BIG DATA CYBERSECURITY

LAB 4 NEW TELEMETRY FOR METRON

ADDING A NEW TELEMETRY SOURCE TO APACHE METRON

Lab Description: Learn to work with telemetry data stored in an access log file of a Squid proxy server. Create a Metron sensor stub to ingest a Squid log file, program Metron to parse the data and index it with Elasticsearch to enable subsequent querying and visualization.

Intrusion detection systems (IDS) are frequently used for monitoring local area network telemetry data in cybersecurity. Telemetry is known as an automated messaging process through which remote endpoints collect a series of measurement data and deliver the collected data to IT systems for monitoring [1]. Another definition states that telemetry refers to the data consisting of a collection of measured values and its transmission from a sensor or a device to a spatially separated location [2]. IDS systems measure and verify transmitted data, which are not essentially used by the transmission protocol. Network telemetry may include temporal data of packet arrival, packet sizes, session times and sizes, amount of dropped packets and so on. Telemetry-based IDSs monitor packets traversing a network, calculate running averages of the telemetry metrics, and produce alerts when network traffic anomalies are detected. [1]

On the other hand, telemetry data may be collected and accumulated in a data lake for further analysis or transferred directly to an evaluation process [2]. Telemetry is frequently understood as time series data, e.g. a sequence of data points with corresponding timestamps accumulated over a period of time [3]. It is known that telemetry data lakes may grow large in size thus making Apache Metron an appropriate tool for the task, since Metron is facilitated by the big data Hadoop ecosystem [4].

Proxy Servers and Squid. A proxy server is a computer server acting as a middle man between a client and a server communicating over the internet. A request issued by a client for a webpage, a CSS file, a Javascript file, etc. is intercepted by a proxy server, which acts as a client in front of the destination server and requests a resource on behalf of the client. A response is returned back to the client, which is presented as if it came directly from the server. This is known as a forward proxy mode of operation. Additionally, a proxy server may be configured to deny or allow communication based on a set of preconfigured rules, e.g. block clients with international IP addresses, block access to video files or other content, and so on. Storing copies of web content in a local cache to save bandwidth and reduce network latency or modifying HTTP requests or responses are among other common applications of a proxy.

Overall, forward proxy mode is frequently applied to shorten web page loading time and decrease network bandwidth, restrict network access to certain resources, monitor users' browsing activities, anonymize users by concealing their IP addresses, monitoring network for the presence of malware activities, etc.

In its reverse mode, a proxy server may improve access to one or more slow webservers via caching its pages and serving clients on behalf of a web server. [5] For example, Flickr was known to use Squid as a reverse proxy server in front of its photo repository. Each proxy server was tuned to preserve objects in their cache for three to five hours and handling approximately one thousand requests per second [6].

Squid proxy server is a free open-source software distributed under GNU General Public License [7]. Squid supports proxy server functionality described above. It is shipped with the squidclient utility, which purpose is for testing access rules or other configuration from a command line interface [5].

Elasticsearch is a search and analytics engine supporting REST API, JSON format, standalone or distributed mode, extendable via Elastic stack providing with security, alerting, monitoring, reporting, graph, machine learning, SQL and presentation features. [8, 9]. The Elastic stack includes Elasticsearch, Kibana—a data visualization tool; Logstash—a tool for ingesting data in multiple formats, e.g. log files, data processing with plugins, e.g. a Grok plugin for parsing unstructured or semi-structured text, commonly present in log files; and Beats—a small footprint agent for sending data to Elasticsearch. [10, 11, 12]

In particular, this assignment employs Grok, a filter plugin for parsing logfile data. Grok uses regular expressions—character sequences defining patterns in textual data—for specifying logfile structure and converting it to a JSON form [13]. Elasticsearch uses JSON-formatted index templates consisting of settings, data mappings and a simple pattern template to format new data indices [14].

Apache Storm is a free open-source distributed application for processing real-time streaming data. It works with Hadoop ecosystem and is designed for scalability, fault tolerance and extensibility. Pipelines responsible for the processing of data streams are known as Storm topologies, which consist of Storm bolts and Storm spouts connected with the use of Storm groupings. In Storm, data streams are unbounded sequences of structured tuples, created and processed in a distributed mode. In a Storm topology, a spout is a source of data, commonly reading tuples from an external source and passing them into the topology. Storm bolts are data transformation units, where a join, data aggregation, filtering or other functionality can be applied. Storm stream groupings specify how data streams are partitioned among Storm bolts. [15]

Apache Kafka is another member of Hadoop ecosystem employed by Metron. Kafka is commonly used for data migration from external systems, such as Oracle or MySQL databases into Hadoop. In general, Kafka is a distributed streaming platform frequently utilized for streaming data in real time between systems or for building applications transforming or reacting to data streams. Kafka organizes streams into categories called topics. Kafka consists of four APIs: Producer for publishing streams into topics, Consumer for subscribing to topics and processing them, Streams for transforming an input stream into an output, and Connector for connecting Kafka topics to external data systems [16].

Metron Parsing Topology takes a sensor stub input, e.g. Squid data, and transforms it into a Metron JSON Object containing several predefined fields, such as source and destination IP address, source and destination ports, protocol, timestamp, message type and so on. Metron Parsing Topology consists of (1) a Storm-**Kafka Spout** transferring data from a Kafka topic into a Storm topology and (2) a Metron Parser Kafka Bolt parsing data and directing it into a Kafka topic in a JSON format. Grok parsers are applied for data processing at this stage. [17, 18]

Metron sensor stubs are services simulating behavior of sensors while reading telemetry data from log files and sending it to Kafka topics. A sensor is designated to process a single data format, e.g. Bro, Snort, Squid, or YAF. Metron is distributed with a sample pcap (packet capture) file, which can be used as a source of stored sensor data. Such files are produced by network monitoring software to capture transmitted packets for later analysis. A user may supply other log files for Metron sensor stubs [19].

Shell variables are used in this assignment to store configuration settings required by Metron and its components for successful communication and operation. Names of these variables are usually self-explanatory and written in all capital letters, e.g. METRON_VERSION stores the version of Metron deployed in the cluster [20].

Curl (typed in lowercase curl) is a command-line utility program for transferring data from or to a server in several supported protocols. In this work, it is used to test REST API via HTTP methods for REST services, such as POST to create data, GET to read data, DELETE to delete data, and PUT or PATCH to replace or modify data [21].

REST API commonly refers to a web service built following the REST software architecture style, REpresentational State Transfer, which follows six principles: (1) Separate data storage from user interface or the client-server principle; (2) Each request from a client to a server must contain all the necessary information or the stateless principle; (3) Data must be labeled as non-cacheable or vice versa—the cacheable principle; (4) Employ four interface constraints: identification of resources, manipulation of resources through representations; self-descriptive messages; and, hypermedia as the engine of application state—the uniform interface principle; (5) Software architecture must be composed of hierarchical layers—the layered system principle; and (6) client functionality can be extended by downloading scripts—the optional code on demand principle. [22]

JSON or the JavaScript Object Notation is a lightweight format for data interchange with two core concepts used to encode the data: collections of key-value pairs and an ordered list of values. [23]

HDFS or Hadoop Distributed Files System is a fault-tolerant distributed file system utilized in Hadoop-powered clusters. It consists of a single Name Node managing namespace and administering client access to files and a number of Data Nodes constituting distributed storage. [24]

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 for each lab to avoid misconfiguration because of the settings kept from other assignments.

LAB EXERCISE/STEP 2

Pull Zookeeper Configuration

Zookeper is a coordinator within an Apache Hadoop cluster. ZooKeeper is a centralized service for maintaining configuration, naming, providing distributed synchronization, and providing group services in distributed applications. Zookeper service itself is distributed and highly reliable.

Pull the Zookeeper configuration information into a local folder using the following command (this must be typed as a single line):

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

Make a screenshot and store it in the submission document.

LAB EXERCISE/STEP 3

Install Squid proxy.

Install Squid proxy by executing the command:

```
sudo yum install -y squid
```

Strart Squid proxy with this command:

```
sudo service squid start
```

Make a screenshot and store it in the submission document.

Switch to the root user in order to run commands as the superuser:

```
sudo su
```

Navigate to the directory with Squid log files:

```
cd /var/log/squid
```

List files in this directory:

```
ls
```

The directory should contain cache.log consisting of error and debug messages as well as the access.log file, which stores information on web resources accessed by Squid users along with other information, such as HTTP status codes, request methods, etc.

LAB EXERCISE/STEP 4

Simulate User Activity and Populate Squid Access Log

At this point the log file should be empty since the Squid proxy server has just been installed on the system. We will simulate activity of proxy server users visiting various URLs with the use of *squidclient*, the Squid testing utility. To do so, run the following lines:

```
squidclient "https://www.nsa.gov/What-We-Do/Cybersecurity/NCX/"

squidclient "https://niccs.us-cert.gov/formal-education/integrating-cybersecurity-classroom"

squidclient "https://www.itpro.co.uk/hacking/30282/what-is-ethical-hacking-white-hat-hackers-explained"

squidclient "https://us.norton.com/internetsecurity-emerging-threats-what-is-the-difference-between-black-white-and-grey-hat-hackers.html"

squidclient "http://metron.apache.org/current-book/index.html"

squidclient "https://www.cisco.com/c/en/us/products/security/advanced-persistent-threat.html"

squidclient "https://www.howtogeek.com/157460/hacker-hat-colors-explained-black-hats-white-hats-and-gray-hats/"

squidclient "https://www.nibusinessinfo.co.uk/content/common-cyber-security-measures"

squidclient "https://www.dhs.gov/topic/cybersecurity"

squidclient "https://www.nec.com/en/global/about/mitatv/16/2.html"

squidclient "https://www.ncsc.gov.uk/active-cyber-defence"

squidclient "https://www.computerweekly.com/news/450432739/Sweden-steps-up-cyber-defence-measures"

squidclient "https://www.cisco.com/c/en/us/products/security/what-is-cybersecurity.html"
squidclient "https://www.diplomatie.gouv.fr/en/french-foreign-policy/defence-security/cyber-security/"

squidclient "https://www.coxblue.com/8-cyber-security-best-practices-for-your-small-to-medium-size-business-smb/"

squidclient "https://www.lawfareblog.com/chinese-perspective-pentagons-cyber-strategy-active-cyber-defense-defending-forward"

squidclient "https://www.cpni.gov.uk/cyber-security"

squidclient "https://www.akamai.com/us/en/resources/cyber-security.jsp"

squidclient "https://www.forbes.com/sites/soorajshah/2019/01/22/government-should-name-and-shame-companies-with-poor-cyber-security-say-academics/#3efeb0525b53"

squidclient "https://www.barracuda.com/glossary/cyber-security"

squidclient "https://docs.hortonworks.com/HDPDocuments/HCP1/HCP-1.6.1/installation/content/introduction_to_metron.html"
```

When the Squid proxy is used in a real-world setting in the forward mode, users would change their web browser settings to specify IP address and port where Squid operates, in order to browse Internet. Thus, Squid would follow its configuration and collect URLs and other related information. We used the *squidclient* utility to simulate users navigating URLs as listed above. Squid access.log should now have several entries, which resemble the lines below.

```
1551302847.374    138 ::1 TCP_MISS/200 14345 GET https://docs.hortonworks.com/HDPDocuments/HCP1/HCP-1.6.1/installation/content/introduction_to_metron.html - HIER_DIRECT/13.32.250.64 text/html

1551302855.636    354 ::1 TCP_MISS/200 75016 GET https://www.nsa.gov/What-We-Do/Cybersecurity/NCX/ - HIER_DIRECT/23.218.61.50 text/html

1551302855.946    303 ::1 TCP_MISS/200 47716 GET https://niccs.us-cert.gov/formal-education/integrating-cybersecurity-classroom - HIER_DIRECT/23.218.69.42 text/html
```

```
1551302857.174   1221 ::1 TCP_MISS/200 96902 GET
https://www.itpro.co.uk/hacking/30282/what-is-ethical-hacking-white-hat-
hackers-explained - HIER_DIRECT/54.247.85.154 text/html
```

Squid format may be customized by including additional or removing existing features. By default, access log has the following attributes:

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

LAB EXERCISE/STEP 5

Create a Kafka Topic for the Squid Data Source

By design, in Metron architecture each new telemetry data source must be accompanied with a Kafka topic, which becomes 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
```

To verify that the squid topic has been successfully created, list all existing Kafka topics:

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

The output of this command should show squid among other topics.

This command will send Squid logs to the Kafka topic:

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

Run this command to ingest Squid logs into Kafka.

```
${HDP_HOME}/kafka-broker/bin/kafka-console-consumer.sh --zookeeper
$ZOOKEEPER --topic squid --from-beginning
```

After it displays approximately 20 lines of output, press the Control C key combination on the keyboard to terminate this command. 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 [26]. 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 [27-29].

```
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
```

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
```

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.

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 in the parser configuration [30]. The Grok Parser is set up to extract complete URLs. For the purpose of this assignment, domain names without subdomains will be sufficient.

Stellar 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 6

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

```
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.

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:

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

The file already contains information. Remember to switch to the Insert mode. 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
```

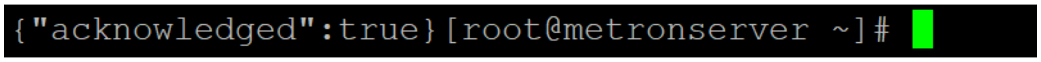
Its output will list a number of configurations in the JSON format. The global config will be shown on the very top of the output. To see it, you will need to scroll up the terminal window.

LAB EXERCISE/STEP 7

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'
```

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.

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

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

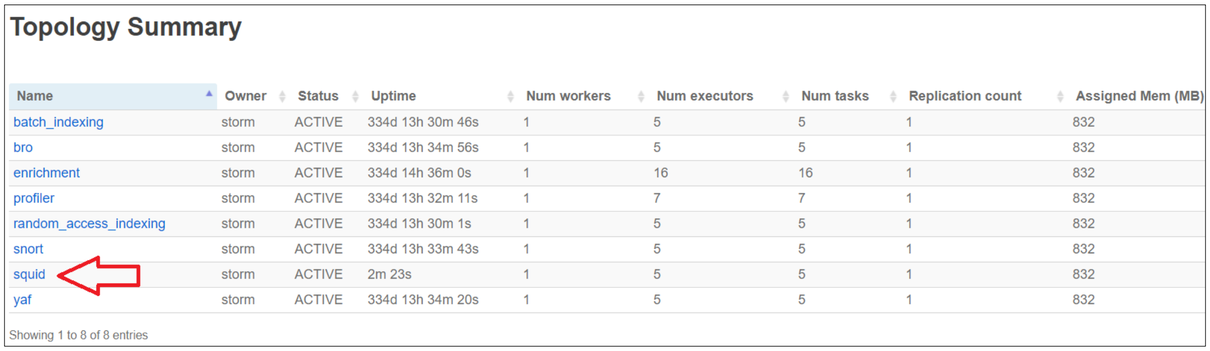
Wait until the script completes. The last line of the output should look similar to this: *[main] INFO o.a.s.StormSubmitter - Finished submitting topology: squid*.

LAB EXERCISE/STEP 8

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 Ipм4 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

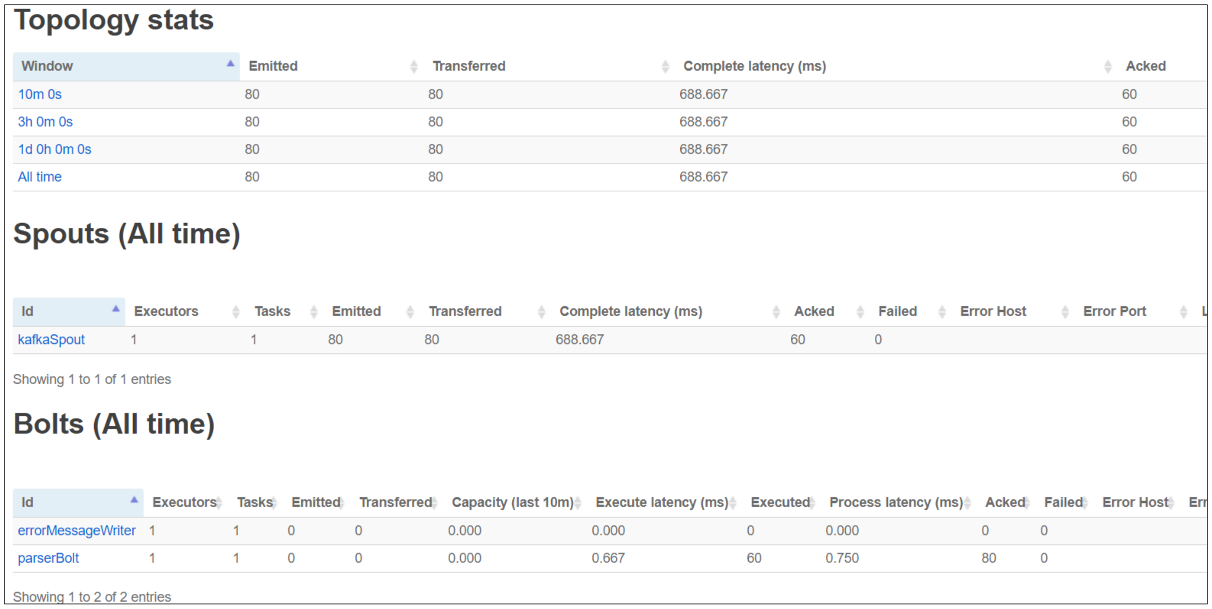
Name	Owner	Status	Uptime	Num workers	Num executors	Num tasks	Replication count	Assigned Mem (MB)
batch_indexing	storm	ACTIVE	334d 13h 30m 46s	1	5	5	1	832
bro	storm	ACTIVE	334d 13h 34m 56s	1	5	5	1	832
enrichment	storm	ACTIVE	334d 14h 36m 0s	1	16	16	1	832
profiler	storm	ACTIVE	334d 13h 32m 11s	1	7	7	1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

Switch back to the terminal window. Now that we have a new running squid parser topology, generate some data to parse by running this command several times:

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

Switch back to the browser window and click the "squid" link which is pointed with a red arrow in the screenshot above. In the Topology stats, Spouts (All time) and Bolts (All time), it should show the number of acknowledged messages proportional to the number of times you executed the command above.



Topology stats

Window	Emitted	Transferred	Complete latency (ms)	Acked
10m 0s	80	80	688.667	60
3h 0m 0s	80	80	688.667	60
1d 0h 0m 0s	80	80	688.667	60
All time	80	80	688.667	60

Spouts (All time)

Id	Executors	Tasks	Emitted	Transferred	Complete latency (ms)	Acked	Failed	Error Host	Error Port	L
kafkaSpout	1	1	80	80	688.667	60	0			

Showing 1 to 1 of 1 entries

Bolts (All time)

Id	Executors	Tasks	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)	Acked	Failed	Error Host	Err
errorMessageWriter	1	1	0	0	0.000	0.000	0	0.000	0	0		
parserBolt	1	1	0	0	0.000	0.667	60	0.750	80	0		

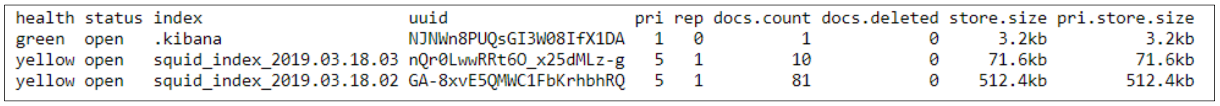
Showing 1 to 2 of 2 entries

LAB EXERCISE/STEP 9

Navigate your web browser to the following URL of Elasticsearch where YOURIPADDRESS should be substituted with an actual IP address of your Metron server:

```
http://YOURIPADDRESS:9200/_cat/indices?v
```

Verify that a squid index has been created:

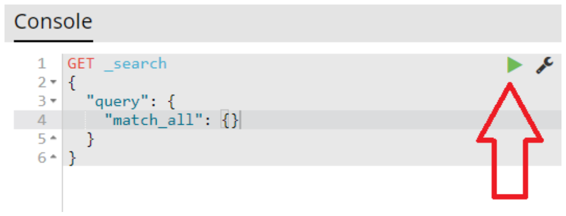


```
health status index                     uuid                   pri rep docs.count docs.deleted store.size pri.store.size
green  open   .kibana                   NJNWn8PUQsGI3W08IfX1DA   1   0          1            0      3.2kb          3.2kb
yellow open   squid_index_2019.03.18.03 nQr0LwwRRt6O_x25dMLz-g   5   1         10            0     71.6kb         71.6kb
yellow open   squid_index_2019.03.18.02 GA-8xvE5QMWC1FbKrhbhRQ   5   1         81            0    512.4kb        512.4kb
```

Then navigate to the Kibana URL (again, substitute YOURIPADDRESS with an actual IP address of your Metron server):

```
http://YOURIPADDRESS:5000/app/kibana#/dev_tools/console?_g=()
```

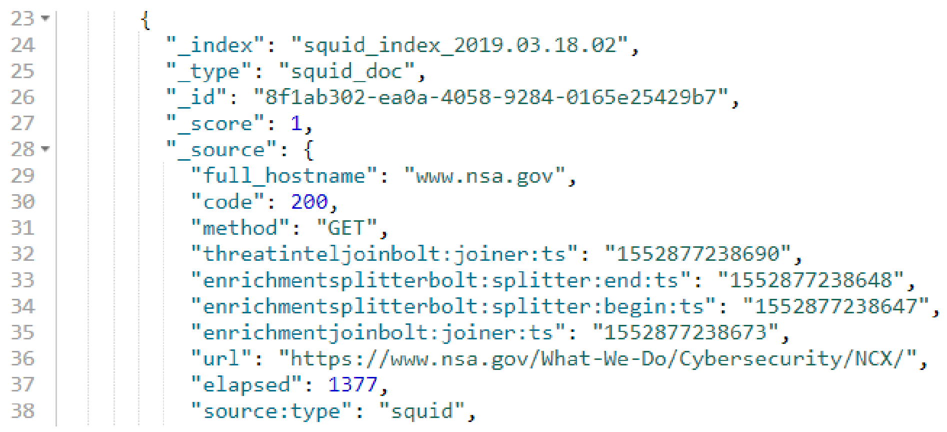
The Console window will contain a JSON request to display all the data.



Console

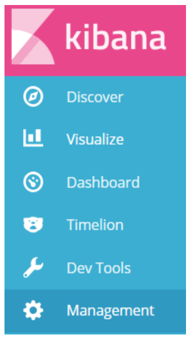
```
GET _search
{
  "query": {
    "match_all": {}
  }
}
```

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.

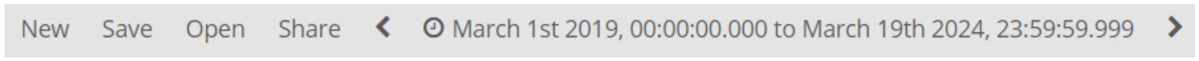


```
      {
        "_index": "squid_index_2019.03.18.02",
        "_type": "squid_doc",
        "_id": "8f1ab302-ea0a-4058-9284-0165e25429b7",
        "_score": 1,
        "_source": {
          "full_hostname": "www.nsa.gov",
          "code": 200,
          "method": "GET",
          "threatinteljoinbolt:joiner:ts": "1552877238690",
          "enrichmentsplitterbolt:splitter:end:ts": "1552877238648",
          "enrichmentsplitterbolt:splitter:begin:ts": "1552877238647",
          "enrichmentjoinbolt:joiner:ts": "1552877238673",
          "url": "https://www.nsa.gov/What-We-Do/Cybersecurity/NCX/",
          "elapsed": 1377,
          "source:type": "squid",
```

It is possible to view messages in the Discover mode as well. Click the Discover menu item in the top left corner.



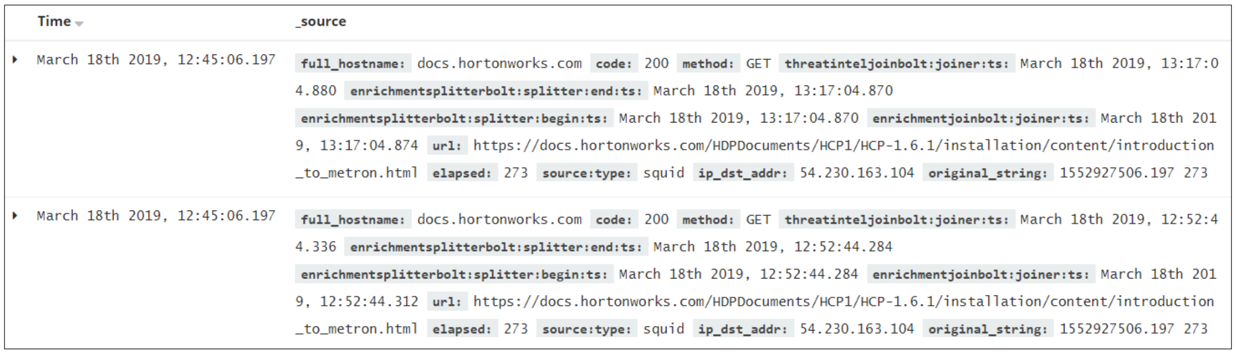
In case no data appear on the screen, change the dates range by clicking the dates in the top right corner. Be sure to include the today's date in the interval.



March 1st 2019, 00:00:00.000 to March 19th 2024, 23:59:59.999

Then click the Go button.

Data should be formatted similar to this.

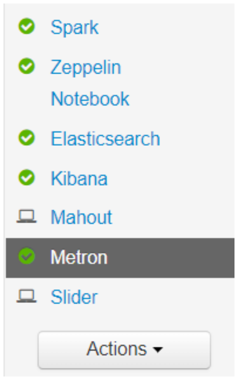


Time	_source
March 18th 2019, 12:45:06.197	**full_hostname:** docs.hortonworks.com **code:** 200 **method:** GET **threatinteljoinbolt:joiner:ts:** March 18th 2019, 13:17:04.880 **enrichmentsplitterbolt:splitter:end:ts:** March 18th 2019, 13:17:04.870 **enrichmentsplitterbolt:splitter:begin:ts:** March 18th 2019, 13:17:04.870 **enrichmentjoinbolt:joiner:ts:** March 18th 2019, 13:17:04.874 **url:** https://docs.hortonworks.com/HDPDocuments/HCP1/HCP-1.6.1/installation/content/introduction_to_metron.html **elapsed:** 273 **source:type:** squid **ip_dst_addr:** 54.230.163.104 **original_string:** 1552927506.197 273
March 18th 2019, 12:45:06.197	**full_hostname:** docs.hortonworks.com **code:** 200 **method:** GET **threatinteljoinbolt:joiner:ts:** March 18th 2019, 12:52:44.336 **enrichmentsplitterbolt:splitter:end:ts:** March 18th 2019, 12:52:44.284 **enrichmentsplitterbolt:splitter:begin:ts:** March 18th 2019, 12:52:44.284 **enrichmentjoinbolt:joiner:ts:** March 18th 2019, 12:52:44.312 **url:** https://docs.hortonworks.com/HDPDocuments/HCP1/HCP-1.6.1/installation/content/introduction_to_metron.html **elapsed:** 273 **source:type:** squid **ip_dst_addr:** 54.230.163.104 **original_string:** 1552927506.197 273

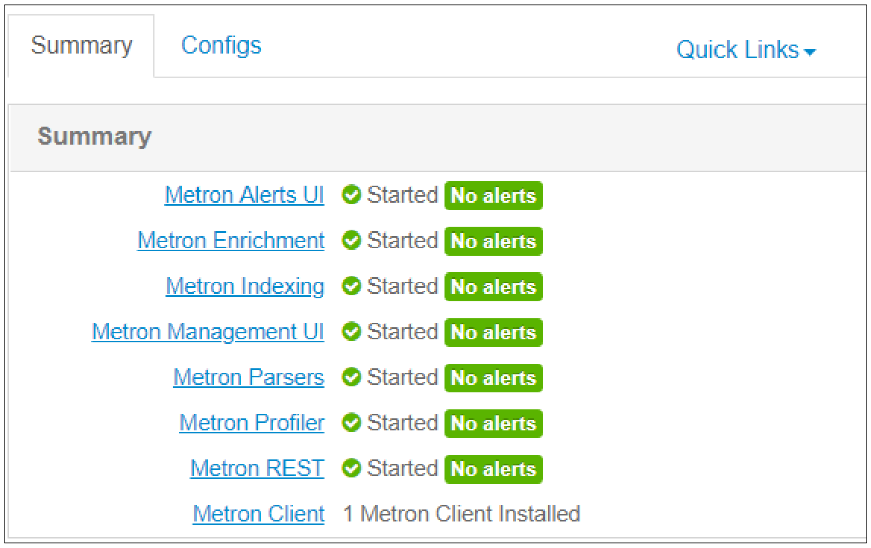
Next, navigate your web browser to the Metron Alert UI Dashboard. First, navigate to Ambari UI (change YOURIPADDRESS to the actual IP address of your Metron server).

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

Ambari UI will request a username and a password. Enter the default username *admin* and the default password *admin*. In the interface, locate Metron in the list of applications on the left-hand side and click it.



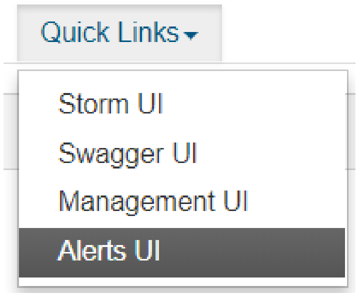
In the right-hand-side part of the window, locate and click the Quick Links menu:


Summary | Configs | Quick Links

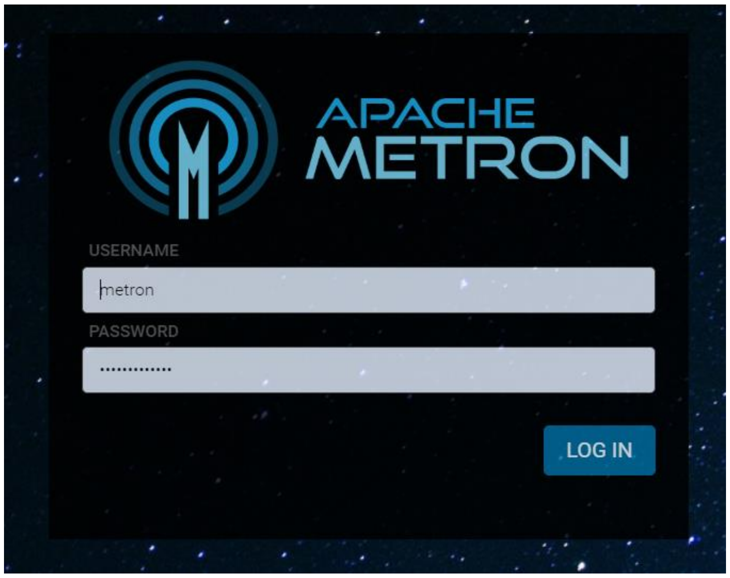
Summary

Metron Alerts UI	✓ Started No alerts
Metron Enrichment	✓ Started No alerts
Metron Indexing	✓ Started No alerts
Metron Management UI	✓ Started No alerts
Metron Parsers	✓ Started No alerts
Metron Profiler	✓ Started No alerts
Metron REST	✓ Started No alerts
Metron Client	1 Metron Client Installed

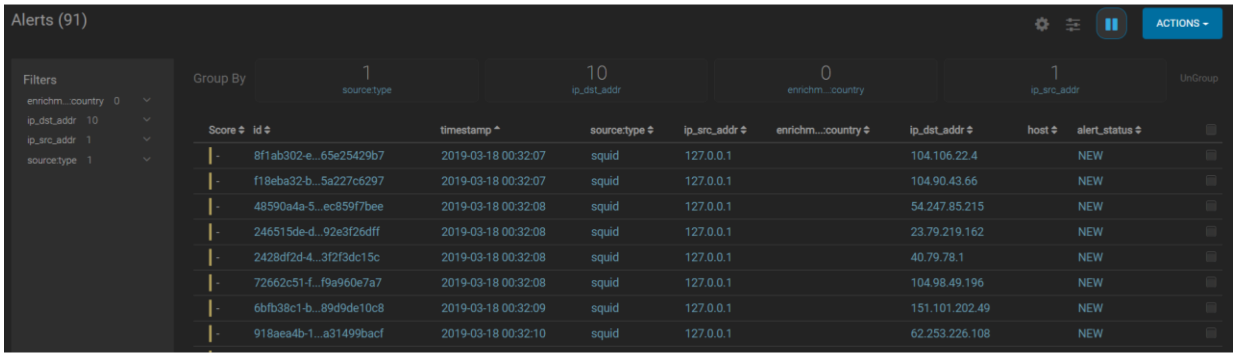
Click the Alerts UI element:



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



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.


Alerts (91)

Filters

enrichm...:country	0
ip_dst_addr	10
ip_src_addr	1
source:type	1

Group By: 1 source:type; 10 ip_dst_addr; 0 enrichm...:country; 1 ip_src_addr

Score	id	timestamp	source:type	ip_src_addr	enrichm...:country	ip_dst_addr	host	alert_status
-	8f1ab302-e...65e25429b7	2019-03-18 00:32:07	squid	127.0.0.1		104.106.22.4		NEW
-	f18eba32-b...5a227c6297	2019-03-18 00:32:07	squid	127.0.0.1		104.90.43.66		NEW
-	48590a4a-5...ec859f7bee	2019-03-18 00:32:08	squid	127.0.0.1		54.247.85.215		NEW
-	246515de-d...92e3f26dff	2019-03-18 00:32:08	squid	127.0.0.1		23.79.219.162		NEW
-	2428df2d-4...3f2f3dc15c	2019-03-18 00:32:08	squid	127.0.0.1		40.79.78.1		NEW
-	72662c51-f...f9a960e7a7	2019-03-18 00:32:08	squid	127.0.0.1		104.98.49.196		NEW
-	6bfb38c1-b...89d9de10c8	2019-03-18 00:32:09	squid	127.0.0.1		151.101.202.49		NEW
-	918aea4b-1...a31499bacf	2019-03-18 00:32:10	squid	127.0.0.1		62.253.226.108		NEW

LAB EXERCISE/STEP 10

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 [31].

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] S. Gaurav and S. Machiraju, *Hardening Azure Applications: Techniques and Principles for Building Large-Scale, Mission-Critical Applications*. New York, NY: Apress, 2018.

[2] O. Michalski and S. Demiliani, *Implementing Azure Cloud Design Patterns*. Birmingham, UK: Packt Publishing, 2018.

[3] A. Ravulavaru, *Enterprise Internet of Things Handbook*. Birmingham, UK: Packt Publishing, 2018.

[4] J. Garrett, *Data Analytics for IT Networks: Developing Innovative Use Cases*. Hoboken, NJ: Cisco Press, 2018.

[5] K. Saini, *Squid Proxy Server 3.1 beginner's guide*, Birmingham, UK: Packt Publishing, 2011.

[6] Squid-cache.org, "Squid use at Flickr," [Online]. Available: http://www.squid-cache.org/Library/flickr.html. [Accessed: Aug. 12, 2019].

[7] WhatIs.com, "Squid proxy server," [Online]. Available: https://whatis.techtarget.com/definition/Squid-proxy-server. [Accessed: Aug. 12, 2019].

[8] Elastic. “The heart of the Elastic Stack,” [Online]. Available: https://www.elastic.co/products/elasticsearch. [Accessed: Aug. 12, 2019].
[9] Elastic. “Meet the core products,” [Online]. Available: https://www.elastic.co/products/stack. [Accessed: Aug. 12, 2019].
[10] Elastic. “Lightweight Data Shippers,” [Online]. Available: https://www.elastic.co/products/beats. [Accessed: Aug. 12, 2019].
[11] Elastic. “Beats Platform Reference. Getting started with Beats,” [Online]. Available: https://www.elastic.co/guide/en/beats/libbeat/current/getting-started.html. [Accessed: Aug. 12, 2019].
[12] Elastic. “What is the difference between Logstash and Beats?,” [Online]. Available: https://www.elastic.co/guide/en/beats/filebeat/1.1/diff-logstash-beats.html. [Accessed: Aug. 12, 2019].
[13] Elastic. “Do you grok Grok?” [Online]. Available: https://www.elastic.co/blog/do-you-grok-grok. [Accessed: Aug. 12, 2019].
[14] Elastic. “Index Templates,” [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/1.7/indices-templates.html. [Accessed: Aug. 12, 2019].
[15] Apache Storm. “Concepts,” [Online]. Available: https://storm.apache.org/releases/current/Concepts.html. [Accessed: Aug. 12, 2019].
[16] Apache Kafka. A distributed streaming platform. “Documentation,” [Online]. Available: https://kafka.apache.org/documentation/. [Accessed: Aug. 12, 2019].
[17] Metron. “Metron JSON Object,” [Online]. Available: https://cwiki.apache.org/confluence/display/METRON/Metron+JSON+Object. [Accessed: Aug. 12, 2019].
[18] Metron. “Parsing Topology,” [Online]. Available: https://cwiki.apache.org/confluence/display/METRON/Parsing+Topology. [Accessed: Aug. 12, 2019].
[19] Apache Metron. “Sensor Stubs,” [Online]. Available: https://metron.apache.org/current-book/metron-deployment/roles/sensor-stubs/index.html. [Accessed: Aug. 12, 2019].
[20] Wikiversity. “Bash programming/Variables,” [Online]. Available: https://en.wikiversity.org/wiki/Bash_programming/Variables. [Accessed: Aug. 12, 2019].

[21] Curl. “curl.1 the man page,” [Online]. Available: https://curl.haxx.se/docs/manpage.html. [Accessed: Aug. 12, 2019].
[22] REST API Tutorial. “What is REST,” [Online]. Available: https://restfulapi.net/. [Accessed: Aug. 12, 2019].
[23] JSON.org. “Introducing JSON,” [Online]. Available: https://www.json.org/. [Accessed: Aug. 12, 2019].
[24] Hadoop. “HDFS Architecture Guide,” [Online]. Available: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html#Introduction. [Accessed: Aug. 12, 2019].
[25] S. Ponomarev and T. Atkison. “Industrial Control System Network Intrusion Detection by Telemetry Analysis,” *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 252-260, 2016.
[26] Apache Metron. “Parsers,” [Online]. Available: https://metron.apache.org/current-book/metron-platform/metron-parsers/index.html. [Accessed: Aug. 12, 2019].
[27] Grok. “http://grok.nflabs.com/WhatIsPattern,” [Online]. Available: http://grok.nflabs.com/WhatIsPattern. [Accessed: Aug. 12, 2019].
[28] Github. “logstash-plugins/logstash-patterns-core,” [Online]. Available: https://github.com/logstash-plugins/logstash-patterns-core/tree/master/patterns. [Accessed: Aug. 12, 2019].
[29] Elastic. “Grok filter plugin,” [Online]. Available: https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html. [Accessed: Aug. 12, 2019].
[30] Apache Metron. “Stellar Language,” https://metron.apache.org/current-book/metron-stellar/stellar-common/index.html. [Accessed: Aug. 12, 2019].
[31] AWS, “AWS Free Tier,” [Online]. Available: https://aws.amazon.com/free/. [Accessed Aug 8, 2019].