HCP Runbook 1

Hortonworks Cybersecurity Platform

Date of Publish: 2018-08-23

http://docs.hortonworks.com

Contents

Introduction to the HCP Runbook....................4

Adding a New Telemetry Data Source....................4
- Prerequisites....................4
- Stream Data into HCP....................5
  - OPTIONAL: Install the Squid Package....................5
  - Install NiFi....................5
  - Create a NiFi Flow to Stream Events to HCP....................5
- Parse the Squid Data Source to HCP....................11
  - Parse the Squid Telemetry Event....................11
  - Verify That the Events Are Indexed....................18
  - Create an Index Template....................18
- Add New Data Source to the Metron Dashboard....................19
  - Configure a New Data Source Index in the Metron Dashboard....................19
  - Review the New Data Source Data....................21

Transform the Squid Message....................22

Enriching Telemetry Events....................24
- Bulk Loading Enrichment Information....................25
  - OPTIONAL: Create a Mock Enrichment Source....................25
  - Configure an Extractor Configuration File....................26
  - Configure Element-to-Enrichment Mapping....................28
  - Run the Enrichment Loader....................28
- Map Fields to HBase Enrichments....................29
- OPTIONAL: Global Configuration....................32
- Verify That the Events Are Enriched....................33

Enriching Threat Intelligence Information....................33
- OPTIONAL: Create a Mock Threat Intel Feed Source....................33
- Configure an Extractor Configuration File....................34
- Using Stellar Data to Transform Threat Intelligence Data....................34
- Configure Element-to-Threat Intelligence Feed Mapping....................36
- Run the Threat Intelligence Loader....................37
- Map Fields to HBase Enrichments....................39
- Verify That the Threat Intel Events Are Enriched....................42

Prioritizing Threat Intelligence....................43
- Prerequisites....................43
- Threat Triage Examples....................43
- Perform Threat Triage....................43
- View Triaged Alerts Using Kafka....................46
- View Triaged Alerts Using the Metron Dashboard....................46

Configuring Indexing.. 46
Default Configuration.. 47
Specify Index Parameters.. 47
Turn off HDFS Writer..49

Setting Up a Profile.. 49
Install Profiler.. 49
Create a Profile.. 50
Profiler Configuration Settings..53
Start the Profiler... 53
Develop Profiles..54
Testing.. 55

Triage Squid Events.. 56
Triage Squid Using the Typosquatting Algorithm...56
Improve Scoring with a Domain Whitelist... 60

Testing.. 61

Introduction to the HCP Runbook

After installing Hortonwork Cybersecurity Platform, you must set up Hortonworks Cybersecurity Platform (HCP) to monitor your data. To set up HCP you must add new telemetry data sources, enrich telemetry events, triage threat intelligence information, and ensure telemetry events are viewable in the user interface.

Unlike other HCP documentation, the Runbook provides detailed examples that are populated with information specific to the Squid data source. Squid is a caching proxy for the Web supporting HTTP, HTTPS, FTP, and more. It reduces bandwidth and improves response times by caching and reusing frequently-requested web pages.

The following sections guide you through how to add the Squid telemetry to HCP. The following procedures use the Management module UI whenever possible. If you would like to see these steps performed using the CLI, see the HCP Administration Guide.

Related Information

squid-cache.org

Adding a New Telemetry Data Source

Part of customizing your Hortonworks Cybersecurity Platform (HCP) configuration is adding a new telemetry data source. Before HCP can process the information from your new telemetry data source, you must use one of the telemetry data collectors to ingest the information into the telemetry ingest buffer.

Prerequisites

Before you add a new telemetry device, you must meet both the HCP and your user requirements.

- Install HDP and HDF, and then install HCP.
- The following use case assumes the following user requirements:
  - Proxy events from Squid logs must be ingested in real-time.
  - Proxy logs must be parsed into a standardized JSON structure suitable for analysis by Metron.
  - In real-time, the Squid proxy events must be enriched so that the domain names contain the IP information.
  - In real-time, the IP within the proxy event must be checked against for threat intelligence feeds.
  - If there is a threat intelligence hit, an alert must be raised.
  - The SOC analyst must be able to view new telemetry events and alerts from Squid.
  - HCP must profile the Squid data and incorporate statistical features of the profile into the threat triage rules.
  - HCP must be able to deploy a machine learning model that derives additional insights from the stream.
  - HCP must incorporate user's machine learning model into user's threat triage rule along with threat intel, static rules, and statistical rules.
  - Set HCP values.

    When you install HCP, you will set up several hosts. You will need the locations of these hosts, along with port numbers, and the Metron version. These values are listed below.

    - KAFKA_HOST = The host where a Kafka broker is installed.
    - ZOOKEEPER_HOST = The host where a ZooKeeper server is installed.
    - PROBE_HOST = The host where your sensor, probes are installed. If don't have any sensors installed, pick the host where a Storm supervisor is running.
    - SQUID_HOST = The host where you want to install SQUID. If you don't care, install SQUID on the PROBE_HOST.

- NIFI_HOST = Host where you will install NIFI. This should be the same host on which you installed Squid.
- HOST_WITH_ENRICHMENT_TAG = The host in your inventory hosts file that you put under the group "enrichment."
- SEARCH_HOST = The host where you have Elastic or Solr running. This is the host in your inventory hosts file that you put under the group "search". Pick one of the search hosts.
- SEARCH_HOST_PORT = The port of the search host where indexing is configured (for example, 9300).
- METRON_UI_HOST = The host where your Metron UI web application is running. This is the host in your inventory hosts file that you put under the group "web."
- METRON_VERSION = The release of the Metron binaries you are working with (for example, 0.2.0BETA-RC2).

Stream Data into HCP

The first step in adding a new data source telemetry is to stream all raw events from the telemetry data source into its own Kafka topic.

OPTIONAL: Install the Squid Package

Prior to adding a new telemetry data source, you must install it. If you have already installed your data source and have a log source, you can skip this section and move to the next one. All procedures in the Runbook use the Squid telemetry data source.

Procedure

1. ssh into $SQUID_HOST,
2. Install and start Squid:

```
sudo yum install squid
sudo service squid start
```

Install NiFi

NiFi is built to automate the flow of data between systems. As a result, NiFi works well to collect, ingest, and push data to HCP. If you haven't already done so, install NiFi.

Important:

NiFi cannot be installed on top of HDP, so you must install NiFi manually to use it with HCP.

Related Information

NiFi Command Line Installation

Create a NiFi Flow to Stream Events to HCP

You can use NiFi to create a data flow to capture events from Squid and push them into HCP. For this task we will use NiFi to create two processors, one TailFile processor that will ingest Squid events and one PutKafka processor that will stream the information to Metron. When we create and define the PutKafka processor, Kafka will create a topic for the Squid data source. We'll then connect the two processors, generate some data in the Squid log, and watch the data flow from one processor to the other.

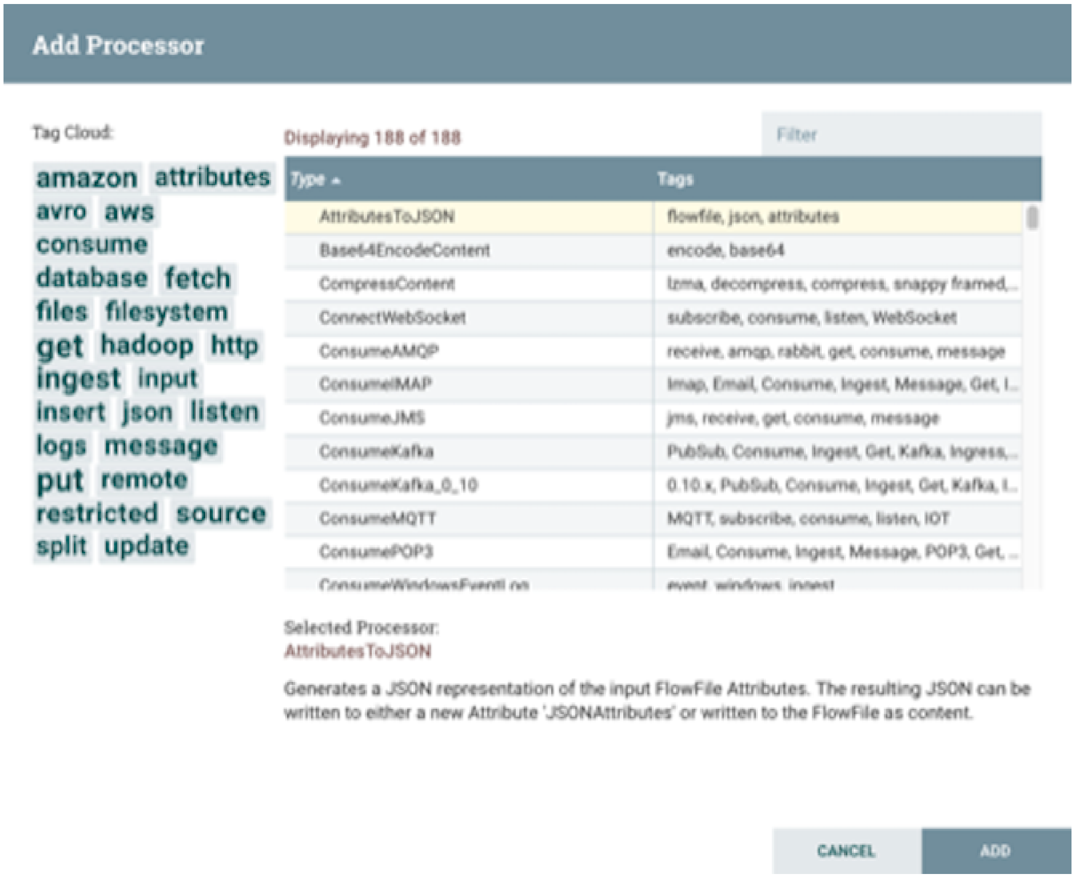
Procedure

1. Drag the first icon on the toolbar

   ⟳

   (the processor icon) to your workspace.

NiFi displays the Add Processor dialog box.



Displaying 188 of 188

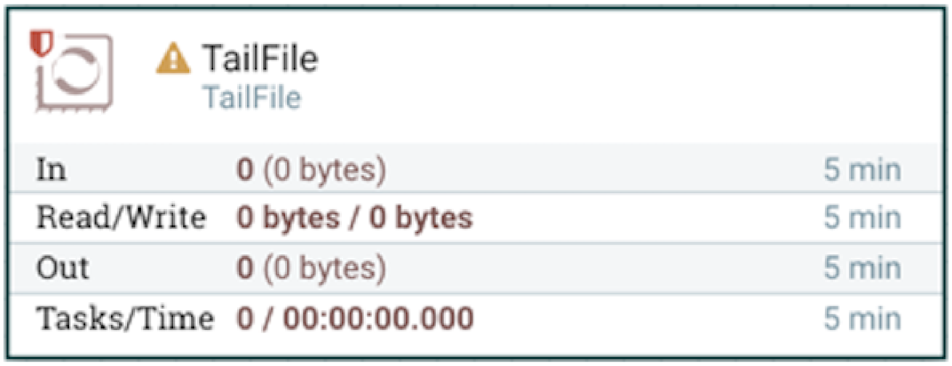
Type ▴	Tags
AttributesToJSON	flowfile, json, attributes
Base64EncodeContent	encode, base64
CompressContent	lzma, decompress, compress, snappy framed,...
ConnectWebSocket	subscribe, consume, listen, WebSocket
ConsumeAMQP	receive, amqp, rabbit, get, consume, message
ConsumeIMAP	Imap, Email, Consume, Ingest, Message, Get, I...
ConsumeJMS	jms, receive, get, consume, message
ConsumeKafka	PubSub, Consume, Ingest, Get, Kafka, Ingress,...
ConsumeKafka_0_10	0.10.x, PubSub, Consume, Ingest, Get, Kafka, I...
ConsumeMQTT	MQTT, subscribe, consume, listen, IOT
ConsumePOP3	Email, Consume, Ingest, Message, POP3, Get, ...
ConsumeWindowsEventLog	event, windows, ingest

Selected Processor:
AttributesToJSON

Generates a JSON representation of the input FlowFile Attributes. The resulting JSON can be written to either a new Attribute 'JSONAttributes' or written to the FlowFile as content.

2. Select the TailFile type of processor and click **Add**.

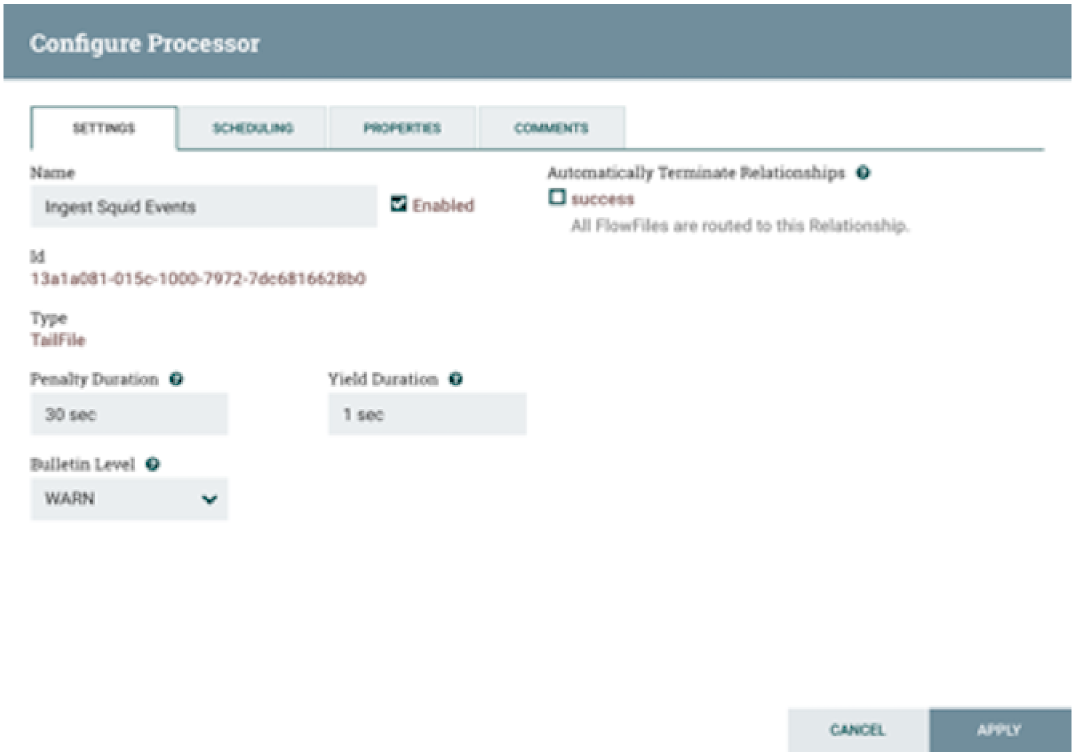
   NiFi displays a new TailFile processor.



In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

3. Right-click the processor icon and select **Configure** to display the **Configure Processor** dialog box.

   a) In the **Settings** tab, change the name to Ingest Squid Events.



Name
Ingest Squid Events

☑ Enabled

Automatically Terminate Relationships
□ success
All FlowFiles are routed to this Relationship.

Id
13a1a081-015c-1000-7972-7dc6816628b0

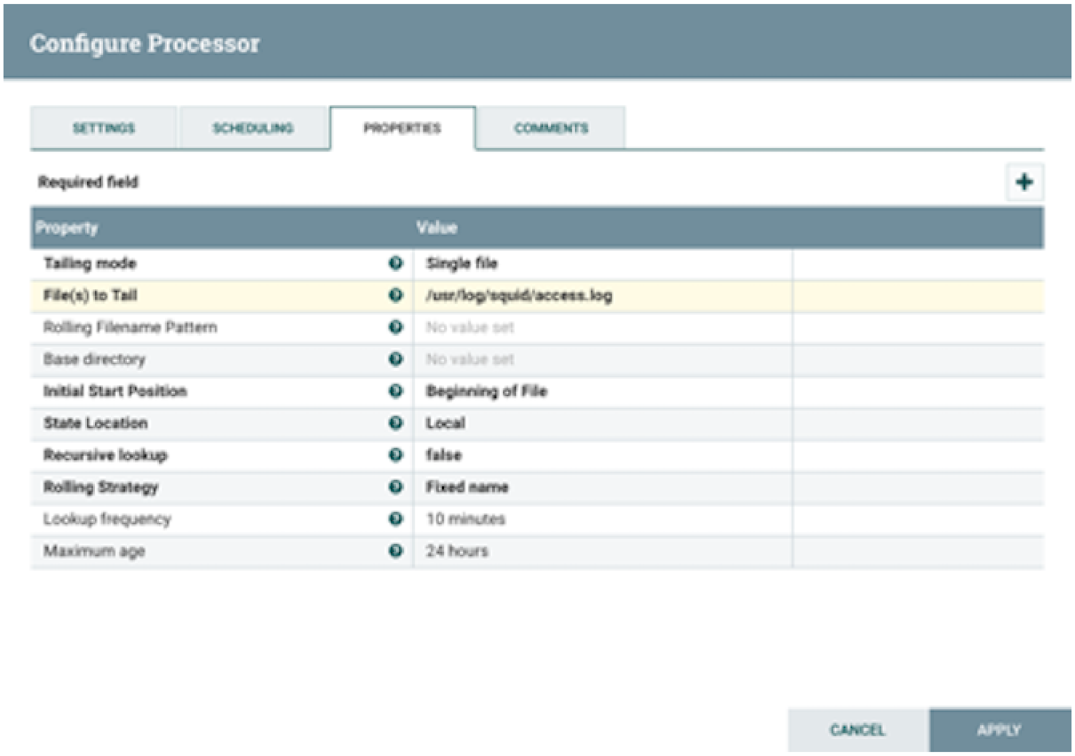
Type
TailFile

Penalty Duration: 30 sec

Yield Duration: 1 sec

Bulletin Level: WARN

b) In the **Properties** tab, enter the path to the squid access.log file in the **Value** column for the **File(s) to Tail** property.

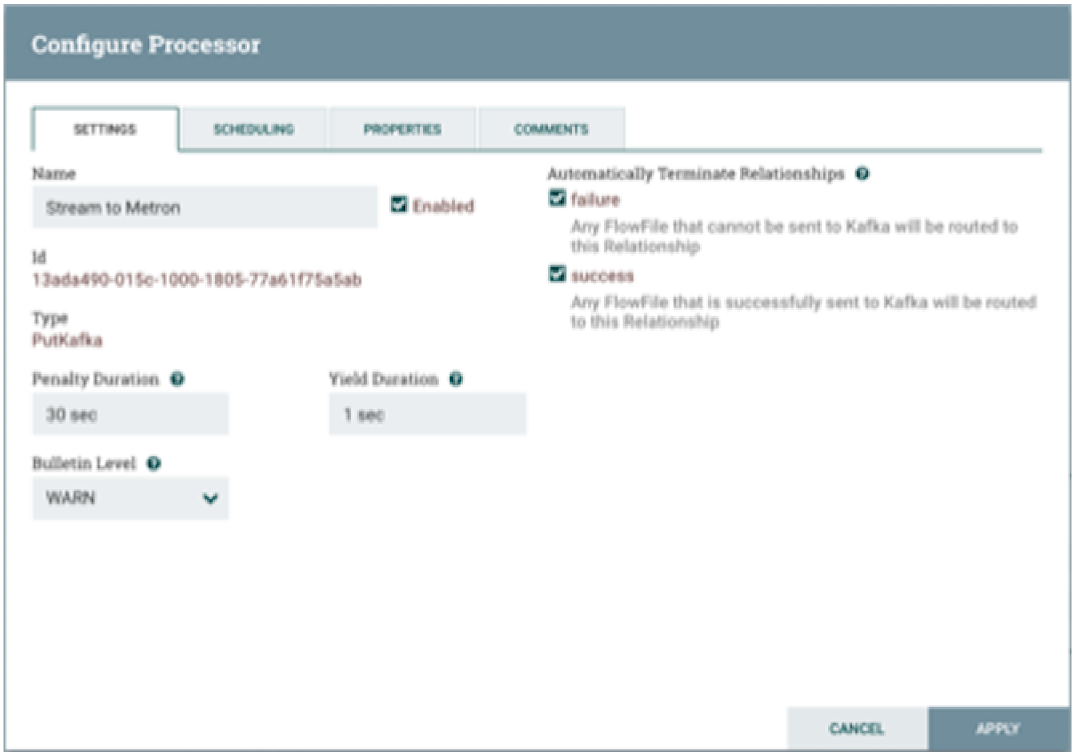


Required field

Property	Value
Tailing mode	Single file
File(s) to Tail	/usr/log/squid/access.log
Rolling Filename Pattern	No value set
Base directory	No value set
Initial Start Position	Beginning of File
State Location	Local
Recursive lookup	false
Rolling Strategy	Fixed name
Lookup frequency	10 minutes
Maximum age	24 hours

4. Click **Apply** to save your changes and dismiss the **Configure Processor** dialog box.
5. Add another processor by dragging the **Processor** icon to the main window.
6. Select the **PutKafka** type of processor and click **Add**.
7. Right-click the processor and select **Configure**.

8. In the Settings tab, change the name to Stream to Metron and then select the Automatically Terminate Relationships check boxes for **Failure** and **Success**.



Configure Processor

Name
Stream to Metron

☑ Enabled

Id
13ada490-015c-1000-1805-77a61f75a5ab

Type
PutKafka

Penalty Duration
30 sec

Yield Duration
1 sec

Bulletin Level
WARN

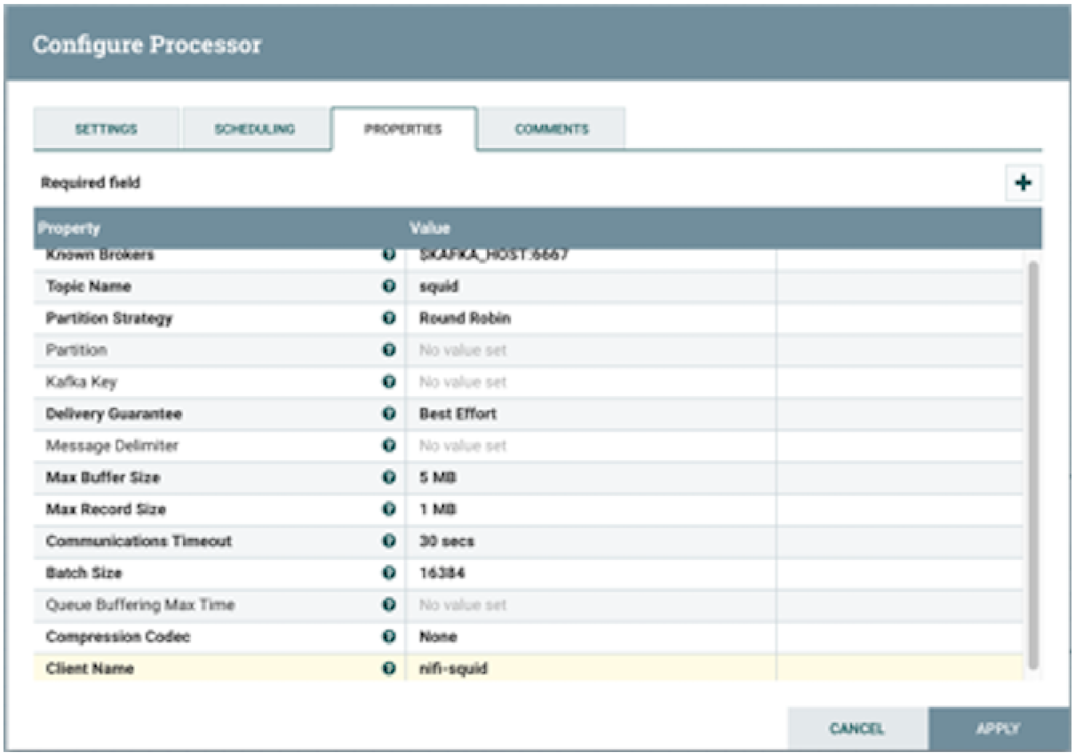
Automatically Terminate Relationships

☑ failure
Any FlowFile that cannot be sent to Kafka will be routed to this Relationship

☑ success
Any FlowFile that is successfully sent to Kafka will be routed to this Relationship

9. In the Properties tab, set the following three properties:

- Known Brokers: $KAFKA_HOST:6667
- Topic Name: squid
- Client Name: nifi-squid



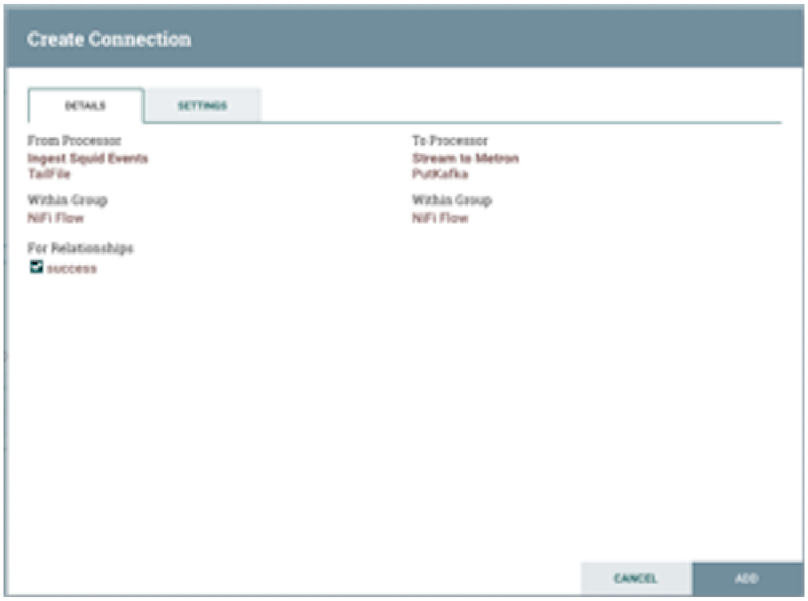
Configure Processor

Required field

Property	Value
Known Brokers	$KAFKA_HOST:6667
Topic Name	squid
Partition Strategy	Round Robin
Partition	No value set
Kafka Key	No value set
Delivery Guarantee	Best Effort
Message Delimiter	No value set
Max Buffer Size	5 MB
Max Record Size	1 MB
Communications Timeout	30 secs
Batch Size	16384
Queue Buffering Max Time	No value set
Compression Codec	None
Client Name	nifi-squid

10. Click **Apply** to save your changes and dismiss the **Configure Processor** dialog box.
11. Create a connection by dragging the arrow from the Ingest Squid Events processor to the Stream to Metron processor.

    NiFi displays a **Create Connection** dialog box.



Create Connection

DETAILS | SETTINGS

From Processor
Ingest Squid Events
TailFile

To Processor
Stream to Metron
PutKafka

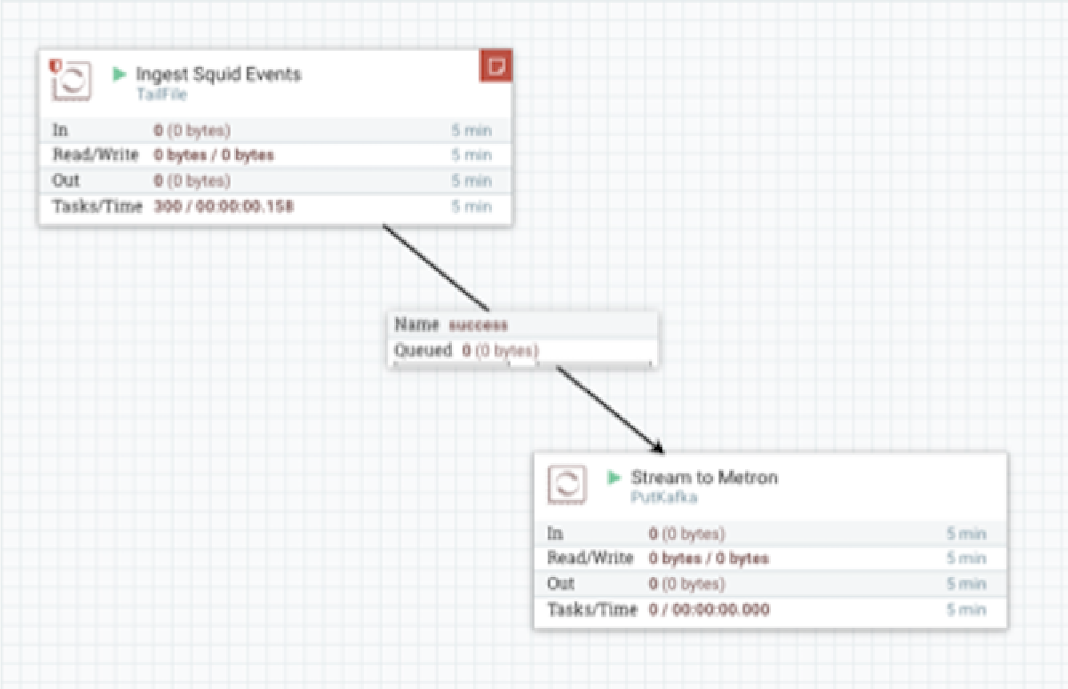
Within Group
NiFi Flow

Within Group
NiFi Flow

For Relationships
☑ success

CANCEL | ADD

12. Click **Add** to accept the default settings for the connection.
13. Press the Shift key and draw a box around both parsers to select the entire flow.



Ingest Squid Events
TailFile

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	300 / 00:00:00.158	5 min

Name success
Queued 0 (0 bytes)

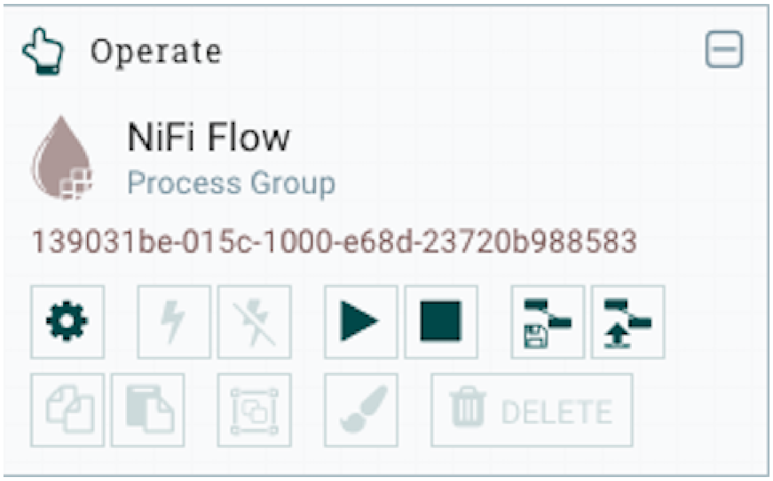
Stream to Metron
PutKafka

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

14. Click

    ▶

    (Start button) in the **Operate** panel.



15. Generate some data using the new data processor client.

a) Use ssh to access the host for the new data source.

b) With Squid started, look at the different log files that get created:

```
sudo su -
cd /var/log/squid
ls
```

The file you want for Squid is the access.log, but another data source might use a different name.

c) Generate entries for the log so you can see the format of the entries.

```
squidclient -h 127.0.0.1 "http://www.atmape.ru"
```

You will see the following data in the access.log file.

```
1481143984.330   1111 127.0.0.1 TCP_MISS/301 714 GET http://
www.atmape.ru/ - DIRECT/212.109.217.71 text/html
```

d) Using the Squid log entries, you can determine the format of the log entries is:

```
timestamp | time elapsed | remotehost | code/status | bytes | method |
 URL rfc931 peerstatus/peerhost | type
```

You should see metrics on the processor of data being pushed into Metron.

16. Look at the Storm UI for the parser topology and you should see tuples coming in.

a) Navigate to Ambari UI.

b) From the **Quick Links** pull-down menu in the upper center of the window, select **Storm UI**.

17. Before leaving this section, run the following commands to fill the access.log with data. You'll need this data when you enrich the telemetry data.

```
squidclient -h 127.0.0.1 "http://www.aliexpress.com/af/shoes.html?
ltype=wholesale&d=y&origin=n&isViewCP=y&catId=0&initiative_id=SB_20160622082445&Searc
squidclient -h 127.0.0.1 "http://www.help.1and1.co.uk/domains-c40986/
transfer-domains-c79878"
squidclient -h 127.0.0.1 "http://www.pravda.ru/science/"
squidclient -h 127.0.0.1 "http://
www.brightsideofthesun.com/2016/6/25/12027078/anatomy-of-a-deal-phoenix-
suns-pick-bender-chriss"
squidclient -h 127.0.0.1 "https://www.microsoftstore.com/store/msusa/
en_US/pdp/Microsoft-Band-2-Charging-Stand/productID.329506400"
squidclient -h 127.0.0.1 "https://tfl.gov.uk/plan-a-journey/"
squidclient -h 127.0.0.1 "https://www.facebook.com/Africa-Bike-
Week-1550200608567001/"
squidclient -h 127.0.0.1 "http://www.ebay.com/itm/02-Infiniti-QX4-Rear-
spoiler-Air-deflector-Nissan-Pathfinder-/172240020293?fits=Make%3AInfiniti
%7CModel%3AQX4&hash=item281a4e2345:g:iMkAAOSwoBtW4Iwx&vxp=mtr"
squidclient -h 127.0.0.1 "http://www.recruit.jp/corporate/english/company/
index.html"
squidclient -h 127.0.0.1 "http://www.lada.ru/en/cars/4x4/3dv/about.html"
squidclient -h 127.0.0.1 "http://www.help.1and1.co.uk/domains-c40986/
transfer-domains-c79878"
squidclient -h 127.0.0.1 "http://www.aliexpress.com/af/shoes.html?
ltype=wholesale&d=y&origin=n&isViewCP=y&catId=0&initiative_id=SB_20160622082445&Searc
```

What to do next

For more information about creating a NiFi data flow, see the NiFi documentation.

Parse the Squid Data Source to HCP

Parsers transform raw data (textual or raw bytes) into JSON messages suitable for downstream enrichment and indexing by HCP. There is one parser for each data source and the information is piped to the Enrichment/Threat Intelligence topology.

Parse the Squid Telemetry Event

Parsers transform raw data into JSON messages suitable for downstream enrichment and indexing by HCP. There is one parser for each data source and HCP pipes the information to the Enrichment/Threat Intelligence topology. You can transform the field output in the JSON messages into information and formats that make the output more useful. For example, you can change the timestamp field output from GMT to your timezone.

The following procedures use the Management module UI whenever possible. If you would like to see these steps performed using the CLI, see the HCP Administration Guide.

Procedure

1. Launch the Management UI:
   a) From the Ambari Dashboard panel, click **Metron**.

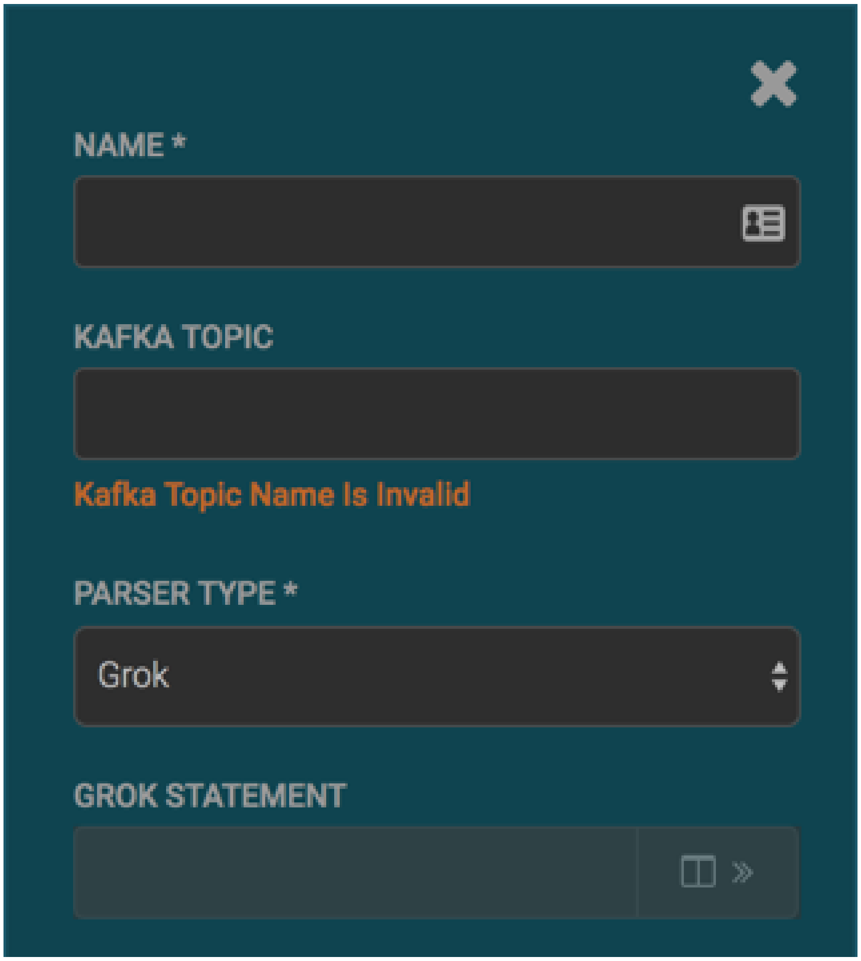
      Make sure you have the **Summary** tab selected.
   b) Select **Management UI** from **Quick Links**.

      The **Management UI** tool should display in a separate browser tab.
2. Click **Sensors** on the left side of the window, under **Operations**.
3. Click

   ✚

   (add button) in the lower right corner of the screen.

   The Management UI displays a panel used to create the new sensor.


NAME *

KAFKA TOPIC

Kafka Topic Name Is Invalid

PARSER TYPE *

Grok

GROK STATEMENT

4. Enter the following information:

```
Name: mysquid
Kafka Topic: mysquid
Parser Type: Grok
```

   If a Kafka topic already exists for the sensor name, the UI displays a message similar to **Kafka Topic Exists. Emitting** and displays the name of the **Parser Type**. You can skip to the next section, Verify Events are Indexed.

5. Create a Grok statement for the new parser:

   a) In the Grok Statement box, click the

      ◫ »

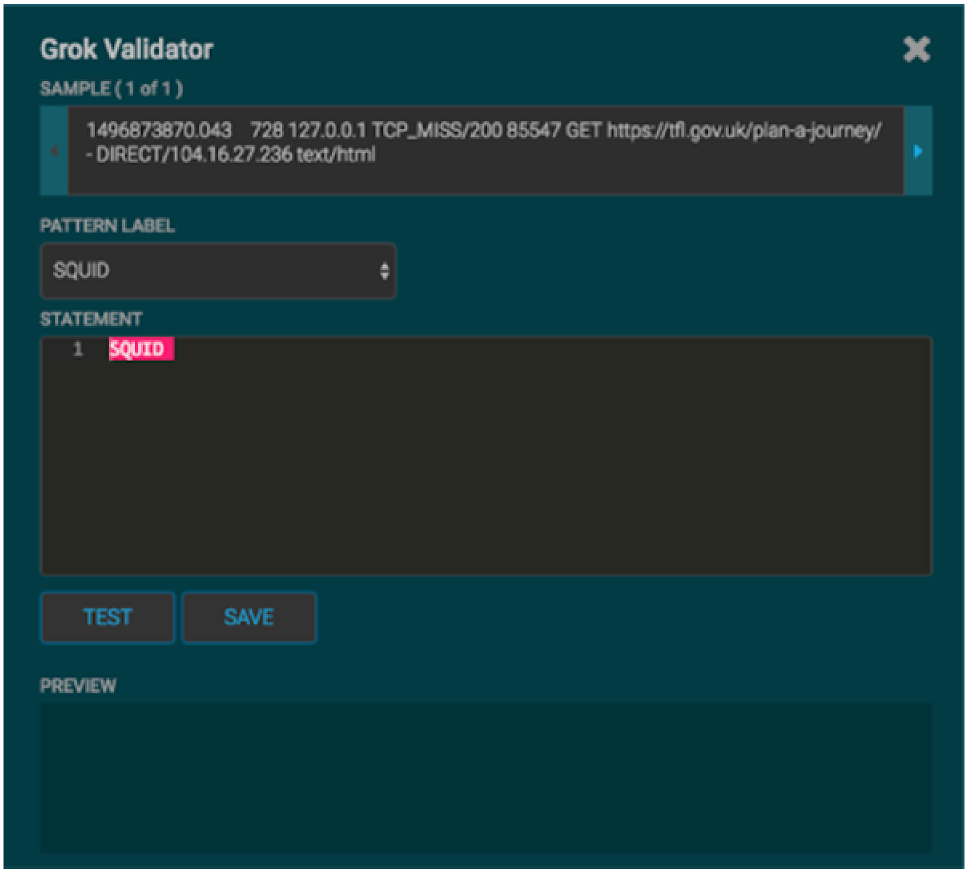
      (expand window) to display the **Grok Validator** panel.

   b) Choose or enter a name for the Grok statement in the **PATTERN LABEL** field.

      For our example, we chose MYSQUID.

   c) Enter a sample log entry for the data source.

      To create sample log entries, see Step 15 in *Create a NiFi Flow to Stream Events to HCP*.



```
1496873870.043    728 127.0.0.1 TCP_MISS/200 85547 GET https://tfl.gov.uk/plan-a-journey/
- DIRECT/104.16.27.236 text/html
```

PATTERN LABEL: SQUID

STATEMENT:

```
SQUID
```

d) Refer to the format of the log entries you determined in Step 11d in *Create a NiFi Flow to Stream Events to HCP*.

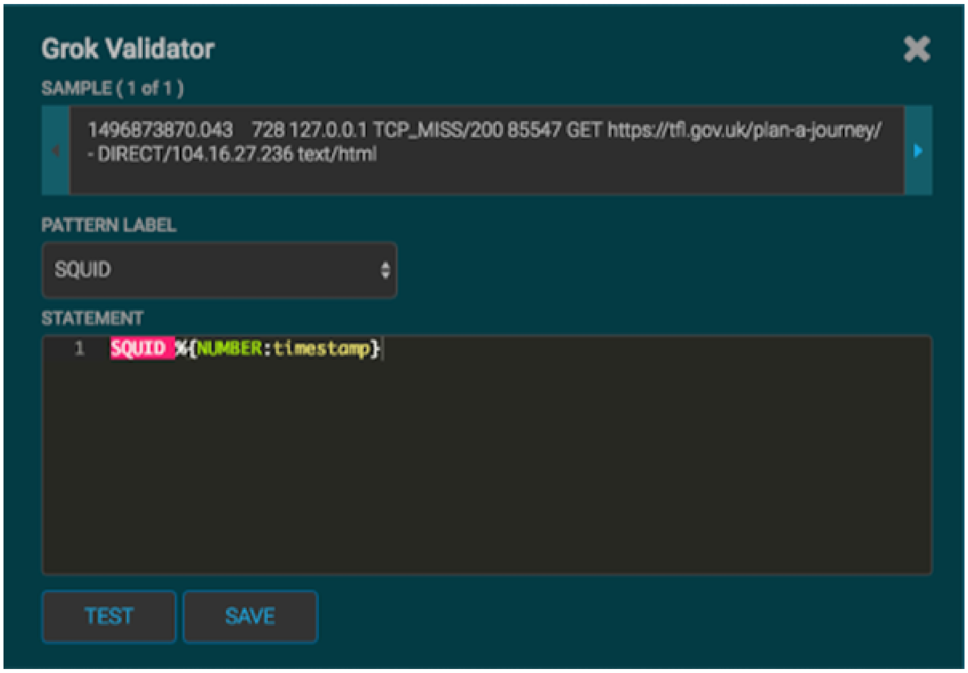
For example, the log entry format for Squid is:

```
timestamp | time elapsed | remotehost | code/status | bytes | method |
 URL rfc931 peerstatus/peerhost | type
```

e) In the **STATEMENT** field, enter the first element in the sensor log file.

For Squid, the first element in the sensor log file is timestamp which is a number, so we enter %{NUMBER:timestamp}.

> **Note:** The Management UI automatically completes partial words in your Grok statement as you enter them.



Grok Validator

SAMPLE (1 of 1)

```
1496873870.043    728 127.0.0.1 TCP_MISS/200 85547 GET https://tfl.gov.uk/plan-a-journey/
- DIRECT/104.16.27.236 text/html
```

PATTERN LABEL

SQUID

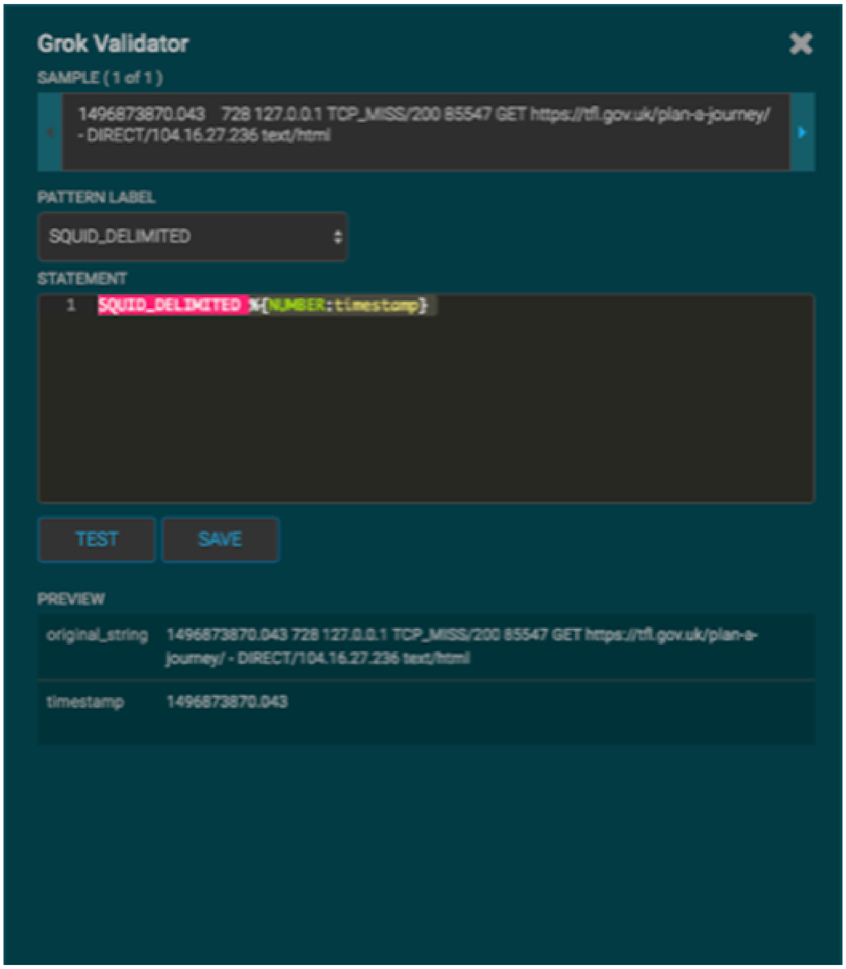
STATEMENT

```
SQUID %{NUMBER:timestamp}
```

f) Click **TEST**.

If the validator finds an error, it displays the error information. If the validation succeeds, it displays the valid mapping in the **PREVIEW** field.

Because you entered the timestamp element, the validator parses the timestamp correctly and leaves the rest of the information as random data.



SAMPLE (1 of 1)

```
1496873870.043   728 127.0.0.1 TCP_MISS/200 85547 GET https://tfl.gov.uk/plan-a-journey/
- DIRECT/104.16.27.236 text/html
```

PATTERN LABEL: SQUID_DELIMITED

STATEMENT

```
SQUID_DELIMITED %{NUMBER:timestamp}
```

PREVIEW

original_string	1496873870.043 728 127.0.0.1 TCP_MISS/200 85547 GET https://tfl.gov.uk/plan-a-journey/ - DIRECT/104.16.27.236 text/html
timestamp	1496873870.043

g) Click **Save**.

h) Verify that the sensor name and topic name are "mysquid" with NO extra spaces or special characters.

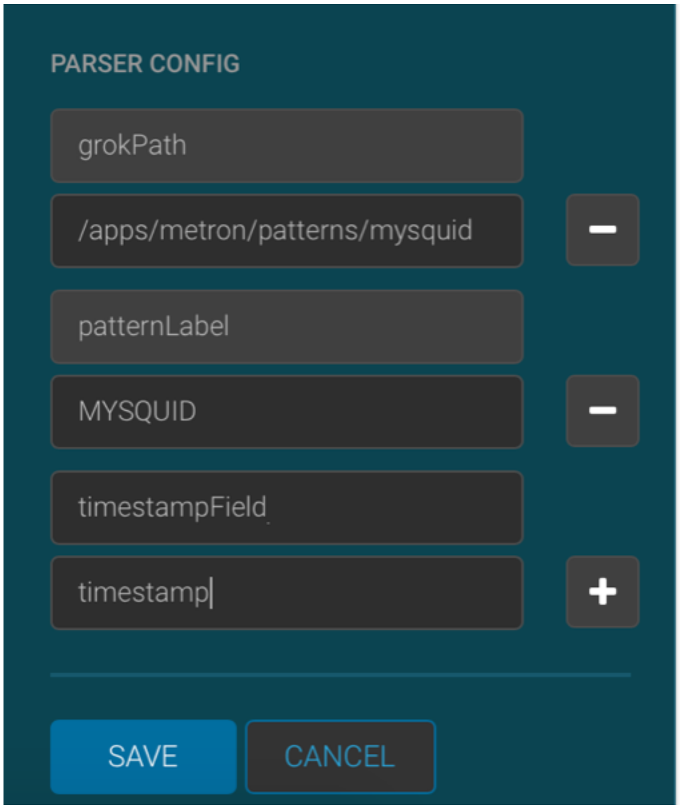
6. Click **Save** on the mysquid sensor.

The mysquid sensor appears in the Management UI. You might need to refresh your screen to see the new sensor.

7. Click the pencil icon to edit the mysquid sensor.

8. Scroll down to the **Parser Config** section.

9. In the first open field, indicated by **enter field**, enter timestampField.



PARSER CONFIG

grokPath: /apps/metron/patterns/mysquid

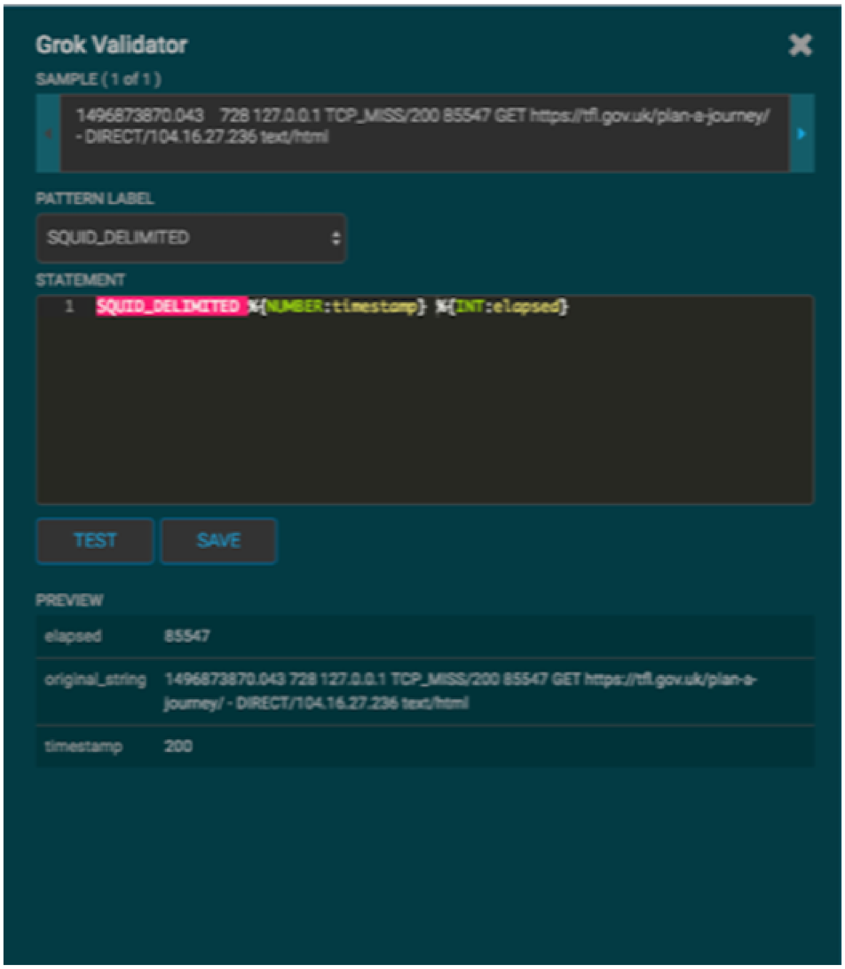
patternLabel: MYSQUID

timestampField: timestamp

10. In next open field, enter timestamp.

11. Click **Save**.

12. Continue to build and test the Grok statement until you have entries for each element in the log entry.
For example, for Squid, the second element is elapsed, so we enter %{INT:elapsed} and click **TEST** again.



SAMPLE (1 of 1)

```
1496873870.043    728 127.0.0.1 TCP_MISS/200 85547 GET https://tfl.gov.uk/plan-a-journey/
- DIRECT/104.16.27.236 text/html
```

PATTERN LABEL: SQUID_DELIMITED

STATEMENT

```
SQUID_DELIMITED %{NUMBER:timestamp} %{INT:elapsed}
```

PREVIEW

elapsed	85547
original_string	1496873870.043 728 127.0.0.1 TCP_MISS/200 85547 GET https://tfl.gov.uk/plan-a-journey/ - DIRECT/104.16.27.236 text/html
timestamp	200

13. When your Grok statement is complete and valid, click **SAVE** to save the Grok statement for the sensor.
14. Click **SAVE** to save the sensor information and add it to the list of Sensors.

    This new data source processor topology ingests from the $Kafka topic and then parses the event with the HCP Grok framework using the Grok pattern. The result is a standard JSON Metron structure that then is added to the "enrichment" Kafka topic for further processing.
15. Test that a Kafka topic has been created for the Squid parser:

    a) Navigate to the following directory:

    ```
    /usr/hdp/current/kafka-broker/bin
    ```

    b) List all of the Kafka topics:

    ```
    ./kafka-topics.sh --zookeeper localhost:2181 --list
    ```

    You should see the following list of Kafka topics:

    - bro
    - enrichments
    - ubdexubg
    - snort
    - mysquid

Verify That the Events Are Indexed

After you finish adding your new data source, you should verify that the data source events are indexed and the output matches any Stellar transformation functions you used.

Procedure

From the Alerts UI, search the source:type filter for squid messages.

By convention, the index where the new messages are indexed is called squid_index_[timestamp] and the document type is squid_doc.

Related Information

Triaging Alerts

Create an Index Template

To work with a new data source data in the Metron dashboard, you need to ensure that the data is landing in the search index (Elasticsearch) with the correct data types. You can achieve this by defining an index template.

Procedure

1. Run the following command to create an index template for the new data source.

```
curl -XPOST $SEARCH_HOST:$SEARCH_PORT/_template/$DATASOURCE_index -d '
{
  "template": "sensor1_index*",
  "mappings": {
    "sensor1_doc": {
      "properties": {
        "timestamp": {
          "type": "date",
          "format": "epoch_millis"
        },
        "ip_src_addr": {
          "type": "ip"
        },
        "ip_src_port": {
          "type": "integer"
        },
        "ip_dst_addr": {
          "type": "ip"
        },
        "ip_dst_port": {
          "type": "integer"
        }
      }
    }
  }
}'
```

The code sample defines an index template for a new sensor called 'sensor1'.

The example assumes the following:

- The template applies to any indices that are named sensor1_index*.
- The index has one document type that must be named sensor1_doc.
- The index contains timestamps.
- The properties section defines the types of each field. This example defines the five common fields that most sensors contain.
- Additional fields can be added following the five that are already defined.

By default, Elasticsearch will attempt to analyze all fields of type string. This means that Elasticsearch will tokenize the string and perform additional processing to enable free-form text search. In many cases, you want to treat each of the string fields as enumerations. This is why most fields in the index template are `not_analyzed`.

2. An index template will only apply for indices that are created after the template is created. Delete the existing indices for the new data source so that new ones can be generated with the index template.

```
curl -XDELETE $SEARCH_HOST:9200/$DATSOURCE*
```

3. Wait for the new data source index to be re-created.

   This might take a minute or two based on how fast the new data source data is being consumed in your environment.

```
curl -XGET $SEARCH_HOST:9200/$DATASOURCE*
```

Add New Data Source to the Metron Dashboard

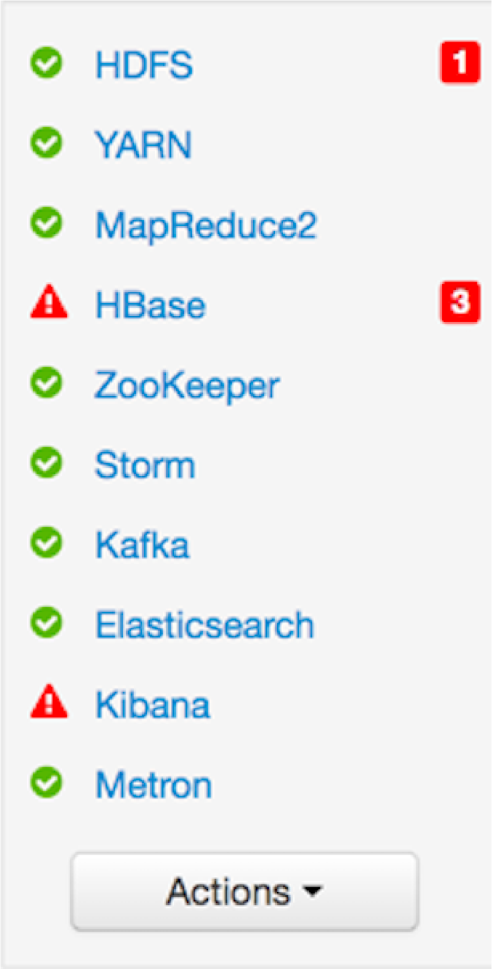
After a new data telemetry source has been added to HCP, you must add it to the Metron dashboard before you can create queries and filters for it and add telemetry panels displaying its data.

Configure a New Data Source Index in the Metron Dashboard

Now that you have an index for the new data source with all of the right data types, you need to tell the Metron dashboard about this index.

Procedure

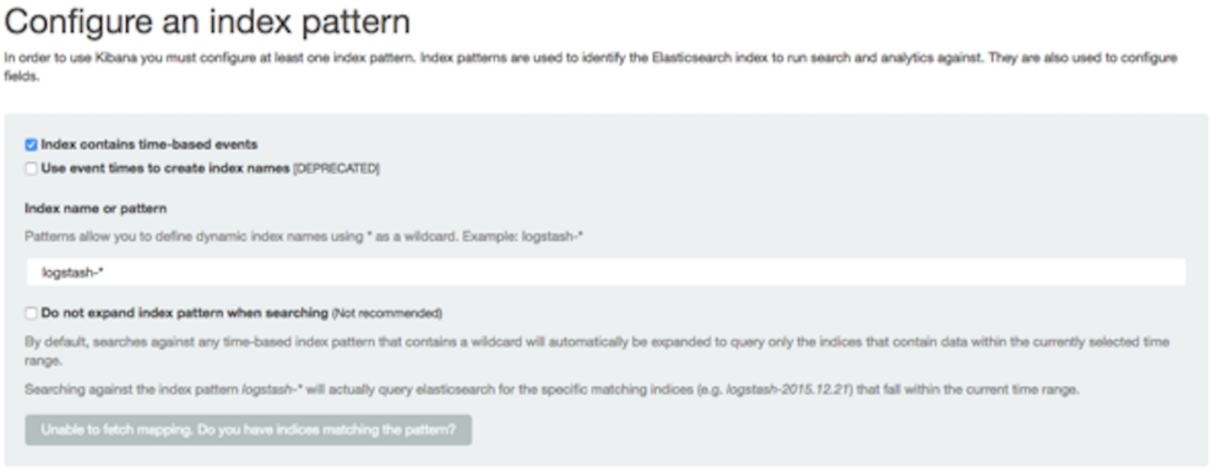
1. Launch the Metron dashboard if you have not already done so: \
   a) From Ambari, click Kibana in the list of quick tasks.



b) Select **Metron UI** from the Quick Links menu in the top center of the window.

2. Click the **Settings** tab on the Metron dashboard.
3. Make sure you have the **Indices** tab selected, then click **+Add New**.

   Kibana displays the **Configure an index pattern** window. Use the index pattern window to identify your telemetry source.



Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

☑ Index contains time-based events

□ Use event times to create index names [DEPRECATED]

Index name or pattern

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

logstash-*

□ Do not expand index pattern when searching (Not recommended)

By default, searches against any time-based index pattern that contains a wildcard will automatically be expanded to query only the indices that contain data within the currently selected time range.

Searching against the index pattern *logstash-** will actually query elasticsearch for the specific matching indices (e.g. *logstash-2015.12.21*) that fall within the current time range.

Unable to fetch mapping. Do you have indices matching the pattern?

4. In the **Index name or pattern** field, enter the name of the index pattern of your data telemetry source.

In most cases the name of the index pattern will match the sensor name. For example, the 'bro' sensor has an index pattern of 'bro-*'.

5. If your data telemetry source does not contain time-based events, clear the **Index contains time-based events** check box.

   If your data telemetry source does contain time-based events, leave the check box as is. Most of your data telemetry sources will contain time-based events.

6. Click **Create** to add the index pattern for your new data telemetry source.

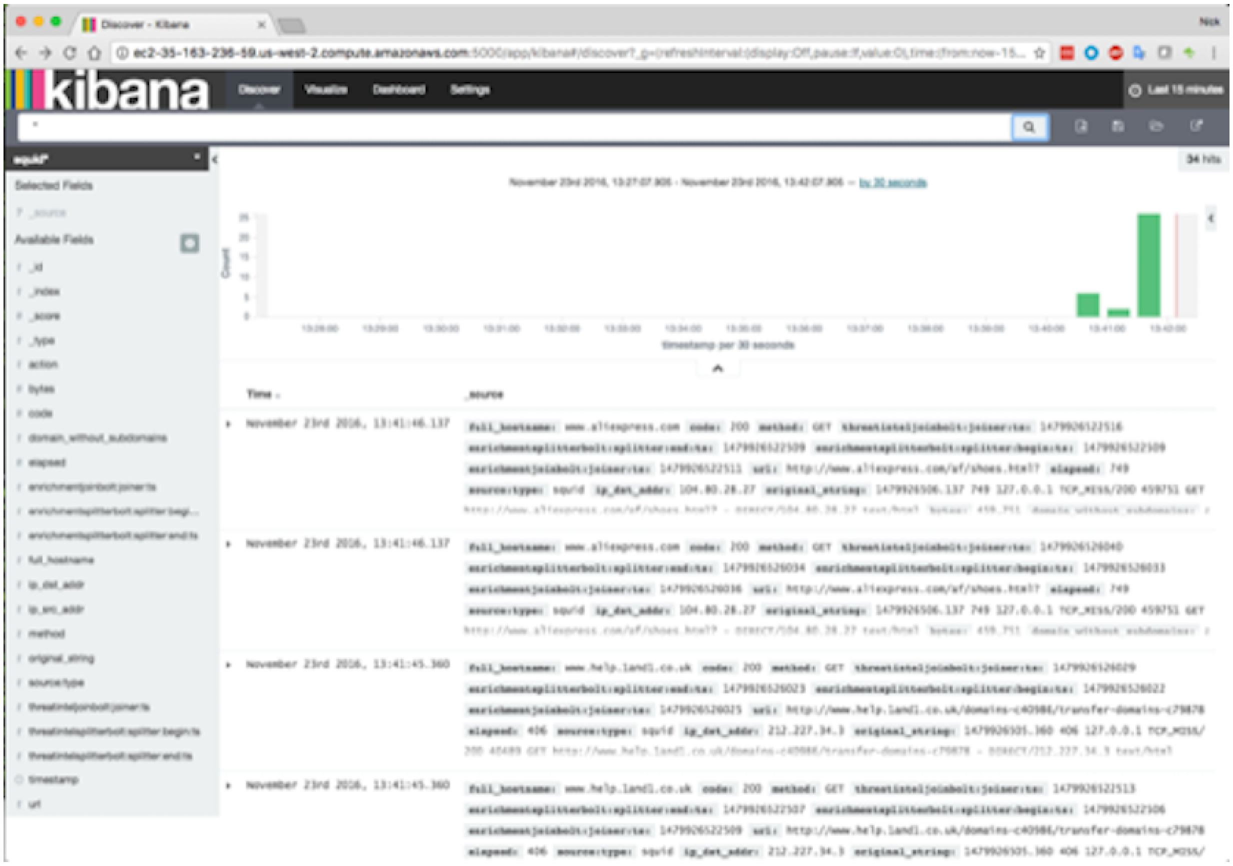
   If you would like this new index pattern to be the default, click the Green Star icon (★).

Review the New Data Source Data

Now that the Metron dashboard is aware of the new data source index, you can look at the data.

Procedure

1. Display the Metron dashboard.
2. Click on the **Discover** tab and then choose the newly created data source index pattern.
3. Click any of the fields in the left column to see a representation of the variety of data for that specific field.
4. Click the Right Facing Arrow icon next to a specific record in the center of the window (the **Document** table) to expand the record and display the available data.



Discover - Kibana
ec2-35-163-236-59.us-west-2.compute.amazonaws.com
kibana
Discover
Visualize
Dashboard
Settings
Last 15 minutes
squid*
Selected Fields
_source
Available Fields
_id
_index
_score
_type
action
bytes
code
domain_without_subdomains
elapsed
enrichmentjoinbolt:joiner:ts
enrichmentsplitterbolt:splitter:begi...
enrichmentsplitterbolt:splitter:end:ts
full_hostname
ip_dst_addr
ip_src_addr
method
original_string
source:type
threatinteljoinbolt:joiner:ts
threatintelsplitterbolt:splitter:begin:ts
threatintelsplitterbolt:splitter:end:ts
timestamp
url
34 hits
November 23rd 2016, 13:27:07.905 - November 23rd 2016, 13:42:07.905 — by 30 seconds
Count
Timestamp per 30 seconds
Time
_source

November 23rd 2016, 13:41:46.137 full_hostname: www.aliexpress.com code: 200 method: GET threatinteljoinbolt:joiner:ts: 1479926522516 enrichmentsplitterbolt:splitter:end:ts: 1479926522509 enrichmentsplitterbolt:splitter:begin:ts: 1479926522509 enrichmentjoinbolt:joiner:ts: 1479926522511 url: http://www.aliexpress.com/af/shoes.html? elapsed: 749 source:type: squid ip_dst_addr: 104.80.28.27 original_string: 1479926506.137 749 127.0.0.1 TCP_MISS/200 459751 GET http://www.aliexpress.com/af/shoes.html? - DIRECT/104.80.28.27 text/html bytes: 459,751 domain_without_subdomains:

November 23rd 2016, 13:41:46.137 full_hostname: www.aliexpress.com code: 200 method: GET threatinteljoinbolt:joiner:ts: 1479926526040 enrichmentsplitterbolt:splitter:end:ts: 1479926526034 enrichmentsplitterbolt:splitter:begin:ts: 1479926526033 enrichmentjoinbolt:joiner:ts: 1479926526036 url: http://www.aliexpress.com/af/shoes.html? elapsed: 749 source:type: squid ip_dst_addr: 104.80.28.27 original_string: 1479926506.137 749 127.0.0.1 TCP_MISS/200 459751 GET http://www.aliexpress.com/af/shoes.html? - DIRECT/104.80.28.27 text/html bytes: 459,751 domain_without_subdomains:

November 23rd 2016, 13:41:45.360 full_hostname: www.help.1and1.co.uk code: 200 method: GET threatinteljoinbolt:joiner:ts: 1479926526029 enrichmentsplitterbolt:splitter:end:ts: 1479926526023 enrichmentsplitterbolt:splitter:begin:ts: 1479926526022 enrichmentjoinbolt:joiner:ts: 1479926526025 url: http://www.help.1and1.co.uk/domains-c40986/transfer-domains-c79878 elapsed: 406 source:type: squid ip_dst_addr: 212.227.34.3 original_string: 1479926505.360 406 127.0.0.1 TCP_MISS/200 40489 GET http://www.help.1and1.co.uk/domains-c40986/transfer-domains-c79878 - DIRECT/212.227.34.3 text/html

November 23rd 2016, 13:41:45.360 full_hostname: www.help.1and1.co.uk code: 200 method: GET threatinteljoinbolt:joiner:ts: 1479926522513 enrichmentsplitterbolt:splitter:end:ts: 1479926522507 enrichmentsplitterbolt:splitter:begin:ts: 1479926522506 enrichmentjoinbolt:joiner:ts: 1479926522509 url: http://www.help.1and1.co.uk/domains-c40986/transfer-domains-c79878 elapsed: 406 source:type: squid ip_dst_addr: 212.227.34.3 original_string: 1479926505.360 406 127.0.0.1 TCP_MISS/

Transform the Squid Message

You can customize your sensor data to provide more meaningful data. For example, you can choose to transform a url to provide the domain name of the outbound connection or the IP address. To do this, you need to add transformation information.

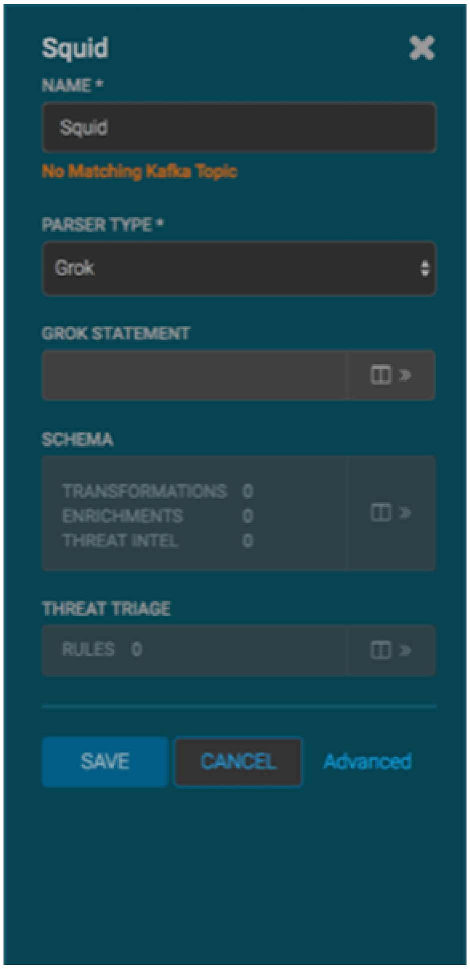
Procedure

1. In the Management module, click

   ✎

   (edit button) for your sensor.

   The Management module displays the schema panel.

   Squid

   NAME *: Squid

   No Matching Kafka Topic

   PARSER TYPE *: Grok

2. In the Schema box, click

   ◫ »

   (expand window button).

   The Management module displays the Schema panel and populates it 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 transformations against this parsed message.

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.

You can apply transformations to an existing field or create a new field. Typically users choose to create and transform a new field, rather than transforming an existing field.

3. To add a new transformation, either click the

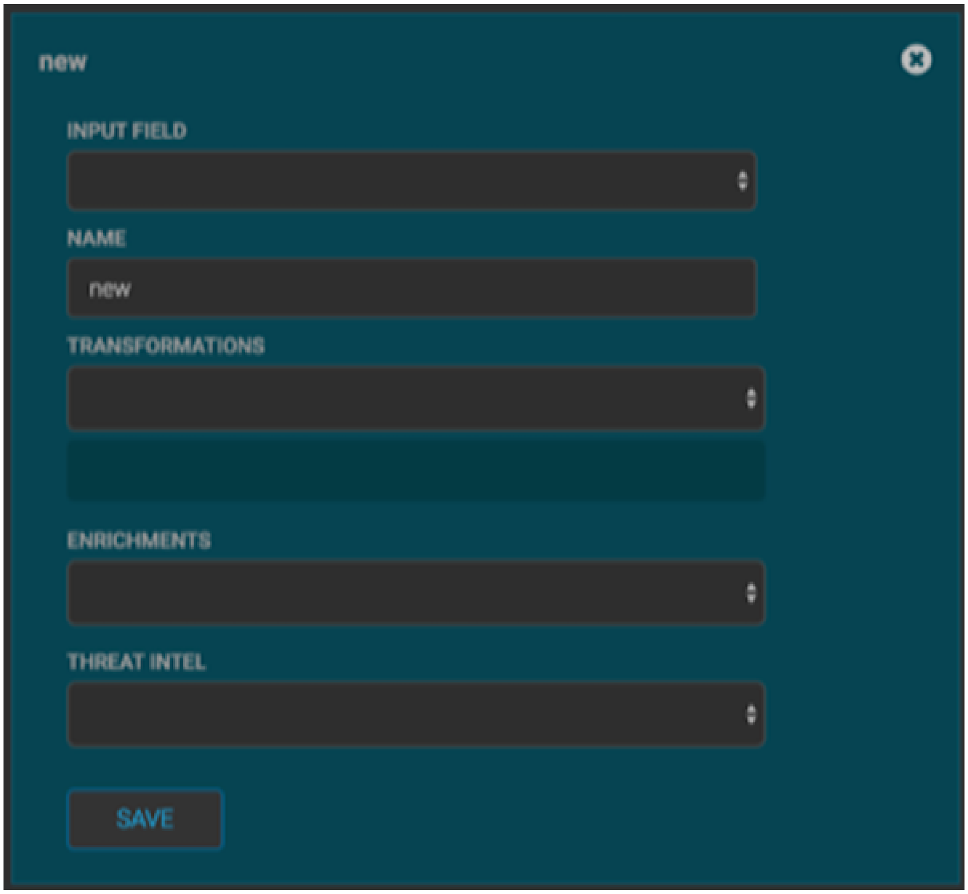
✎

next to a field or click the

+

(plus sign) at the bottom of the **Schema** panel.

The module displays a new dialog box for your transformations.



new

INPUT FIELD

NAME

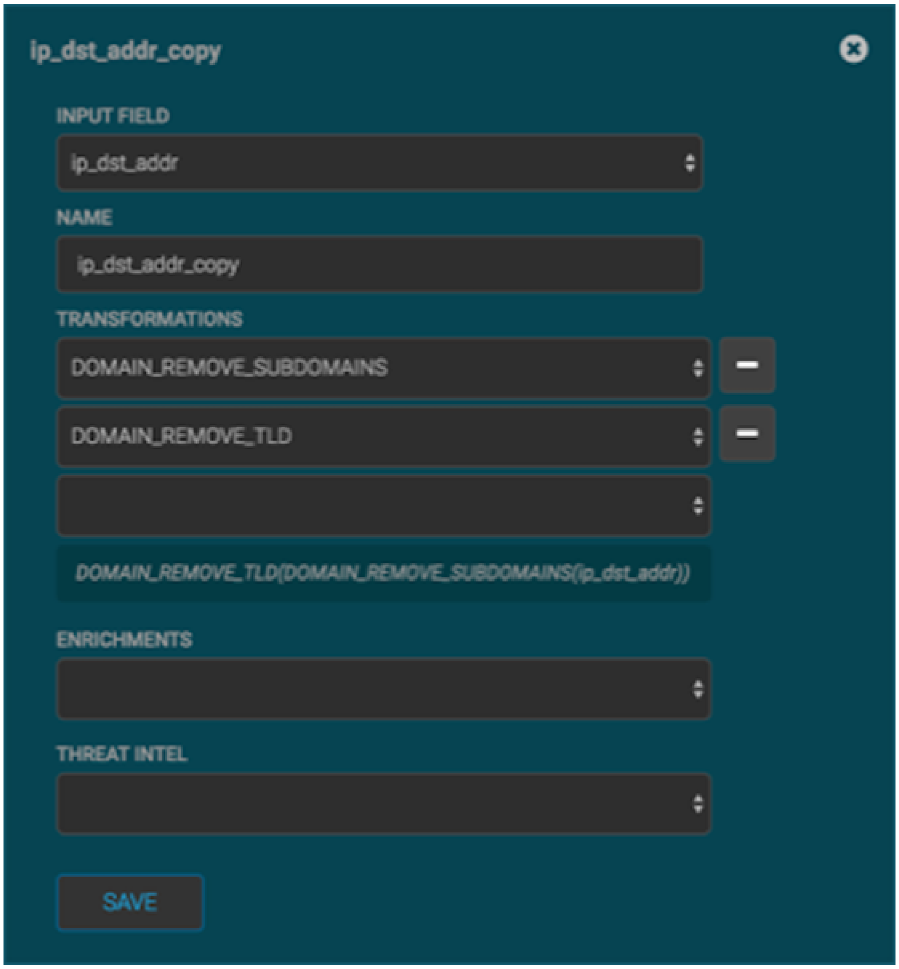
new

TRANSFORMATIONS

ENRICHMENTS

THREAT INTEL

4. Choose the field you want to transform from the **INPUT FIELD** box, enter the name of the new field in the **NAME** field, and then choose a function with the appropriate parameters in the **TRANSFORMATIONS** box. You can apply more than transformation to the input field.



ip_dst_addr_copy

INPUT FIELD
ip_dst_addr

NAME
ip_dst_addr_copy

TRANSFORMATIONS
DOMAIN_REMOVE_SUBDOMAINS
DOMAIN_REMOVE_TLD

DOMAIN_REMOVE_TLD(DOMAIN_REMOVE_SUBDOMAINS(ip_dst_addr))

ENRICHMENTS

THREAT INTEL

5. Click **SAVE** to save your additions.

   The Management module populates the Transforms field with the number of transformations applied to the sensor.

   If you change your mind and want to remove a transformation, click "-" next to the field.
6. Click **SAVE** in the parser panel to save the transformation information.

Enriching Telemetry Events

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). HCP 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.

HCP 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.

HCP provides two types of enrichment:

- Telemetry events
- Threat intelligence information

HCP 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 HCP, the enrichment store (which for HCP 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.

HCP 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:

Related Information

OPTIONAL: Global Configuration

Enriching Threat Intelligence Information

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

Related Information

Bulk Loading Threat Intelligence

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. 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:

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" : "$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" }

5. 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.

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" : "ENRICHMENT"
            ,"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 HCP enrichment store (HBase) and store the enrichment configuration in ZooKeeper.

Procedure

1. 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.

HCP 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.

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

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

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

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

4. 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.

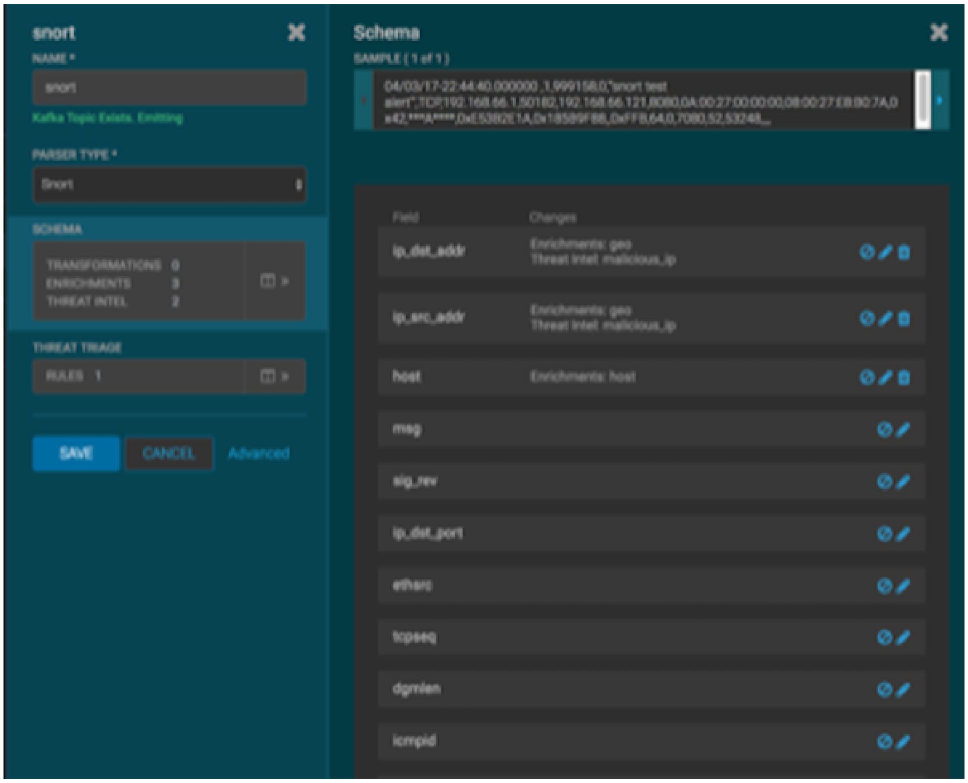
2. Select the new sensor from the list of sensors on the main window.
3. Click the pencil icon in the list of tool icons

   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.

   snort

   NAME * snort

   Kafka Topic Exists. Emitting

   PARSER TYPE * Snort

   SCHEMA

   TRANSFORMATIONS 0
   ENRICHMENTS 3
   THREAT INTEL 2

   THREAT TRIAGE

   RULES 1

   Schema

   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:00:27:00:00:00,08:00:27:E8:B0:7A,0
   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

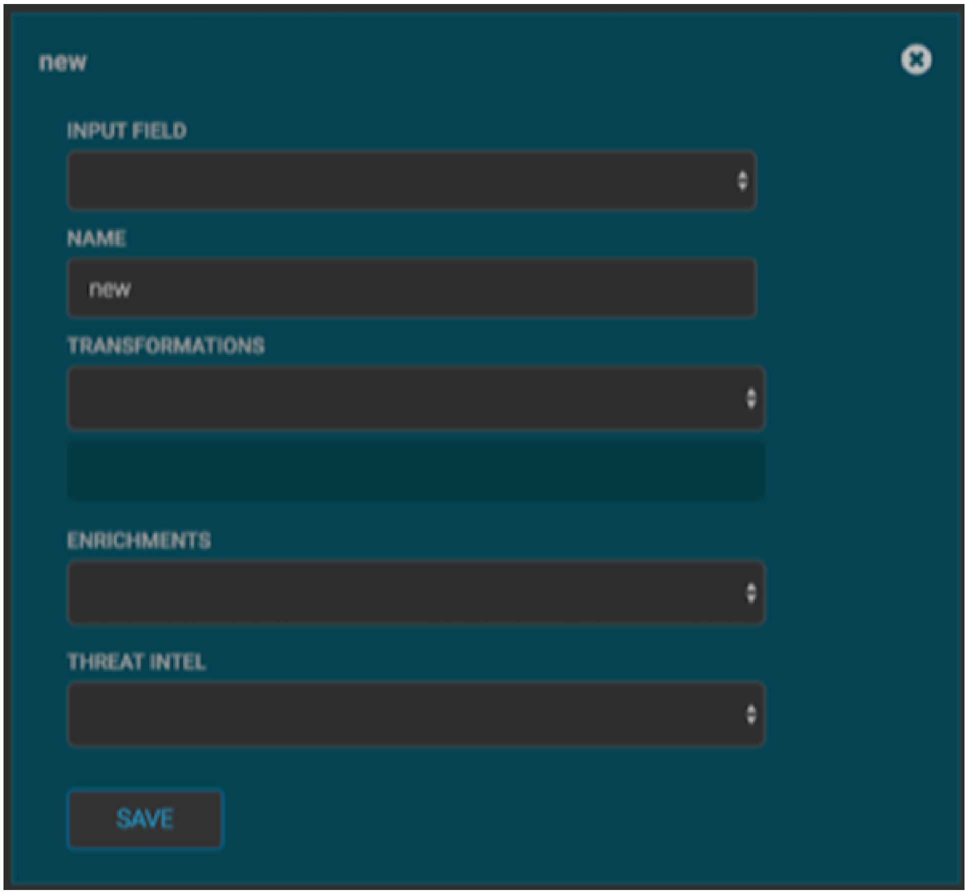
   (edit icon) next to a field to apply transformations to that field. Or click

   (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



new

INPUT FIELD

NAME

new

TRANSFORMATIONS

ENRICHMENTS

THREAT INTEL

SAVE

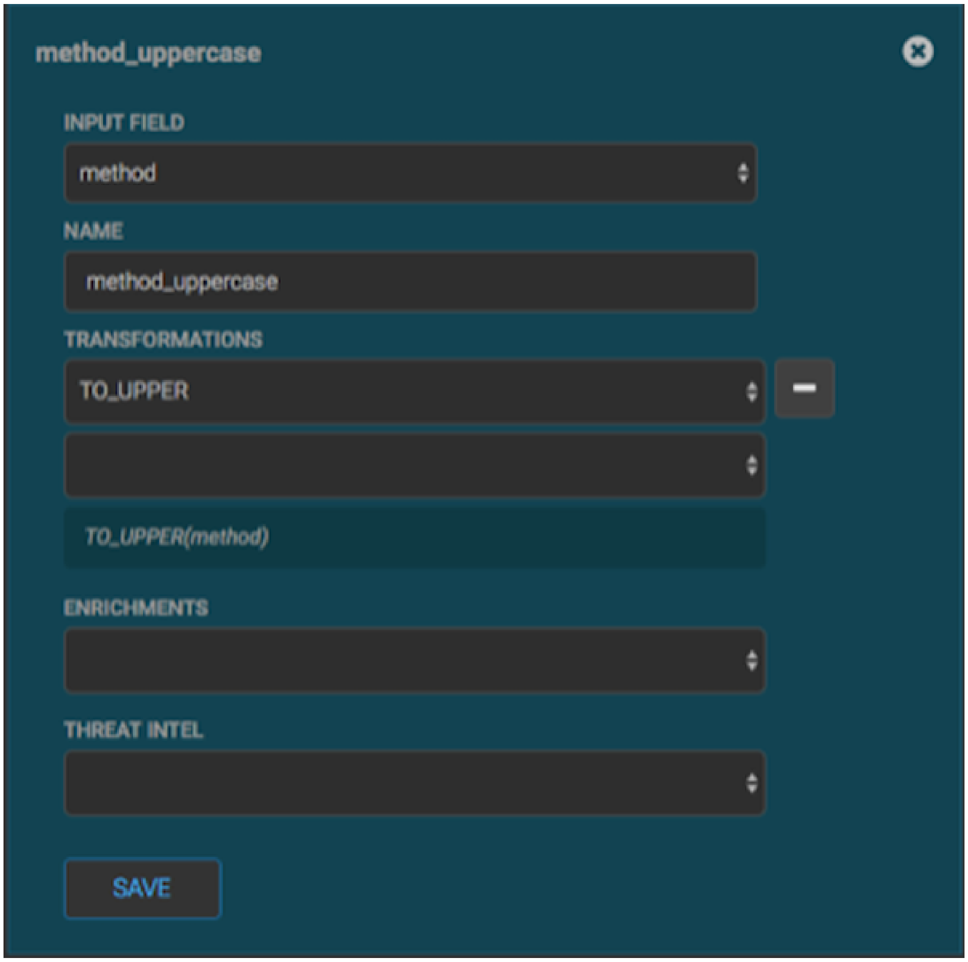
6. In the dialog box, enter the name of the new field in the **NAME** field, choose an input field from the **INPUT FIELD** box, and choose your transformation from the **TRANSFORMATIONS** field or enrichment from the **ENRICHMENTS** field.

   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



method_uppercase

INPUT FIELD: method

NAME: method_uppercase

TRANSFORMATIONS: TO_UPPER

TO_UPPER(method)

ENRICHMENTS

THREAT INTEL

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 toplogy's writer batching settings.

Related Information

Understanding Global Configuration

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.

Related Information

Triaging Alerts

Enriching Threat Intelligence Information

You can enrich your threat intelligence information just like you enriched your telemetry information.

You can choose to skip this section and come back to it later if you don't want to enrich your threat intelligence information at this time.

Metron provides an extensible framework to plug in threat intel sources. Each threat intel source has two components: an enrichment data source and an enrichment bolt. The threat intelligence feeds are loaded into a threat intelligence store similar to how the enrichment feeds are loaded. The keys are loaded in a key-value format. The key is the indicator and the value is the JSON formatted description of what the indicator is.

We recommend using a threat feed aggregator such as Soltra to dedup and normalize the feeds via STIX/Taxii. Metron provides an adapter that is able to read Soltra-produced STIX/Taxii feeds and stream them into HBase, which is the preferred data store to back high-speed threat intel lookups on HCP. HCP additionally provides a flat file and STIX bulk loader that can normalize, dedup, and bulk load or poll threat intel data into HBase even without the use of a threat feed aggregator.

Related Information

Soltra

OPTIONAL: Create a Mock Threat Intel Feed Source

Hortonworks Cybersecurity Platform (HCP) is designed to work with STIX/Taxii threat feeds, but can also be bulk loaded with threat data from a CSV file. In this example, we will explore the CSV example. The same loader framework that is used for enrichment here is used for threat intelligence. Similar to enrichments, we need to set up a data.csv file, the extractor config JSON, and the enrichment config JSON.
For this example, we use a Zeus malware tracker list located here: https://zeustracker.abuse.ch/blocklist.php?download=domainblocklist.

Procedure

1. Log into the $HOST_WITH_ENRICHMENT_TAG as root.
2. Copy the contents from the Zeus malware tracker list link to a file called domainblocklist.csv.

```
curl https://zeustracker.abuse.ch/blocklist.php?download=domainblocklist
 | grep -v "^#" | grep -v "^$" | grep -v "^https" | awk '{print
 $1",abuse.ch"}' > domainblocklist.csv
```

Configure an Extractor Configuration File

You use the extractor configuration file 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 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" : {
          "domain" : 0
          ,"source" : 1
      }
      ,"indicator_column" : "domain"
      ,"type" : "zeusList"
      ,"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 threatintel_extractor_config_temp.json -o threatintel_extractor_config.json

Using Stellar Data to Transform Threat Intelligence Data

You can transform and filter threat intelligence data as it is loaded into HBase by using Stellar extractor properties in the extractor configuration file. This task is optional.

Procedure

1. Transform and filter the threat intelligence 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",`

The following example uses a CSV list of top domains as an enrichment and filters the value metadata, as well as the indicator column, with each of the supported Stellar expressions:

```
{
 "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" }

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. 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" }

4. Remove any non-ASCII characters:

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

Configure Element-to-Threat Intelligence Feed Mapping

You 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 threatintel_config_temp.json at $METRON_HOME/config.

```
{
  "zkQuorum" : "localhost:2181"
 ,"sensorToFieldList" : {
    "squid" : {
           "type" : "THREAT_INTEL"
          ,"fieldToEnrichmentTypes" : {
             "domand_without_subdomains" : [ "zeusList" ]
                                      }
           }
                        }
}
```

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

You must specify the following:

- The ZooKeeper quorum which holds the cluster configuration
- The mapping between the fields in the enriched documents and the enrichment types.

This configuration allows the ingestion tools to update ZooKeeper post-ingestion so that the enrichment topology can take advantage immediately of the new type.

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

```
iconv -c -f utf-8 -t ascii threat_intel_config_temp.json -o
 threat_intel_config.json
```

Run the Threat Intelligence Loader

Now that you have the threat intel source, threat intel extractor, and threat intel mapping config defined, you can run the loader to move the data from the threat intel source to the Metron threat intel Store and store the enrichment config in ZooKeeper.

> **Note:** There is a special configuration parameter to the Extractor config that is only considered during this loader:
>
> | | |
> |---|---|
> | **inputFormatHander** | This specifies 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 where each line indicates a unit of information which can 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. |

Procedure

1. Log into $HOST_WITH_ENRICHMENT_TAG as root.
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
 zeusList_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 were properly ingested into HBase:

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

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

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

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

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

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:

```
ssh <Nifi Host>

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

```
tail -f access.log
```

c) Generate some data by entering the following:

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

6. Generate some data by using the Squid client to execute http requests:

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

Map Fields to HBase Enrichments

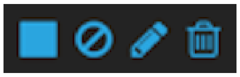
Now that you have data flowing into the HBase table, you need to ensure that the enrichment topology can be used to enrich the data flowing past. You can refine the parser output through transformations, enrichments, and threat intelligence.

Each of the parser outputs is added or modified in the Schema field.

Procedure

1. Select the new sensor from the list of sensors on the main window.

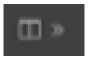
2. Click the pencil icon in the list of tool icons



for the new sensor.

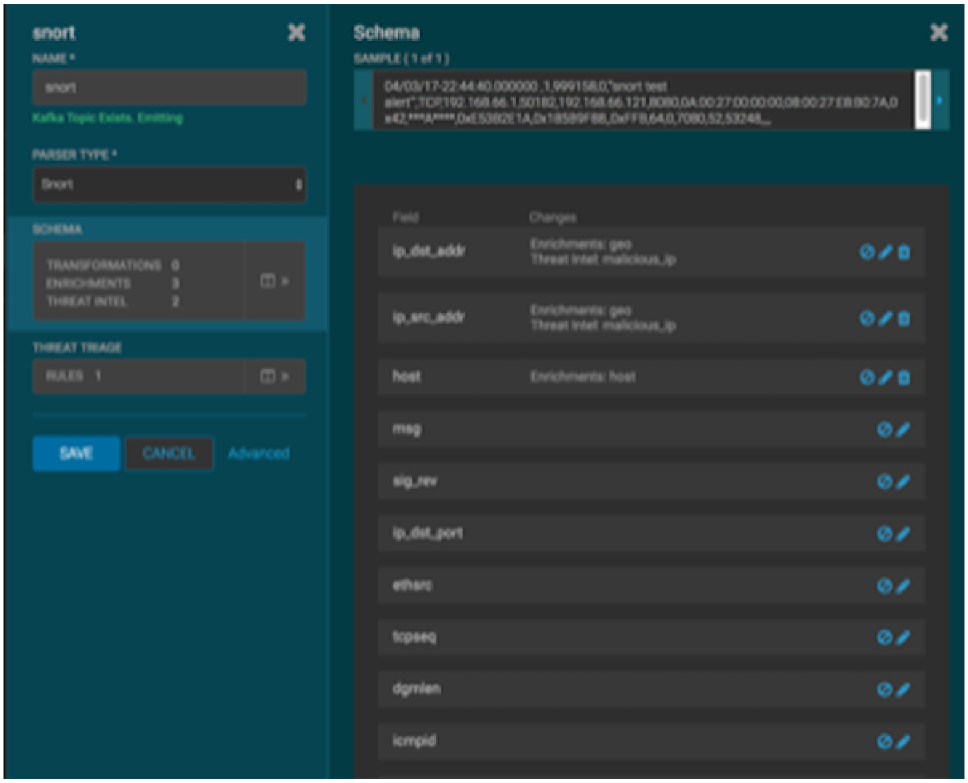
The Management Module displays the sensor panel for the new sensor.

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.



snort

NAME *

snort

Kafka Topic Exists. Emitting

PARSER TYPE *

Snort

SCHEMA

TRANSFORMATIONS 0
ENRICHMENTS 3
THREAT INTEL 2

THREAT TRIAGE

RULES 1

Schema

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:00:27:00:00:00,08:00:27:E8:B0:7A,0
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	

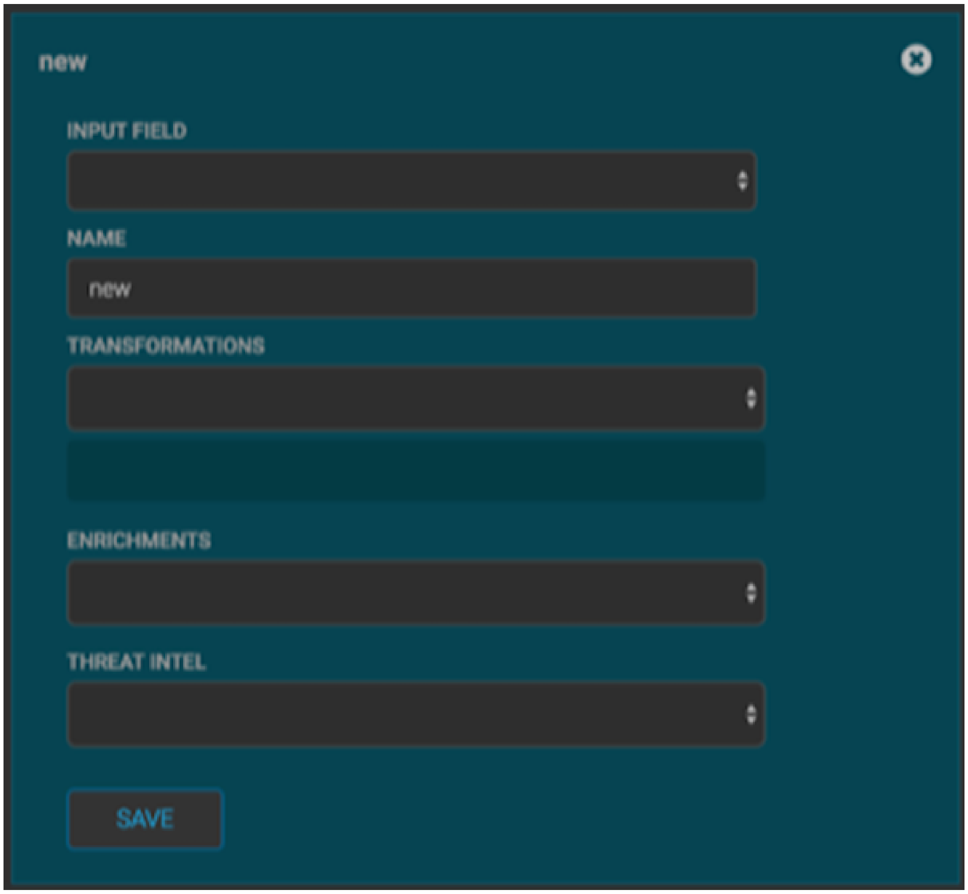
The Sample field, at the top of the panel, displays a parsed version of a sample message from the sensor. The Management module will test your transformations against these parsed messages.

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 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

INPUT FIELD

NAME

new

TRANSFORMATIONS

ENRICHMENTS

THREAT INTEL

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 **TRANSFORMATIONS** field or enrichment from the **ENRICHMENTS** field.

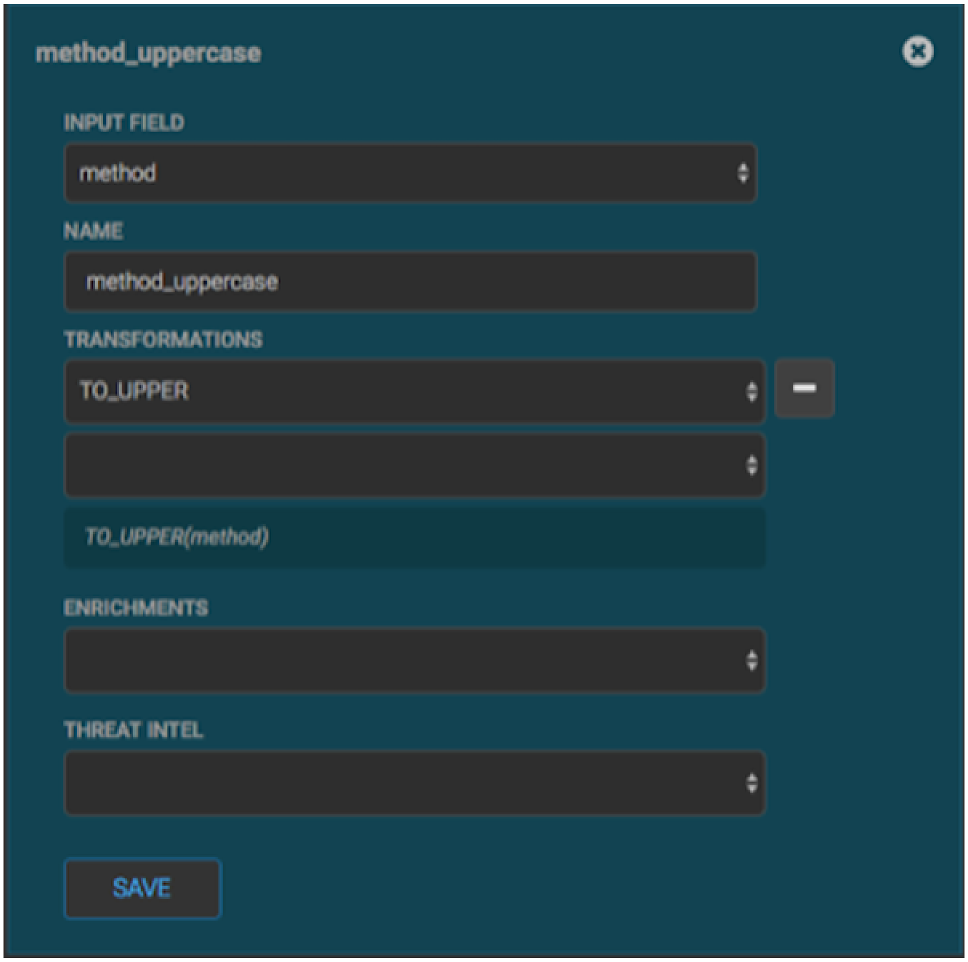
   For example, to create a new field showing the lower case version of the method field, do the following:

   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:



method_uppercase

INPUT FIELD: method

NAME: method_uppercase

TRANSFORMATIONS: TO_UPPER

TO_UPPER(method)

ENRICHMENTS

THREAT INTEL

6. Click SAVE to save your changes.
7. You can suppress fields from showing in 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.

Verify That the Threat Intel 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.

Related Information

Triaging Alerts

Prioritizing Threat Intelligence

Not all threat intelligence indicators are equal. Some require immediate response, while others can be dealt with or investigated as time and availability permits. As a result you need to triage and rank threats by severity.

In Hortonworks Cybersecurity Platform (HCP), you assign severity by associating possibly complex conditions with numeric scores. Then, for each message, you use a configurable aggregation function to evaluate the set of conditions and to aggregate the set of numbers for matching conditions. This aggregated score is added to the message in the threat.triage.level field.

Prerequisites

Before you can prioritize a threat intelligence enrichment, you must ensure that the enrichment is working properly.

Threat Triage Examples

Threat triage rules identify the conditions in the data source data flow and associate alert scores with those conditions.

Following are some examples of threat triage rules:

Rule 1	If a threat intelligence enrichment type is alerted, imagine that you want to receive an alert score of 5.
Rule 2	If the URL ends with neither .com nor .net, then imagine that you want to receive an alert score of 10.

Perform Threat Triage

To create a threat triage rule configuration, you must first define your rules. These rules identify the conditions in the data source data flow and associate alert scores with those conditions.

Procedure

1. Click the

   ✎

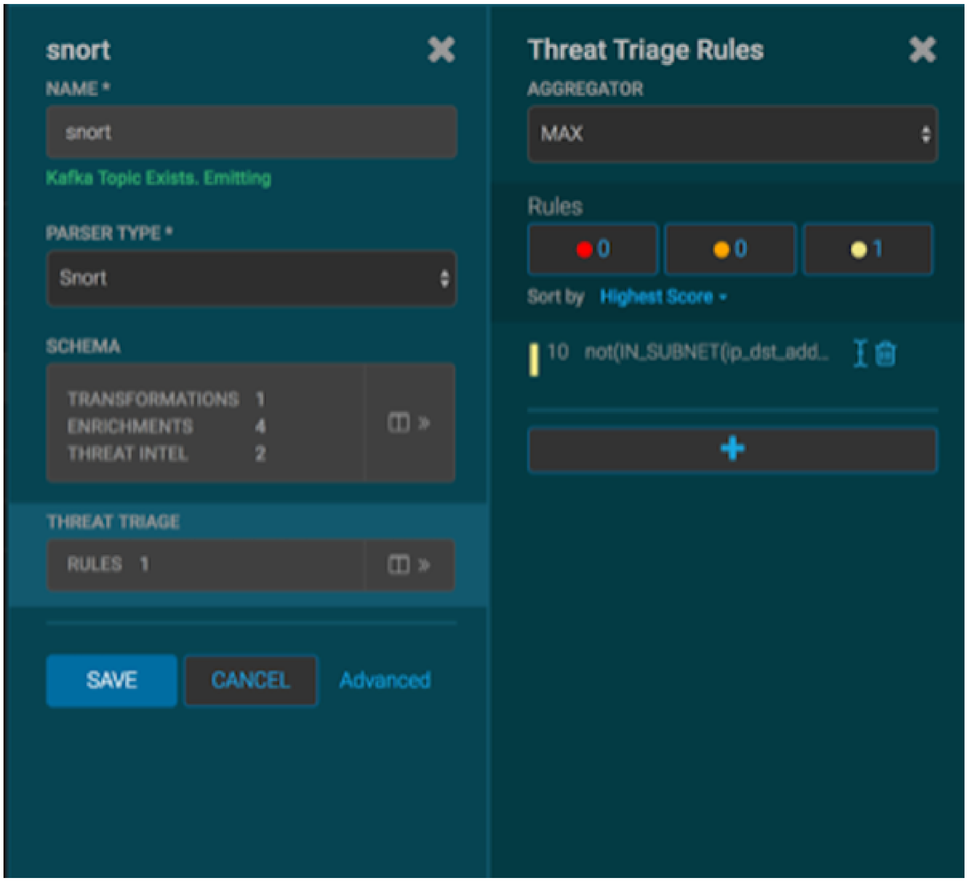
   (edit button) for your sensor.
2. In the Threat Triage field, click the

   ◫ »

   icon (expand window).

   The module displays the Threat Triage Rules panel.

   Threat Triage Rules Panel



snort

NAME * snort

Kafka Topic Exists. Emitting

PARSER TYPE * Snort

SCHEMA

TRANSFORMATIONS 1
ENRICHMENTS 4
THREAT INTEL 2

THREAT TRIAGE

RULES 1

Threat Triage Rules

AGGREGATOR MAX

Rules

● 0 ● 0 ● 1

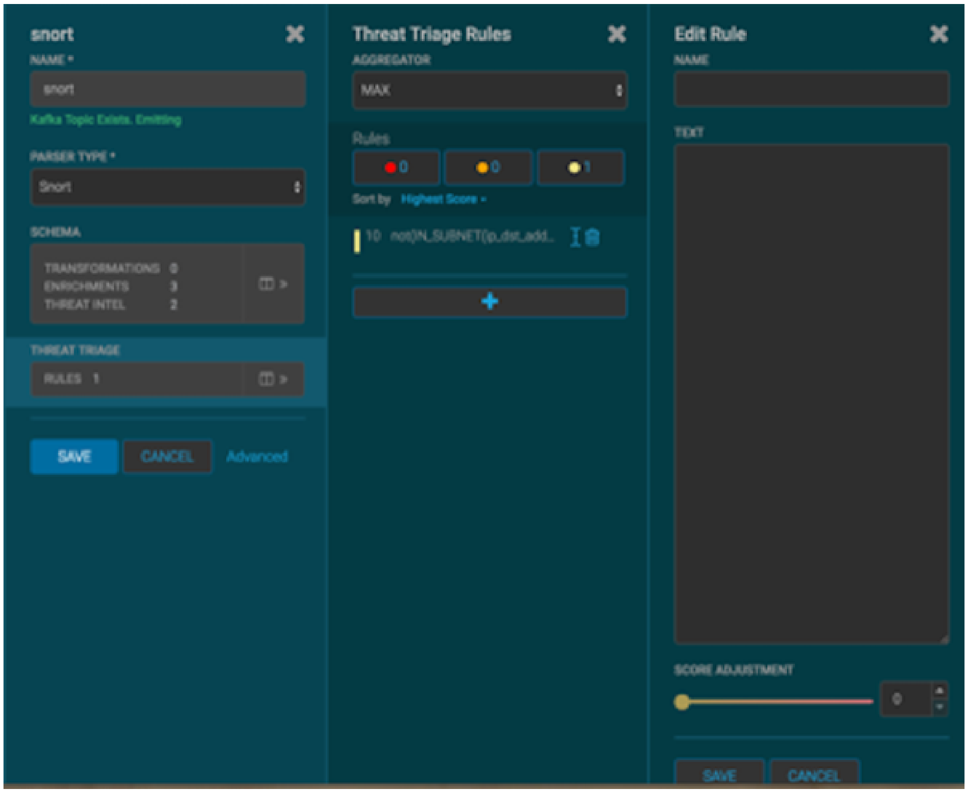
Sort by Highest Score

10 not(IN_SUBNET(ip_dst_add...

3. Click the + button to add a rule.

   The module displays the **Edit Rule** panel.

   Edit Rule Panel



4. Assign a name to the new rule by entering the name in the NAME field.
5. In the Text field, enter the syntax for the new rule.

```
Exists(IsAlert)
```

6. Use the **SCORE ADJUSTMENT** slider to choose the threat score for the rule.
7. Click **SAVE** to save the new rule.

   The new rule is listed in the Threat Triage Rules panel.
8. Choose how you want to aggregate your rules by choosing a value from the Aggregator menu.

   You can choose between:

MAX	The maximum of all of the associated values for matching queries.
MIN	The minimum of all of the associated values for matching queries.
MEAN	the mean of all of the associated values for matching queries.
POSITIVE_MEAN	The mean of the positive associated values for the matching queries.

9. You can use the **Rules** section and the **Sort by** pull down menu below the **Rules** section to filter how threat triages display.

   For example, to display only high levels alerts, click the box containing the red indicator. To sort the high level alerts from highest to lowest, choose **Highest Score** from the **Sort by** pull down menu.
10. Click **SAVE** on the Sensor panel to save your changes.

View Triaged Alerts Using Kafka

You can view triaged alerts in the indexing topic in Kafka.

Procedure

1. List the Kafka topics to find the threat triage alert panel:

```
/usr/hdp/current/kafka-broker/bin/kafka-topics.sh --zookeeper
 $ZOOKEEPER_HOST:2181 --list
```

2. View the threat triage alert Kafka topic:

```
cd $METRON_HOME/bin/.stellar
THREAT_TRIAGE_PRINT(conf)
```

The topic should appear similar to the following:

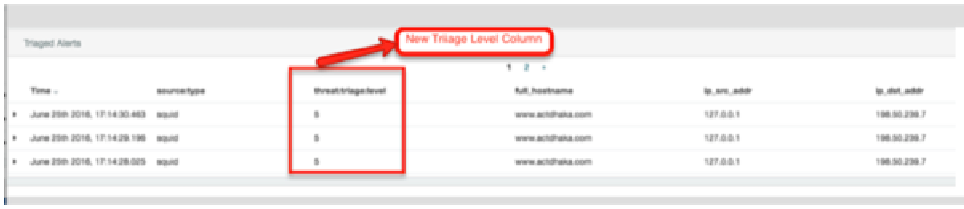
```
> THREAT_TRIAGE_PRINT(conf)
####################################################################################
# Name              # Comment # Triage Rule
                       # Score # Reason

                    #
####################################################################################
# Abnormal DNS Port #         # source.type == "bro" and protocol == "dns"
 and ip_dst_port != 53 # 10    # FORMAT("Abnormal DNS Port: expected: 53,
 found: %s:%d", ip_dst_addr, ip_dst_port) #
####################################################################################
```

View Triaged Alerts Using the Metron Dashboard

You can view triaged alerts in the triaged alert panel in the HCP Metron dashboard.

The following figure shows you an example of a triaged alert panel in the Hortonworks Cybersecurity Platform (HCP) Metron dashboard. For URLs from cnn.com, no threat alert is shown, so no triage level is set. Notice the lack of a threat.triage.level field:

Investigation Module Triaged Alert Panel



Time	source:type	threat:triage:level	full_hostname	ip_src_addr	ip_dst_addr
June 25th 2016, 17:14:30.463	squid	5	www.actdhaka.com	127.0.0.1	198.50.239.7
June 25th 2016, 17:14:29.196	squid	5	www.actdhaka.com	127.0.0.1	198.50.239.7
June 25th 2016, 17:14:28.025	squid	5	www.actdhaka.com	127.0.0.1	198.50.239.7

Configuring Indexing

The indexing topology is a topology dedicated to taking the data from a topology that has been enriched and storing the data in one or more supported indices. More specifically, the enriched data is ingested into Kafka, written in an indexing batch or bolt with a specified size, and sent to one or more specified indices. The configuration is intended to configure the indexing used for a given sensor type (for example, snort).

Currently, Hortonworks Cybersecurity Platform (HCP) supports the following indices:

- Elasticsearch

- Solr
- HDFS under /apps/metron/enrichment/indexed

Depending on how you start the indexing topology, it can have HDFS and either elasticsearch or SOLR writers running.

Just like the Global Configuration file, the Indexing Configuration file format is a JSON file stored in ZooKeeper and on disk at $METRON_HOME/config/zookeeper/ indexing.

Within the sensor-specific configuration, you can configure the individual writers. The parameters currently supported are:

index	The name of the index to write to (defaulted to the name of the sensor).
batchSize	The size of the batch that is written to the indices at once (defaulted to 1).
enabled	Whether the index or writer is enabled (default true).

Default Configuration

If you do not configure the individual writers, the sensor-specific configuration will use the default values.

You can choose to use this default configuration by either not creating the Indexing Configuration file or by entering the following in the file. You can name the file anything you like, for example index_config.json, but it must be located at $METRON_HOME/config/zookeeper/indexing.

```
{
}
```

If a writer configuration is unspecified, then a warning is indicated in the Storm console. For example, WARNING: Default and (likely) unoptimized writer config used for hdfs writer and sensor squid. You can ignore this warning message if you intend to use the default configuration.

This default configuration uses the following configuration:

- elasticsearch writer
  - index name the same as the sensor
  - batch size of 1
  - enabled
- hdfs writer
  - index name the same as the sensor
  - batch size of 1
  - enabled

Specify Index Parameters

You can to specify the parameters for the writers rather than using the default values using the HCP Management Module.

Procedure

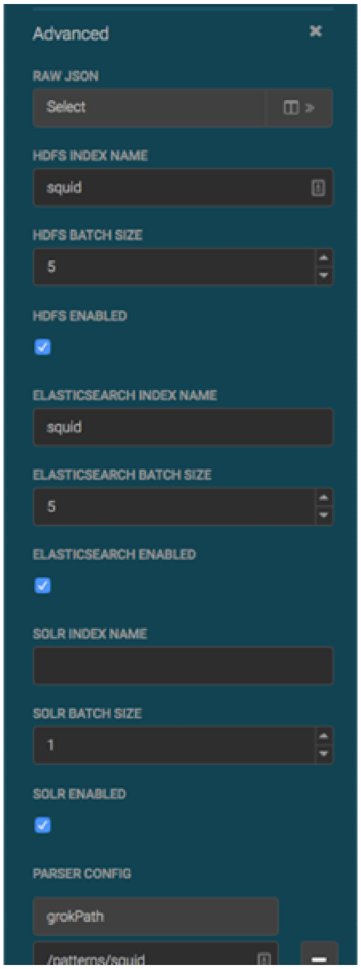
1. Edit your sensor by clicking

   ✎

   (the edit button) next your sensor in the Management Module.
2. Click the **Advanced** button next to **Save** and **Cancel**.

   The Management Module expands the panel to display the Advanced fields.

   Management Module Advanced Panel

   Advanced ✖

   RAW JSON: Select

   HDFS INDEX NAME: squid

   HDFS BATCH SIZE: 5

   HDFS ENABLED: ☑

   ELASTICSEARCH INDEX NAME: squid

   ELASTICSEARCH BATCH SIZE: 5

   ELASTICSEARCH ENABLED: ☑

   SOLR INDEX NAME:

   SOLR BATCH SIZE: 1

   SOLR ENABLED: ☑

   PARSER CONFIG

   grokPath

   /patterns/squid

3. Enter index configuration information for your sensor.
4. Click the **Raw JSON** field and set the alert field to "type": "nested":

   ```
   },
    "alert": {
    "type": "nested"
   }
   ```

   If this field is not set, Elasticsearch can throw an error and the field will not be queryable.
5. Click **Save** to save your changes and push your configuration to ZooKeeper.

Turn off HDFS Writer

You can turn off the HDFS writer when you are configuring and testing your system.

Procedure

Turn off the HDFS index or writer using the following syntax in the index.json file.

```
{
 "elasticsearch": {
 "index": "foo",
 "enabled" : true
 },
 "hdfs": {
 "index": "foo",
 "batchSize": 100,
 "enabled" : false
 }
}
```

Setting Up a Profile

A profile describes the behavior of an entity on a network. An entity can be a server, user, subnet, or application. Once you generate a profile defining what normal behavior looks like, you can build models that identify anomalous behavior.

Install Profiler

Data scientists use the Profiler to describe the behavior of entities on a network. The first step in setting up a Profiler is to install it.

Procedure

1. Build the Metron RPMs (see Building the RPMs).

   You might have already built the Metron RPMs when core Metron was installed.

   ```
   $ find metron-deployment/ -name "metron-profiler*.rpm"
   metron-deployment//packaging/docker/rpm-docker/RPMS/noarch/metron-
   profiler-0.4.1-201707131420.noarch.rpm
   ```

2. Copy the Profiler RPM to the installation host.

   The installation host must be the same host on which HCP was installed. Depending on how you installed HCP, the Profiler RPM might have already been copied to this host with the other HCP RPMs.

   [root@$METRON_HOME ~]# find /localrepo/ -name "metron-profiler*.rpm" /localrepo/metron-profiler-0.4.0-201707112313.noarch.rpm

3. Install the RPM.

   ```
   [root@$METRON_HOME ~]# rpm -ivh metron-profiler-*.noarch.rpm
   Preparing...                ############################################
    [100%]
      1:metron-profiler        ############################################
    [100%]
   ```

   ```
   [root@$METRON_HOME ~]# rpm -ql metron-profiler
   ```

```
/usr/metron
/usr/metron/0.4.1
/usr/metron/0.4.1/bin
/usr/metron/0.4.1/bin/start_profiler_topology.sh
/usr/metron/0.4.1/config
/usr/metron/0.4.1/config/profiler.properties
/usr/metron/0.4.1/flux
/usr/metron/0.4.1/flux/profiler
/usr/metron/0.4.1/flux/profiler/remote.yaml
/usr/metron/0.4.1/lib
/usr/metron/0.4.1/lib/metron-profiler-0.4.0-uber.jar
```

4. Create a table within HBase that will store the profile data. By default, the table is named profiler with a column family P.

   ```
   $ /usr/hdp/current/hbase-client/bin/hbase shell
   hbase(main):001:0> create 'profiler', 'P'
   ```

   The table name and column family must match the Profiler's configuration (see Metron Profiler for Storm or Metron Profiler for Spark).

5. Edit the configuration file located at $METRON_HOME/config/profiler.properties.

   ```
   kafka.zk=node1:2181
   kafka.broker=node1:6667
   ```

   Change kafka.zk to refer to ZooKeeper in your environment.

   Change kafka.broker to refer to a Kafka Broker in your environment.

6. Start the Profiler topology.

   ```
   $ cd $METRON_HOME
   $ bin/start_profiler_topology.sh
   ```

   At this point the Profiler is running and consuming telemetry messages. We have not defined any profiles yet, so it is not doing anything very useful. The next section walks you through the steps to create your very first "Hello, World!" profile.

Related Information

Building the RPMs

Create a Profile

After you install Profiler, you must define the profile and upload the definition to ZooKeeper.

Procedure

1. Create a table within HBase that will store the profile data.

   ```
   $ /usr/hdp/current/hbase-client/bin/hbase shell
   hbase(main):001:0> create 'profiler', 'P'
   ```

   The table name and column family must match the Profiler's configuration.

2. Define the profile in a file located at $METRON_HOME/config/zookeeper/profiler.json.

   The following example JSON creates a profile that simply counts the number of messages per ip_src_addr during each sampling interval.

   ```
   {
     "profiles": [
   ```

```
    {
      "profile": "test",
      "foreach": "ip_src_addr",
      "init":    { "count": 0 },
      "update":  { "count": "count + 1" },
      "result":  "count"
    }
  ]
}
```

The following table lists the supported profile elements:

Name	Description	
profile	Required	A unique name identifying the profile. The field is treated as a string.
foreach	Required	A separate profile is maintained 'for each' of these. This is effectively the entity that the profile is describing. The field is expected to contain a Stellar expression whose result is the entity name. For example, if ip_src_addr then a separate profile would be maintained for each unique IP source address in the data; 10.0.0.1, 10.0.0.2, etc.
onlyif	Optional	An expression that determines if a message should be applied to the profile. A Stellar expression that returns a Boolean is expected. A message is only applied to a profile if this expression is true. This allows a profile to filter the messages that get applied to it.

groupBy	Optional	One or more Stellar expressions used to group the profile measurements when persisted. This is intended to sort the Profile data to allow for a contiguous scan when accessing subsets of the data.
		The 'groupBy' expressions can refer to any field within aorg.apache.metron.profiler.ProfileMeasurement. A common use case would be grouping by day of week. This allows a contiguous scan to access all profile data for Mondays only. Using the following definition would achieve this.
		`"groupBy":`
		`[ "DAY_OF_WEEK()" ]`

init	Optional	One or more expressions executed at the start of a window period. A map is expected where the key is the variable name and the value is a Stellar expression. The map can contain 0 or more variables/expressions. At the start of each window period the expression is executed once and stored in a variable with the given name.
		`"init": {`
		`  "var1": "0",`
		`  "var2": "1"`
		`}`

Name	**Description**	

update	Required	One or more expressions executed when a message is applied to the profile. A map is expected where the key is the variable name and the value is a Stellar expression. The map can include 0 or more variables/ expressions. When each message is applied to the profile, the expression is executed and stored in a variable with the given name.
		`"update": {`
		`  "var1": "var1 + 1",`
		`  "var2": "var2 + 1"`
		`}`

result	Required	A Stellar expression that is executed when the window period expires. The expression is expected to summarize the messages that were applied to the profile over the window period. The expression must result in a numeric value such as a Double, Long, Float, Short, or Integer. For more advanced use cases, a profile can generate two types of results. A profile can define one or both of these result types at the same time. • profile: A required expression that defines a value that is persisted for later retrieval. • triage: An optional expression that defines values that are accessible within the Threat Triage process.
expires	Optional	A numeric value that defines how many days the profile data is retained. After this time, the data expires and is no longer accessible. If no value is defined, the data does not expire.

3. Upload the profile definition to ZooKeeper:

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

4. Start the Profiler topology:

```
$ bin/start_profiler_topology.sh
```

5. Ensure that test messages are being sent to the Profiler's input topic in Kafka.

   The Profiler will consume messages from the inputTopic defined in the Profiler's configuration.
6. Check the HBase table to validate that the Profiler is writing the profile.

```
$ /usr/hdp/current/hbase-client/bin/hbase shell
hbase(main):001:0> count 'profiler'
*** Output Information ***
```

   Remember that the Profiler is flushing the profile every 15 minutes. You will need to wait at least this long to start seeing profile data in HBase.
7. Use the Profiler Client to read the profile data.

The following example PROFILE_GET command reads data written by the sample profile given above, if 10.0.0.1 is one of the input values for ip_src_addr. For more information on using the API client, refer to Accessing Profiles.

```
$ bin/stellar -z $ZOOKEEPER_HOST:2181

[Stellar]>>> PROFILE_GET( "test", "10.0.0.1", PROFILE_FIXED(30,
 "MINUTES"))
[451, 448]
```

Profiler Configuration Settings

The Profiler is installed during the Hortonworks Cybersecurity Platform (HCP) installation and runs as an independent Storm topology. The configuration for the Profiler topology is stored in ZooKeeper at /metron/topology/profiler. These properties also exist in the default installation of HCP at $METRON_HOME/config/zookeeper/profiler.json. The profiler values can be changed on disk and then uploaded to ZooKeeper using $METRON_HOME/bin/zk_load_configs.sh.

You might need to work with your Platform Engineer to modify or tune the Profiler values.

Settings.	**Description**
profiler.workers	The number of worker processes to create for the topology.
profiler.executors	The number of executors to spawn per component.
profiler.input.topic	The name of the Kafka topic from which to consume data.
profiler.output.topic	The name of the Kafka topic to which profile data is written. Only used with profiles that use the triage result field.
profiler.period.duration	The duration of each profile period. This value should be define along with profiler.period.duration.units.
profiler.period.duration.units	The units used to specify the profile period duration. This value should be defined along with profiler.period.duration.
profiler.ttl	If a message has not been applied to a Profile in this period of time, the Profile will be forgotten and its resources will be cleaned up. This value should be defined along with profiler.ttl.units.
profiler.ttl.units	The units used to specify the profiler.ttl.
profiler.hbase.salt.divisor	A salt is prepended to the row key to help prevent hotspotting. This constant is used to generate the sale. Ideally, this constant should be roughly equal to the number of nodes in the HBase cluster.
profiler.hbase.table	The name of the HBase table that profiles are written to.
profiler.hbase.column.family	The column family used to store profiles.
profiler.hbase.batch	The number of puts that are written in a single batch.
profiler.hbase.flush.interval.seconds	The maximum number of seconds between batch writes to HBase.

Start the Profiler

After you install and configure the Profiler, you can start the profiler.

Procedure

Start the Profiler using the following command:

```
$METRON_HOME/bin/start_profiler_topology.sh
```

Develop Profiles

Troubleshooting issues when programming against a live stream of data can be difficult. The Stellar REPL (an interactive top level or language shell) is a powerful tool to help work out the kinds of enrichments and transformations that are needed. The Stellar REPL can also be used to help when developing profiles for the Profiler.

Procedure

1. Take a first pass at defining your profile.

   For example, in the editor copy/paste the basic Hello, World profile below.

```
[Stellar]>>> conf := SHELL_EDIT()
[Stellar]>>> conf
{
  "profiles": [
    {
      "profile": "hello-world",
      "onlyif":  "exists(ip_src_addr)",
      "foreach": "ip_src_addr",
      "init":    { "count": "0" },
      "update":  { "count": "count + 1" },
      "result":  "count"
    }
  ]
}
```

2. Initialize the Profiler.

```
[Stellar]>>> profiler := PROFILER_INIT(conf)
[Stellar]>>> profiler
org.apache.metron.profiler.StandAloneProfiler@4f8ef473
```

3. Create a message to simulate the type of telemetry that you expect to be profiled.

   For example, in the editor copy/paste the JSON below.

```
[Stellar]>>> message := SHELL_EDIT()
[Stellar]>>> message
{
   "ip_src_addr": "10.0.0.1",
   "protocol": "HTTPS",
   "length": "10",
   "bytes_in": "234"
}
```

4. Apply some telemetry messages to your profiles. The following applies the same message 3 times.

```
[Stellar]>>> PROFILER_APPLY(message, profiler)
org.apache.metron.profiler.StandAloneProfiler@4f8ef473

[Stellar]>>> PROFILER_APPLY(message, profiler)
org.apache.metron.profiler.StandAloneProfiler@4f8ef473

[Stellar]>>> PROFILER_APPLY(message, profiler)
org.apache.metron.profiler.StandAloneProfiler@4f8ef473
```

5. Flush the Profiler to see what has been calculated.

```
[Stellar]>>> values := PROFILER_FLUSH(profiler)
[Stellar]>>> values
```

```
[{period={duration=900000, period=1669628, start=1502665200000,
 end=1502666100000},
   profile=hello-world, groups=[], value=3, entity=10.0.0.1}]
```

A flush is what occurs at the end of each 15 minute period in the Profiler. The result is a list of profile measurements. Each measurement is a map containing detailed information about the profile data that has been generated.

This profile counts the number of messages by IP source address. Notice that the value is '3' for the entity '10.0.0.1' as we applied 3 messages with an 'ip_src_addr' of '10.0.0.1'. There will always be one measurement for each [profile, entity] pair.

6. If you are unhappy with the data that has been generated, then 'wash, rinse and repeat' this process. After you are satisfied with the data being generated by the profile, then follow the Getting Started guide to use the profile against your live, streaming data in a Metron cluster.

Testing

After you install, configure, and start Profiler, you should validate that the Porfiler is working correctly.

Procedure

1. Login to the server hosting Metron.

2. Enter the following command:

```
[root@node1 0.3.0]# bin/stellar -z $ZOOKEEPER_HOST:2181
Stellar, Go!
Please note that functions are loading lazily in the background and will
 be unavailable until loaded fully.
{es.clustername=metron, es.ip=node1, es.port=9300,
 es.date.format=yyyy.MM.dd.HH}

[Stellar]>>> ?PROFILE_GET
Functions loaded, you may refer to functions now...
PROFILE_GET
Description: Retrieves a series of values from a stored profile.

Arguments:
    profile - The name of the profile.
    entity - The name of the entity.
    durationAgo - How long ago should values be retrieved from?
    periods - The list of profile periods to grab. These are ProfilePeriod
 objects.
    groups - Optional, must correspond to the 'groupBy' list used in
 profile creation - List (in square brackets) of groupBy values used to
 filter the profile. Default is the empty list, meaning groupBy was not
 used when creating the profile.
    config_overrides - Optional - Map (in curly braces) of name:value
 pairs, each overriding the global config parameter of the same name.
 Default is the empty Map, meaning no overrides.
Returns: The profile measurements.

[Stellar]>>> PROFILE_GET('test','192.168.138.158', 1, 'HOURS') [12078.0,
 8921.0, 12131.0]
```

In the preceeding example, we use the Stellar Shell to replicate the execution environment of Stellar running in a Storm topology, like Metron's Parser or Enrichment topology. Replace 'node1:2181' with the URL to a ZooKeeper Broker.

The client API call above retrieves the past hour of the test profile for the entity 192.168.138.158.

Triage Squid Events

Security event triage rules determine which events require further follow up and which events can be archived without further investigation. HCP processes many events every day so effective triage helps analysts focus on the most important events.

The two components of security event triage are:

- Determine if the event is an alert.
- If the event is an alert, assign a score. If the event is not an alert, it is not scored.

Triage Squid Using the Typosquatting Algorithm

For this example, we use a very simple triage rule to detect typosquatting. Typosquatting uses common domain misspellings to install malicious web content.

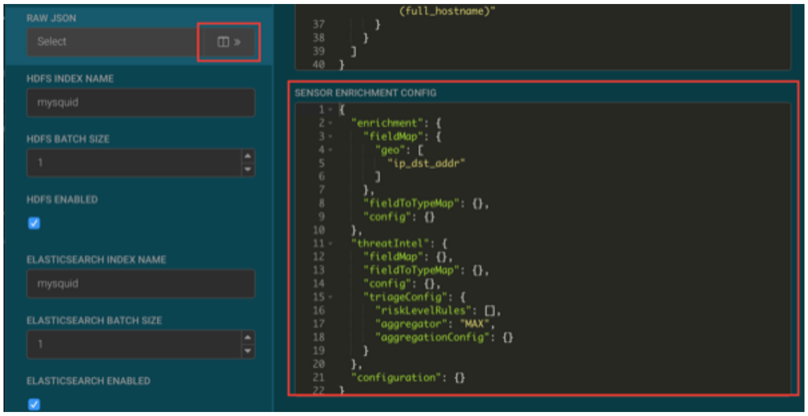
Procedure

1. Display the Management module user interface.
2. Select the Squid sensor from the list of sensors on the main window.
3. Click the pencil icon in the list of tool icons

   for the sensor.

   The Management Module displays the Squid sensor panel.
4. Click the **Advanced** button.
5. Click

   (expand window button) next to the **RAW JSON** field.

   RAW JSON

   Select

   HDFS INDEX NAME

   mysquid

   HDFS BATCH SIZE

   1

   HDFS ENABLED

   ☑

   ELASTICSEARCH INDEX NAME

   mysquid

   ELASTICSEARCH BATCH SIZE

   1

   ELASTICSEARCH ENABLED

   ☑

   ```
                 (full_hostname)"
           }
         }
       ]
   }
   ```

   SENSOR ENRICHMENT CONFIG

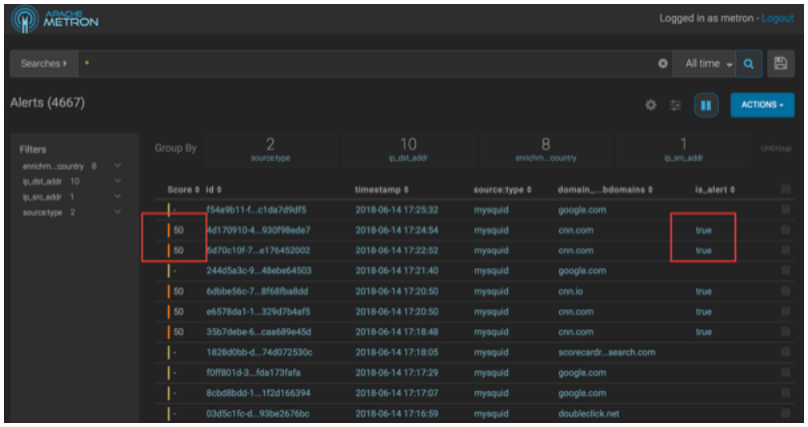
   ```
   {
     "enrichment": {
       "fieldMap": {
         "geo": [
           "ip_dst_addr"
         ]
       },
       "fieldToTypeMap": {},
       "config": {}
     },
     "threatIntel": {
       "fieldMap": {},
       "fieldToTypeMap": {},
       "config": {},
       "triageConfig": {
         "riskLevelRules": [],
         "aggregator": "MAX",
         "aggregationConfig": {}
       }
     },
     "configuration": {}
   }
   ```
6. Replace the JSON information in the **SENSOR ENRICHMENT CONFIG** section with the following JSON information:

```
{
 "enrichment": {
  "fieldMap": {
   "geo": [
    "ip_dst_addr"
   ],
```

```
   "stellar": {
    "config": [
     "domain_without_tld := DOMAIN_REMOVE_TLD(domain_without_subdomains)",
     "is_potential_typosquat := BLOOM_EXISTS(OBJECT_GET('/tmp/reference/
alexa10k_filter.ser'), domain_without_tld)",
     "domain_without_tld := null"
    ]
   }
  },
  "fieldToTypeMap": {},
  "config": {}
 },
 "threatIntel": {
  "fieldMap": {
   "stellar": {
    "config": [
     "is_alert := (exists(is_alert) && is_alert) ||
 is_potential_typosquat"
    ]
   }
  },
  "fieldToTypeMap": {},
  "config": {},
  "triageConfig": {
   "riskLevelRules": [
    {
     "name": "Alexa 10k Typosquat Bloom",
     "comment": "Inspect a bloom filter with potentially typosquatted
 domains from the top Alexa 10k",
     "rule": "is_potential_typosquat != null && is_potential_typosquat",
     "score": 50,
     "reason": "FORMAT('%s is a potential typosquatted domain from the top
 10k domains from alexa', domain_without_subdomains)"
    }
   ],
   "aggregator": "MAX",
   "aggregationConfig": {}
  }
 },
 "configuration": {}
}
```

7. Click **SAVE** below the JSON information.
8. Click **SAVE** at the bottom of the Squid sensor configuration panel.
9. Enter cnn.com or nsp.com in the browser connected to the HCP proxy.
10. Display the Alerts UI.

    In the Score column, you will see events with non-zero scores and the **is_alert** field set to **true**.



Score ⇕	id ⇕	timestamp ⇕	source:type ⇕	domain_...bdomains ⇕	is_alert ⇕
-	f54a9b11-f...c1da7d9df5	2018-06-14 17:25:32	mysquid	google.com	
50	4d170910-4...930f98ede7	2018-06-14 17:24:54	mysquid	cnn.com	true
50	5d70c10f-7...e176452002	2018-06-14 17:22:52	mysquid	cnn.com	true
-	244d5a3c-9...48ebe64503	2018-06-14 17:21:40	mysquid	google.com	
50	6dbbe56c-7...8f68fba8dd	2018-06-14 17:20:50	mysquid	cnn.io	true
50	e6578da1-1...329d7b4af5	2018-06-14 17:20:50	mysquid	cnn.com	true
50	35b7debe-6...caa689e45d	2018-06-14 17:18:48	mysquid	cnn.com	true
-	1828d0bb-d...74d072530c	2018-06-14 17:18:05	mysquid	scorecardr...search.com	
-	f0ff801d-3...fda173fafa	2018-06-14 17:17:29	mysquid	google.com	
-	8cbd8bdd-1...1f2d166394	2018-06-14 17:17:07	mysquid	google.com	
-	03d5c1fc-d...93be2676bc	2018-06-14 17:16:59	mysquid	doubleclick.net	

11. If you want to view the columns as they appear in the screen shot, complete the following:

a) Click the gear icon on the left of the **Actions** button.

b) Check the **Score**, **id**, **timestamp**, **source:type**, **domain_without_subdomains**, and **is_alert** fields.

c) Uncheck all other columns.

d) Click **SAVE**.

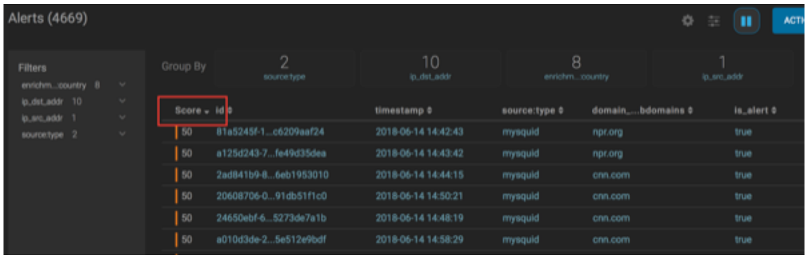
By default, the columns after the **score**, **id**, and **timestamp** are displayed from left to right in alphabetical order by field name.

e) To change the column order, click the up or down button to move the columns displayed up or down.

f) Click **SAVE**.

12. Click the **Score** header to sort the events ascending by Score.

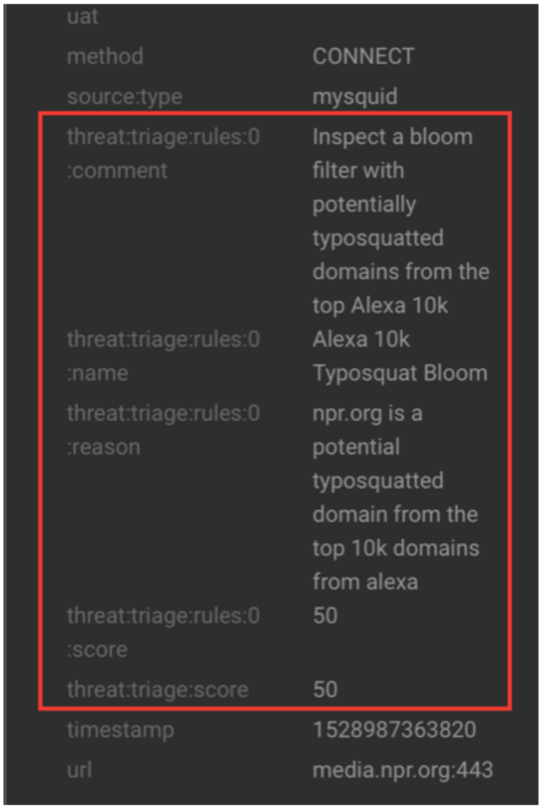
Click again to sort descending by Score. A downward arrow appears next to the **Score** header when sorted descending by Score.



Score ▾	id ⇕	timestamp ⇕	source:type ⇕	domain_...bdomains ⇕	is_alert ⇕
50	81a5245f-1...c6209aaf24	2018-06-14 14:42:43	mysquid	npr.org	true
50	a125d243-7...fe49d35dea	2018-06-14 14:43:42	mysquid	npr.org	true
50	2ad841b9-8...6eb1953010	2018-06-14 14:44:15	mysquid	cnn.com	true
50	20608706-0...91db51f1c0	2018-06-14 14:50:21	mysquid	cnn.com	true
50	24650ebf-6...5273de7a1b	2018-06-14 14:48:19	mysquid	cnn.com	true
50	a010d3de-2...5e512e9bdf	2018-06-14 14:58:29	mysquid	cnn.com	true

13. Click between the columns of one of the Scored alerts to view the alert details.

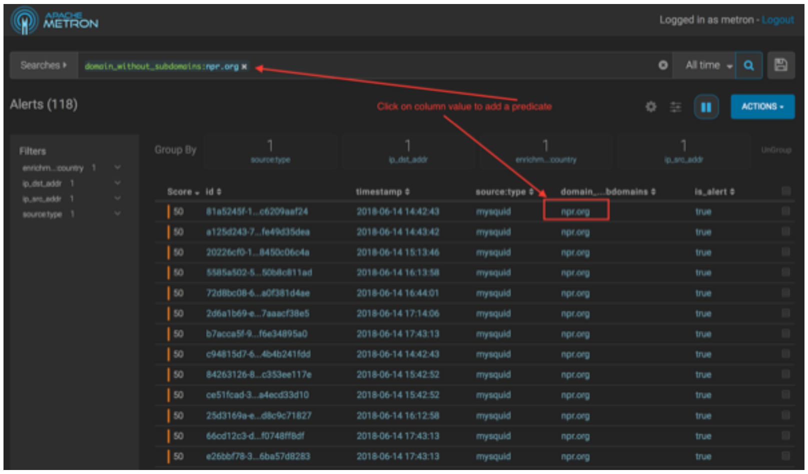
The fields beginning with **threat:triage:rules** show the results of all the triage rules. The **threat:triage:score** field is the aggregated score of the event. If there are more than one triage rule, this field will contain the score combining the results from all the rules. The **is_alert** field is set only if the triage rules indicate the event is an alert.



uat	
method	CONNECT
source:type	mysquid
threat:triage:rules:0:comment	Inspect a bloom filter with potentially typosquatted domains from the top Alexa 10k
threat:triage:rules:0:name	Alexa 10k Typosquat Bloom
threat:triage:rules:0:reason	npr.org is a potential typosquatted domain from the top 10k domains from alexa
threat:triage:rules:0:score	50
threat:triage:score	50
timestamp	1528987363820
url	media.npr.org:443

14. To see all the alerts for a particular domain, click the domain name.

The Alerts UI displays only the alerts with the selected domain name.



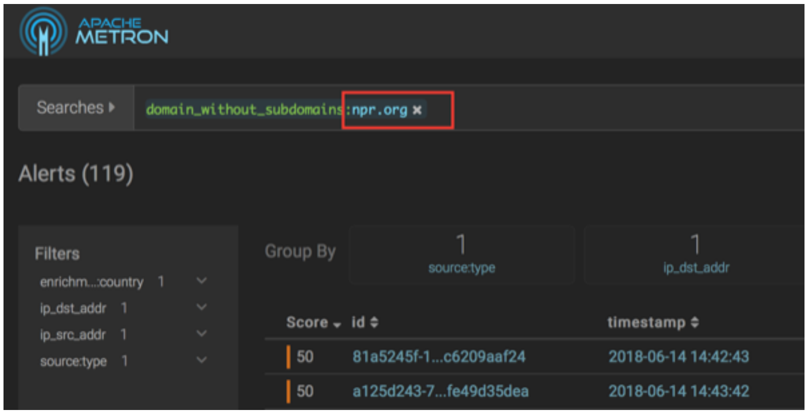
domain_without_subdomains:npr.org ✕

Click on column value to add a predicate

Score	id	timestamp	source:type	domain_...bdomains	is_alert
50	81a5245f-1...c6209aaf24	2018-06-14 14:42:43	mysquid	npr.org	true
50	a125d243-7...fe49d35dea	2018-06-14 14:43:42	mysquid	npr.org	true
50	20226cf0-1...8450c06c4a	2018-06-14 15:13:46	mysquid	npr.org	true
50	5585a502-5...50b8c811ad	2018-06-14 16:13:58	mysquid	npr.org	true
50	72d8bc08-6...a0f381d4ae	2018-06-14 16:44:01	mysquid	npr.org	true
50	2d6a1b69-e...7aaacf38e5	2018-06-14 17:14:06	mysquid	npr.org	true
50	b7acca5f-9...f6e34895a0	2018-06-14 17:43:13	mysquid	npr.org	true
50	c94815d7-6...4b4b241fdd	2018-06-14 14:42:43	mysquid	npr.org	true
50	84263126-8...c353ee117e	2018-06-14 15:42:52	mysquid	npr.org	true
50	ce51fcad-3...a4ecd33d10	2018-06-14 15:42:52	mysquid	npr.org	true
50	25d3169a-e...d8c9c71827	2018-06-14 16:12:58	mysquid	npr.org	true
50	66cd12c3-d...f0748ff8df	2018-06-14 17:43:13	mysquid	npr.org	true
50	e26bbf78-3...6ba57d8283	2018-06-14 17:43:13	mysquid	npr.org	true

15. To remove a filter, click **x** next to the filter.

To view all events, click **x** on the Searches field.



Searches ▸ domain_without_subdomains:npr.org ✖

Alerts (119)

Filters

enrichm...:country	1
ip_dst_addr	1
ip_src_addr	1
source:type	1

Group By: 1 source:type; 1 ip_dst_addr

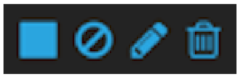
Score	id	timestamp
50	81a5245f-1...c6209aaf24	2018-06-14 14:42:43
50	a125d243-7...fe49d35dea	2018-06-14 14:43:42

Improve Scoring with a Domain Whitelist

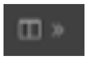
Once you have identified and investigated a potential typosquatted domain and found that it is legitimate, you can stop future alerts by using a domain whitelist enrichment.

Procedure

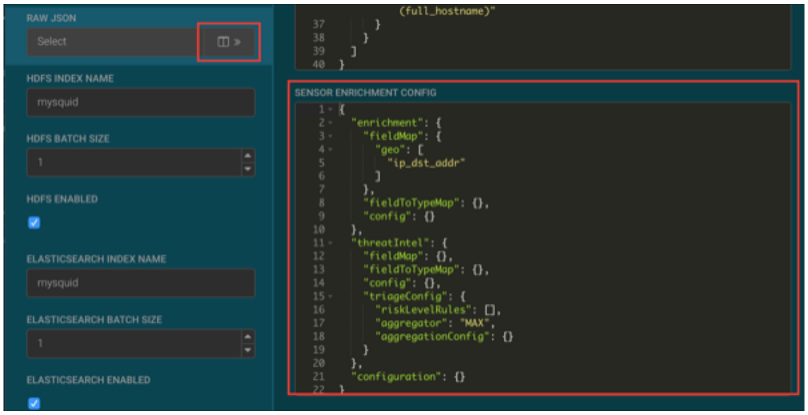
1. Display the Management module UI.
2. Select the Squid sensor from the list of sensors on the main window.
3. Click the pencil icon in the list of tool icons

   for the Squid sensor.
4. Click **Advanced**.
5. Click

   (expand window button) next to the **RAW JSON** field.

   RAW JSON: Select

   HDFS INDEX NAME: mysquid

   HDFS BATCH SIZE: 1

   HDFS ENABLED: ☑

   ELASTICSEARCH INDEX NAME: mysquid

   ELASTICSEARCH BATCH SIZE: 1

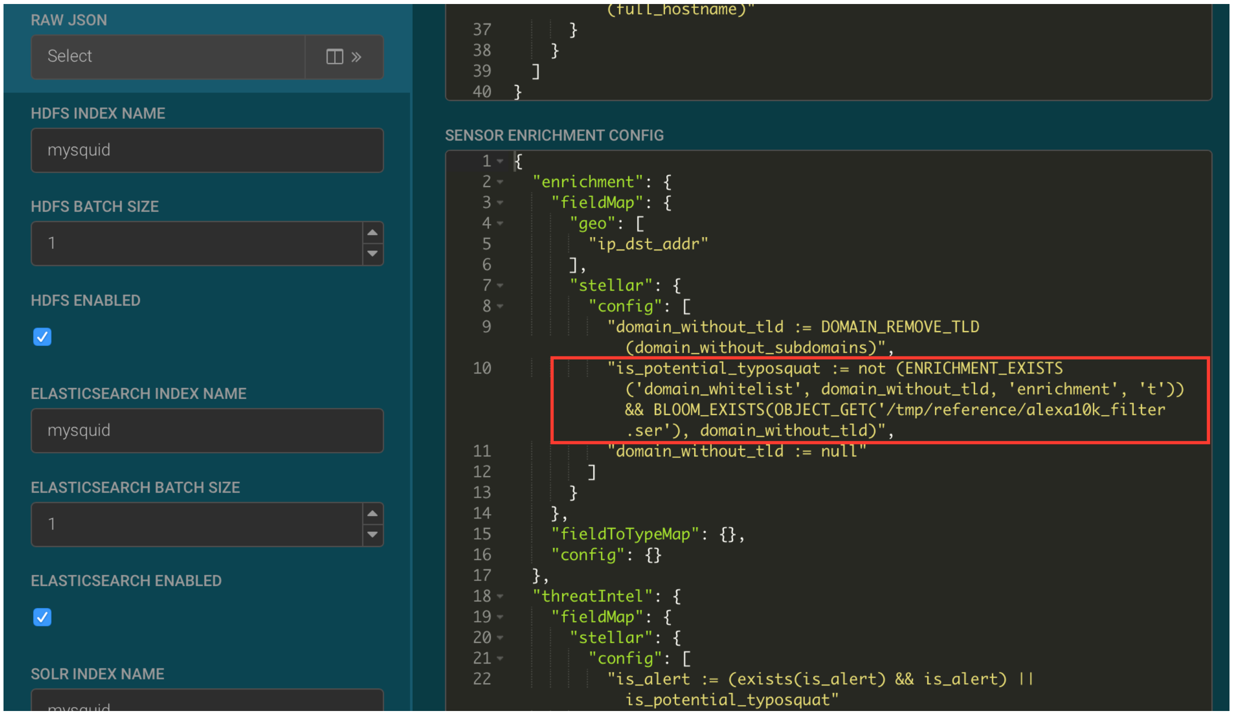
   ELASTICSEARCH ENABLED: ☑

   ```
                 (full_hostname)"
         }
       }
     ]
   }
   ```

   SENSOR ENRICHMENT CONFIG

   ```
   {
     "enrichment": {
       "fieldMap": {
         "geo": [
           "ip_dst_addr"
         ]
       },
       "fieldToTypeMap": {},
       "config": {}
     },
     "threatIntel": {
       "fieldMap": {},
       "fieldToTypeMap": {},
       "config": {},
       "triageConfig": {
         "riskLevelRules": [],
         "aggregator": "MAX",
         "aggregationConfig": {}
       }
     },
     "configuration": {}
   }
   ```
6. Replace the is_potential_typosquat information with the following:

   ```
   "is_potential_typosquat := not (ENRICHMENT_EXISTS('domain_whitelist',
    domain_without_tld, 'enrichment', 't')) && BLOOM_EXISTS(OBJECT_GET('/tmp/
   reference/alexa10k_filter.ser'), domain_without_tld)",
   ```



RAW JSON

Select

HDFS INDEX NAME

mysquid

HDFS BATCH SIZE

1

HDFS ENABLED

☑

ELASTICSEARCH INDEX NAME

mysquid

ELASTICSEARCH BATCH SIZE

1

ELASTICSEARCH ENABLED

☑

SOLR INDEX NAME

```
            (full_hostname)"
        }
      }
    ]
  }
```

SENSOR ENRICHMENT CONFIG

```
{
  "enrichment": {
    "fieldMap": {
      "geo": [
        "ip_dst_addr"
      ],
      "stellar": {
        "config": [
          "domain_without_tld := DOMAIN_REMOVE_TLD
            (domain_without_subdomains)",
          "is_potential_typosquat := not (ENRICHMENT_EXISTS
            ('domain_whitelist', domain_without_tld, 'enrichment', 't'))
            && BLOOM_EXISTS(OBJECT_GET('/tmp/reference/alexa10k_filter
            .ser'), domain_without_tld)",
          "domain_without_tld := null"
        ]
      }
    },
    "fieldToTypeMap": {},
    "config": {}
  },
  "threatIntel": {
    "fieldMap": {
      "stellar": {
        "config": [
          "is_alert := (exists(is_alert) && is_alert) ||
            is_potential_typosquat"
```

7. Click **SAVE** below the JSON panel.
8. Click **SAVE** at the bottom of the Squid sensor configuration panel.
9. Open cnn.com or npr.com in the browser connected to the HCP proxy.
10. Open the Alerts UI.
11. Click on the **timestamp** column header until the events are sorted descending by timestamp.

    Proxy events to cnn.com and npr.org are no longer alerts.

Testing

After you install, configure, and start Profiler, you should validate that the Porfiler is working correctly.

Procedure

1. Login to the server hosting Metron.
2. Enter the following command:

```
[root@node1 0.3.0]# bin/stellar -z $ZOOKEEPER_HOST:2181
Stellar, Go!
Please note that functions are loading lazily in the background and will
 be unavailable until loaded fully.
{es.clustername=metron, es.ip=node1, es.port=9300,
 es.date.format=yyyy.MM.dd.HH}

[Stellar]>>> ?PROFILE_GET
Functions loaded, you may refer to functions now...
PROFILE_GET
Description: Retrieves a series of values from a stored profile.

Arguments:
    profile - The name of the profile.
    entity - The name of the entity.
    durationAgo - How long ago should values be retrieved from?
    periods - The list of profile periods to grab. These are ProfilePeriod
 objects.
    groups - Optional, must correspond to the 'groupBy' list used in
 profile creation - List (in square brackets) of groupBy values used to
 filter the profile. Default is the empty list, meaning groupBy was not
 used when creating the profile.
    config_overrides - Optional - Map (in curly braces) of name:value
 pairs, each overriding the global config parameter of the same name.
 Default is the empty Map, meaning no overrides.
Returns: The profile measurements.

[Stellar]>>> PROFILE_GET('test','192.168.138.158', 1, 'HOURS') [12078.0,
 8921.0, 12131.0]
```

In the preceeding example, we use the Stellar Shell to replicate the execution environment of Stellar running in a Storm topology, like Metron's Parser or Enrichment topology. Replace 'node1:2181' with the URL to a ZooKeeper Broker.

The client API call above retrieves the past hour of the test profile for the entity 192.168.138.158.