Threat Triage

In the previous section, Application of Threat Intel Fields, we walked through how to load threat intel data into Metron and then apply those threat intels in realtime as telemetry events are streamed through the platform.

The problem, however, is that not all threat intelligence indicators are made equal. Some require immediate response, whereas others can be dealt with or investigated as time and availability permits. What we need is the ability to triage and rank threats by severity.

Now that we know what we should do, the next question is how to accomplish it; in other words, we must define what exactly we mean when we say "severity." The capability, as implemented in Metron, is accomplished by providing the ability to associate possibly complex conditions to numeric scores. Then, for each message, the set of conditions are evaluated and the set of numbers for matching conditions are aggregated via a configurable aggregation function. This aggregated score is added to the message in the `threat.triage.level`. Let's dig a bit deeper into this and provide an example.

- Metron Query Language Explained
- Threat Triage Configuration Explained
- Step 1: Setup and Prerequisites
- Step 2: Create the Threat Triage Rule Configuration
- Step 3: Upload the Threat Triage Configuration to Zookeeper
- Step 4: View Triaged/Scored Alerts
    - View Non Threat Data
    - Threat Data from alamman.com has a triage level of 5
    - Threat Data from atmape.ru has a triage level of 10
    - Metron UI Triaged Alerts Panel

Metron Query Language Explained

The heart of the problem is how we define a "condition." In Metron, we provide a custom domain specific language for defining conditions.

The query language supports the following:

- Referencing fields in the enriched JSON
- Simple boolean operations:
    - `and, &&`
    - `not`
    - `or, ||`
- Determining whether a field exists (via `exists`)
- The ability to have parenthesis to make order of operations explicit
- A fixed set of functions which take strings and return boolean. Currently:
    - `IN_SUBNET(ip, cidr1, cidr2, ...)`
    - `IS_EMPTY(str)`
    - `STARTS_WITH(str, prefix)`
    - `ENDS_WITH(str, suffix)`
    - `REGEXP_MATCH(str, pattern)`
- A fixed set of string-to-string transformation functions. Currently:
    - `TO_LOWER`
    - `TO_UPPER`
    - `TRIM`

Consider, for example, the following JSON message:

```
{
    ...
  "src_ip_addr" : "192.168.0.1"
 ,"is_local" : true
    ...
}
```

Consider the query:

```
IN_SUBNET( src_ip_addr, '192.168.0.0/24') or src_ip_addr in [ '10.0.0.1', '10.0.0.2' ] or exists
(is_local)
```

This evaluates to true precisely when one of the following is true for a message:

- The value of the `src_ip_addr` field is in the `192.168.0.0/24` subnet
- The value of the `src_ip_addr` field is `10.0.0.1` or `10.0.0.2`

- The field `is_local` exists

More information can be found here: Metron Query Language

Threat Triage Configuration Explained

Now that we have the ability to define conditions, for each sensor we need to associate these conditions to scores. Since this is a per-sensor configuration, this fits nicely within the sensor enrichment configuration held in Zookeeper. This configuration fits well within the `threatIntel` section of the configuration like so:

```
{
  ...
  ,"threatIntel" : {
            ...
           , "triageConfig" : {
                     "riskLevelRules" : [
{
                                 “name” : “ "
                                 “comment” : “  "
                                 “rule”: "  ”
                                 “score” :
  }

              ,"aggregator" : "MAX"
               ,"aggregationConfig" : { }
                      }
            }
}
```

- `riskLevelRules` correspond to the set of condition-to-numeric-level mappings that define the threat triage for this particular sensor.
  - name: The name of the threat triage rule.
  - comment: A comment describing the rule.
  - rule: The rule, represented as a Stellar statement.
  - score: Associated threat triage score for the rule.
- `aggregator` is an aggregation function that takes all non-zero scores representing the matching queries from `riskLevelRules` and aggregates them into a single score. The current supported aggregation functions are the following:
  - `MAX` : The max of all of the associated values for matching queries.
  - `MIN` : The min 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.

Step 1: Setup and Prerequisites

1. Complete the instructions in Adding a new Telemetry Data Source.
2. Make sure the following variables are configured based on your environment:

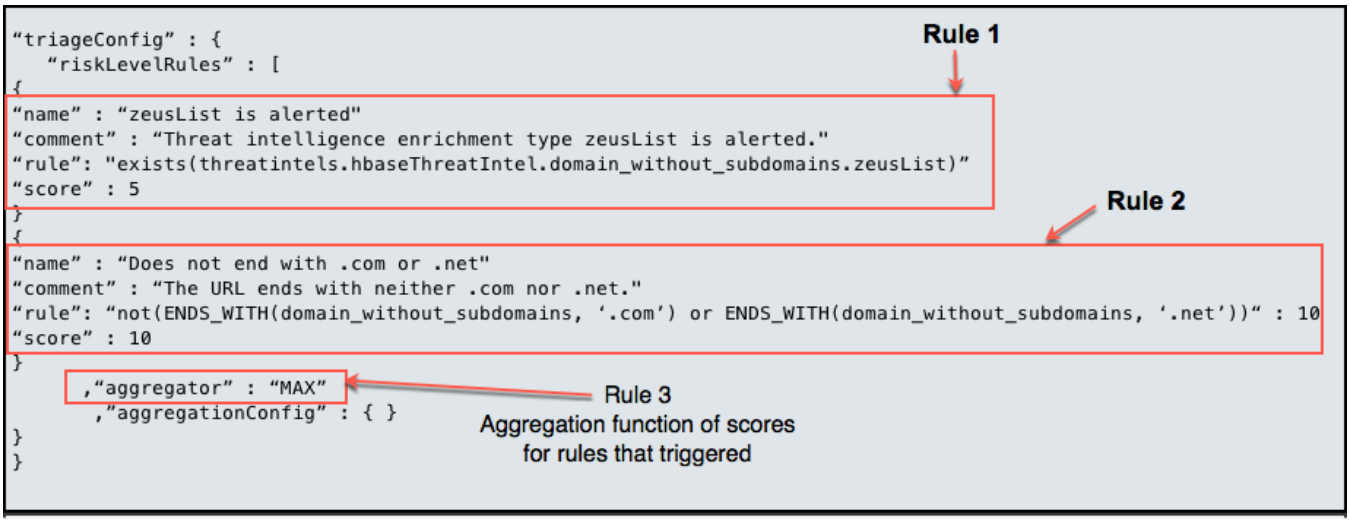
   - KAFKA_HOST = The host where a Kafka broker is installed.
   - ZOOKEEPER_HOST = The host where a Zookeeper server is installed.
   - PROBE_HOST = The host where your sensor, probes are installed. If don't have any sensors installed, pick the host where a Storm supervisor is running.
   - SQUID_HOST = The host where you want to install SQUID. If you don't care, just install SQUID on the PROBE_HOST.
   - NIFI_HOST = Host where you will install NIFI. You want this this to be 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. (e.g., 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. (e.g., 0.2.0BETA-RC2)

Step 2: Create the Threat Triage Rule Configuration

In the previous article, Application of Threat Intel Feed, we left off with a working threat intelligence enrichment. Now, let's see if we can triage those threats for the Squid data flowing through. In particular, let's triage the threat alerts for the `squid`sensor data that are higher under the following conditions:

- Rule 1: If the threat intel enrichment type `zeusList` as defined in the previous article is alerted, then we want to consider that an alert score of 5.
- Rule 2: If the `url` is neither a `.com` nor a `.net`, then we want to consider that an alert score of 10.
- Rule 3: For each message, the triage score is the maximum score across all conditions.

For each message we will assign the maximum score across all conditions as the triage score. This translates into the following configuration:



Step 3: Upload the Threat Triage Configuration to Zookeeper

In order to apply this triage configuration, we must modify the configuration for the `squid` sensor in the enrichment topology.

1. Log into $HOST_WITH_ENRICHMENT_TAG as root.
2. We need to modify `/usr/metron/$METRON_RELEASE/config/zookeeper/sensors/squid.json.` However, since the configuration in Zookeeper may be out of sync with the configuration on disk, we must make sure they are in sync by downloading the Zookeeper configuration first:

   ```
   /usr/metron/$METRON_RELEASE/bin/zk_load_configs.sh -m PULL -z $ZOOKEEPER_HOST:2181 -f -o /usr/metron
   /$METRON_RELEASE/config/zookeeper
   ```

3. Validate that the enrichment config for Squid exists.

   ```
   cat /usr/metron/$METRON_RELEASE/config/zookeeper/enrichments/squid.json
   ```

4. Edit the configuration. In `/usr/metron/$METRON_RELEASE/config/zookeeper/enrichments/squid.json` add the following to the triageConfig section to the threat intel section.

   ```
   "threatIntel" : {
       "fieldMap" : {
         "hbaseThreatIntel" : [ "domain_without_subdomains" ]
       },
       "fieldToTypeMap" : {
         "domain_without_subdomains" : [ "zeusList" ]
       },
       "config" : { },
       "triageConfig" : {
         "riskLevelRules" : {
            "exists(threatintels.hbaseThreatIntel.domain_without_subdomains.zeusList)" : 5
                 , "not(ENDS_WITH(domain_without_subdomains, '.com') or ENDS_WITH(domain_without_subdomains, '.
   net'))" : 10
   ```

```
                        }
        ,"aggregator" : "MAX"
        ,"aggregationConfig" : { }
                    }
                }
    }
```

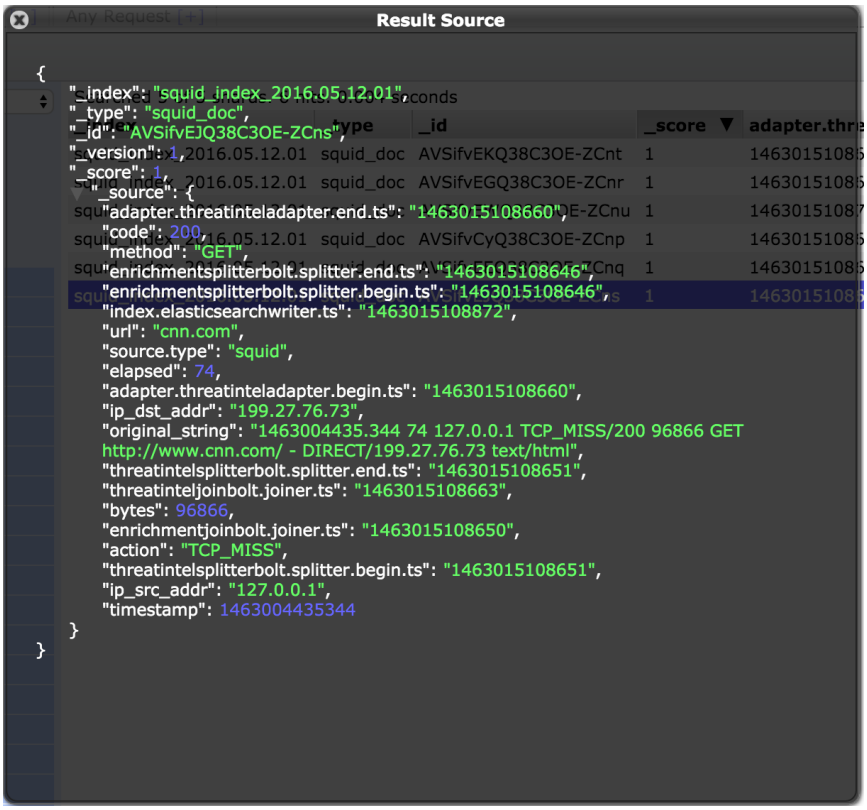
5. Ensure that the `aggregator` field indicates `MAX`.
6. After modifying the configuration, we can push the configuration back to Zookeeper and have the enrichment topology pick it up with live data by running the following:

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

Step 4: View Triaged/Scored Alerts

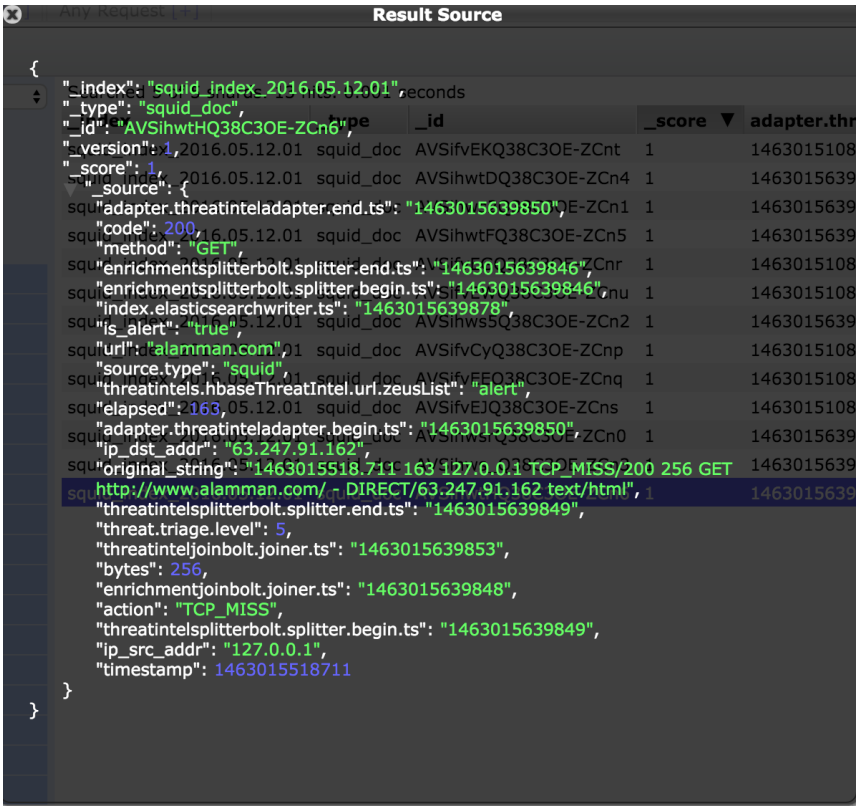
View Non Threat Data

For URL's from cnn.com, we see no threat alert, so no triage level is set. Notice the lack of a `threat.triage.level` field.



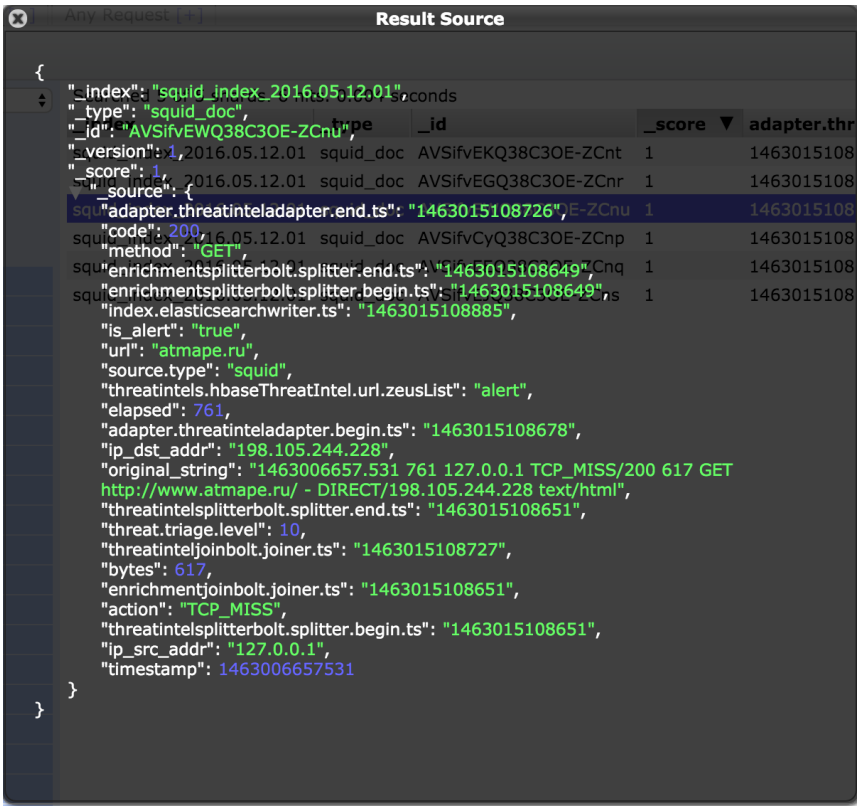
Threat Data from alamman.com has a triage level of 5

Because alamman.com is a malicious host from the `zeusList` threat intel feed but is a `.com` address, it's assigned a `threat.triage.level` of 5.



Threat Data from atmape.ru has a triage level of 10

Because atmape.ru is both a malicious host from the `zeusList` threat intel feed as well as a non `.com` and non `.net` address, it's assigned a `threat.triage.level` of 10.



Metron UI Triaged Alerts Panel