Metron Tutorial - Fundamentals Part 4: Pluggable Threat Intelligence

Now that we know how to add telemetries and enrichments, as well as how to setup a test framework and troubleshoot them, let's move on to the last step of this blog series and talk about adding threat intelligence. 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. Similarly to enrichments we need to setup a data.csv file, the extractor config JSON and the enrichment config JSON.

~~For this example we will be using a Zeus malware tracker list located here: https://zeustracker.abuse.ch/blocklist.php?download=domainblocklist~~

Update 8/23/19 - The Zeus tracker list was discontinued on July 8, 2019.

For this example we will be using a Squid blacklist malware tracker list located here: https://www.squidblacklist.org/downloads/dg-malicious.acl

```
curl -o domainblocklist.txt https://www.squidblacklist.org/downloads/dg-malicious.acl
```

Similarly to enrichment we will need to process this feed into a CSV so we can bulk load it into HBase. After we process the feed (here is a sample script for doing so):

```
cat domainblocklist.txt | grep -v "^#" | grep -v "^$" | grep -v "^https" | awk '{print $1",squidblacklist.org"}' > domainblocklist.csv
```

And produce our domainblocklist.csv that would would look as follows (lets focus on the two specific domains from the list):

```
....
accounts-google.ru,squidblacklist.org
webtahmin.com,squidblacklist.org
.....
```

Now that we have the CSV of threat intel extracted we need to define our threat intel configs similarly to how we defined them for enrichment.

Now let's define our threat intel enrichment config by placing the following in a file named threatintel_config_temp.json. Replace $ZOOKEEPER with your quorum:

```
{
  "zkQuorum" : "$ZOOKEEPER"
 ,"sensorToFieldList" : {
    "squid" : {
          "type" : "THREAT_INTEL"
         ,"fieldToEnrichmentTypes" : {
            "domain_without_subdomains" : [ "squidBlacklist" ]
         }
    }
  }
}
```

Again we need to remove non ascii characters we run this:

```
iconv -c -f utf-8 -t ascii threatintel_config_temp.json -o threatintel_config.json
```

And now we define the extractor config and place it in a file named threatintel_extractor_config_temp.json:

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

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

And to remove the non-ascii characters we run the following:

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

Now we run the following command to bulk load the threat intel:

```
${METRON_HOME}/bin/flatfile_loader.sh -n threatintel_config.json -i domainblocklist.csv -t threati
ntel -c t -e threatintel_extractor_config.json
```

This command will modify the squid enrichment config in Zookeeper to include the threat intel enrichment as well as import the threat intel data to HBase to a table named "threatintel". There should be around 168k records added.

```
[root@node1: ~]
# echo "count 'threatintel'" | hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.1.2.2.6.5.1175-1, r897822d4dd5956ca186974c10382e9094683fa29, Thu Jun 20 17:08:24 UTC
2019

count 'threatintel'
Current count: 1000, row: \x01l\xA9M\xB4\x8F]p~\x9E\x9B\x0Ceg\xD5M\x00\x0EsquidBlacklist\x00\x15lp
.autocleantools.com
Current count: 2000, row:
\x02\xFB\x92\xBEC\x83G\xD7\x853\x02GX\xF9\xD7d\x00\x0EsquidBlacklist\x00\x09kraken.cc
Current count: 3000, row: \x04\x8A*\x83
(\xF7P\xBD7Y\x13\xE6\xBD\xBA\xCA\xE4\x00\x0EsquidBlacklist\x00\x085inv.biz

...

Current count: 166000, row: \xFBqYw\x19\xF8>_P9US\xED\xAFW\xF1\x00\x0EsquidBlacklist\x00\x0Brosehi
ll.hu
Current count: 167000, row: \xFC\xF8\xD3\x03\xA7\xCE\x1E\x086Sfd@Sw\x12\x00\x0EsquidBlacklist\x00\
x14selfpackshipping.com
Current count: 168000, row:
\xFEyE\xD1\x03gG\xF5\xE7T\x9B\xDD\x8F\xE1\xBB\xBB\x00\x0EsquidBlacklist\x00\x11timetodoright.org
168979 row(s) in 13.3170 seconds

168979
```

You should see a parser config that looks like the following:

```
[root@node1: ~]
# ${METRON_HOME}/bin/zk_load_configs.sh -m DUMP -z $ZOOKEEPER -c PARSER -n squid

PARSER Config: squid

{
  "parserClassName": "org.apache.metron.parsers.GrokParser",
```

```
  "sensorTopic": "squid",
  "parserConfig": {
    "grokPath": "/patterns/squid",
    "patternLabel": "SQUID_DELIMITED",
    "timestampField": "timestamp"
  },
  "fieldTransformations" : [
    {
      "transformation" : "STELLAR"
    ,"output" : [ "full_hostname", "domain_without_subdomains" ]
    ,"config" : {
      "full_hostname" : "URL_TO_HOST(url)"
      ,"domain_without_subdomains" : "DOMAIN_REMOVE_SUBDOMAINS(full_hostname)"
                }
    }
                            ]
}
```

And an enrichment config that looks like this:

```
[root@node1: ~]
# ${METRON_HOME}/bin/zk_load_configs.sh -m DUMP -z $ZOOKEEPER -c ENRICHMENT -n squid

ENRICHMENT Config: squid

{
  "enrichment" : {
    "fieldMap" : {
      "hbaseEnrichment" : [ "domain_without_subdomains" ]
    },
    "fieldToTypeMap" : {
      "domain_without_subdomains" : [ "whois" ]
    },
    "config" : { }
  },
  "threatIntel" : {
    "fieldMap" : {
      "hbaseThreatIntel" : [ "domain_without_subdomains" ]
    },
    "fieldToTypeMap" : {
      "domain_without_subdomains" : [ "squidBlacklist" ]
    },
    "config" : { },
    "triageConfig" : {
```

```
      "riskLevelRules" : [ ],
      "aggregator" : "MAX",
      "aggregationConfig" : { }
    }
  },
  "configuration" : { }
}
```

We'll want to maintain a current set of local configs to continue working from, so we'll want to pull them locally. To pull these modifications locally, execute the following:

```
${METRON_HOME}/bin/zk_load_configs.sh -m PULL -z $ZOOKEEPER -o ${METRON_HOME}/config/zookeeper -f
```

(Optional) Now let's drop the Elasticserach squid indexes.

```
curl -XDELETE "http://${ELASTICSEARCH}:9200/squid*"
```

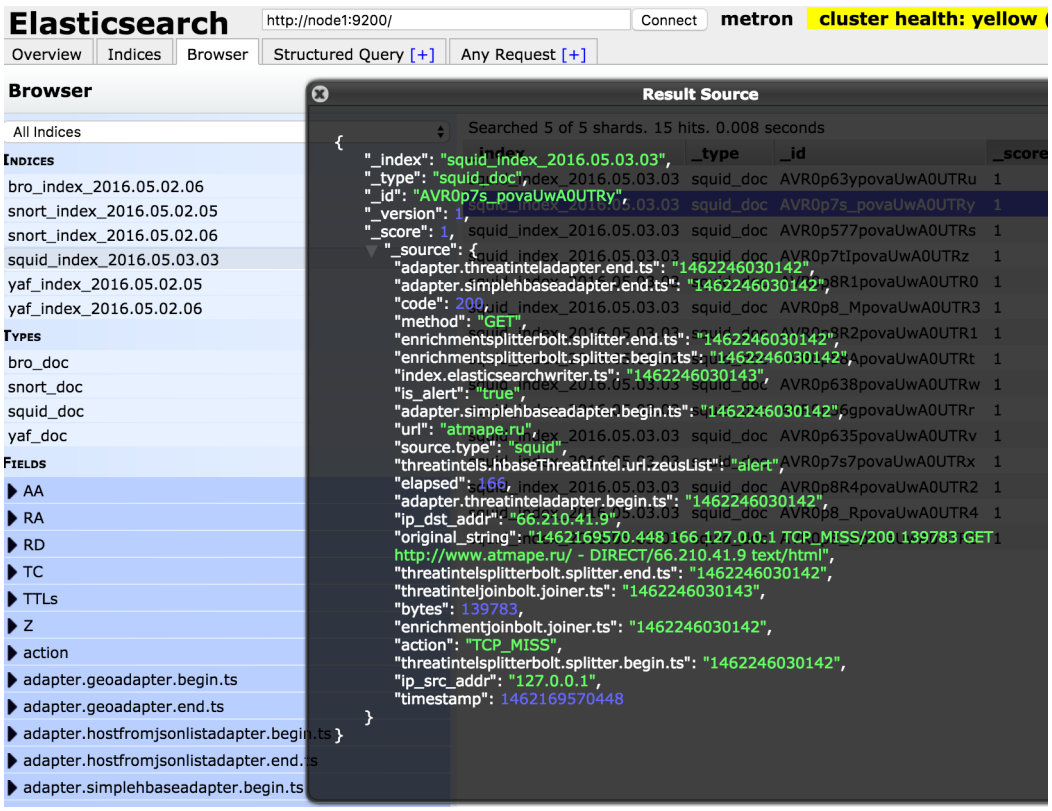
After dropping the indexes we re-ingest. Let's trigger on two of the domains we ingested (note, this list is constantly changing, so verify these domains do in fact exist in the domainblocklist.csv before triggering the squidclient. If either/both are not in the list, choose another domain):

```
squidclient http://kapriz-podolsk.ru
squidclient http://webtahmin.com
```

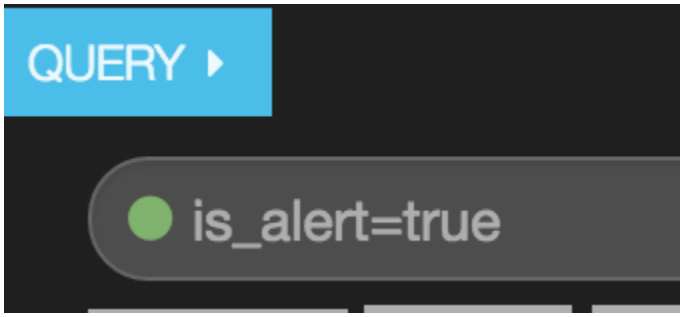
Push the new squid log entries into the squid Kafka topic:

```
tail -f /var/log/squid/access.log -n 2 | ${HDP_HOME}/kafka-broker/bin/kafka-console-producer.sh --
broker-list $BROKERLIST --topic squid
```

When the logs are ingested we get messages that has a hit against threat intel:



Notice a couple of characteristics about this message. It has is_alert=true, which designates it as an alert message. It also tells us which field received a hit against threat intel (url.zeusList). Now that we have alerts coming through we need to visualize them in Kibana. First, we need to setup a pinned query to look for messages where is_alert=true:



And then once we point the alerts table to this pinned query it looks like this: