimum version of `2.7.5` and the following packages (use `pip` to install these packages):

- ruamel.yaml
- threatqsdk

threatqsdk is installed typically by making changes to the pip.conf file as follows:

```
[global]
       index-url = https://system-updates.threatq.com/pypi
         extra-index-url = https://username:password
         @extensions.threatq.com/threatq/integrations
         https://username:password@extensions.threatq.com/threatq/sdk
```

After satisfying the above requirements, the integration can be installed as follows:

```
python create_bitsight_cdf.py -c tq.config -y bitsight.yaml -f '{"username": {"value": "<YOURUSER>",
"label": "Username"}}'
```

ThreatQ Versions 4.7 Through 4.8

The environment and dependency instructions remain the same. The installation script differs and is given below.

```
python bulk_create_cdf.py -c tq.config -y bit-
sight.yaml -f '{"username": {"value": "<YOURUSER>",
"label": "Username"}}'
```

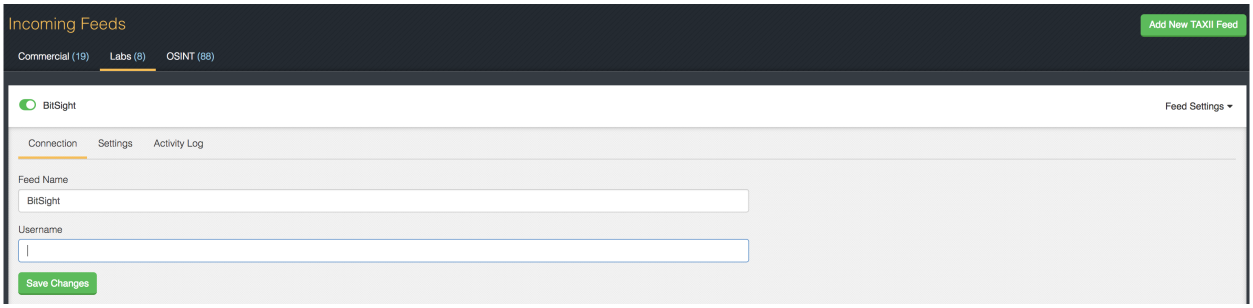
ThreatQ Version 4.9 or Later

The following artisan command on the platform will install the connector.

```
sudo php artisan threatq:feed-install bitsight.yaml
```

ThreatQ User Interface Configuration

The connector installs as a feed under Labs as shown below. The BitSight button must be enabled for the feed to begin ingesting data.

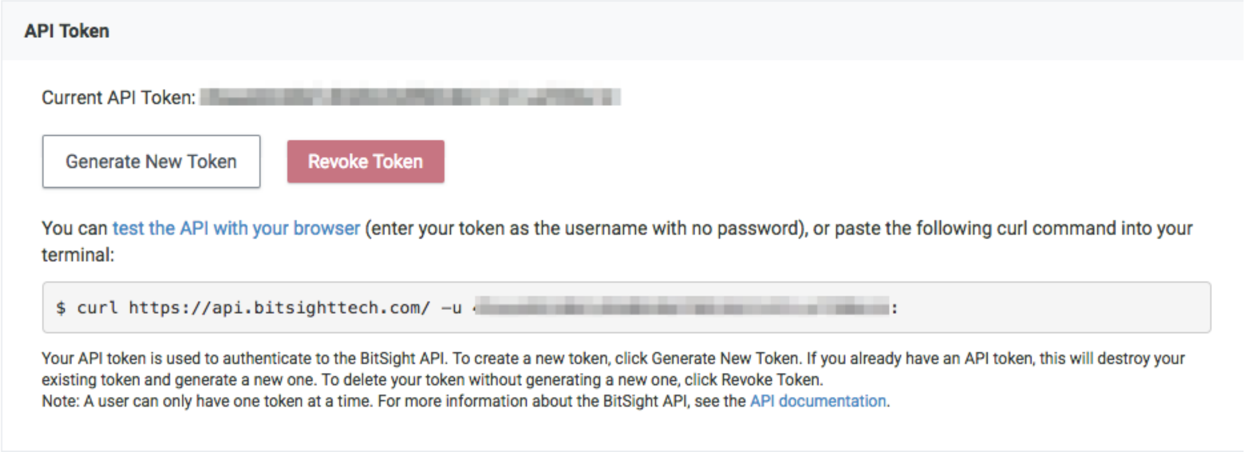


Feed Name

BitSight

Username

For the field **Username**, use the API key instead. The API key can be created in the user's Bitsight account as shown below.



API Token

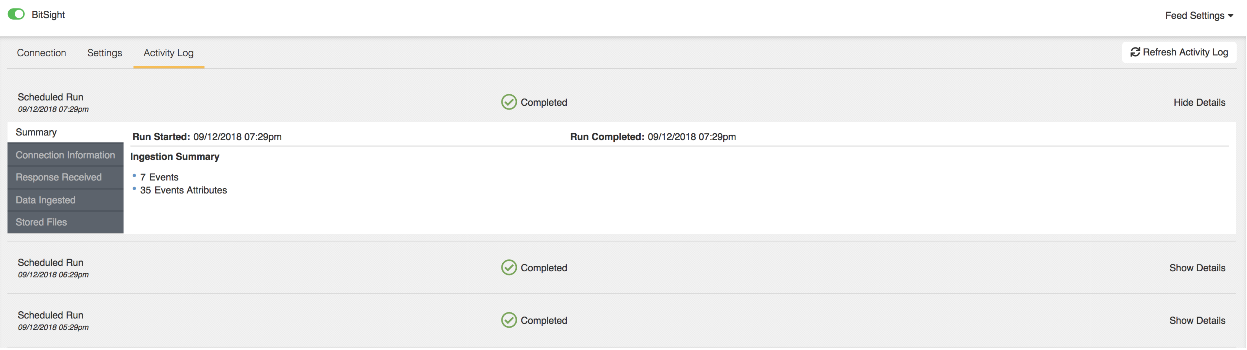
Current API Token: [redacted]

You can test the API with your browser (enter your token as the username with no password), or paste the following curl command into your terminal:

```
$ curl https://api.bitsighttech.com/ -u [redacted]:
```

Your API token is used to authenticate to the BitSight API. To create a new token, click Generate New Token. If you already have an API token, this will destroy your existing token and generate a new one. To delete your token without generating a new one, click Revoke Token.
Note: A user can only have one token at a time. For more information about the BitSight API, see the API documentation.

The status of the feed is available under **Activity Log** as shown below.



Scheduled Run
09/12/2018 07:29pm
Completed
Hide Details

Run Started: 09/12/2018 07:29pm
Run Completed: 09/12/2018 07:29pm

Ingestion Summary

- 7 Events
- 35 Events Attributes

Scheduled Run
09/12/2018 06:29pm
Completed
Show Details

Scheduled Run
09/12/2018 05:29pm
Completed
Show Details

Known Issues

If users are using a ThreatQ Version before 4.9.0, we recommend that you check the json arguments being passed (after the `-f` flag) using a `json validator` tool, since the script does not do any validation. If the json arguments were passed incorrectly, the UI feeds page is unable to parse those arguments and the whole feeds page fails to load. If ever this problem occurred regardless while using the script above, drop the connector from the database.