ThreatQuotient for ELK Stack

October 4, 2018

Version 1.0.0

11400 Commerce Park Dr
Suite 200,
Reston, VA
20191, USA
https://www.threatq.com/
Support: support@threatq.com
Sales: sales@threatq.com

Contents

CONTENTS **2**

LIST OF FIGURES AND TABLES **3**

ABOUT THIS THREATQUOTIENT FOR ELK STACK **4**

History 4
Review 4

1 INTRODUCTION **4**

1.1 Preface 4
1.2 Audience 4
1.3 Scope 4
1.4 Assumptions 5

2 IMPLEMENTATION OVERVIEW **6**

2.1 Prerequisites 6
2.2 Security and Privacy 6
2.3 Setting up the Integration 7
2.4 Configuring the connector 8

3 ALERT PARSING CONFIGURATION **10**

3.1 Logstash Filters 11
3.1.1 Translation Filters *11*
3.1.2 Clone Filters *12*
3.2 Dashboard Installation (Optional) 13
3.2.1 ThreatQ Dashboard *13*
3.2.2 ThreatQ Sightings Dashboard *13*
3.2.3 Installation *13*

4 CRON **15**

4.1.1 Setting Up the CRONJOB *15*

APPENDIX A: SQUID PROXY HIT EXAMPLE **16**

TRADEMARKS AND DISCLAIMERS **18**

List of Figures and Tables

FIGURE 1: TIME ZONE LIST EXAMPLE 6
FIGURE 2: TIME ZONE CHANGE EXAMPLE 6
FIGURE 3: INSTALLING FROM THE THREATQUOTIENT REPOSITORY (EXAMPLE OUTPUT) 7
FIGURE 4: CREATING INTEGRATION DIRECTORIES (EXAMPLE) 7
FIGURE 5: RUNNING THE INTEGRATION 8
FIGURE 6: RUNNING THE INTEGRATION WITH -N FLAG 8
FIGURE 7: THREATQ UI CONFIGURATION 8
FIGURE 8: ALERT PARSING CONFIGURATION 10
FIGURE 9: TRANSLATION FILTER 11
FIGURE 10: THREATQ CLONE FILTER 12
FIGURE 11: THREATQ SIGHTING OUTPUT 12
FIGURE 12: KIBANA MANAGEMENT ~SAVED OBJECTS 13
FIGURE 13: KIBANA MANAGEMENT ~IMPORT 13
FIGURE 14: KIBANA MANAGEMENT ~THREATQ DASHBOARDS 14
FIGURE 15: COMMAND LINE CRONTAB COMMAND 15
FIGURE 16: COMMAND LINE CRONTAB TQ-ELK-SYNC-DATA COMMAND 15

TABLE 1: THREATQUOTIENT FIREWALL PORTS INFORMATION 5
TABLE 2: THREATQUOTIENT USER INFORMATION 6

1 Introduction

1.1 Application Function

ELK is an extremely flexible document store and search platform that ingests files and data from a myriad of sources. In today's changing cybersecurity context, ELK is becoming a popular SIEM. Given ELK's flexibility in design and ingestion of data, a bi-directional integration can be performed in many ways.

This integration provides several key benefits, including the ease of installation and maintenance for customers.

This integration uses the following integration points:

- Logstash Translation Filters
- Logstash Clone filters
- Elasticsearch threatq index
- Elasticsearch threatq-sightings indices
- Xpack Watcher Alert API

1.2 Preface

The purpose of this ThreatQuotient for ELK Stack implementation guide is to provide the information necessary to implement the ThreatQuotient for ELK Stack application. Although it may be used as such, this document is not specifically intended as a site reference guide. It is assumed that the implementation engineer has experience installing and commissioning ThreatQuotient Apps and integrations covered within the document, as well as the experience necessary to troubleshoot at a basic level.

1.3 Audience

This document is intended for use by the following parties:

1. ELK system administrators & engineers
2. Security engineers

1.4 Scope

This document only covers the implementation of the ThreatQuotient for ELK Stack.

1.5 Assumptions

The following criteria is assumed to be in place and functional to allow the implementation of the ThreatQuotient for Elk Stack application into the managed estate:

- All ThreatQuotient equipment is online and in service.
- All required firewall ports have been opened.

Table 1: ThreatQuotient Firewall Ports Information

Source	Destination	Port	Description
Installation Point	ThreatQ	443	Connections to ThreatQ via SSL
	ELK	9200	Connections from where the integration is installed and Elasticsearch (default port 9200)

- All equipment is powered from permanent power supplies.
- A clock source of sufficient accuracy is connected to the network and the network is using it as the primary clock source.

This integration requires:

- ThreatQ version of 4.5 or greater
- ELK version of 6.3 or greater

If an Xpack license is available, then the alerts API can be used. If not, then the alerts API is unavailable.

This integration is only suitable for RHEL and RHEL variants (CentOS, Scientific Linux, etc). As such, only ELK servers of this type are supported. Contact support@threatq.com to find out if another Linux variant is supported.

2 Implementation Overview

2.1 Prerequisites

Throughout this implementation document, there we will refer to several files and directories, some of which will be symbolic, and others may change depending on specifics of the environmental setup.

Ensure all ThreatQ devices are set to the correct time (UTC is recommended), time zone and date, and using a clock source available to all devices.

To identify which time zone is closest to your present location, use the `timedatectl` command with the `list-timezones` command line option. For example, to list all available time zones in Europe, type:

Figure 1: Time Zone List Example

```
timedatectl list-timezones | grep Europe
Europe/Amsterdam
Europe/Athens
Europe/Belgrade
Europe/Berlin
```

To change the time zone to UTC, type as root:

Figure 2: Time Zone Change Example

```
timedatectl set-timezone UTC
```

Recommended user account that are required for the installation.

Table 2: ThreatQuotient User Information

Example Username	System	Description
elk@<company>.com	ThreatQ	A dedicated user for the ELK integration is recommended. This user is where the indicator searches will be created and are the credentials to use during installation.
threatq@<company>.com	ELK	If Xpack Security is being used, this would be a user that has access to read, write, create, and delete indices, as well as have access to the Watcher API. This is the user that ThreatQ will use to synchronize ThreatQ data in ELK.

2.2 Security and Privacy

Passwords have not been provided in this document. Please contact your project team for this information, if required.

All engineers are reminded that all data belonging and pertaining to the business is confidential and should not be disclosed to any unauthorized parties.

2.3 Setting up the Integration

The integration works by installing the python wheel file onto the ELK server itself and provides three commands once installed.

These commands are:

- Synchronize data in a saved Indicator Search with the threatq index.
- Get triggered alerts from the Xpack Watcher API and store them in ThreatQ.
- Use the same saved Indicator Searches to create files that can be used via Logstash's Translation filter.

Each level of the integration is optional, and customers can choose to integrate all three (recommended) or integrate only those pieces that fulfill their operational needs.

It is possible to install this onto the ThreatQ platform specifically, but a provision would need to be made for a mechanism to move the created Translation files from the ThreatQ instance to the ELK Logstash servers.

2.3.1 From the ThreatQuotient Repository

To install this ThreatQuotient for ELK Stack from the ThreatQuotient repository with YUM credentials, complete the following steps:

1. Install the ThreatQuotient for ELK Stack application by using the following commands:

Figure 3: Installing from The ThreatQuotient Repository (Example Output)

```
sudo pip install -i
https://<USERNAME>:<PASSWORD>@extensions.threatq.com/threatq/integrations tq-conn-
elk
Collecting tq-conn-elk
  Downloading https://extensions.threatq.com/threatq/integrations-
dev/+f/5e1/cc6d2ba4931da/tq_conn_elk-1.0.0.tar.gz
Requirement already satisfied (use --upgrade to upgrade): threatqcc>=1.3.1 in
/usr/lib/python2.7/site-packages (from tq-conn-elk)
Collecting elasticsearch==6.3.0 (from tq-conn-elk)
  Downloading
https://extensions.threatq.com/root/pypi/+f/24c/93ba3bb078328/elasticsearch-6.3.0-
py2.py3-none-any.whl (119kB)
    100% |████████████████████████████████| 122kB 300kB/s
Requirement already satisfied (use --upgrade to upgrade): pyparsing in
/usr/lib/python2.7/site-packages (from packaging->cryptography==1.8.1->tq-conn-elk)
Requirement already satisfied (use --upgrade to upgrade): pycparser in
/usr/lib/python2.7/site-packages (from cffi>=1.4.1->cryptography==1.8.1->tq-conn-
elk)
Installing collected packages: urllib3, elasticsearch, tq-conn-elk
  Found existing installation: urllib3 1.10.2
    Uninstalling urllib3-1.10.2:
      Successfully uninstalled urllib3-1.10.2
  Running setup.py install for tq-conn-elk ... done
Successfully installed elasticsearch-6.3.0 tq-conn-elk-1.0.0 urllib3-1.23
```

2. Once the application has been installed, use the `mkdir -p` command to create a directory structure for all configuration, logs and files. See example below:

Figure 4: Creating Integration Directories (Example)

```
mkdir -p /opt/tq-integrations/elk
mkdir -p /opt/tq-integrations/elk/config
mkdir -p /opt/tq-integrations/elk/logs
```

A driver called `tq-elk-sync-data` is installed.

3. Issue the following command to initialize the integration:

```
$> tq-elk-sync-data -c /file/path/to/config/ -ll /file/path/to/logs/ -f
```

During this initial execution, several prompts will be displayed for the following information:

- **ThreatQ Host:** Hostname or IP Address of the ThreatQ server.
    - If this is a hostname, it must be resolvable on the installation point.
- **Client ID:** This is the OAuth Management value found in **Settings** > **OAuth Management.**
- **E-Mail Address:** This is the e-mail address of the ***ThreatQ*** user for this integration.
    - This should be a dedicated user (e.g. elk@threatq.com).
- **Password:** This is the Password for the above ***ThreatQ*** user.
- **Status:** This is the default status of newly created IoCs.

Figure 5: Running the Integration

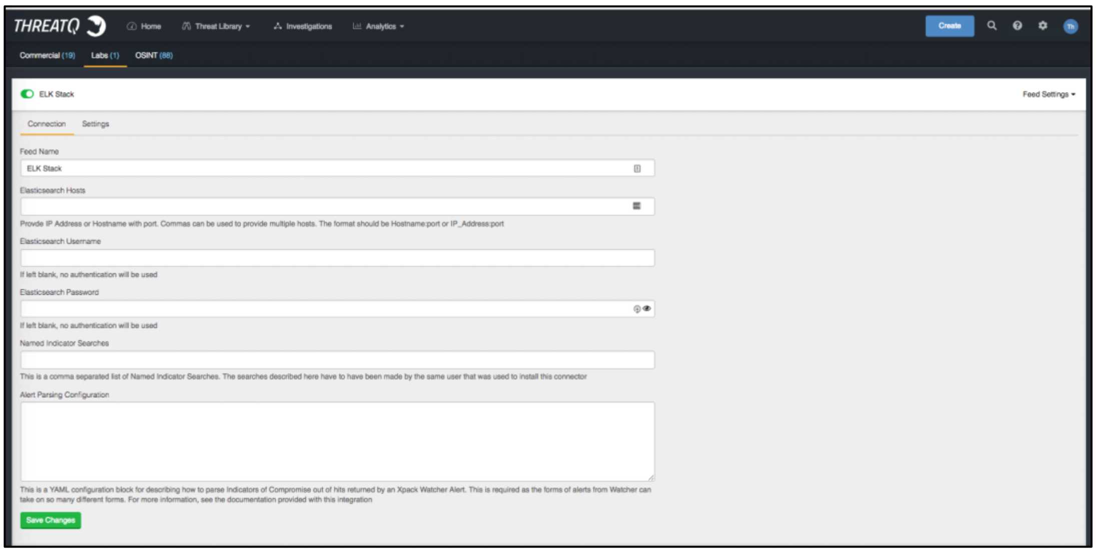
```
$> tq-elk-sync-data -c /file/path/to/config/ -ll /file/path/to/logs/ -f
/file/path/to files/ -v 3  --files files
ThreatQ Host: <ThreatQ Host IP or Hostnme >
Client ID: <ClientID>
E-Mail Address: <EMAIL ADDRESS>
Password: <PASSWORD>
Status: Active
Connector configured.  Set information in UI.
```

The driver will run once, where it will connect to the ThreatQ instance and install the user interface component of the connector.

2.4 Configuring the connector

After the installation is complete, navigate to the ThreatQ user interface to **Settings > Incoming Feeds > Labs** and locate the ELK Stack entry.

Figure 6: ThreatQ UI Configuration



ELK Stack

Feed Name
ELK Stack

Elasticsearch Hosts
Provde IP Address or Hostname with port. Commas can be used to provide multiple hosts. The format should be Hostname:port or IP_Address:port

Elasticsearch Username
If left blank, no authentication will be used

Elasticsearch Password
If left blank, no authentication will be used

Named Indicator Searches
This is a comma separated list of Named Indicator Searches. The searches described here have to have been made by the same user that was used to install this connector

Alert Parsing Configuration
This is a YAML configuration block for describing how to parse Indicators of Compromise out of hits returned by an Xpack Watcher Alert. This is required as the forms of alerts from Watcher can take on so many different forms. For more information, see the documentation provided with this integration

Complete the form with the following information:

1. **Elasticsearch Hosts**: This is a coma separated list of Elasticsearch hosts for use with this integration. These hosts should be hostnames or IP Addresses that can be resolved/reached from the installation point. These should be in the form hostname:port or ip address:port. For instance, if your Elasticsearch node is at `192.168.0.2` and running the default port of 9200, then `192.168.0.2:9200` should be entered.
2. **Elasticsearch Username:** If Xpack Security is being used, this supports HTTP Basic Auth style authentication. This would be the Elasticsearch user that should be used to synchronize data between the systems. If Xpack Security is disabled, leave this blank.
3. **Elasticsearch Password:** If the above Username is provided, then the password must also be provided. This can be left blank if Xpack Security is disabled.
4. **Named Indicator Searches:** This is a comma-separated list of saved Indicator Searches. (**Magnifying Glass > Indicator Search**) that should be used as the export to synchronize data with.
5. **Alert Parsing Configuration**: See the Alert Parsing Configuration section below.

Once complete, click **Save Changes** and ensure that the toggle next the name is enabled.

3 Alert Parsing Configuration

Xpack Alerting is extremely flexible, and as such, the parsing method for this extremely flexible parsing must also be flexible. To accommodate this flexibility, the **Alert Parsing Configuration** field has been added. In this field, the **Watcher ID** is given as the main key, and then the hits, the IoC description, and the IoC Attributes description are given so that hits can be processed properly.

As an example, it is assumed we have a Squid Proxy server, and are monitoring the logs for matches against the ThreatQ data via the translation filters (See the Logstash Filters Section). This allows us to use the following parsing configuration:

Figure 7: Alert Parsing Configuration

```
---
  threatq-squid-sightings:
    hits: "result.input.payload.hits.hits"
    attributes:
      _source.status_code: "ELK Squid Status Code"
      _source.path: "ELK Original File Path"
      _source.@timestamp: "ELK Seen at"
      _soruce.hierarchy_code: "SQUID Hierarchy Code"
      _id: "ELK Event ID"
    iocs:
      _source.server: "IP Address"
      _source.url: "URL"
```

A corresponding match would appear like the text found in Appendix A: Squid Proxy Hit Example:

Viewing the parsing configuration, there are three main components:

- **Hits**: This is the path in the watcher output to the hits that should be recorded as IoCs. In this case, the hits are results.input.payload.hits.hits, but this will vary for any watcher. configuration. The best way to develop this for your installation is to simulate a hit via the Watcher configuration page in Elk (**Management > Watcher**).
- **iocs**: These are the individual Indicators of Compromise that caused the trigger. In the case above we say that the field `_source.server` should contain an IP Address and that `_source.url` should contain a URL. When this alert is parsed, those two fields will be used to create 2 IoCs in ThreatQ and link them to a triggered event.
- **attributes:** These are attributes in the data stream that should be parsed and recorded for each IoC. The form is path: "Attribute Name" and the value will be whatever is in that path. If a path doesn't exist, it will be skipped.

3.1 Logstash Filters

Logstash provides a way to parse and pre-analyze logs, then store them in several places, including the Elasticsearch stores. Since Logstash can do pre-analysis work, it is possible to enrich data in the log stream and store that data as part of the original data. To do this, a Logstash Translation filter is used.

It is also possible to store sightings of ThreatQ data directly. To do this, after the data has been enriched via the Translation filter, a Clone filter is used to split the data, clean it, and store it in a second sightings index, to be analyzed separately.

3.1.1 Translation Filters

Included below is an example configuration for a Logstash Translation filter. This can be configured in a variety of ways, but the below configuration is the expected configuration for items downstream.

Figure 8: Translation Filter

```
translate {
            dictionary_path => "/opt/logstash_filters/ThreatQ Translation File --
IP Address.yml"
            field => server
            destination => threatq_data
            add_field => { "threatq_match_field" => "server"}
            add_tag => ["threatq_match"]
        }
kv {
            allow_duplicate_values => false
            field_split => "|"
            target => "threatq_match_data"
            remove_field => "threatq_data"
            source => "threatq_data"
        }
```

Here, the files generated by the **tq-elk-create-enrichment-files** command are placed in the `/opt/logstash_filters` directory. In this location, an IP Address specific file is used to check the field `server` from the upstream parsing.

If that IP Address is in the `ThreatQ Translation File – IP Address.yml` file, it will grab pertinent information and add it to the `threatq_data` field.

The `kv` filter is then used to parse that stored information into its subsidiary parts.

These are then added to the `threatq_match_data` sub-object, which contains the following information:

- ThreatQ ID
- Link back to ThreatQ for that IoC
- Sources
- Adversaries
- Score
- Indicator Type

This information is then added into the stream so that sightings of ThreatQ data can easily be found.

3.1.2 Clone Filters

As the number of sightings of ThreatQ data rises, containing a footprint of these in their own index would be beneficial for analysis and roll up reporting. To do this, use a `clone` filter. The following configuration will show a clone filter that copies the data, cleans the parsed event information, and keeps only the relevant ThreatQ data.

This will speed up analysis of Sightings greatly.

Figure 9: ThreatQ Clone Filter

```
if "threatq_match" in [tags] {
            clone {
                clones => [ "threatq_sighting" ]
                add_tag => [ "threatq_match_clone" ]
            }
            if [type] == "threatq_sighting" {
                prune {
                   whitelist_names => ["^message$", "threatq_match_data",
"threatq_match_field", "@timestamp", "^timestamp$", "^type$", "^tag$"]
                }
            }
        }
```

In the above example, this takes data that has been matched to ThreatQ data in one of the Translation Filter Files, and then clones this and then strips it of nearly all of the original data (except the message field) and then sends it upstream.

To store this data, an output for this type of event ({{threatq_sighting}}) would have to be added. It should look like the following:

Figure 10: ThreatQ Sighting Output

```
if [type] == "threatq_sighting" {
        elasticsearch {
            hosts => ["localhost:9200"]
            index => "threatq-sightings-%{+YYYY.MM.dd}"
        }
```

Once complete, the data will fill the threatq-sightings index, which is used to fuel the **ThreatQ Sightings Dashboard**.

Run the following commands manually before creating your index patterns or creating dashboards, once they have run without error and have been populated, they can then be added to the cronjob (see the CRON Section).

Figure 11: ThreatQ Manual Run Commands

```
tq-elk-sync-data -c -c /file/path/to/config/ -ll /file/path/to/logs/ -f
/file/path/to files/ -v 3  --files files

tq-elk-get-alerts -c /file/path/to/config/ -ll /file/path/to/logs/ -f /file/path/to
files/ -v 3  --files files

tq-elk-create-enrichment-files -c /file/path/to/config/ -ll --enrichment-dest
/opt/logstash_filters/ -v3
```

3.2 Dashboard Installation (Optional)

A powerful capability of ELK is to visualize data in a dashboard that can auto update, showing in near real time the threats and issues that a SOC faces. To aid in this, two dashboards are provided with this integration.

Ensure that index patterns for `threatq` and `threatq-sightings*` are created.

3.2.1 ThreatQ Dashboard

The ThreatQ Dashboard shows analytics for the `threatq` index. This shows only the data from ThreatQ and how it breaks down statistically in several ways.

The dashboard will not aid in the normal Hunt Missions of the SOC, but it will show the data in the Translation Files and what data is available in ELK from ThreatQ.

This is important as the breakdown can be fine-tuned by changing the Indicator Searches that are used to export data to ELK, providing a litmus test for search configurations.

3.2.2 ThreatQ Sightings Dashboard

If the Translation and Clone Logstash filters are configured for some or all of the incoming data to Logstash, this dashboard will show, the sightings of ThreatQ information in near real time.

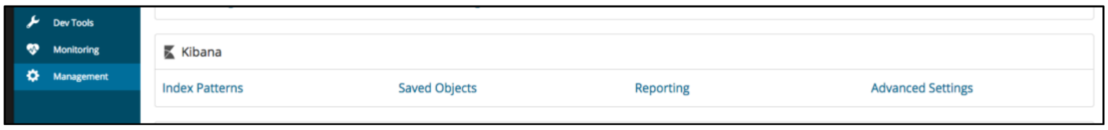
This is important as it relays information on which Threat Intelligence sources are most efficacious, which adversaries may be operating, and the ThreatQ score of the IoC that triggered the sighting.

All of this can be used to hone the data that is stored in ELK and eventually hone the information exported to other systems, like Firewalls, IDS, IPS, and so on.

3.2.3 Installation

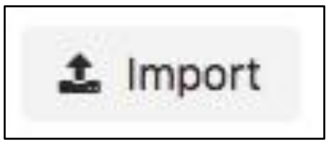
Both dashboards are located in one file, and will be installed simultaneously. To do this, as a Kibana Administrator, navigate to **Management** > **Saved Objects**:

Figure 12: Kibana Management ~Saved Objects



Click on Import:

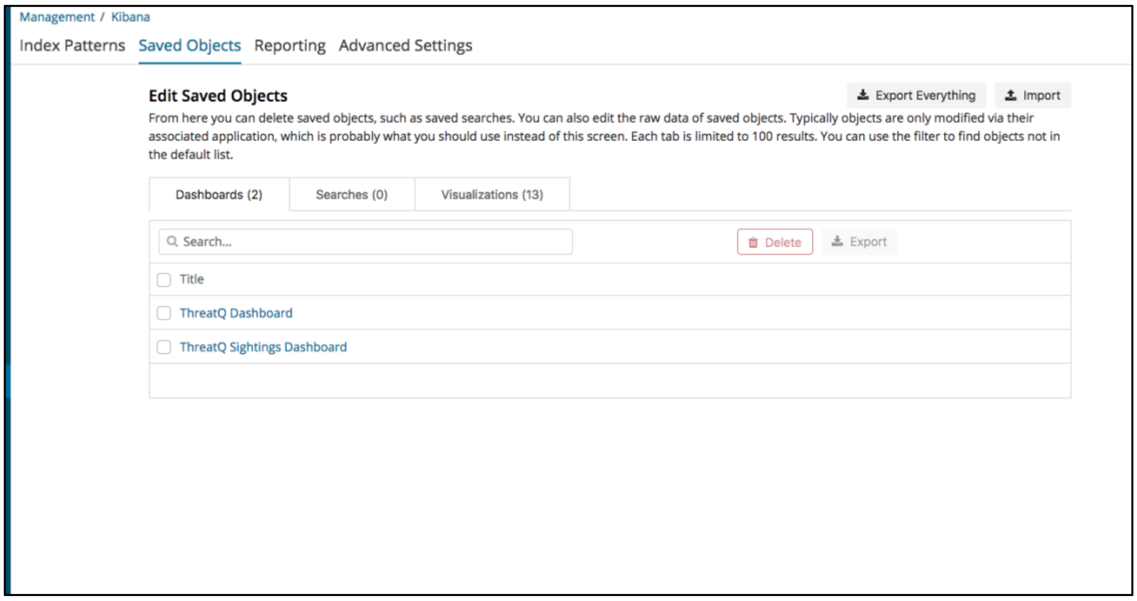
Figure 13: Kibana Management ~Import



Select the `threatq_dashboards.json` file and upload it.

After completion, you should now have the following dashboards available:

Figure 14: Kibana Management ~ThreatQ Dashboards


Management / Kibana

Index Patterns | Saved Objects | Reporting | Advanced Settings

Edit Saved Objects

Export Everything | Import

From here you can delete saved objects, such as saved searches. You can also edit the raw data of saved objects. Typically objects are only modified via their associated application, which is probably what you should use instead of this screen. Each tab is limited to 100 results. You can use the filter to find objects not in the default list.

Dashboards (2) | Searches (0) | Visualizations (13)

Search... | Delete | Export

Title
ThreatQ Dashboard
ThreatQ Sightings Dashboard

⚠ When these are loaded for the first time, **ensure** that index patterns for `threatq` and `threatq-sightings*` are created.

4 CRON

To run this script on a reoccurring basis use CRON or some other system schedule. The argument in the cron script ***must*** specify the config and log locations.

Threat Intelligence data changes frequently, as new data enters the system and old data stales out, these changes should be exported to ELK.

To accomplish this, one of the three commands are executed via cron or some other scheduling system to repeatedly refresh the data.

The following three commands are described below:

- **tq-elk-sync-data:** This synchronizes data between saved Indicator Searches and the threatq index in Elasticsearch.
- **tq-elk-get-alerts:** This synchronizes triggered alerts that have their internal conditions met with ThreatQ as Events of type, Sighting.
- **tq-elk-create-enrichment-files:** This produces a set of files, based off of the saved Indicator Searches, to be used with Logstash Filters.

Each of these should be added to {{cron}} or another task scheduler to refresh the data in the individual components. This can be run multiple times a day and should not be run more often than once per hour.

4.1.1 Setting Up the CRONJOB

1. Login via a CLI terminal session to you ThreatQ host.
2. Input the commands below.

   Figure 15: Command Line Crontab Command

   ```
   $> crontab -e
   ```

   This will enable the editing of the crontab, using vi.

   Depending on how often you wish the cronjob to run, you will need to adjust the time to suit the environment.
3. Input the commands below – this example shows every **10,15 & 20 minutes past the Hour.**

   Figure 16: Command Line Crontab tq-elk-sync-data Command

```
10 * * * * tq-elk-sync-data -c /file/path/to/config/ -ll /file/path/to/logs/ -v3
15 * * * * tq-elk-get-alerts -c /file/path/to/config/ -ll /file/path/to/logs/ -v3
20 * * * * tq-elk-create-enrichment-files -c /file/path/to/config/ -ll
/file/path/to/logs/ --enrichment-dest /opt/logstash_filters/ -v3
```

To run this script on a reoccurring basis, use CRON or some other on system schedule. CRON is shown below.

The argument in the cron script ***must*** specify the config and log locations.

Appendix A: Squid Proxy Hit Example

Example of a Squid Proxy match.

The text contained within has been edited and should only serve as a reference.

```
{
  "_index": ".watcher-history-7-2018.08.14",
  "_type": "doc",
  "_id": "threatq-squid-sightings_6d57dded-f167-43c2-ab62-c39a8726f327-2018-08-
14T12:41:57.487Z",
  "_score": 1,
  "_source": {
    "watch_id": "threatq-squid-sightings",
    "node": "2QNng82MStyeJKKfyu8JgA",
    "state": "executed",
    "status": {
      "state": {
        "active": true,
        "timestamp": "2018-08-13T17:31:40.161Z"
      },
      "last_checked": "2018-08-14T12:41:57.487Z",
      "last_met_condition": "2018-08-14T12:41:57.487Z",
      "actions": {
        "my-logging-action": {
          "ack": {
            "timestamp": "2018-08-14T12:41:57.487Z",
            "state": "ackable"
          },
          "last_execution": {
            "timestamp": "2018-08-14T12:41:57.487Z",
            "successful": true
          },
          "last_successful_execution": {
            "timestamp": "2018-08-14T12:41:57.487Z",
            "successful": true
          }
        }
      },
      "execution_state": "executed",
      "version": -1
    },
    "trigger_event": {
      "type": "schedule",
      "triggered_time": "2018-08-14T12:41:57.486Z",
      "schedule": {
        "scheduled_time": "2018-08-14T12:41:57.363Z"
      }
    },
    "input": {
      "search": {
        "request": {
          "search_type": "query_then_fetch",
          "indices": [
            "squid-access-*"
          ],
          "types": [],
          "body": {
            "query": {
              "bool": {
                "must": [
                  {
                    "exists": {
                      "field": "threatq_match_field"
                    }
                  },
                  {
                    "range": {
                      "@timestamp": {
                        "gte": "now-10m",
                        "lte": "now"
                      }
                    }
                  }
                ]
              }
            }
          }
        }
      }
    },
      "condition": {
        "type": "compare",
        "status": "success",
        "met": true,
        "compare": {
          "resolved_values": {
            "ctx.payload.hits.total": 79
          }
        }
      },
      "actions": [
        {
          "id": "my-logging-action",
          "type": "logging",
          "status": "success",
          "logging": {
            "logged_text": "There are 79 documents in your index. Threshold is 10."
          }
        }
      ]
    },
    "messages": []
  },
  "fields": {
    "result.execution_time": [
      "2018-08-14T12:41:57.487Z"
    ],
    "trigger_event.schedule.scheduled_time": [
      "2018-08-14T12:41:57.363Z"
    ],
    "trigger_event.triggered_time": [
      "2018-08-14T12:41:57.486Z"
    ]
  }
}
```

Trademarks and Disclaimers