BIG DATA CYBERSECURITY

LAB 6 THREAT INTELLIGENCE

Cyber Threat Triage

Lab Description: Metron Cybersecurity Operations Center (SOC) may process data from multiple log files and network sensors, which may overwhelm a cybersecurity analyst due to the large number of produced alerts. To facilitate their ranking, an approach was borrowed from the healthcare field, known as triage—the sorting of patients (as in an emergency room) according to the urgency of their need for care [1]. In cybersecurity, triage is commonly done by severity. Alerts are assigned scores according to their severity and those with the highest potential damage are processed first [2].

Metron is capable of processing two community-supported cybersecurity data formats for prevention and mitigation attacks, TAXII—the Trusted Automated eXchange of Indicator Information, and STIX—the Structured Threat Information eXpression. TAXII is an application layer protocol for exchanging cyberthreat information over HTTPS, while STIX is a language and serialization format for describing and storing cyberthreat data [3, 4]. Additionally, Metron is capable of processing custom data formats given that a description is provided in a CSV form.

This assignment explores the use of Metron with a custom cybersecurity data collected by the ZeuS Tracker, which builds a blocklist of IP addresses and domain names related to the ZeuS malware [5], which has been used to secretly capture victims' bank account numbers, passwords, and other information. Subsequently, hackers used it to take over victims' bank accounts and make unauthorized wire transfers through middlemen's accounts [6]. The global awareness of ZeuS began in 2007 when it was detected in the U.S. Department of Transportation and several corporate networks [7]. In 2010, FBI arrested a number of people charged with stealing millions of dollars from victimized accounts. Zeus source code was published online in 2011 [8] and facilitated spawning several malicious descendants [9]. More than a hundred of ZeuS command and control servers were reported online in 2019 [5].

Lab Files that are Needed: metron-cluster.ppk.

LAB EXERCISE/STEP 1

In the AWS cloud, create a new Metron virtual machine (VM) comprising a single-node Hadoop cluster as described in Lab 1. Establish an SSH connection with the VM as specified in Steps 1 through 4 of Lab 2. Note: it is recommended setting up a new VM to avoid misconfiguration because of the settings kept from other assignments.

LAB EXERCISE/STEP 2

To ingest ZeuS Tracker blocklist with Metron, we need to provide with a specification file in CSV format, a JSON file to configure a data extractor and another JSON file with enrichment configuration. The blocklist is available for download at [5]. If the blocklist is unavailable, download the Gameover Zeus DGA sample dataset from [12]. Begin by switching to the root user, changing to the home directory and downloading the blocklist file:

```
sudo su
cd ~
curl -o domainblocklist.txt
https://zeustracker.abuse.ch/blocklist.php?download=domainblocklist
```

After executing the commands above, domainblocklist.txt will be created in the home directory. To inspect its contents do:

```
vi domainblocklist.txt
```

Exit the file by typing

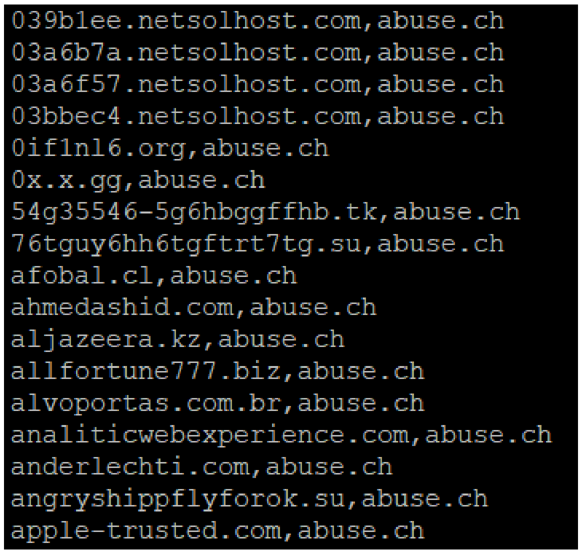
```
:q
```

LAB EXERCISE/STEP 3

The following Linux shell command will read the contents of domainblocklist.txt, process it by removing comment lines and empty lines, append abuse.ch to each data line, and store the output in CSV format as domainblocklist.csv.

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

The domainblocklist.csv should look similar to this:



```
039b1ee.netsolhost.com,abuse.ch
03a6b7a.netsolhost.com,abuse.ch
03a6f57.netsolhost.com,abuse.ch
03bbec4.netsolhost.com,abuse.ch
0if1nl6.org,abuse.ch
0x.x.gg,abuse.ch
54g35546-5g6hbggffhb.tk,abuse.ch
76tguy6hh6tgftrt7tg.su,abuse.ch
afobal.cl,abuse.ch
ahmedashid.com,abuse.ch
aljazeera.kz,abuse.ch
allfortune777.biz,abuse.ch
alvoportas.com.br,abuse.ch
analiticwebexperience.com,abuse.ch
anderlechti.com,abuse.ch
angryshippflyforok.su,abuse.ch
apple-trusted.com,abuse.ch
```

Exit the file by typing

```
:q
```

Now, we need to remove ASCII characters from the blocklist.csv with the command below:

```
iconv -c -f utf-8 -t ascii domainblocklist.csv -o domainblocklist_clean.csv
```

LAB EXERCISE/STEP 4

Next, we need to define our threat intel enrichment configuration. To do so, create a new file named *threatintel_config_temp.json*

```
vi threatintel_config_temp.json
```

Switch to the Insert mode by pressing the i button. Then paste the following JSON content into it:

```
{
  "zkQuorum" : "metronserver.localdomain:2181","
 ,"sensorToFieldList" : {
    "squid" : {
           "type" : "THREAT_INTEL"
          ,"fieldToEnrichmentTypes" : {
             "domain_without_subdomains" : [ "zeusList" ]
```

```
      }
   }
}
```

Now, save the file and exit by pressing Escape and then :wq

To avoid errors in Metron, the file we have created must be stripped off of non-ASCII characters. To do so, run the command below:

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

It is time to create a JSON file with the extractor configuration:

```
vi threatintel_extractor_config_temp.json
```

Press the i button to switch to the Insert mode. And paste the configuration below into the file:

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

Press Escape then type :wq and press Enter to save and exit the file. Again, to avoid Metron errors we need to clean the file by removing the non-ASCII characters by running the command below. The cleaned file will be stored as treatintel_extractor_config.json.

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

LAB EXERCISE/STEP 5

Run the command below invoking the Metron’s flat file loader to load the domainblocklist.csv file with threat intelligence data. This data will be read following the specification in threatintel_config.json, then it will be processed as specified in threatintel_extractor_config.json and stored in the threatintel HBase table.

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

To verify the content of the *threatintel* table in HBase, run the following command:

```
echo "scan 'threatintel'" | hbase shell
```

Make a screenshot and store it in the submission document.

While inspecting results, pick three domains from the output and save them using your favorite text editor. We will use them in the following steps. In this manual, we will be using the three domains shown below: *reserve.jumpingcrab.com*, *mygoodness.in.ua* and *wasabi.mine.nu*. Please note that at the time when you will be completing the assignment, these URLs might not be in the blocklist. To avoid an error, please pick the domains which are present in the blocklist at the time of completing this assignment.

LAB EXERCISE/STEP 6

Testing Metron Alerts

In this part of the assignment, we will emulate an infected computer navigating to the three domains from the blacklist, which would mean communication with one of the ZeuS command and control servers. These actions will be recorded in the Squid log file. Then we will pass the Squid log to Metron for analysis. Metron will detect malicious activity and raise alerts.

First, we need to install the Squid proxy server. Execute the following 0command to install Squid.

```
yum install squid -y
```

Start Squid by adjusting a general command to start a service in CentOS 7:

```
service squid start
```

Run the commands below, which are needed to emulate an infected computer navigating to the three ZeuS servers through the Squid proxy server.

```
squidclient http://reserve.jumpingcrab.com
squidclient http://mygoodness.in.ua
squidclient http://wasabi.mine.nu
```

Make a screenshot and store it in the submission document.

LAB EXERCISE/STEP 7

Create a Kafka topic for the Squid data source since Metron architecture needs a Kafka topic for each new telemetry data source. The Kafka topic will become a part of a Metron data stream. To create a new topic, execute the following command:

```
${HDP_HOME}/kafka-broker/bin/kafka-topics.sh --zookeeper $ZOOKEEPER
--create --topic squid --partitions 1 --replication-factor 1

$METRON_HOME/bin/start_parser_topology.sh -z $ZOOKEEPER -s squid
```

Verify that the *squid* topic is listed among existing Kafka topics:

```
${HDP_HOME}/kafka-broker/bin/kafka-topics.sh --zookeeper $ZOOKEEPER
--list
```

Make a screenshot and store it in the submission document.

LAB EXERCISE/STEP 8

Now we are ready to tackle the Metron parsing topology setup. Commonly, log files contain semi-structured data—custom rules are required to extract information. Metron offers parsers for log files ingestion—software components transforming raw data into the JSON format [10]. In this assignment, we employ a Grok-based parser to extract data from Squid logs. Grok uses text and regular expression patterns to match textual data in log files or other data sources. Information can be extracted from the default Squid access log by applying the Grok code listed below [11-13].

```
SQUID_DELIMITED %{NUMBER:timestamp}[^0-9]*%{INT:elapsed}
%{IP:ip_src_addr} %{WORD:action}/%{NUMBER:code}
%{NUMBER:bytes} %{WORD:method} %{NOTSPACE:url}[^0-
9]*(%{IP:ip_dst_addr})?
```

By design, Metron requires parses to be stored in */apps/metron/patterns/* directory in HDFS. To place the Grok parser in that directory, we need to create a file in local Linux file system containing the Grok code above, save that file and then copy it to HDFS. To create a file in local file system and open it with the vi text editor:

```
vi /tmp/squid
```

Press the i button on the keyboard to switch vi to the Insert mode allowing to enter or paste new text. Now copy and paste (or type) the Squid code listed above into the new file. To save the results, press the Escape button and the following key combination:

```
:wq
```

LAB EXERCISE/STEP 9

We need to switch to the hdfs user, because this user has access privileges for the HDFS file system. To do so:

```
su - hdfs
```

Run this command to copy the file from the local file system /tmp/squid to the /apps/metron/patterns directory in HDFS:

```
hadoop fs -put -f /tmp/squid /apps/metron/patterns/
```

And switch back to the root user:

```
exit
```

Make a screenshot and store it in the submission document.

LAB EXERCISE/STEP 10

Now that the Grok pattern is staged in HDFS we need to define a parser configuration for the Metron Parsing Topology, which is responsible for processing a sensor input in its native format and converting it to the

Metron JSON format. The parsing topology is formed by two components: (1) a Storm Kafka Spout for reading from a Kafka topic and sending data to a Storm topology and (2) Metron parser Kafka Bolt for parsing messages and sending them into a Kafka enrichment topic. Apache Zookeeper is responsible for preserving the configurations. Thus, the sensor configuration must be uploaded there after it has been created.

We need to ensure that the Zookeeper configuration stored on disk is in sync with the one in memory. To do so, we will sync it by executing the PULL command as shown below:

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

Create a Squid Grok parser configuration file:

```
vi ${METRON_HOME}/config/zookeeper/parsers/squid.json
```

Switch to the insert mode by pressing the i button. Check the file content. It must look exactly as listed below. If it is different, make corrections accordingly.

```
{
  "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)"
                   }
     } ]
}
```

To exit the Insert mode, press Escape. To save the file and exit, type:

```
:wq
```

And press Enter. In the parser configuration above, the fieldTransformations element uses the Stellar language, specific to Apache Metron, in the parser configuration [14]. The Grok Parser is set up to extract complete URLs. For the purpose of this assignment, domain names without subdomains will be sufficient. Stellar, the Metron Transformation Language, will be used for this field transformation. It supports multiple network-related string-processing functions and other operations. In the configuration above, the URL_TO_HOST function is applied to extract a hostname from a URL. For example, URL_TO_HOST('http://www.yahoo.com/foo') would yield 'www.yahoo.com'. Another function, DOMAIN_REMOVE_SUBDOMAINS deletes subdomains from a URL, e.g. DOMAIN_REMOVE_SUBDOMAINS('mail.yahoo.com') yields 'yahoo.com'. Thus, two new fields are added to each message, "full_hostname" and "domain_without_subdomains".

LAB EXERCISE/STEP 11

Now, we need to setup index types and batch sizes by adding the lines below to the squid.json file. Edit the file for editing:

```
vi ${METRON_HOME}/config/zookeeper/indexing/squid.json
```

Switch to the insert mode by pressing i. Now, paste the lines into the file:

```
{
  "hdfs":{
    "index":"squid",
    "batchSize":5,
    "enabled":true
  },
  "elasticsearch":{
    "index":"squid",
    "batchSize":5,
    "enabled":true
  },
  "solr":{
    "index":"squid",
    "batchSize":5,
    "enabled":true
  }
}
```

Press the Escape button to exit the Insert mode and type

```
:wq
```

To save and exit the file.

LAB EXERCISE/STEP 12

In order to ensure the source IP and destination IP addresses are in the proper format, we will validate the messages. To do so, we will edit a global Metron configuration in the JSON format and push it into Zookeeper, which will perform the validation. Open the file for editing:

```
vi ${METRON_HOME}/config/zookeeper/global.json
```

The file already contains information. Remember to switch to the Insert mode by pressing i. Then, locate the line containing *"parser.error.topic" : "indexing",* make a new line right below it, and paste the following lines for IPv4 validation there:

```
"fieldValidations":[
  {
    "input":[
      "ip_src_addr",
      "ip_dst_addr"
    ],
    "validation":"IP",
    "config":{
      "type":"IPV4"
    }
  }
],
```

After pasting, content indentation may be lost. Correct formatting to keep the content readable. Then save and exit the file. Metron has a script to upload configurations to Zookeeper. To upload the changes we have just completed, run the command below. Notice that it uses the PUSH option.

```
${METRON_HOME}/bin/zk_load_configs.sh -i
${METRON_HOME}/config/zookeeper -m PUSH -z $ZOOKEEPER
```

In order to verify that the configuration has been changed successfully, execute the same script with the DUMP option:

```
${METRON_HOME}/bin/zk_load_configs.sh -m DUMP -z $ZOOKEEPER
```

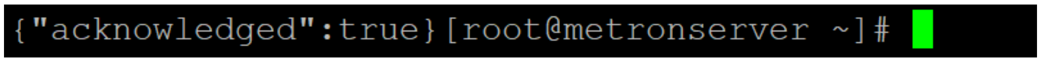
Its output will list a number of configurations in the JSON format. The global config will be shown on the very top of the output. To see it, you will need to scroll up the terminal window.

LAB EXERCISE/STEP 13

Now, install an Elasticsearch template for your new sensor so that we can effectively query results in the Metron Alerts UI. Run the following command.

```
curl -XPUT 'http://metronserver.localdomain:9200/_template/squid_index'
-d '
{
  "template": "squid_index*",
  "mappings": {
    "squid_doc": {"dynamic_templates": [
      {
        "geo_location_point": {
          "match": "enrichments:geo:*:location_point",
          "match_mapping_type": "*",
          "mapping": {"type": "geo_point"
          }
        }
      },
      {
        "geo_country": {
          "match": "enrichments:geo:*:country",
          "match_mapping_type": "*",
          "mapping": {
            "type": "keyword"
          }
        }
      },
      {
        "geo_city": {
          "match": "enrichments:geo:*:city",
          "match_mapping_type": "*",
          "mapping": {
            "type": "keyword"
          }
        }
      },
      {
        "geo_location_id": {
          "match": "enrichments:geo:*:locID",
          "match_mapping_type": "*",
          "mapping": {
            "type": "keyword"
          }
        }
      },
      {
        "geo_dma_code": {
          "match": "enrichments:geo:*:dmaCode",
          "match_mapping_type": "*",
          "mapping": {
            "type": "keyword"
          }
        }
      },
      {
        "geo_postal_code": {
          "match": "enrichments:geo:*:postalCode",
          "match_mapping_type": "*",
          "mapping": {
            "type": "keyword"
          }
        }
      },
      {
        "geo_latitude": {
          "match": "enrichments:geo:*:latitude",
          "match_mapping_type": "*",
          "mapping": {
            "type": "float"
          }
        }
      },
      {
        "geo_longitude": {
          "match": "enrichments:geo:*:longitude",
          "match_mapping_type": "*",
          "mapping": {
            "type": "float"
          }
        }
      },
      {
        "timestamps": {
          "match": "*:ts",
          "match_mapping_type": "*",
          "mapping": {
            "type": "date",             "format": "epoch_millis"
          }
        }
      },
      {
        "threat_triage_score": {
          "mapping": {
            "type": "float"
          },
          "match": "threat:triage:*score",           "match_mapping_type": "*"
        }
      },
      {
        "threat_triage_reason": {
          "mapping": {
            "type": "text",
            "fielddata": "true"
          },
          "match": "threat:triage:rules:*:reason",
  "match_mapping_type": "*"
        }
      },
      {
        "threat_triage_name": {
          "mapping": {
            "type": "text",
            "fielddata": "true"
          },
          "match": "threat:triage:rules:*:name",
  "match_mapping_type": "*"
        }
      }
      ],
      "properties": {
        "timestamp": {
          "type": "date",           "format": "epoch_millis"
        },
        "source:type": {           "type": "keyword"
        },
        "ip_dst_addr": {
          "type": "ip"
        },
        "ip_dst_port": {          "type": "integer"
        },
        "ip_src_addr": {
          "type": "ip"
        },
        "ip_src_port": {          "type": "integer"
        },
        "alert": {          "type": "nested"
        },
        "guid": {          "type": "keyword"
        }
      }
     }
    }
  }
  '
```

The output of this command should display an acknowledgement in the following format:



```
{"acknowledged":true}[root@metronserver ~]#
```

Let’s verify that the new template installed successfully by running the command:

```
curl -XGET
'http://metronserver.localdomain:9200/_template/squid_index?pretty'
```

Make a screenshot and store it in the submission document.

The output of this command will be in the JSON format. Most likely, it will not fit on the screen. To verify, scroll all the way up where you should see “squid_index” and the dynamic templates as entered above. This template serves two purposes. First, it establishes default mappings for metron-specific types such as timestamps. Second, it creates types for properties that will come from the parsed data such as ip_src_addr.

LAB EXERCISE/STEP 14

Start the new squid parser topology by executing the following line:

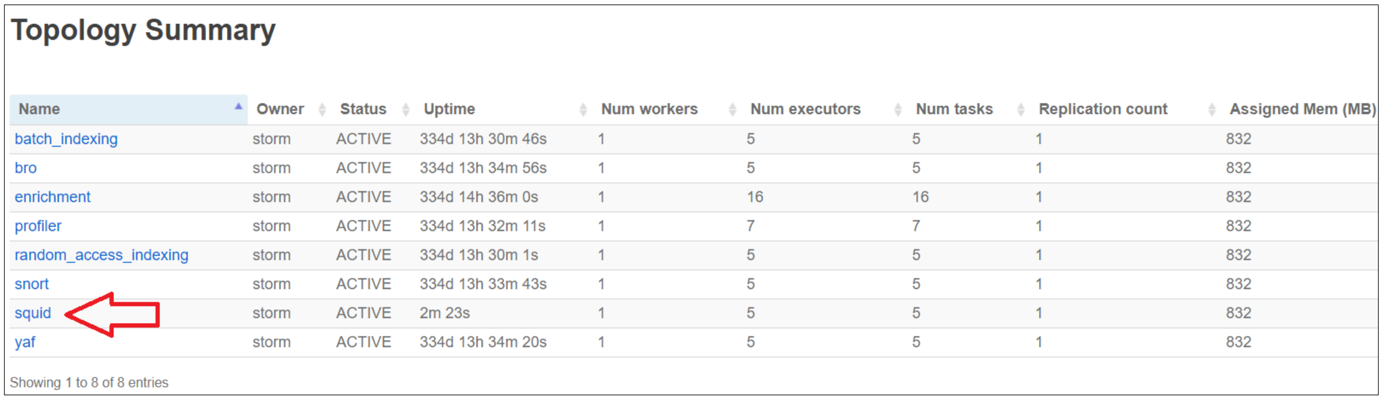
```
${METRON_HOME}/bin/start_parser_topology.sh -k $BROKERLIST -z
$ZOOKEEPER -s squid
```

Wait until the script completes. The last line of the output should look similar to this: *[main] INFO o.a.s.StormSubmitter - Finished submitting topology: squid*.

Navigate your web browser to the squid parser topology in the Storm UI at

```
http://YOURIPADDRESS:8744/index.html
```

where YOURIPADDRESS should be substituted with an actual IP address of your Metron server, e.g. http://18.188.213.228:8744/index.html. Verify the topology is in ACTIVE status and does not display errors:



Topology Summary

Name	Owner	Status	Uptime	Num workers	Num executors	Num tasks	Replication count	Assigned Mem (MB)
batch_indexing	storm	ACTIVE	334d 13h 30m 46s	1	5	5	1	832
bro	storm	ACTIVE	334d 13h 34m 56s	1	5	5	1	832
enrichment	storm	ACTIVE	334d 14h 36m 0s	1	16	16	1	832
profiler	storm	ACTIVE	334d 13h 32m 11s	1	7	7	1	832
random_access_indexing	storm	ACTIVE	334d 13h 30m 1s	1	5	5	1	832
snort	storm	ACTIVE	334d 13h 33m 43s	1	5	5	1	832
squid	storm	ACTIVE	2m 23s	1	5	5	1	832
yaf	storm	ACTIVE	334d 13h 34m 20s	1	5	5	1	832

Showing 1 to 8 of 8 entries

LAB EXERCISE/STEP 15

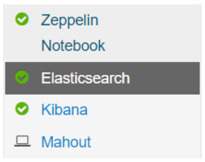
Switch back to the terminal window. Now that we have a new running squid parser topology, send the Squid log data for parsing. To do so, push the last three entries from the Squid log into the Kafka squid topic:

```
tail /var/log/squid/access.log -n 3 | ${HDP_HOME}/kafka-broker/bin/kafka-console-producer.sh --broker-list $BROKERLIST --topic squid
```

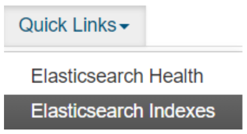
Navigate your web browser to the Ambari interface.

```
http://YOURIPADDRESS:8080
```

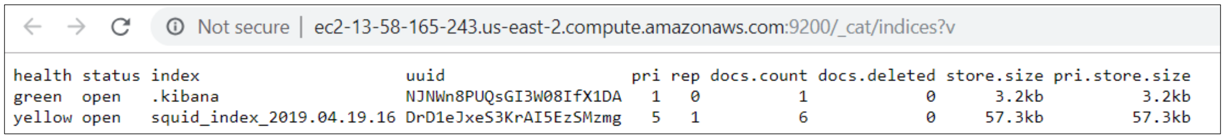
If needed, enter the default username *admin* and default password, which is also *admin*. In Ambari, scroll down to the Elasticsearch menu item and click it.



Locate the Quick Links in the left-hand-side menu in the middle of the screen. Click it and then click Elasticsearch Indexes.



This will open a webpage showing statistics on Elasticsearch indexes. Verify that a squid index has been created:

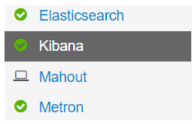

Not secure | ec2-13-58-165-243.us-east-2.compute.amazonaws.com:9200/_cat/indices?v

```
health status index                    uuid                   pri rep docs.count docs.deleted store.size pri.store.size
green  open   .kibana                  NJNWn8PUQsGI3W08IfX1DA   1   0          1            0      3.2kb          3.2kb
yellow open   squid_index_2019.04.19.16 DrD1eJxeS3KrAI5EzSMzmg   5   1          6            0     57.3kb         57.3kb
```

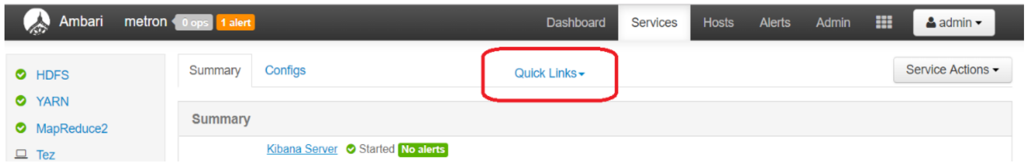
Make a screenshot and store it in the submission document.

LAB EXERCISE/STEP 16

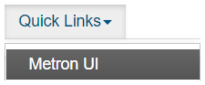
In your web browser, navigate to Ambari interface, locate and click Kibana in the left-hand-side menu.



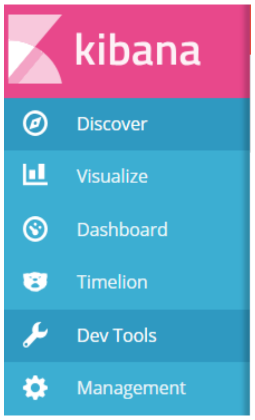
In the new screen, locate the Quick Links menu and click it.



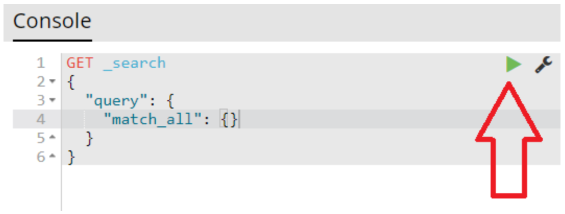
In this menu, click the Metron item.



Kibana interface will open. There, click the DevTools item in the left-hand-side menu.

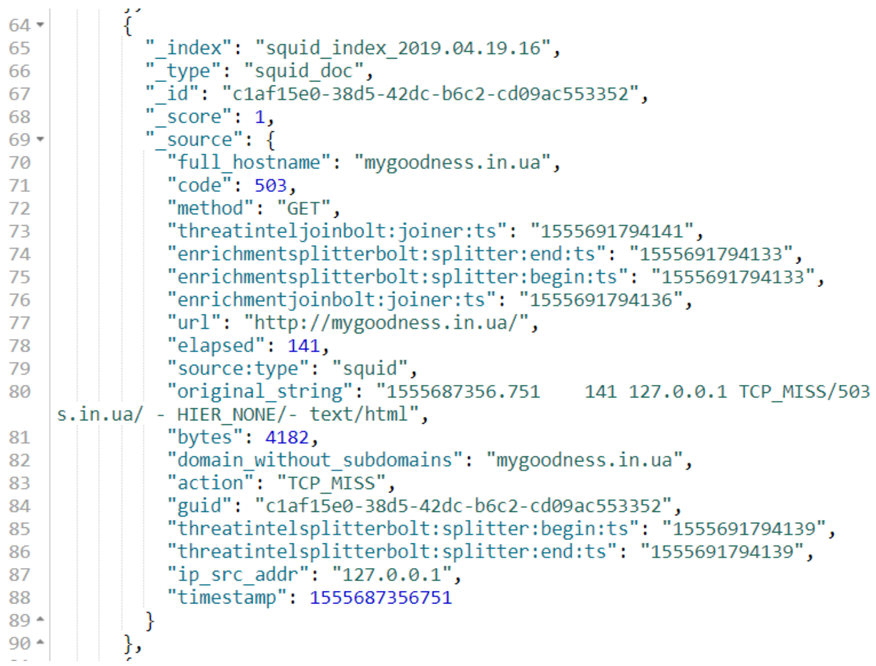


This will open the Dev Tools screen. Click the Get to Work button. The Console will resize to full screen. Console contains a JSON request to display all the data.



```
GET _search
{
  "query": {
    "match_all": {}
  }
}
```

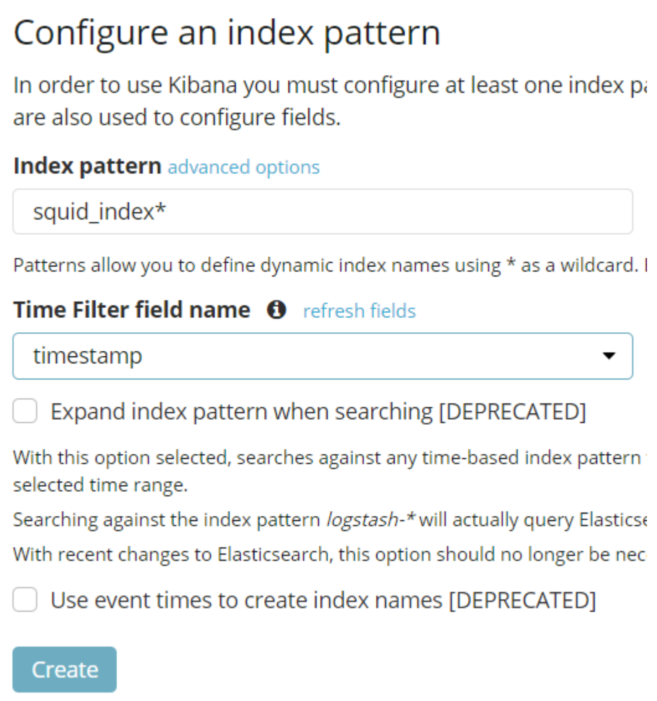
Click the green triangle pointed by the red arrow on the screenshot above. It will display parsed messages in the right-hand-side window. Be sure to scroll to the squid index. Messages within the squid index will contain those URLs classified as hits (malicious activity) according to the ZeuS data we supplied to Metron earlier in this work.



```
      {
        "_index": "squid_index_2019.04.19.16",
        "_type": "squid_doc",
        "_id": "c1af15e0-38d5-42dc-b6c2-cd09ac553352",
        "_score": 1,
        "_source": {
          "full_hostname": "mygoodness.in.ua",
          "code": 503,
          "method": "GET",
          "threatinteljoinbolt:joiner:ts": "1555691794141",
          "enrichmentsplitterbolt:splitter:end:ts": "1555691794133",
          "enrichmentsplitterbolt:splitter:begin:ts": "1555691794133",
          "enrichmentjoinbolt:joiner:ts": "1555691794136",
          "url": "http://mygoodness.in.ua/",
          "elapsed": 141,
          "source:type": "squid",
          "original_string": "1555687356.751    141 127.0.0.1 TCP_MISS/503
s.in.ua/ - HIER_NONE/- text/html",
          "bytes": 4182,
          "domain_without_subdomains": "mygoodness.in.ua",
          "action": "TCP_MISS",
          "guid": "c1af15e0-38d5-42dc-b6c2-cd09ac553352",
          "threatintelsplitterbolt:splitter:begin:ts": "1555691794139",
          "threatintelsplitterbolt:splitter:end:ts": "1555691794139",
          "ip_src_addr": "127.0.0.1",
          "timestamp": 1555687356751
        }
      },
```

LAB EXERCISE/STEP 17

It is possible to setup Metron so that the new data will be supplied constantly from Squid log files. Then, another interface will be more useful. Click Management in the left-hand-side menu. Then configure an index pattern by entering *squid_index** into the Index pattern field and pick *timestamp* in the dropdown list of the Time Filter field name element. Then click the Create button.



Configure an index pattern

In order to use Kibana you must configure at least one index pa
are also used to configure fields.

Index pattern advanced options

squid_index*

Patterns allow you to define dynamic index names using * as a wildcard. I

Time Filter field name refresh fields

timestamp

Expand index pattern when searching [DEPRECATED]

With this option selected, searches against any time-based index pattern
selected time range.

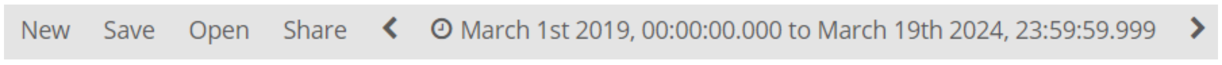
Searching against the index pattern *logstash-** will actually query Elasticse

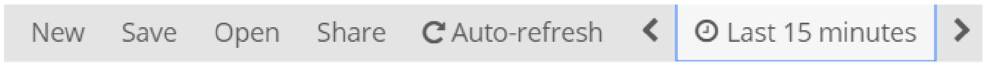
With recent changes to Elasticsearch, this option should no longer be nec

Use event times to create index names [DEPRECATED]

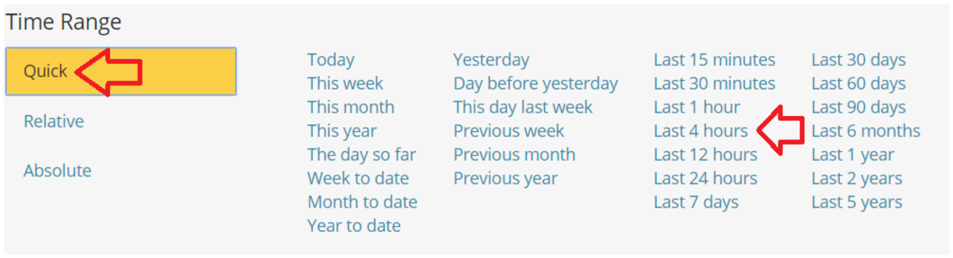
Create

Click the Discover item in the left-hand-side menu. In case no data appear on the screen, i.e. "No results found" message is displayed, change the dates range by clicking the dates or Last 15 minutes in the top right corner.

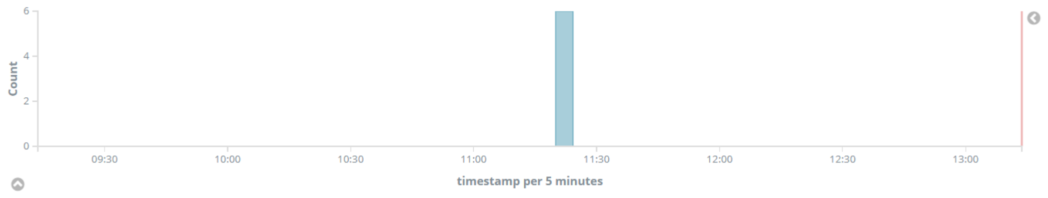

New
Save
Open
Share
March 1st 2019, 00:00:00.000 to March 19th 2024, 23:59:59.999



The Time Range menu will appear on the screen. Click Quick then click Last 4 hours.

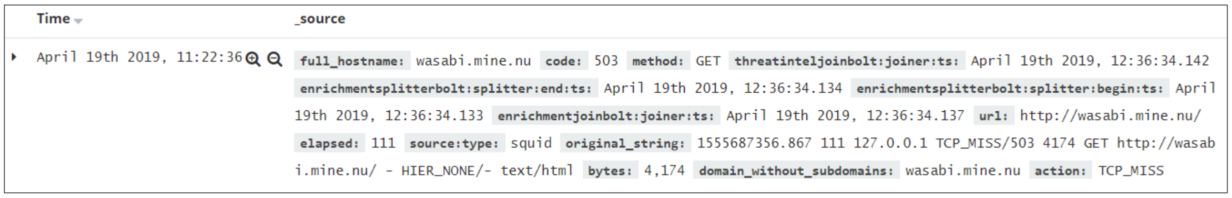


The new screen will show a bar chart with number of alerts per five-minute interval.



Make a screenshot and store it in the submission document.

Detailed information on indexed events will be displayed below the chart.



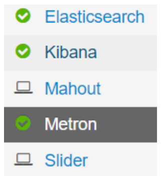
Time	_source
April 19th 2019, 11:22:36	full_hostname: wasabi.mine.nu code: 503 method: GET threatinteljoinbolt:joiner:ts: April 19th 2019, 12:36:34.142 enrichmentsplitterbolt:splitter:end:ts: April 19th 2019, 12:36:34.134 enrichmentsplitterbolt:splitter:begin:ts: April 19th 2019, 12:36:34.133 enrichmentjoinbolt:joiner:ts: April 19th 2019, 12:36:34.137 url: http://wasabi.mine.nu/ elapsed: 111 source:type: squid original_string: 1555687356.867 111 127.0.0.1 TCP_MISS/503 4174 GET http://wasabi.mine.nu/ - HIER_NONE/- text/html bytes: 4,174 domain_without_subdomains: wasabi.mine.nu action: TCP_MISS

LAB EXERCISE/STEP 18

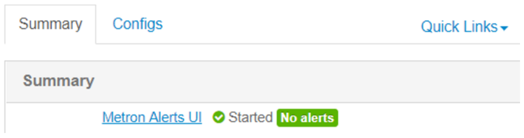
Next, navigate your web browser to the Ambari UI (change YOURIPADDRESS to the actual IP address of your Metron server).

```
http://YOURIPADDRESS:8080
```

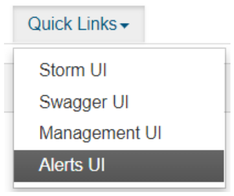
Then locate and click Metron in the list of applications on the left-hand side.



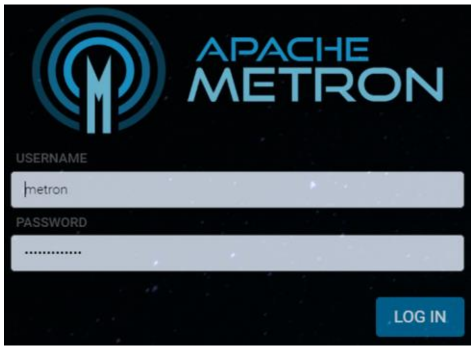
In the new screen, locate the Quick Links menu and click it.



Click the Alerts UI element:



This will take you to the Metron Alerts UI login screen.



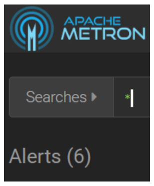
USERNAME

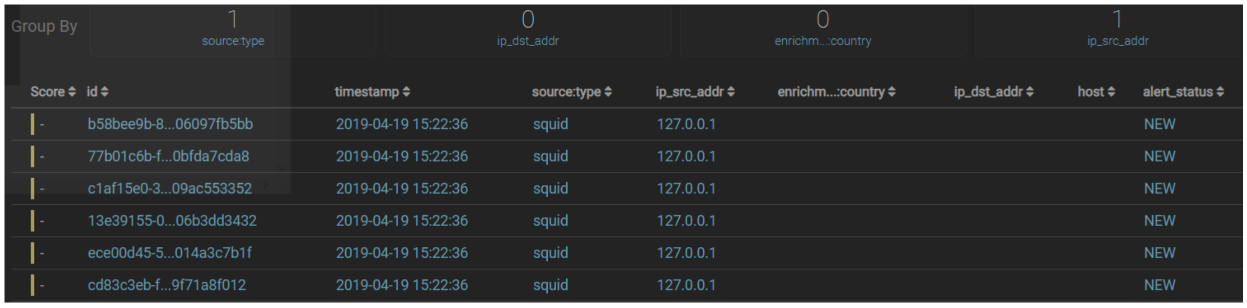
metron

PASSWORD

••••••••••••••

Enter default username *metron* and default password *Smoothmetron2* and click the LOG IN button. The screen will display the list of generated alerts, which can be manipulated with the actions available in the graphical user interface.



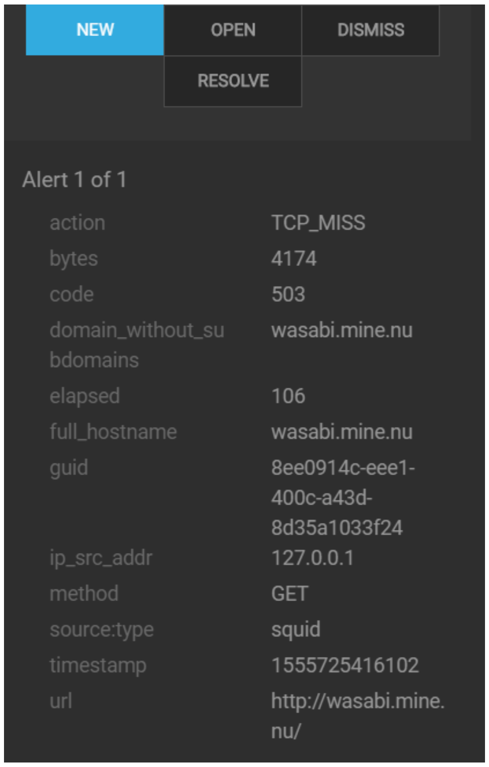


Group By

1	0	0	1
source:type	ip_dst_addr	enrichm...:country	ip_src_addr

Score	id	timestamp	source:type	ip_src_addr	enrichm...:country	ip_dst_addr	host	alert_status
-	b58bee9b-8...06097fb5bb	2019-04-19 15:22:36	squid	127.0.0.1				NEW
-	77b01c6b-f...0bfda7cda8	2019-04-19 15:22:36	squid	127.0.0.1				NEW
-	c1af15e0-3...09ac553352	2019-04-19 15:22:36	squid	127.0.0.1				NEW
-	13e39155-0...06b3dd3432	2019-04-19 15:22:36	squid	127.0.0.1				NEW
-	ece00d45-5...014a3c7b1f	2019-04-19 15:22:36	squid	127.0.0.1				NEW
-	cd83c3eb-f...9f71a8f012	2019-04-19 15:22:36	squid	127.0.0.1				NEW

Make a screenshot and store it in the submission document.



Alert 1 of 1

action	TCP_MISS
bytes	4174
code	503
domain_without_subdomains	wasabi.mine.nu
elapsed	106
full_hostname	wasabi.mine.nu
guid	8ee0914c-eee1-400c-a43d-8d35a1033f24
ip_src_addr	127.0.0.1
method	GET
source:type	squid
timestamp	1555725416102
url	http://wasabi.mine.nu/

LAB EXERCISE/STEP 19

Be sure to terminate your AWS Metron VM to avoid budget depletion. Resources created in a cloud environment under your account have associated costs. AWS offers more than 60 products at a free tier with associated free tier usage limits [15].

PUZZLER

Install Zeppelin notebooks shipped with metron by Navigating to Ambari → Metron → Service Actions → Install Zeppelin Notebooks. Then, navigate to Ambari → Zeppelin Notebook → Quick Links → Zeppelin UI. Login using admin as a username and as a password. A welcome screen will open. Locate the Hello World Tutorial and click it. This Zeppelin Notebook is distributed with Metron. This assignment explores network intrusion events using Python, Hive and Spark. Zeppelin notebooks consist of paragraphs, which can be executed either altogether or one by one. Read the paragraphs thoroughly. There is no need to re-run the paragraphs in this tutorial, since they contain the output as well. Each paragraph may contain description, code and output for the code if any.

What to submit

Submit a Word (or other text editor) document with embedded screenshots made as requested in the assignment and a brief description for each screenshot.

References

[1] Merriam-Webster, "Triage," [Online]. Available: https://www.merriam-webster.com/dictionary/triage. [Accessed: Aug. 12, 2019].

[2] SANS Institute. Information Security Reading Room "Triaging Alerts with Threat Indicators," [Online]. Available: https://www.sans.org/reading-room/whitepapers/threatintelligence/triaging-alerts-threat-indicators-37945. [Accessed: Aug. 12, 2019].

[3] STIX TAXII. "Sharing threat intelligence just got a lot easier!," [Online]. Available: https://oasis-open.github.io/cti-documentation/. [Accessed: Aug. 12, 2019].

[4] U.S. Department of Homeland Security. "Information Sharing Specifications for Cybersecurity," [Online]. Available: https://www.us-cert.gov/Information-Sharing-Specifications-Cybersecurity. [Accessed: Aug. 12, 2019].

[5] ZeuS Tracker, "ZeuS Tracker," [Online]. Available: https://zeustracker.abuse.ch/blocklist.php?download=domainblocklist. [Accessed: Mar. 1, 2019].

[6] The Federal Bureau of Investigations. "Cyber Banking Fraud. Global Partnerships Lead to Major Arrests" [Online]. Available: https://archives.fbi.gov/archives/news/stories/2010/october/cyber-banking-fraud. [Accessed: Aug. 12, 2019].

[7] J. Finkle, "Hackers steal U.S. government, corporate data from PCs," [Online]. Available: https://www.reuters.com/article/us-internet-attack/hackers-steal-u-s-government-corporate-data-from-pcs-idUSN1638118020070717. Reuters, July 17, 2017.

[8] Github, "Visgean/Zeus," [Online]. Available: https://github.com/Visgean/Zeus. [Accessed: Aug. 12, 2019].

[9] ISMG Network. Bank Info Security. "Zeus Banking Trojan Spawn: Alive and Kicking," [Online]. Available: https://www.bankinfosecurity.com/zeus-banking-trojan-spawn-alive-kicking-a-10471. [Accessed: Aug. 12, 2019].

[10] Apache Metron. “Parsers,” [Online]. Available: https://metron.apache.org/current-book/metron-platform/metron-parsers/index.html. [Accessed: Aug. 12, 2019].
[11] Grok. “Create a Grok Pattern,” [Online]. Available: http://grok.nflabs.com/WhatIsPattern. [Accessed: Aug. 12, 2019].
[12] Github. “logstash-plugins/logstash-patterns-core,” [Online]. Available: https://github.com/logstash-plugins/logstash-patterns-core/tree/master/patterns. [Accessed: Aug. 12, 2019].
[13] Elastic. “Grok filter plugin,” [Online]. Available: https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html. [Accessed: Aug. 12, 2019].
[14] Apache Metron. “Stellar Language,” [Online]. Available: https://metron.apache.org/current-book/metron-stellar/stellar-common/index.html. [Accessed: Aug. 12, 2019].
[15] AWS, “AWS Free Tier,” [Online]. Available: https://aws.amazon.com/free/. [Accessed Aug 8, 2019].
[16] SecRepo.com, “Samples of Security Related Data,” [Online]. Available: https://www.secrepo.com/. [Accessed Aug 8, 2019].