Big Data Security and Visulization
Prof: Nikolaos Kolokotronis

John Papadopoulos (ID: 202220600135),
Xerovasilas Leonidas ID: (2022202204023),
Asimoglou Menelaos ID: (2022202200001)

July 2023

1 Introduction

The goal of this project is to develop a distributed Intrusion Detection System (IDS) designed to identify threats within complex network structures. To achieve this, we're leveraging Zeek, a renowned platform for network event security monitoring. Complementing Zeek, we've also integrated Suricata, a rule-based network threat detection software, to bolster our IDS capabilities.

Our system adopts a passive architecture that does not actively counter threats. Instead, it focuses on detecting and reporting these threats at the earliest possible stage. To facilitate comprehensive data collection, analytics, and visualization, we've selected the Elastic platform. This platform not only supports a wide range of integrations but also delivers real-time query responses and visualizations.

The robustness of our IDS is further validated by conducting an attack simulation, thus ensuring its readiness for real-world threats.

2 Methodology

The primary IDS node operates on a virtual machine, equipped with 8.6 GiB RAM, running atop VMware Workstation (v15) on a Windows 7 host. This virtual machine runs Ubuntu 22.04 LTS and includes a bridged virtual Ethernet adapter, thereby facilitating access within the local network. Specifically, this virtual machine is assigned the IP address 192.168.1.9.

This node utilizes Elasticsearch (v7.17.11) for data analytics and Kibana (v7.17.11) for data visualization. Both services are configured in their respective YAML files – *elasticsearch.yaml* located in the *$ES_HOME* directory and *kibana.yaml* housed in the *$KIBANA_HOME* directory.

On this machine, Zeek (zeek-lts v5.04) is configured in clustered mode, but due to the presence of a single virtual Ethernet adapter (ens33), only one worker is employed. Zeek's configuration files consist of the *node.cfg* file in the

$ZEEK_HOME/etc directory (defining the deployment mode) and the *local.zeek* file in the *$ZEEK_HOME/share/zeek/site/* directory (setting parameters and policies such as Zeek log output in JSON format). This JSON format setting is activated by inserting the *@load policy/tuning/json-logs.zeek* line into the *local.zeek* file.

Finally, Zeek is deployed using the *zeekctl deploy* command.

In addition to Zeek, our virtual machine also hosts Suricata IDS (v6.0.4). For Suricata to function optimally, it's imperative to download rules that encompass the latest types of threats. This can be accomplished using the Oinkmaster tool. The configuration file for Oinkmaster, termed *oinkmaster.conf* and located in the */etc* directory, can be set up to fetch these rules from the Emerging Threats website. Upon proper configuration, the Oinkmaster tool can be executed with suitable arguments to deposit the rule files into the *rules* subdirectory located within the Suricata installation directory. Within this installation directory, the Suricata configuration file, *suricata.yaml*, resides. It is in this file that we incorporate the downloaded rules under the rules section and configure the correct interfaces (in this case, the virtual Ethernet port ens33).

To facilitate the transmission of logs to the Elastic platform, Filebeat (v7.17.11) is meticulously configured. Initially, the Zeek and Suricata modules are enabled. Following this, Filebeat is set up through *filebeat.yaml* to interact seamlessly with Elasticsearch and Kibana. Subsequently, the modules are configured to fetch the requisite log files - Zeek logs, which are housed in the *$ZEEK_HOME/logs/current* directory, and the Suricata eve log, located in the */var/log/suricata* directory.

To ensure the operation of at least three Zeek workers, we leverage a second, physically separate machine that is connected to the same network, serving as an additional IDS node. This machine runs Fedora 38 and houses Zeek (v5.09), set up in cluster mode with two workers—one each for the wireless interface (wlp3s0) and the Ethernet interface (eno1). In addition to Zeek, Suricata (v6.0.12) is installed on this secondary machine, configured with identical rules as the primary node but now accounting for the two available interfaces. Similarly, Filebeat (v7.11.1) is also installed and set up to communicate with the Elasticsearch and Kibana services on the main node (IP address 192.168.1.9). Furthermore, the Zeek and Suricata modules are properly configured for efficient log fetching.

3 Results

3.1 Overview

After configuring the IDS nodes we will take an overview at the logs produced by Zeek (v5.09) and Suricata (v6.0.12) from all nodes collected by Filebeat (v7.11.1).

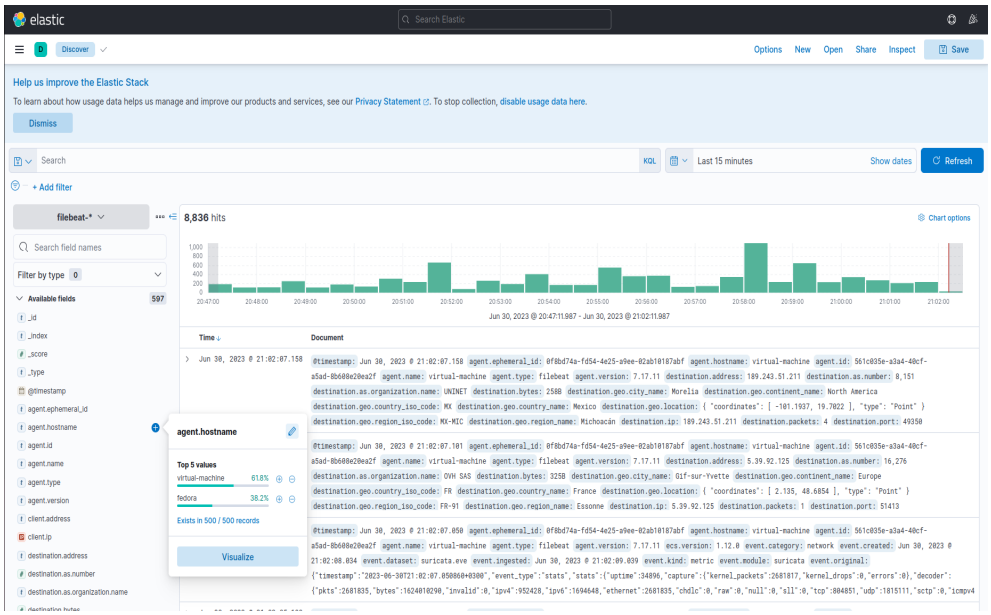

Figure 1: Filebeat overview

Here[1] we can see that most of logged traffic is generated by the main IDS node and the rest logged traffic is generated by the secondary one.

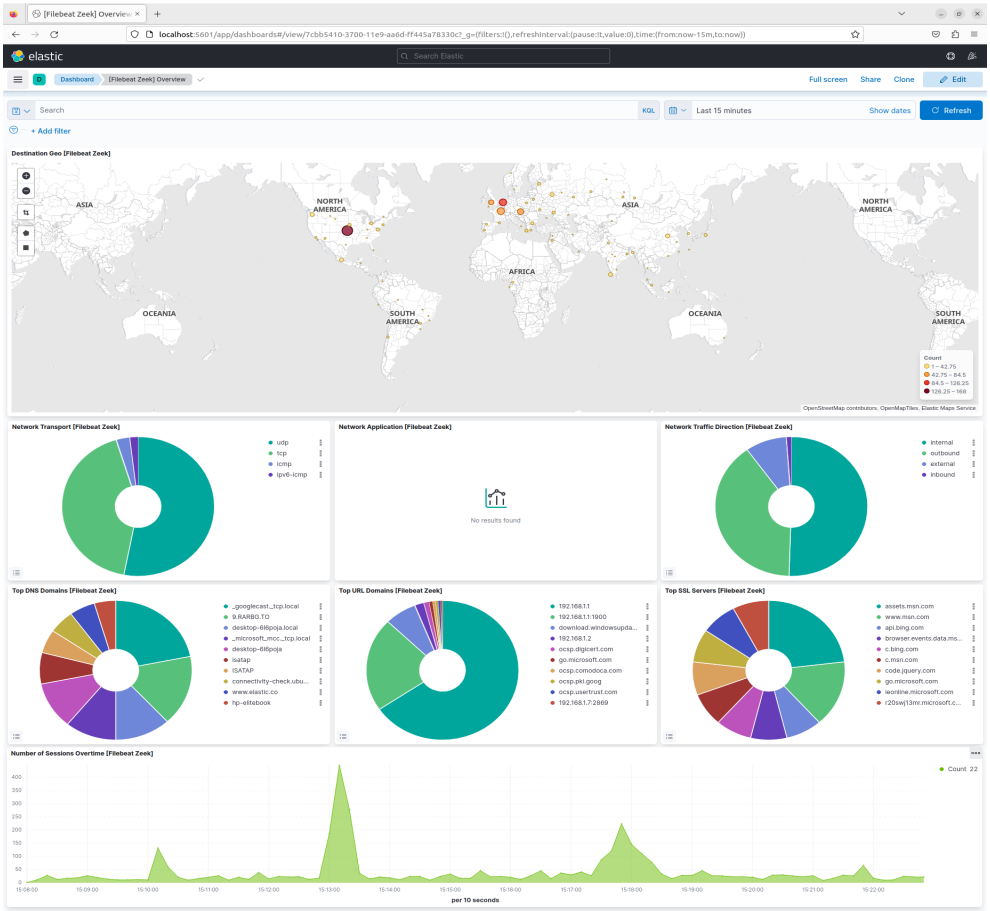

Figure 2: zeek kibana dashboard.

Moving on to the Kibana dashboard displaying Zeek overview [2], it is evident that the majority of the external traffic originates from the United States, the Netherlands, and France. Most of this traffic utilizes the UDP protocol, with the predominant traffic stemming from the Googlecast protocol. The latter is typically employed by browser plugins such as those in Google Chrome, which periodically scan the network for Googlecast-compatible devices. A noteworthy observation is that Zeek also logs traffic emanating from the host of the virtual machine (for example, the Windows update website). However, it appears that Zeek does not log a significant amount of traffic from the broader internet, with most logged traffic pertaining to intranet communications.

3.2 Attack

For the attack scenario the infection monkey tool is used. We will simulate a ransomware infection through the use of an exploit (if successful). Our target is to test our IDS in order to detect the attempted attack at the earliest possible point. For convenience, Infection Monkey is installed on our secondary Zeek node, with the primary IDS node serving as the target of the attack. To accomplish this, we download the most recent appimage of Infection Monkey for Linux (v2.2.1) and execute it. Following this, we proceed to use the browser version of Monkey Island.

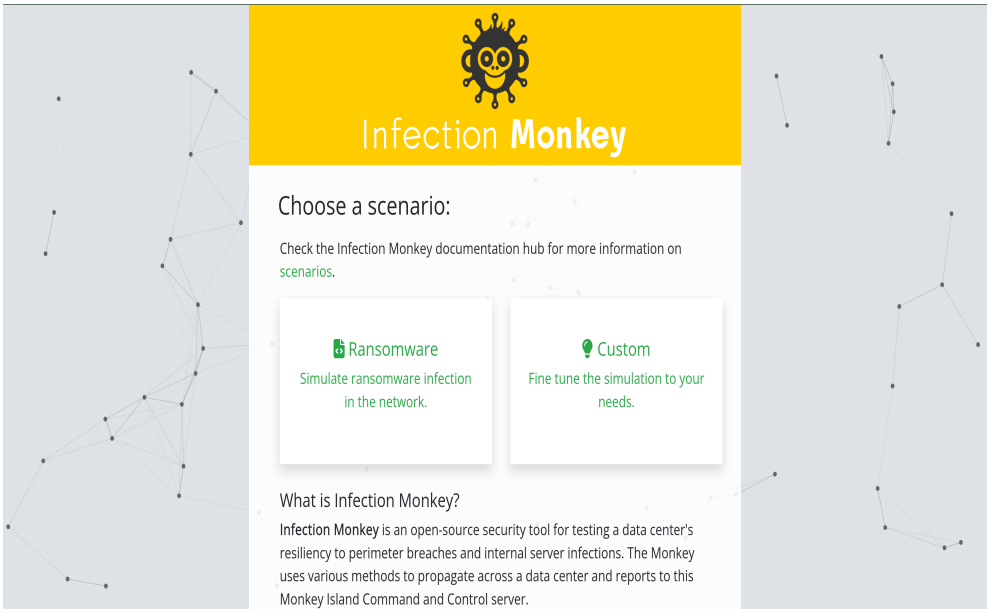

Figure 3: Monkey island startup screen.

On the first start it is required to create a user for the monkey island and when its done we choose the ransomware infection simulation scenario[3] and then we choose most of the exploiters and we enable the agent network scanning option[4].

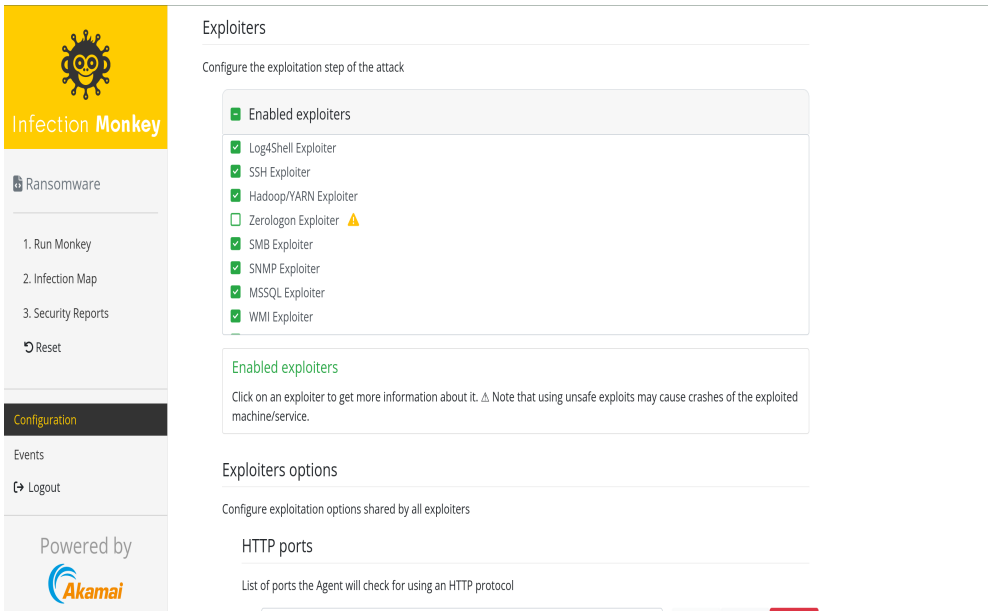

Figure 4: Attack options screen, here the selected exploits are shown.

Subsequently, we initiate the 'monkey' from within Monkey Island and await its network scanning operation. The monkey will strive to exploit the target machine[**?**].

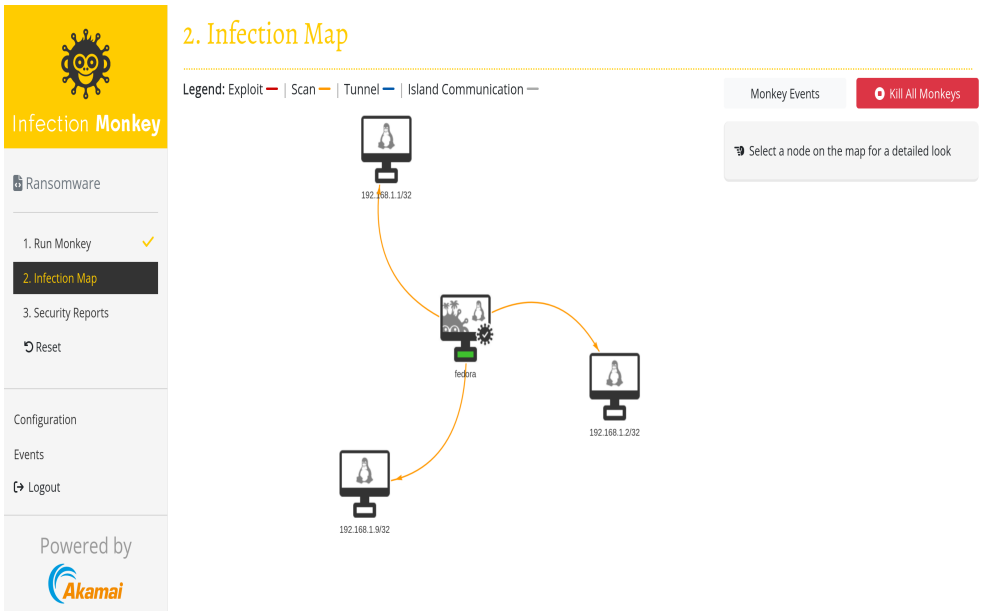

Figure 5: Monkey network scanning and exploitation.

Following a brief period, we turn our attention to the primary IDS Kibana dashboard to review Suricata alerts6. It becomes apparent that Suricata successfully detected Monkey's exploit attempt. Interestingly, the attempted exploit sought to leverage a critical vulnerability in log4j for remote code execution (CVE-2021-44228). Although the exploit—and consequently, the attack—didn't succeed, our IDS timely identified the potential threat. If our IDS had not been passive, it could even have automatically prevented this exploit attempt.

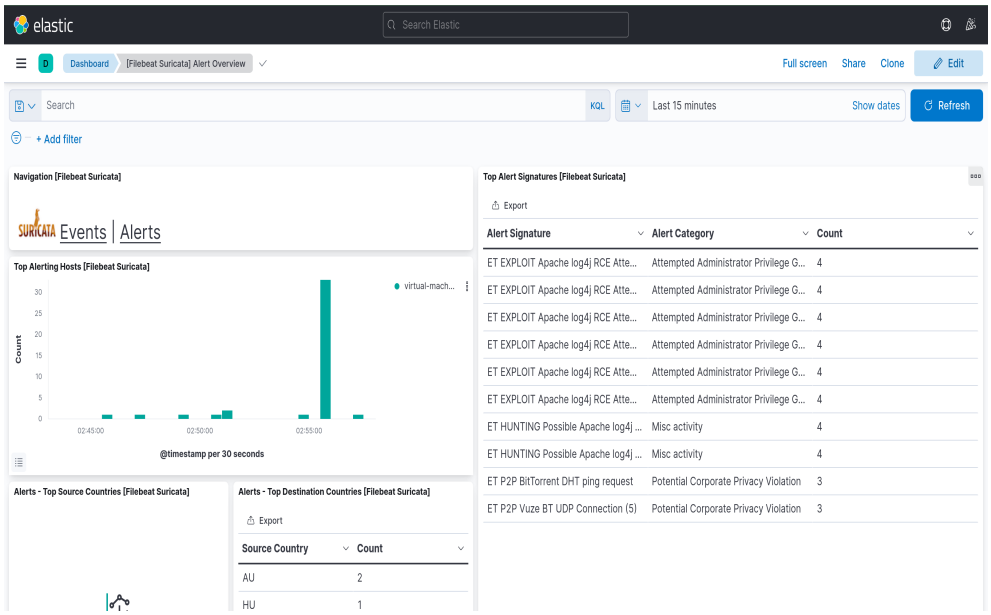

Figure 6: Suricata kibana dashboard.

4 Conclusions

Despite the attack's failure, this simulation underscores the reliability of our IDS in promptly and efficiently detecting attempted network intrusions at their nascent stages. The design also provides future scope for active intervention, potentially leveraging active dropout rules and even machine learning techniques, to further bolster security measures and intrusion detection capabilities.

5 Bonus-Future work

5.1 Elastic ML Anomaly detection

We have also used the Elastic machine learning feature on kibana in order to enchance our IDS via machine learning anomaly detection. We managed to detect anomalies on the occasions of high count of network denies, high count of network events, high count by destination country and rare destination country. The anomaly index scales from 0 to 100.
Here[7] it can be seen that we have detected a high anomaly rate on some occasions, particularly on June 25 or June 29 for example.

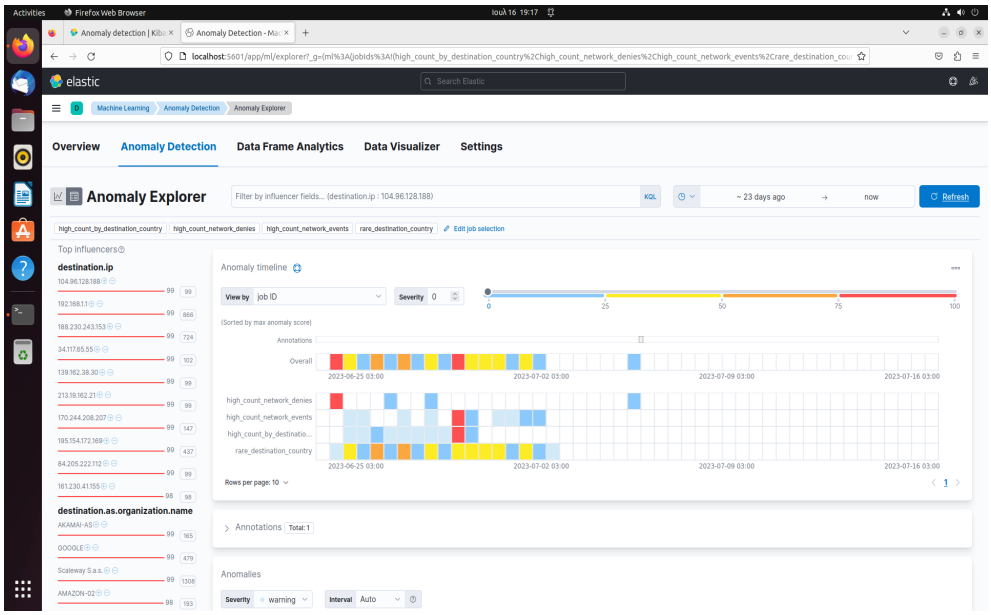

Figure 7: Kibana anomaly detection.

5.2 Zeek docker container

Moreover we have installed zeek-lts (v.5.0.4) on a ubuntu 22.04 LTS based docker container based on the main node along with filebeat (V7.17.11) to send out the gathered Zeek logs to elasticsearch stack which is also located on the main node. Zeek on this docker container was installed in clustered mode with a worker on its virtual Ethernet port (eth0).[8] In this case the docker container's IP address is 127.17.0.2 and its hostname is "79e9aa68felf". The gathered zeek logs from the docker container as well as all connected hosts can be seen on the kibana filebeat overview dashboard.[9]

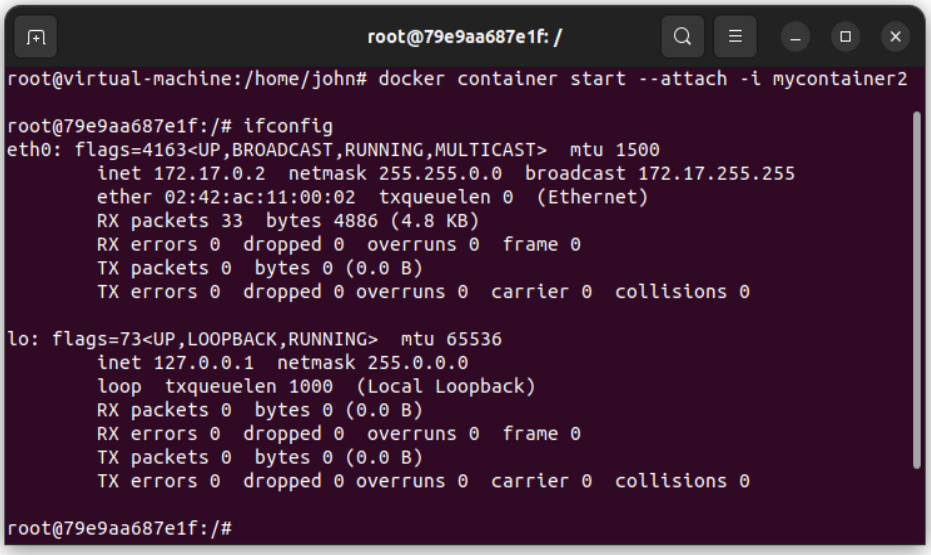

Figure 8: The docker container showing its virtual adapters.

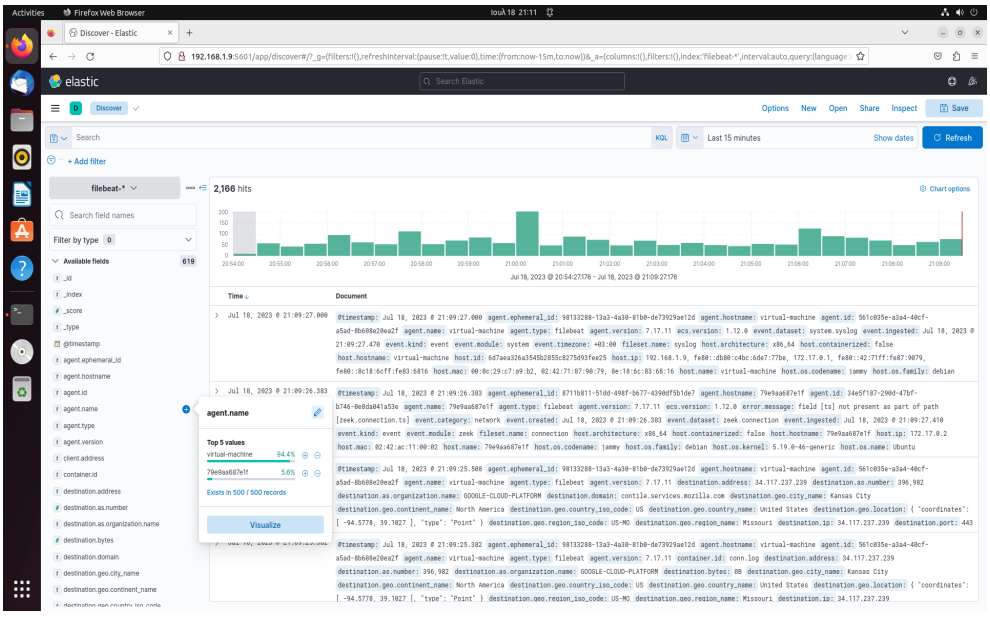

Figure 9: Filebeat overview(2)