ID2221 - Project report

Group Oana & Iulia
24 October 2021

Oana-Andreea Butaru (butaru@kth.se)
Iulia-Lidia Oltean (oltean@kth.se)
Camilo-Alfonso Rodríguez-Garzón (carg2@kth.se)

1 Introduction

In the IT security department of every organization, the CIA triad (confidentiality, integrity, and availability) offers the main objective of every project planned and implemented. Particularly, there is a specific kind of attack, known as Denial of Service (DoS) that affects the availability of the systems. There are several ways to do this: opening as much connections as it is possible (known as DoS slowloris), sending slow data not slow enough to time out (DoS Slowhttptest), generating unique and obfuscated traffic volumes (DoS Hulk) or some more specific such as to take advantage of some of parameters used in HTTP (DoS GoldenEye) - these ITsecurity guys definitely do not lack creativity naming attacks.

The detection of these attacks usually requires computation and processing of massive amounts of data, and that is one reason for using the Spark framework to try to discover these attacks using real data provided by the Canadian Institute for Cybersecurity [1]. The main questions to be answered: Is it possible to use this dataset, processes it and visualize it in order to detect when the attack was done? Is it possible to distinguish the underlying infrastructure using only the data?

2 Tools

- Scala & Python - development languages;
- Spark - for data processing;
- Kafka - for the communication between the producer and the consumer;
- Cassandra - for storing the data;
- Apache Zeppelin - for data visualization;
- Docker - for virtualization of Cassandra and Apache Zeppelin;
- Google Cloud Platform - for the working environment.

3 Data

The dataset used was provided by the Canadian Institute of Cybersecurity [1]. It contains information on the common attacks, which resembles the true real-world data (PCAP format). The data was captured for a period of 5 days, starting at 9 a.m., Monday, July 3rd, 2017 and ending at 5 p.m. on Friday July 7th, 2017. However, the amount of data is too big, so for the processing it is decided to select the Wednesday, 5th of July, which has a size of 12GB. For this project it is decided to use the source and destination addresses and ports, frame size and timestamp.

4 Methodology and algorithm

The architecture of the system is represented in Figure 1.

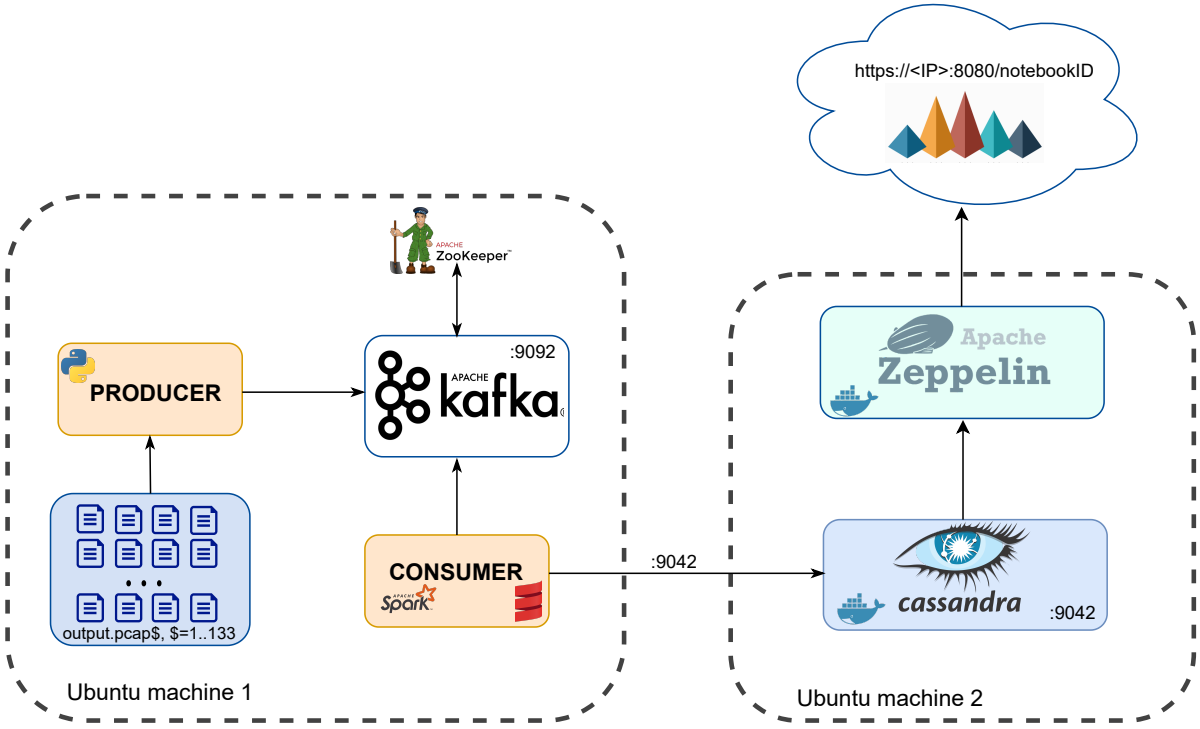

Figure 1: Implemented Architecture. 2 Ubuntu VMs are deployed in Google Cloud. The Ubuntu machine 1 has 8 vCPUs, 32 GB memory and 80 GB SSD. The 2nd one has 2 vCPUs, 4 GB memory and 20 GB SSD. Source: own elaboration.

The producer, which was developed in Python, reads the PCAP file and parses it. Because the original PCAP file was too big and its reading lasted too long, the file was splitted into smaller chunks, of 100MB each. These smaller files are read one after another. To split the PCAP file, the command used was:

```
$ tcpdump -r Wednesday-WorkingHours.pcap -w output.pcap -C 100
```

This command preserves the integrity of the original PCAP file. From each one of the packets in the smaller files, the timestamp, source IP address, source port, destination IP address, destination port and frame size is extracted and sent to Kafka. Figure 2 shows a small part of the output of the producer, with the above mentioned information parsed.

```
root@ubuntussd:/apps/finalproject/datetry#
root@ubuntussd:/apps/finalproject/datetry# python3 producer.py
Created Producer

../pcap-files/output.pcap1
2017-07-05 11:45:48
2017-07-05 11:45:48,803713,192.168.10.14,13.107.4.50,49533,80,60
2017-07-05 11:45:48
2017-07-05 11:45:48,803755,13.107.4.50,192.168.10.14,80,49533,2318
2017-07-05 11:45:48
2017-07-05 11:45:48,803757,13.107.4.50,192.168.10.14,80,49533,1514
2017-07-05 11:45:48
2017-07-05 11:45:48,803947,13.107.4.50,192.168.10.14,80,49533,2974
2017-07-05 11:45:48
2017-07-05 11:45:48,803960,192.168.10.14,13.107.4.50,49533,80,60
2017-07-05 11:45:48
2017-07-05 11:45:48,804154,13.107.4.50,192.168.10.14,80,49533,1514
2017-07-05 11:45:48
```

Figure 2: Python Program running

A topic `final` was created in Kafka, to which the producer writes. The consumer, which was developed as a Spark Streaming application in Scala, reads the data from Kafka. In order to determine when an attack has taken place, the number of connections and the minimum, average and maximum frame size, per IP source and IP destination pair are being investigated. A screenshot of one batch is shown in Figure 3. After that, the data is persisted into a Cassandra table, as shown in Figure 4.

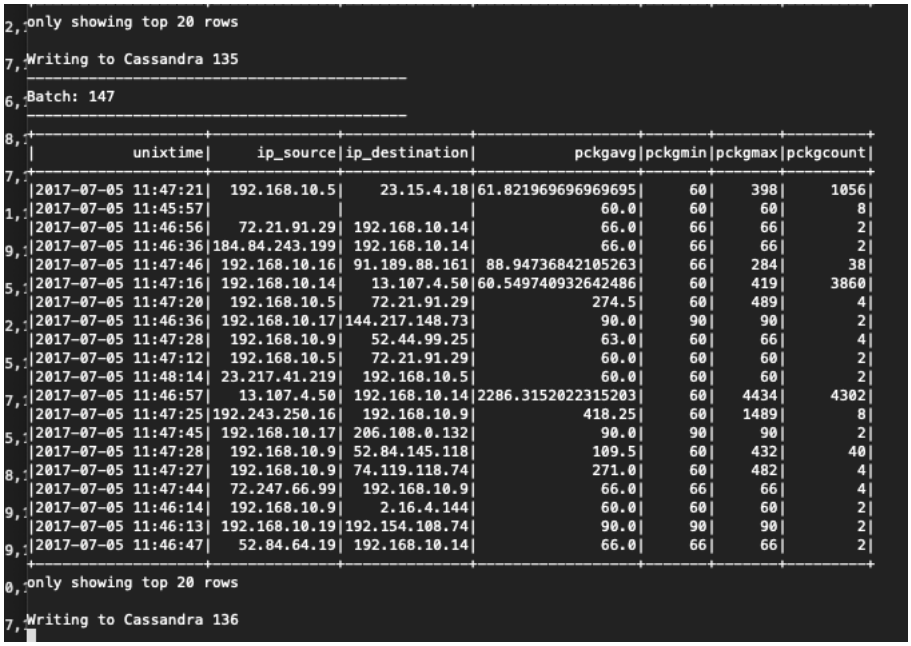

Figure 3: Scala Program running

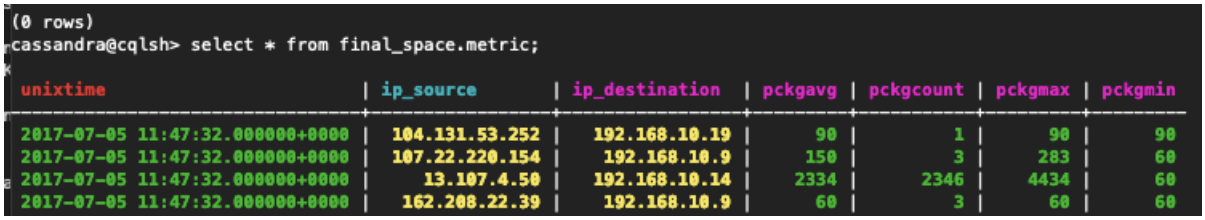

Figure 4: Cassandra DB

Apache Zeppelin was used for data visualization. Using the Cassandra interpreter, Scala was connected to the Cassandra database and then it is possible to graph data from the database. Zeppelin's cron scheduler was used in order to interrogate the database every minute and update the graphs.

Figure 5 shows the Google Cloud Platform resources used. The VM called 'ubuntussd' was used for the producer, consumer and Kafka. The VM named 'dbweb' was used for Cassandra and Apache Zeppelin, which run inside Docker containers, as pictured in Figure 6.

Figure 5: Virtual Machines running in Google Cloud Platform.

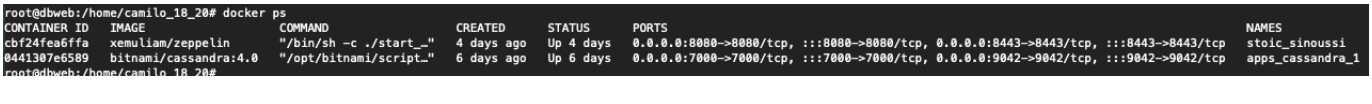

Figure 6: Dockers running in the second machine.

After reviewing the pcap file, it is decided to group by unixtime, ip_source and ip_destination because in every second there can be thousands of events. In this way it is possible to graph. Figure 7 shows a graph filtered by ip_destination.

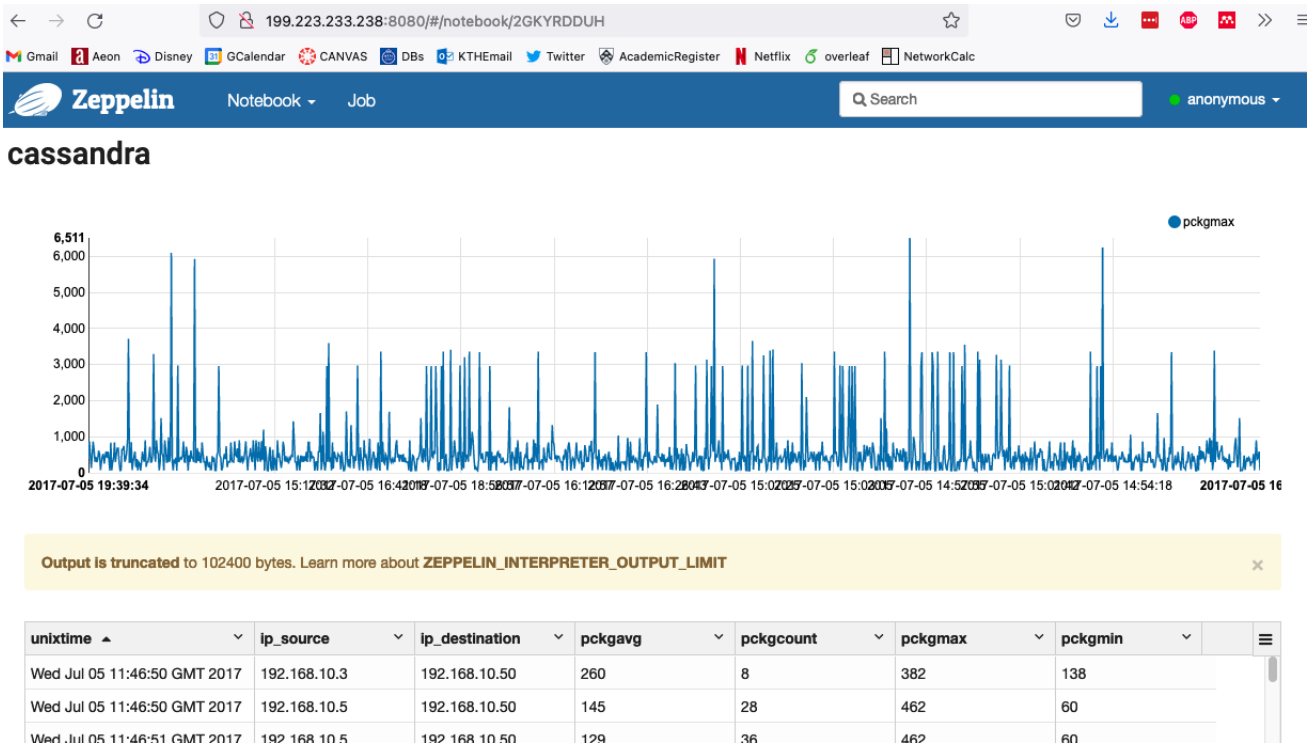

Figure 7: Final graph result filtering only by IP 192.168.10.50.

5 Final Remarks

- As an initial approximation, the infrastructure and configuration implemented gathers, processes and displays information from different pcap files. For IT Security professionals this system could serve as the basis for exhaustive analysis of network packets.
- The construction of the system only used open-source software. Regarding monetary costs, the only one was the operational expense (or Opex) of having the 2 VMs up and running. An 8cpu 32GBRAM VM costs around US$196.67/month and a 2vcpu 4GB RAM costs around US$25.46/month in Google Cloud Platform. For a total of 222,13 USD/month. For a startup, it could be useful.
- The scope of this project was the first part in a complex system. The tools implemented here by itself do not spot attacks. It is important to mention that specialized operators or trained AI tools should do this job.
- Complementing the previous item, IT Security attacks differ in mode, tools, operations, scope and almost every single possible aspect can be changed. This required a comprehensive knowledge to configure the tools.
- As a curious remark, trying to open the original file of 12 GB of size in a Windows Server machine with 2 vcpu, 32 GB RAM and 50 GB SSD took around 20 minutes. With the proposed solution, it is possible to visualize the results almost instantly.

Bibliography

[1] Canadian Institute for Cybersecurity. Intrusion detection evaluation dataset (cic-ids2017). `https://www.unb.ca/cic/datasets/ids-2017.html`. Accessed: 2021-09-30.