CCP Enriching Telemetry Events 2.0.1

Enriching Telemetry Events

Date of publish: 2017-11-06

https://docs.cloudera.com/

Legal Notice

Contents

Enriching Telemetry Events Overview

After the raw security telemetry events have been parsed and normalized, you need to enrich the data elements of the normalized event.

Enrichments add external data from data stores (such as HBase). CCP uses a combination of HBase, Storm, and the telemetry messages in json format to enrich the data in real time to make it relevant and consumable. You can use this enriched information immediately rather than needing to hunt in different silos for the relevant information.

CCP supports two types of configurations: global and sensor specific. The sensor specific configuration configures the individual enrichments and threat intelligence enrichments for a given sensor type (for example, squid). This section describes sensor specific configurations.

CCP provides two types of enrichment:

- Telemetry events
- Threat intelligence information

CCP provides the following telemetry enrichment sources but you can add your own enrichment sources to suit your needs:

- Asset
- GeoIP
- User

Prior to enabling an enrichment capability within CCP, the enrichment store (which for CCP is primarily HBase) must be loaded with enrichment data. The dataload utilities convert raw data sources to a primitive key (type, indicator) and value and place it in HBase.

CCP supports three types of enrichment loaders:

- Bulk load from HDFS via MapReduce
- Taxii Loader
- Flat File ingestion

For simplicity's sake, we use the bulk loader to load enrichments:

Bulk Loading Enrichment Information

Bulk loading is used to load information that does not change frequently. For example, bulk loading is ideal for loading from an asset database on a daily or even weekly basis because you don't typically change the number of assets on a network very often.

Enrichment data can be bulk loaded from the local file system, HDFS. The enrichment loader transforms the enrichment into a JSON format that is understandable to Metron. The loading framework has additional capabilities for aging data out of the enrichment stores based on time. Once the stores are loaded, an enrichment bolt that can interact with the enrichment store can be incorporated into the enrichment topology.

You can bulk load enrichment information from the following sources:

- CSV Flat File Ingestion
- HDFS via MapReduce
- Taxii Loader

OPTIONAL: Create a Mock Enrichment Source

For our runbook demonstration, we create a mock enrichment source. In your production environment you will want to use a genuine enrichment source.

Procedure

1. As root user, log into $HOST_WITH_ENRICHMENT_TAG.

   ```
   sudo -s $HOST_WITH_ENRICHMENT_TAG
   ```

2. Copy and paste the following data into a file called whois_ref.csv in $METRON_HOME/config. This CSV file represents our enrichment source.

   ```
   google.com, "Google Inc.", "US", "Dns Admin",874306800000
   work.net, "", "US", "PERFECT PRIVACY, LLC",788706000000
   capitalone.com, "Capital One Services, Inc.", "US", "Domain
    Manager",795081600000
   cisco.com, "Cisco Technology Inc.", "US", "Info Sec",547988400000
   cnn.com, "Turner Broadcasting System, Inc.", "US", "Domain Name
    Manager",748695600000
   news.com, "CBS Interactive Inc.", "US", "Domain Admin",833353200000
   nba.com, "NBA Media Ventures, LLC", "US", "C/O Domain
    Administrator",786027600000
   espn.com, "ESPN, Inc.", "US", "ESPN, Inc.",781268400000
   pravda.com, "Internet Invest, Ltd. dba Imena.ua", "UA", "Whois privacy
    protection service",806583600000
   hortonworks.com, "Hortonworks, Inc.", "US", "Domain
    Administrator",1303427404000
   microsoft.com, "Microsoft Corporation", "US", "Domain
    Administrator",673156800000
   yahoo.com, "Yahoo! Inc.", "US", "Domain Administrator",790416000000
   rackspace.com, "Rackspace US, Inc.", "US", "Domain Admin",903092400000
   1and1.co.uk, "1 & 1 Internet Ltd","UK", "Domain Admin",943315200000
   ```

   Make sure you don't have an empty newline character as the last line of the CSV file, as that will result in a null pointer exception.

Configure an Extractor Configuration File

The extractor configuration file is used to bulk load the 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 enrichment source.

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

3. Create an extractor configuration file called extractor_config.json at $METRON_HOME/config and populate it with the enrichment source schema.

   For example:

   ```
   {
    "config" : {
      "columns" : {
         "domain" : 0
         ,"owner" : 1
         ,"home_country" : 2
         ,"registrar": 3
   ```

```
        ,"domain_created_timestamp": 4
    }
    ,"indicator_column" : "domain"
    ,"type" : "whois"
    ,"separator" : ","
  }
  ,"extractor" : "CSV"
}
```

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

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

Note: The extractor_config.json file is not stored anywhere by the loader. This file is used once by the bulk loader to parse the enrichment dataset. If you would like to keep a copy of this file, be sure to save a copy to another location.

Using Stellar Properties to Transform Enrichment Data

You can transform and filter enrichment data as it is loaded into HBase by using Stellar extractor properties in the extractor configuration file. This feature is available to all extractor types. This task is optional.

Procedure

1. Transform and filter the enrichment data as it is loaded into HBase by using the following Stellar extractor properties in theextractor_config.json file at $METRON_HOME/config:

Extractor Property	Description	Example

value_transform	Transforms fields defined in the columns mapping with Stellar transformations. New keys introduced in the transform are added to the key metadata.	`"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" : "node1:2181",
    "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"
}
```

For example, if you have a top-list.csv containing the following information:

```
1,google.com
2,youtube.com
...
```

when you run a file import with the above data and extractor configuration you should receive the following results in two extracted data records:

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

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

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

Note: The extractor_config.json file is not stored anywhere by the loader. This file is used once by the bulk loader to parse the enrichment dataset. If you would like to keep a copy of this file, be sure to save a copy to another location.

3. To access properties that reside in the global configuration file, provide a ZooKeeper quorum via the zk_quorum property.

For example, if the global configuration looks like the following:

```
{
    "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" }
```

Configure Element-to-Enrichment Mapping

We now need to configure what element of a tuple should be enriched with what enrichment type. This configuration is stored in ZooKeeper.

Procedure

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

```
sudo -s $METRON_HOME
```

2. Copy and paste the following into a file called enrichment_config_temp.json at $METRON_HOME/config.

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

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

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

Run the Enrichment Loader

After the enrichment source and enrichment configuration are defined, you must run the loader to move the data from the enrichment source to the CCP enrichment store (HBase) and store the enrichment configuration in ZooKeeper.

Procedure

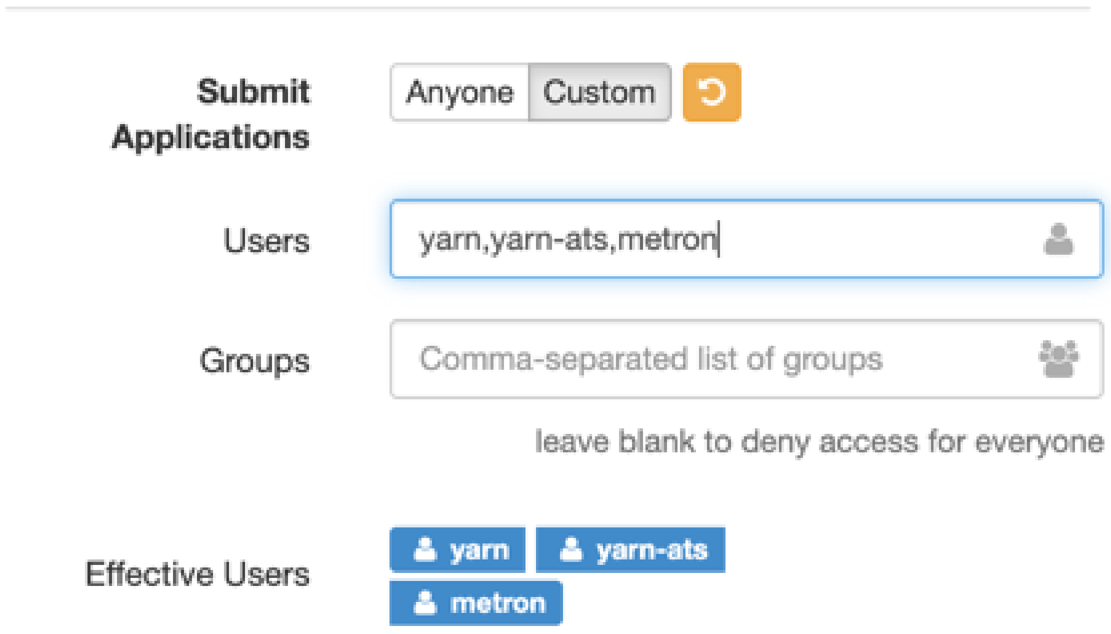
1. If you will be bulk loading enrichments using the MR mode, add the metron user to the access control list of users and groups in the Yarn Queue Manager.

   You do not need to perform this step if you use the LOCAL mode.

   a) In Ambari, click ▦ (Ambari views) and choose **Yarn Queue Manager**.
   b) In the upper left corner of the window, under **Add Queue**, select **default**.

      Ambari displays the default information on the right side of the window.
   c) By **Submit Applications**, click **Custom**.
   d) In the **Users** field, add metron.

For example:


Submit Applications: Anyone [✓ Custom]

Users: yarn,yarn-ats,metron

Groups: Comma-separated list of groups

leave blank to deny access for everyone

Effective Users: yarn, yarn-ats, metron

2. Use the loader to move the enrichment source to the enrichment store in ZooKeeper.

   Perform the following from the location containing your extractor and enrichment configuration files and your enrichment source. In our example, this information is located at $METRON_HOME/config.

   ```
   $METRON_HOME/bin/flatfile_loader.sh -n enrichment_config.json -i
    whois_ref.csv -t enrichment -c t -e
    extractor_config.json
   ```

   The parameters for the utility are as follows:

Short Code	Long Code	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.

CCP loads the enrichment data into Apache HBase and establishes a ZooKeeper mapping. The data is extracted using the extractor and configuration defined in the extractor_config.json file and populated into an HBase table called enrichment.

3. Verify that the logs were properly ingested into HBase:

   ```
   hbase shell
   scan 'enrichment'
   ```

4. Verify that the ZooKeeper enrichment tag was properly populated:

   ```
   $METRON_HOME/bin/zk_load_configs.sh -m DUMP -z $ZOOKEEPER_HOST:2181
   ```

5. Generate some data by using the Squid client to execute requests.

   a) Use ssh to access the host for Squid.

   b) Start Squid and navigate to /var/log/squid:

   ```
   sudo service squid start
   sudo su -
   cd /var/log/squid
   ```

   c) Generate some data by entering the following:

   ```
   squidclient http://www.cnn.com
   ```

Map Fields to HBase Enrichments

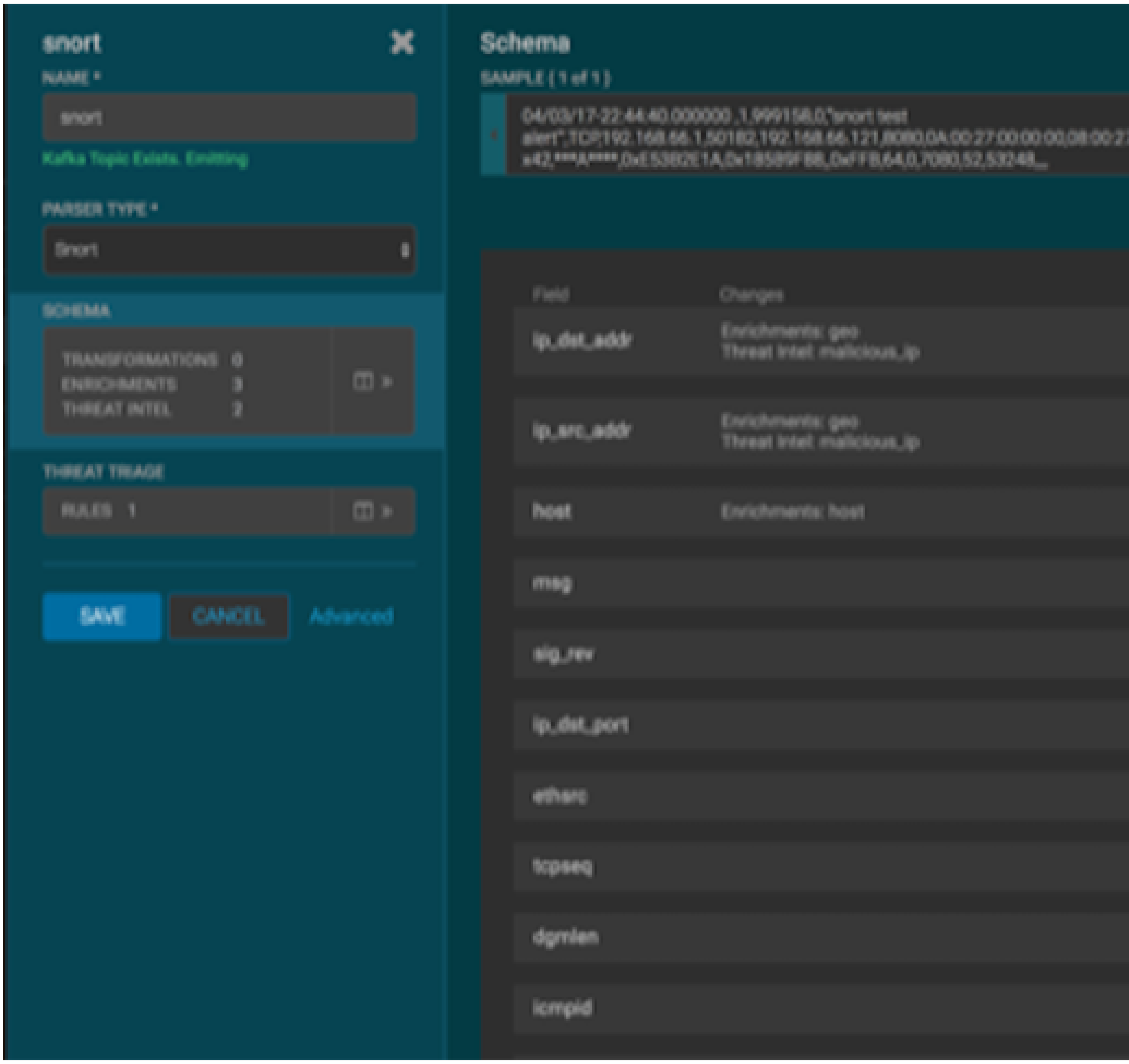
Now that you have data flowing into the HBase table, you need to ensure that the enrichment topology can be used to enrich the data flowing past. You can refine the parser output using transformation, enrichment, and threat intelligence.
Each of the parser outputs is added or modified in the Schema field.

Procedure

1. Display the Management module UI.
2. Select the new sensor from the list of sensors on the main window.
3. 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.
4. In the Schema box, click (expand window button).

   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 transformations against these parsed messages.



NAME *

snort

Kafka Topic Exists. Emitting

PARSER TYPE *

Snort

SCHEMA

TRANSFORMATIONS 0
ENRICHMENTS 3
THREAT INTEL 2

THREAT TRIAGE

RULES 1

SAMPLE (1 of 1)

```
04/03/17-22:44:40.000000 ,1,999158,0,"snort test
alert",TCP,192.168.66.1,50182,192.168.66.121,8080,0A:00:27:00:00:00,08:00:2
x42,***A****,0xE5382E1A,0x18589F88,0xFFB,64,0,7080,52,53248,,,
```

Field	Changes
ip_dst_addr	Enrichments: geo Threat Intel: malicious_ip
ip_src_addr	Enrichments: geo Threat Intel: malicious_ip
host	Enrichments: host
msg	
sig_rev	
ip_dst_port	
ethsrc	
tcpseq	
dgmlen	
icmpid	

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

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.

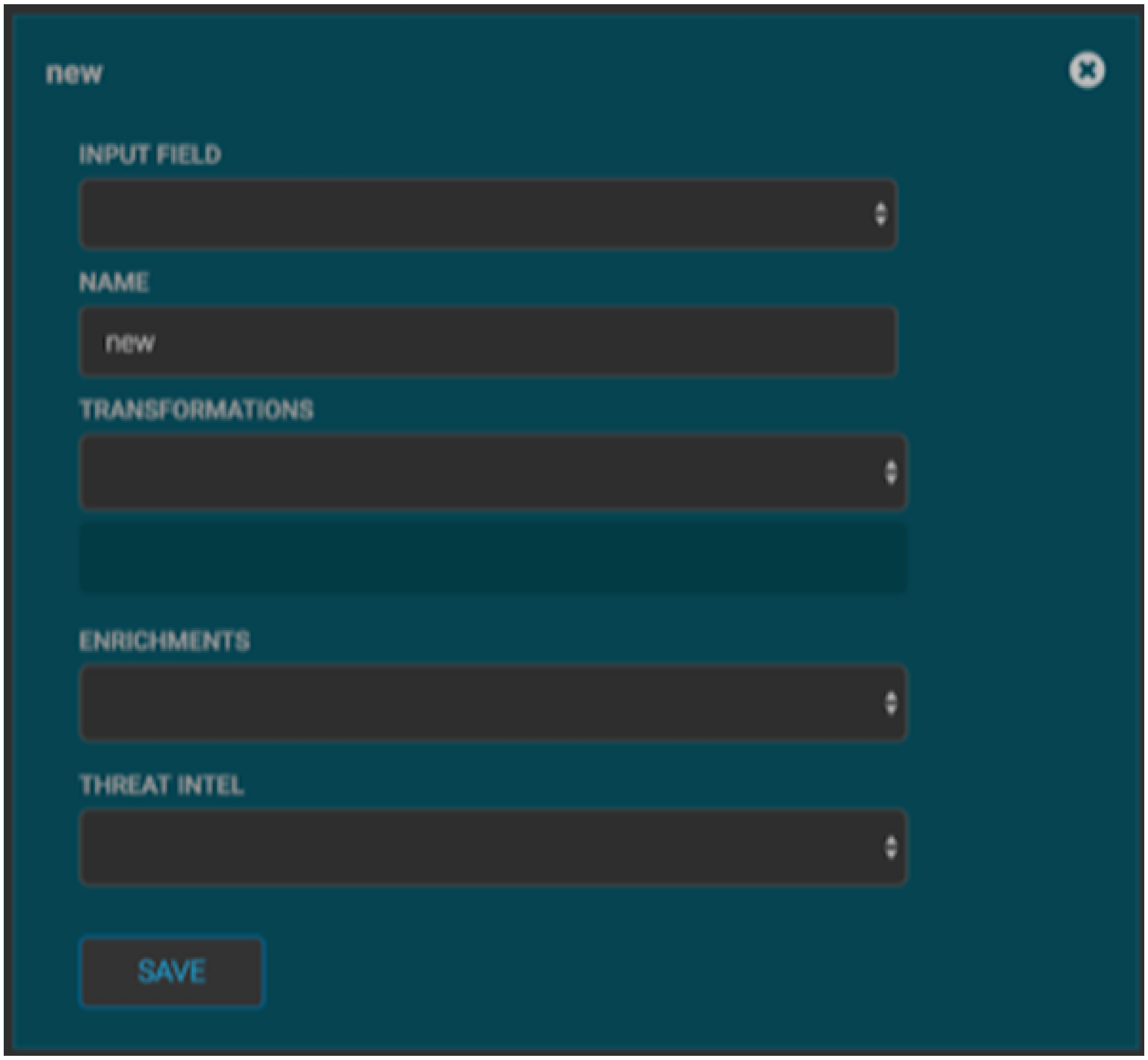
5. You can apply transformations to an existing field or create a new field. Click the (edit icon) next to a field to apply transformations to that field. Or click

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

Typically users store transformations in a new field rather than overriding existing fields.

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

New Schema Information Panel



NAME: new

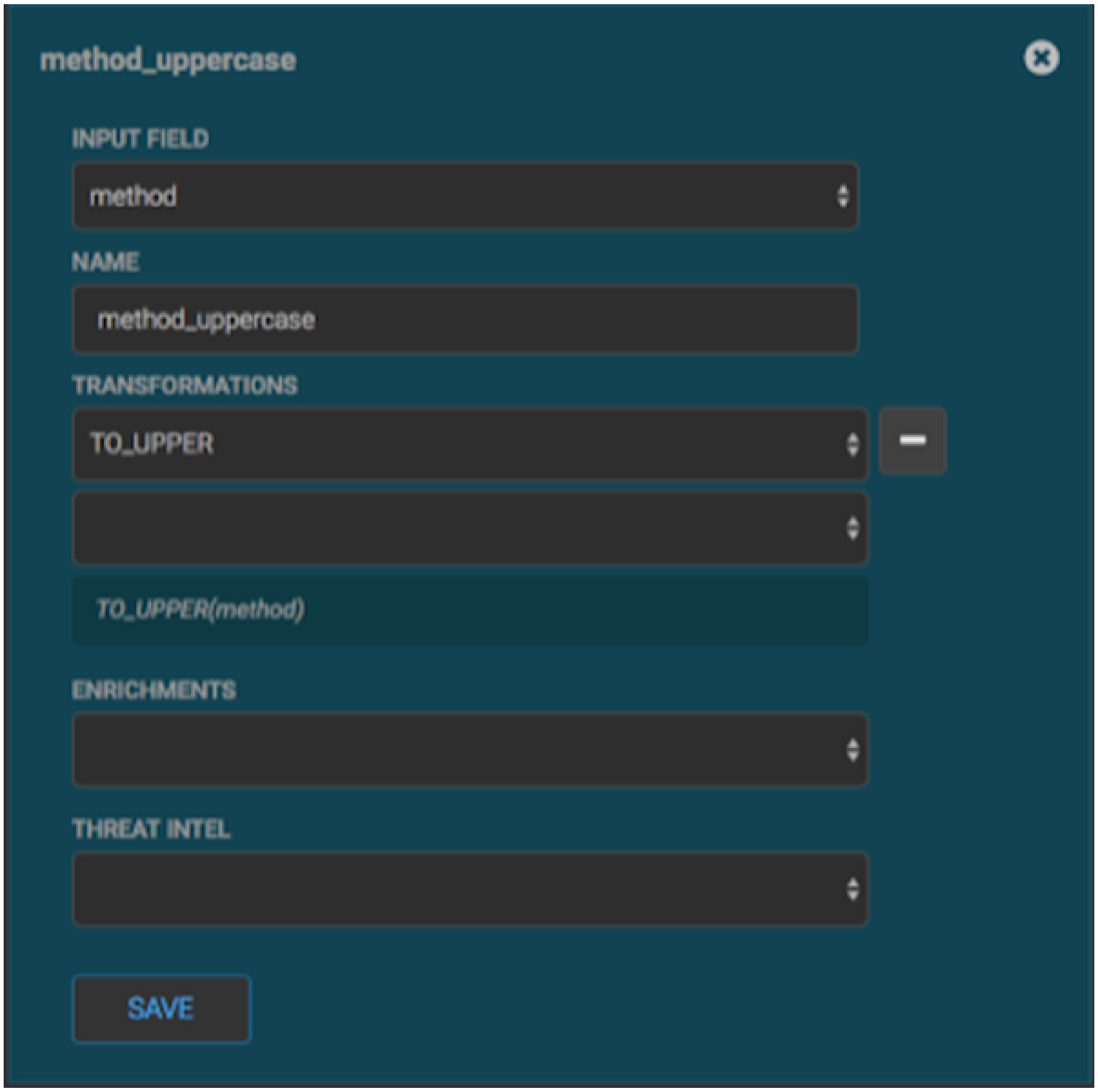
6. In the dialog box, enter the name of the new field in the **NAME** field, choose an input field from the **INPUT FIELD** box, and choose your transformation from the **TRANSFORMATIONS** field or enrichment from the **ENRICHMENTS** field.

   For example, to create a new field showing the lower case version of the method field, do the following:

   a) Enter method-uppercase in the **NAME** field.
   b) Choose method from the **INPUT FIELD**.
   c) Choose TO_UPPER in the **TRANSFORMATIONS** field.

Your new schema information panel should look like this:

Populated New Schema Information Panel



7. Click SAVE to save your changes.
8. You can suppress fields from showing in the Index by clicking (suppress icon).
9. Click SAVE to save the changed information.

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

OPTIONAL: Global Configuration

The global configuration file is a repository of properties that can be used by any configurable component in the system.

The global configuration file can be used to assign a property to multiple parser topologies. This type of enrichment can save you time by applying common enrichments to all of your sensors. The global configuration file can also be used to assign properties to enrichments and the profiler which each use a single topology. For example, you can use the global configuration to configure the enrichment topology's writer batching settings.

Verify That the Events Are Enriched

After you finish enriching your new data source, you should verify that the output matches your enrichment information. By convention, the index where the new messages are indexed is called squid_index_[timestamp] and the document type is squid_doc.

Procedure

From the Alerts UI, search the source:type filter for squid messages and ensure that they display your enrichment information.