Enriching With Threat Intelligence Information

Date of Publish: 2018-10-15

http://docs.hortonworks.com

Contents

Enriching with Threat Intelligence Information

The threat intelligence topology takes a normalized JSON message and cross references it against threat intelligence information, tags it with alerts if appropriate, runs the results against the scoring component of machine learning models where appropriate, and stores the telemetry in a data store.

Prior to configuring threat intelligence, you must meet the following requirements:

- Choose your threat intelligence sources
- As a best practice, install a threat intelligence feed aggregator, such as SoltraEdge
- Mark messages as threats based on data in external data stores
- Mark threat alerts with a numeric triage level based on a set of Stellar rules

You can bulk load threat intelligence information from the following sources:

- CSV Ingestion
- HDFS through MapReduce
- Taxii Loader

Bulk Loading CSV Threat Intelligence Sources

Although Hortonworks Cybersecurity Platform (HCP) is designed to work with STIX/Taxii threat feeds, you can also be bulk loaded with threat data from a CSV file.

The shell script $METRON_HOME/bin/flatfile_loader.sh reads data from local disk and loads the threat intelligence data into an HBase table. This loader uses the special configuration parameter inputFormatHandler to specify how to consider the data. The two implementations are BY_LINE and org.apache.metron.dataloads.extractor.inputformat. WholeFileFormat.

The default is BY_LINE, which makes sense for a list of CSVs in which each line indicates a unit of information to be imported. However, if you are importing a set of STIX documents, then you want each document to be considered as input to the Extractor.

Start the user parser topology by running the following:

```
$METRON_HOME/bin/start_parser_topology.sh -s user -z $ZOOKEEPER_HOST:2181 -k
 $KAKFA_HOST:6667
```

The parser topology listens for data streaming in and pushes the data to HBase. Now you have data flowing into the HBase table, but you need to ensure that the enrichment topology can be used to enrich the data flowing past.

The parameters for the utility are as follows:

Short Code	Long Code	Is Required?	Description
-h		No	Generate the help screen/set of options
-e	--extractor_config	Yes	JSON document describing the extractor for this input data source
-t	--hbase_table	Yes	The HBase table to import into
-c	--hbase_cf	Yes	The HBase table column family to import into

Short Code	Long Code	Is Required?	Description
-i	--input	Yes	The input data location on local disk. If this is a file, then that file will be loaded. If this is a directory, then the files will be loaded recursively under that directory.
-l	--log4j	No	The log4j properties file to load
-n	--enrichment_config	No	The JSON document describing the enrichments to configure. Unlike other loaders, this is run first if specified.

HDFS via MapReduce

The shell script $METRON_HOME/bin/flatfile_loader.sh will kick off the MapReduce job to load data stated in HDFS into an HBase table. The following is as example of the syntax:

```
                          $METRON_HOME/bin/flatfile_loader.sh -i /tmp/
top-10k.csv -t enrichment -c t -e ./extractor.json -m MR
```

The parameters for the utility are as follows:

Short Code	Long Code	Is Required?	Description
-h		No	Generate the help screen/set of options
-e	--extractor_config	Yes	JSON document describing the extractor for this input data source
-t	--hbase_table	Yes	The HBase table to import into
-c	--hbase_cf	Yes	The HBase table column family to import into
-i	--input	Yes	The input data location on local disk. If this is a file, then that file will be loaded. If this is a directory, then the files will be loaded recursively under that directory.
-l	--log4j	No	The log4j properties file to load
-n	--enrichment_config	No	The JSON document describing the enrichments to configure. Unlike other loaders, this is run first if specified.

Configure a CSV Extractor Configuration File

After you have chosen a threat intelligence feed source, you must configure an extractor configuration file that describes the source.

Procedure

1. Log in as root user to the host on which Metron is installed.
2. Create a file called extractor_config_temp.json and add the following content:

```
{
"config" : {
   "columns" : {
```

```
        "domain" : 0
        ,"source" : 1
    }
    ,"indicator_column" : "domain"
    ,"type" : "zeusList"
    ,"separator" : ","
  }
  ,"extractor" : "CSV"
}
```

3. You can transform and filter the enrichment data as it is loaded into HBase by using Stellar extractor properties in the extractor configuration file. HCP supports the following Stellar extractor properties:

value_transform

Transforms fields defined in the columns mapping with Stellar transformations. New keys introduced in the transform are added to the key metadata. For example:

```
"value_transform" : {
   "domain" :
 "DOMAIN_REMOVE_TLD(domain)"
```

value_filter

Allows additional filtering with Stellar predicates based on results from the value transformations. In the following example, records whose domain property is empty after removing the TLD are omitted.

```
"value_filter" : "LENGTH(domain) >
 0",
  "indicator_column" : "domain",
```

indicator_transform

Transforms the indicator column independent of the value transformations. You can refer to the original indicator value by using indicator as the variable name, as shown in the following example. In addition, if you prefer to piggyback your transformations, you can refer to the variable domain, which allows your indicator transforms to inherit transformations done to this value during the value transformations.

```
"indicator_transform" : {
   "indicator" :
 "DOMAIN_REMOVE_TLD(indicator)"
```

indicator_filter

Allows additional filtering with Stellar predicates based on results from the value transformations. In the following example, records whose indicator value is empty after removing the TLD are omitted.

```
"indicator_filter" :
 "LENGTH(indicator) > 0",
  "type" : "top_domains",
```

If you include all of the supported Stellar extractor properties in the extractor configuration file, it will look similar to the following:

```
{
```

```
  "config" : {
    "zk_quorum" : "$ZOOKEEPER_HOST:2181",
    "columns" : {
       "rank" : 0,
       "domain" : 1
    },
    "value_transform" : {
       "domain" : "DOMAIN_REMOVE_TLD(domain)"
    },
    "value_filter" : "LENGTH(domain) > 0",
    "indicator_column" : "domain",
    "indicator_transform" : {
       "indicator" : "DOMAIN_REMOVE_TLD(indicator)"
    },
    "indicator_filter" : "LENGTH(indicator) > 0",
    "type" : "top_domains",
    "separator" : ","
  },
  "extractor" : "CSV"
}
```

Running a file import with the above data and extractor configuration will result in the following two extracted data records:

Indicator	Type	Value
google	top_domains	{ "rank" : "1", "domain" : "google" }
yahoo	top_domains	{ "rank" : "2", "domain" : "yahoo" }

4. To access properties that reside in the global configuration file, provide a ZooKeeper quorum via the zk_quorum property. If the global configuration looks like "global_property" : "metron-ftw", enter the following to expand the value_transform:

```
"value_transform" : {
    "domain" : "DOMAIN_REMOVE_TLD(domain)",
     "a-new-prop" : "global_property"
 },
```

The resulting value data will look like the following:

Indicator	Type	Value
google	top_domains	{ "rank" : "1", "domain" : "google", "a-new-prop" : "metron-ftw" }
yahoo	top_domains	{ "rank" : "2", "domain" : "yahoo", "a-new-prop" : "metron-ftw" }

5. Remove any non-ASCII characters:

```
iconv -c -f utf-8 -t ascii extractor_config_temp.json -o
 extractor_config.json
```

6. Configure the mapping for the element-to-threat intelligence feed.

   This step configures which element of a tuple to cross-reference with which threat intelligence feed. This configuration is stored in ZooKeeper.

   a) Log in as root user to the host that has Metron installed.

   b) Cut and paste the following file into a file called enrichment_config_temp.json":

```
{
     "zkQuorum" : "$ZOOKEEPER_HOST:2181"
    ,"sensorToFieldList" : {
     "$DATASOURCE" : {
```

```
          "type" : "THREAT_INTEL"
         ,"fieldToEnrichmentTypes" : {
              "domain_without_subdomains" : [ "zeusList" ]
          }
      }
   }
}
```

c) Remove the non-ASCII characters:

```
iconv -c -f utf-8 -t ascii enrichment_config_temp.json -o
 enrichment_config.json
```

Configure CSV Mapping for the Intelligence Feed

After you configure an extractor configuration file, you must configure which element of a tuple to cross-reference with which threat intelligence feed. This configuration is stored in ZooKeeper.

Procedure

1. On the host with Metron installed, log in as root.
2. Cut and paste the following file into a file called enrichment_config_temp.json:

```
{
     "zkQuorum" : "$ZOOKEEPER_HOST:2181"
    ,"sensorToFieldList" : {
     "$DATASOURCE" : {
          "type" : "THREAT_INTEL"
         ,"fieldToEnrichmentTypes" : {
              "domain_without_subdomains" : [ "zeusList" ]
          }
      }
   }
}
```

3. Remove any non-ASCII invisible characters in the pasted syntax in Step 2:

```
iconv -c -f utf-8 -t ascii enrichment_config_temp.json -o
 enrichment_config.json
```

Run the CSV Threat Intel Loader

After you define the threat intelligence source, threat intelligence extractor, and threat intelligence mapping configuration, you must run the loader to move the data from the threat intelligence source to the Metron threat intelligence store and to store the enrichment configuration in ZooKeeper.

Procedure

1. Log into $HOST_WITH_ENRICHMENT_TAG as root and navigate to $METRON_HOME/config.
2. Use the loader to move the enrichment source to the enrichment store in ZooKeeper:

```
$METRON_HOME/bin/flatfile_loader.sh -n threatintel_config.json
 -i domainblocklist_ref.csv -t threatintel -c t -e
 threatintel_extractor_config.json
```

This command modifies the Squid enrichment config in ZooKeeper to include the threat intel enrichment.

The parameters for the utility are as follows:

-b,--batchSize <SIZE>	The batch size to use for HBase puts

-c,--hbase)cf <CF>	HBase column family to ingest the data into. This must be column family t.
-e,--extractor_config <JSON_FILE>	JSON Document describing the extractor for this input data source
-h,--help	Generate Help screen
-i,--input <FILE>	The CSV File to load
-l,--log4j <FILE>	The log4j properties file to load
-m,--import_mode <MODE>	The Import mode to use: LOCAL,MR.Default: LOCAL
-n,--enrichment_config <JSON_FILE>	JSON Document describing the enrichment configuration details. This is used to associate an enrichment type with a field type in ZooKeeper.
-p,--threads <NUM_THREADS>	The number of threads to use when extracting data. The default is the number of cores of your machine.
-q,--quiet	Do not update progress
-t,--hbase_table <TABLE>	HBase table to ingest the data into.

The data is populated into an HBase table called enrichment.

3. Verify that the logs are properly ingested to HBase:

```
hbase shell
scan 'threatintel'
```

You should see a configuration for the sensor that looks something like the following:

4. Generate some data to populate the Metron dashboard.

Bulk Loading Threat Intelligence Sources Using STIX/TAXII

Hortonworks Cybersecurity Platform (HCP) is designed to work with STIX/TAXII threat feeds. OpenTAXII is a Python implementation of TAXII services that delivers a rich feature set and friendly pythonic API.

TAXII loader is a stand-alone Java application that never stops streaming cyber threat intelligence information. You can use the shell script $METRON_HOME/bin/threatintel_taxii_load.sh to poll a Taxii server for STIX documents and ingest them into HBase.

It is common for the Taxii server to be an aggregation server such as Soltra Edge.

To set up an opentaxii service on HCP, you must deploy an opentaxii role, create an extractor file and a connection file, and push the feeds to HBase. In addition to the Enrichment and Extractor configs described in the following sections, this loader requires a configuration file describing the connection information to the Taxii server. The following is an example of a configuration file:

```
                        {
   "endpoint" : "http://localhost:8282/taxii-discovery-service"
  ,"username" : "guest"
  ,"password" : "guest"
  ,"type" : "DISCOVER"
  ,"collection" : "guest.Abuse_ch"
  ,"table" : "threat_intel"
  ,"columnFamily" : "cf"
  ,"allowedIndicatorTypes" : [ "domainname:FQDN", "address:IPV_4_ADDR" ]
}
```

where:

endpoint — The URL of the endpoint

type — POLL or DISCOVER depending on the endpoint.

collection — The Taxii collection to ingest

table — The HBase table to import into

columnFamily — The column family to import into

allowedIndicatorTypes — an array of acceptable threat intelligence types (see the "Enrichment Type Name" column of the Stix table above for the possibilities).

The parameters for the utility are as follows:

Short Code	Long Code	Is Required?	Description
-h		No	Generate the help screen/set of options
-e	--extractor_config	Yes	JSON Document describing the extractor for this input data source
-c	--taxii_connection_config	Yes	The JSON config file to configure the connection
-p	--time_between_polls	No	The time between polling the Taxii server in milliseconds. (default: 1 hour)
-b	--begin_time	No	Start time to poll the Taxii server (all data from that point will be gathered in the first pull). The format for the date is yyyy-MM-dd HH:mm:ss

Short Code	Long Code	Is Required?	Description
-l	--log4j	No	The Log4j Properties to load
-n	--enrichment_config	No	The JSON document describing the enrichments to configure. Unlike other loaders, this is run first if specified.

Fetch Hail a TAXII Feeds

After you install your TAXII provider, you must fetch the latest Hail a TAXII feeds into the TAXII server. Hail a TAXII.com is a repository of Open Source Cyber Threat intelligence feeds in STIX format.

Before you begin

Set up your TAXII provider. Refer to your TAXII provider documentation for more information.

Procedure

1. Fetch the lastest Hail a TAXII feeds into the TAXII server:

```
service opentaxii sync <service-name> [YYYY-MM-DD]
For example:
service opentaxii sync guest.phishtank_com
service opentaxii sync guest.Abuse_ch 2016-08-01
```

   Note: The date (YYYY-MM-DD) indicates the time from when the threat intel feeds is to be pulled. If not suffixed, then the sync command picks up feeds available for the current day.

2. Repeate Step 1 for all subscribed services.

Configure TAXII Extractor Configuration File

After you fetch the latest Hail a TAXII feeds to the TAXII server, you must create an extractor configuration file to bulk load the threat intelligence enrichment store into HBase.

Procedure

1. Log in as root to the host on which Metron is installed.

```
sudo -s $METRON_HOME
```

2. Determine the schema of the threat intelligence source.

   The schema of our mock enrichment source is domain|owner|registeredCountry|registeredTimestamp.

3. Create an extractor configuration file called threatintel_extractor_config_temp.json at $METRON_HOME/config and populate it with the threat intelligence source schema:

```
{
"config" : {
   "columns" : {
      "ip" : 0
   }
   ,"indicator_column" : "ip"
   ,"type" : "malicious_ip"
   ,"separator" : ","
  }
  ,"extractor" : "STIX"
}
```

4. Remove any non-ASCII invisible characters that might have been included if you copy and pasted:

   iconv -c -f utf-8 -t ascii threatintel_extractor_config_temp.json -o threatintel_extractor_config.json

Configure TAXII Connection Configuration File

In addition to the Extractor configuration file, this TAXII loader requires a configuration file describing the connection information to the TAXII server.

Procedure

1. Log in as root to the host on which Metron is installed.

```
sudo -s $METRON_HOME
```

2. Create a connection configuration file called threatintel_connection_config_temp.json at $METRON_HOME/config and populate it with the threat intelligence source schema:

```
{
  "endpoint" : "http://localhost:9000/services/discovery"
  ,"username" : "guest"
  ,"password" : "guest"
  ,"type" : "DISCOVER"
  ,"collection" : "guest.MalwareDomainList_Hostlist"
  ,"table" : "threatintel"
  ,"columnFamily" : "t"
  ,"allowedIndicatorTypes" : [ "domainname:FQDN", "address:IPV_4_ADDR" ]
}
```

where:

endpoint — The URL of the endpoint

type — POLL or DISCOVER depending on the endpoint.

collection — The Taxii collection to ingest

table — The HBase table to import into

columnFamily — The column family to import into

allowedIndicatorTypes — an array of acceptable threat intelligence types (see the "Enrichment Type Name" column of the Stix table above for the possibilities).

The parameters for the utility are as follows:

Short Code	Long Code	Is Required?	Description
-h		No	Generate the help screen/set of options
-e	--extractor_config	Yes	JSON Document describing the extractor for this input data source
-c	--taxii_connection_config	Yes	The JSON config file to configure the connection
-p	--time_between_polls	No	The time between polling the Taxii server in milliseconds. (default: 1 hour)
-b	--begin_time	No	Start time to poll the Taxii server (all data from that point will be gathered in the first pull). The format for the date is yyyy-MM-dd HH:mm:ss

Short Code	Long Code	Is Required?	Description
-l	--log4j	No	The Log4j Properties to load
-n	--enrichment_config	No	The JSON document describing the enrichments to configure. Unlike other loaders, this is run first if specified.

3. Remove any non-ASCII invisible characters that might have been included if you copy and pasted:
   iconv -c -f utf-8 -t ascii threatintel_connection_config_temp.json -o threatintel_connection_config.json

Push Hail a TAXII Feeds to HBase

After you create the extractor configuration and connection configuration files, you can push the Hail a TAXII feeds from the TAXII server into HBase.

Procedure

1. Push the Hail a TAXII feeds from the TAXII server into HBase:

```
$METRON_HOME/bin/threatintel_taxii_load.sh -b <START_TIME> -c
 $METRON_HOME/config/threatintel_connection_config.json
  -e $METRON_HOME/config/threatintel_extractor_config.json -p
 <TIME_INTERVAL_MSECS>
For example:$METRON_HOME/bin/threatintel_taxii_load.sh -b "2016-08-01
 00:00:00" -c $METRON_HOME/config/threatintel_connection_config.json -e
 $METRON_HOME/config/threatintel_extractor_config.json -p 10000
```

2. Remove any non-ASCII invisible characters that might have been included if you copy and pasted:
   iconv -c -f utf-8 -t ascii threatintel_extractor_config_temp.json -o threatintel_extractor_config.json

Verify Threat Intelligence Feeds in HBase

After you push the Hail a TAXII feeds to HBASE, you should check HBase for the threat intelligence information.

Procedure

Query the HBase table to check for threat intelligence feeds:

```
echo "scan 'threatintel'" | hbase shell
```

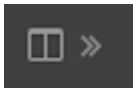
Map Fields to HBase Threat Intel by Using the Management Module

Defining the threat intelligence topology is very similar to defining the transformation and enrichment topology.

Procedure

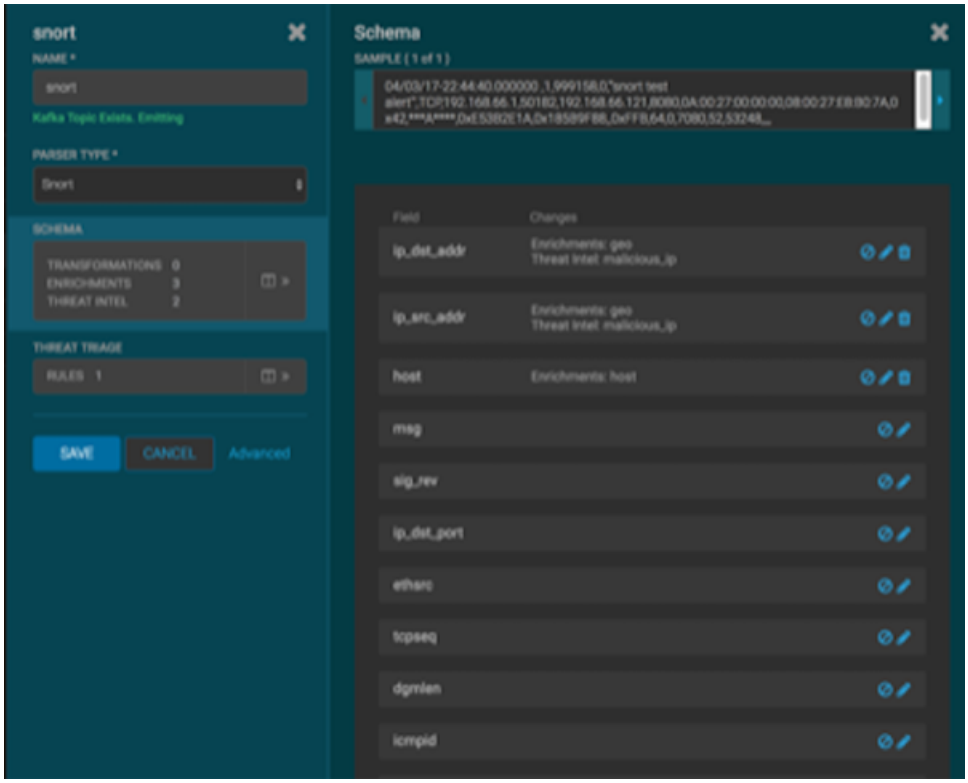
1. Select the new sensor from the list of sensors on the main window.
2. Click the pencil icon in the list of tool icons  for the new sensor.
   The Management module displays the sensor panel for the new sensor.

3. In the Schema box, click



(expand window button).

The Management module displays a second panel and populates the panel with message, field, and value information.



The Sample field, at the top of the panel, displays a parsed version of a sample message from the sensor. The Management module will test your threat intelligence against these parsed messages.

You can use the right and left arrow buttons in the Sample field to view the parsed version of each sample message available from the sensor.

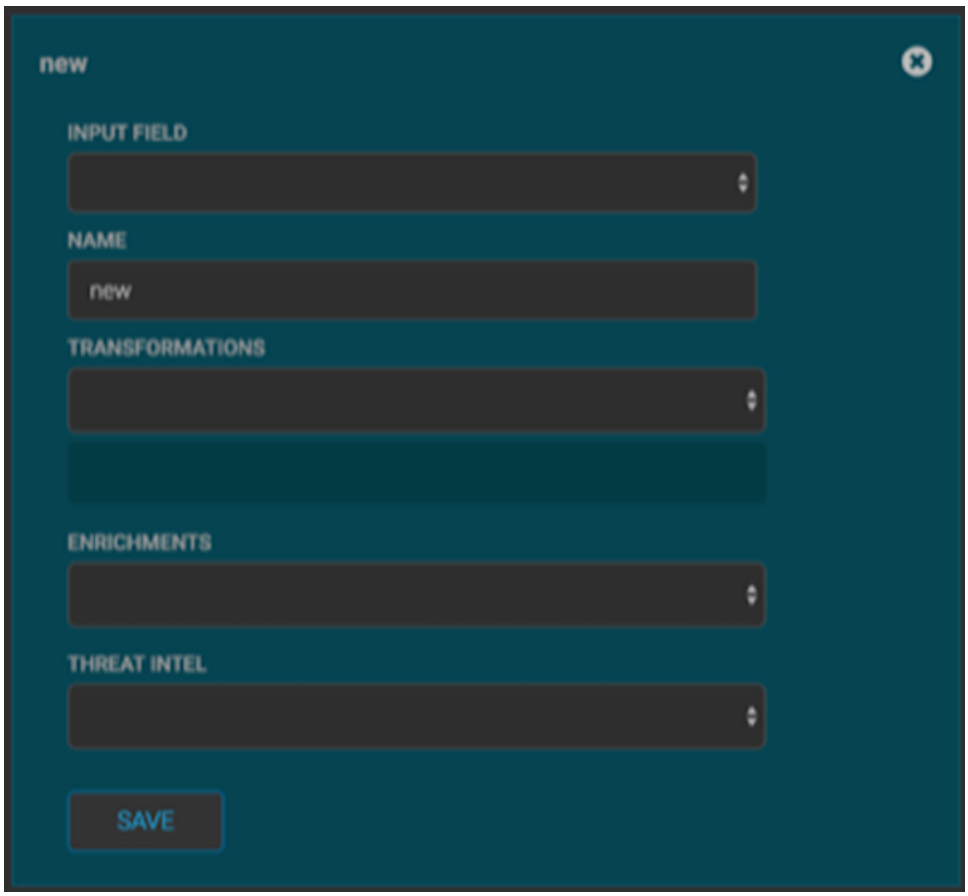
4. You can apply threat intelligence to an existing field or create a new field. Click the

(edit icon) next to a field to apply transformations to that field. Or click

(plus sign) at the bottom of the Schema panel to create new fields.

Typically users choose to create and transform a new field, rather than transforming an existing field.

For both options, the Management module expands the panel with a dialog box containing fields in which you can enter field information.



5. In the dialog box, enter the name of the new field in the **NAME** field, choose an input field from the **INPUT FIELD** box, and choose your transformation from the **THREAT INTEL** field .
6. Click SAVE to save your changes.
7. You can suppress fields from the Index by clicking

   (suppress icon).
8. Click SAVE to save the changed information.

   The Management module updates the Schema field with the number of changes applied to the sensor.

What to do next

After you have finished enriching the telemetry events, ensure that the enriched data is displaying on the Metron dashboard.

Map Fields to HBase Threat Intel by Using the CLI

Defining the threat intelligence topology is very similar to defining the transformation and enrichment topology.

Procedure

1. Edit the new data source threat intelligence configuration at $METRON_HOME/config/zookeeper/enrichments/$DATASOURCE to associate the ip_src_addr with the user enrichment.

For example:

```
{
  "index" : "squid",
  "batchSize" : 1,
  "enrichment" : {
    "fieldMap" : {
      "hbaseEnrichment" : [ "ip_src_addr" ]
    },
    "fieldToTypeMap" : {
      "ip_src_addr" : [ "whois" ]
    },
    "config" : { }
  },
  "threatIntel" : {
    "fieldMap" : { },
    "fieldToTypeMap" : { },
    "config" : { },
    "triageConfig" : {
      "riskLevelRules" : { },
      "aggregator" : "MAX",
      "aggregationConfig" : { }
    }
  },
  "configuration" : { }
}
```

2. Push this configuration to ZooKeeper:

```
$METRON_HOME/bin/zk_load_configs.sh -m PUSH -z $ZOOKEEPER_HOST:2181 -i
 $METRON_HOME/zookeeper
```

What to do next

After you have finished enriching the telemetry events, ensure that the enriched data is displaying on the Metron dashboard.

Create a Streaming Threat Intel Feed Source

Streaming intelligence feeds are incorporated slightly differently than data from a flat CSV file. Because you are defining a streaming source, you need to define a parser topology to handle the streaming data. In order to do that, you will need to create a file in $METRON_HOME/config/zookeeper/parsers/user.json.

Procedure

1. Define a parser topology to handle the streaming data:

```
touch $METRON_HOME/config/zookeeper/parsers/user.json
```

2. Populate the file with the parser topology definition.

   For example:

```
{
 "parserClassName" : "org.apache.metron.parsers.csv.CSVParser"
 ,"writerClassName" :
 "org.apache.metron.enrichment.writer.SimpleHbaseEnrichmentWriter"
 ,"sensorTopic":"user"
 ,"parserConfig":
```

```
 {
    "shew.table" : "threatintel"
   ,"shew.cf" : "t"
   ,"shew.keyColumns" : "ip"
   ,"shew.enrichmentType" : "user"
   ,"columns" : {
      "user" : 0
     ,"ip" : 1
                }
 }
}
```

where

parserClassName	The parser name.
writerClassName	The writer destination. For streaming parsers, the destination is SimpleHbaseEnrichmentWriter.
sensorTopic	Name of the sensor topic.
shew.table	The simple HBase enrichment writer (shew) table to which we want to write.
shew.cf	The simple HBase enrichment writer (shew) column family.
shew.keyColumns	The simple HBase enrichment writer (shew) key.
shew.enrichmentType	The simple HBase enrichment writer (shew) enrichment type.
columns	The CSV parser information. For our example, this information is the user name and IP address.

This file fully defines the input structure and how that data can be used in enrichment.

3. Push the configuration file to ZooKeeper:
   a) Create a Kafka topic:

```
/usr/hdp/current/kafka-broker/bin/kafka-topics.sh --create --zookeeper
 $ZOOKEEPER_HOST:2181 --replication-factor 1 --partitions 1 --topic user
```

   When you create the Kafka topic, consider how much data will be flowing into this topic.
   b) Push the configuration file to ZooKeeper.

```
$METRON_HOME/bin/zk_load_configs.sh -m PUSH -z $ZOOKEEPER_HOST:2181 -i
 $METRON_HOME/config/zookeeper
```

4. Edit the new data source enrichment configuration at $METRON_HOME/config/zookeeper/enrichments/$DATASOURCE to associate the ip_src_addr with the user enrichment.

   For example:

```
{
 "enrichment" : {
   "fieldMap" : {
     "hbaseEnrichment" : [ "ip_src_addr" ]
   },
   "fieldToTypeMap" : {
```

```
      "ip_src_addr" : [ "user" ]
    },
    "config" : { }
  },
  "threatIntel" : {
    "fieldMap" : { },
    "fieldToTypeMap" : { },
    "config" : { },
    "triageConfig" : {
      "riskLevelRules" : { },
      "aggregator" : "MAX",
      "aggregationConfig" : { }
    }
  },
  "configuration" : { }
}
```

5. Push this configuration to ZooKeeper:

```
$METRON_HOME/bin/zk_load_configs.sh -m PUSH -z $ZOOKEEPER_HOST:2181 -i
 $METRON_HOME/config/zookeeper
```