Application of Threat Intel Feeds

In the previous article of the series, Enriching Telemetry Events, we walked through how to enrich a domain element of a given telemetry event with WhoIs data like home country, company associated with domain, etc. In this article, we will enrich with a special type of data called threat intel feeds. When a given telemetry event matches data in a threat Intel feed, the system generates an alert.

Again, the customers requirement are the following:

1. The proxy events from Squid logs must be ingested in real-time.
2. The proxy logs must be parsed into a standardized JSON structure that Metron can understand.
3. In real-time, the squid proxy event must be enriched so that the domain named are enriched with the IP information
4. **In real-time, the IP with in the proxy event must be checked for threat intel feeds.**
5. **If there is a threat intel hit, an alert must be raised.**
6. The end user must be able to see the new telemetry events and the alerts from the new data source.
7. All of this requirements must be implemented easily without writing any new Java code.

In this article, we will walk you through how to meet requirements 4 and 5.

Threat Intel Framework Explained

Metron currently provides an extensible framework to plug in threat intel sources. Each threat intel source has two components: an enrichment data source and and enrichment bolt. The threat intelligence feeds are bulk loaded and streamed 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 Metron. Metron additionally provides a flat file and Stix bulk loader that can normalize, dedup, and bulk load or stream threat intel data into HBase even without the use of a threat feed aggregator.

The following diagram illustrates the architecture:

blocked URL

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 a Mock Threat Intel Feed Source

Metron 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 will use a Zeus malware tracker list located here: https://zeustracker.abuse.ch/blocklist.php?download=domainblocklist.

1. Log into the $HOST_WITH_ENRICHMENT_TAG as root.
2. Let's copy the contents from that 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
```

Step 3: Configure an Extractor Config File

1. Log into the $HOST_WITH_ENRICHMENT_TAG as root.
2. Now that we have the "Threat Intel Feed Source," we need to configure an extractor config file that describes the source. Create a file called extractor_config_temp.json and add the following content to it.

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

3. Run the following command to remove the non-ascii characters:

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

Step 4: Configure Element to Threat Intel Feed Mapping

We now need to configure which element of a tuple and which threat intel feed with which to cross-reference.This configuration will be stored in Zookeeper.

1. Log into the $HOST_WITH_ENRICHMENT_TAG as root.
2. Cut and paste this file into a file called "enrichment_config_temp.json."

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

3. Because copying and pasting from this blog will include some non-ascii invisible characters, run the following command to strip them out: iconv -c -f utf-8 -t ascii enrichment_config_temp.json -o enrichment_config.json

Step 5: Run the Threat Intel Loader

Now that we have the threat intel source, threat intel exractor, and threat intel mapping config defined, we 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.

1. Log into the $HOST_WITH_ENRICHMENT_TAG as root.
2. Run the loader.
```
/usr/metron/$METRON_RELEASE/bin/flatfile_loader.sh -n enrichment_config.json -i domainblocklist.csv -t
threatintel -c t -e extractor_config.json
```

3. The previous command adds the threat intel data into HBase and establishes a Zookeeper mapping. The data is populated into an HBase table called threatintel. To verify that the logs were properly ingested into HBase, run the following command:
```
hbase shell
scan 'threatintel'
```

4. Now check if the Zookeeper enrichment tag was properly populated:

```
/usr/metron/$METRON_RELEASE/bin/zk_load_configs.sh -m DUMP -z $ZOOKEEPER_HOST:2181
```

5. You should see a config for the Squid sensor something like the following:

```
ENRICHMENT Config: squid
{
  "index" : "squid",
  "batchSize" : 1,
  "enrichment" : {
    "fieldMap" : {
      "hbaseEnrichment" : [ "domain_without_subdomains" ]
    },
    "fieldToTypeMap" : {
      "domain_without_subdomains" : [ "whois" ]
    },
    "config" : { }
  },
  "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" : { }
                                }
                            }
  },
  "configuration" : { }
}
```

Threat Intel Config

6. Generate some data by using the Squid client to execute http requests. (Do this about 20 times.)
   `squidclient http://www.actdhaka.com`

Step 6: View the Threat Alerts in Metron UI

Now that we have configured real-time threat intel cross referencing so that alerts get generated when there is a hit for the Squid sensor, let's render these alerts on the Metron UI. We will be adding two new panels to visualize the Squid Alerts.

Creating a Threat Intel Hits Count Panel

1. Log into the Metron UI Dashboard: http://METRON_UI_HOST:5000.
2. Select "Visualize" Tab --> Select "Metric" Visualization"= --> Select "From a new search" for Search Source --> Select "squid*" index source.
3. In the search box, enter "is_alert = true" then execute the search.
4. Click the Save disk icon on the top right and name the Visualization "Threat Intel Hits," then click Save.
5. Select "Dashboard" Tab --> Click the plus icon --> Select "Visualization" tab --> Search for "Squid Event Count" --> Select it.
   The visualization will be added to the bottom of the dashboard.
6. Click the save icon on the top right to save the dashboard.

Creating an Alert Detail Panel

1. Log into the Metron UI Dashboard: http://METRON_UI_HOST:5000.
2. Select "Discover" Tab --> Select the "squid*" index.
3. Search only for alerts in the Squid index.
   a. Type the following in search:
      "is_alert = true"
   b. Click the search icon
4. Now we only need to select a subset of the fields that we want to display in the detail panel. In the left hand panel under "Available Fields", add the following fields:

full_hostname
ip_src_addr
ip_dst_addr
original_string
method
type

Dashboard with the Two Panels

The following is what the new dashboard will look like with these two new panels.