Conceptual Documentation

Big Data Analytics

PfSense Firewall Analysis

Jonas Färmann, Dena Karini, Lorenz Wackenhut

Examiner:	Prof. Dr. Renner
Submission Date:	28.01.2021

Table of Contents

Table of Figures

Tables

1 Approach

Our approach is based on the CRISP-DM model (Figure 1), which is used as a systematic standard guideline for Data Mining projects. The term CRISP-DM is an abbreviation for Cross Industry Standard Process for Data Mining and is applicable throughout many industries. Prior to adaption, the model was modified for our purposes. We adjusted the content by omitting the modeling step as it is not applicable outside of a machine learning context. (Wirt and Hipp 2000)

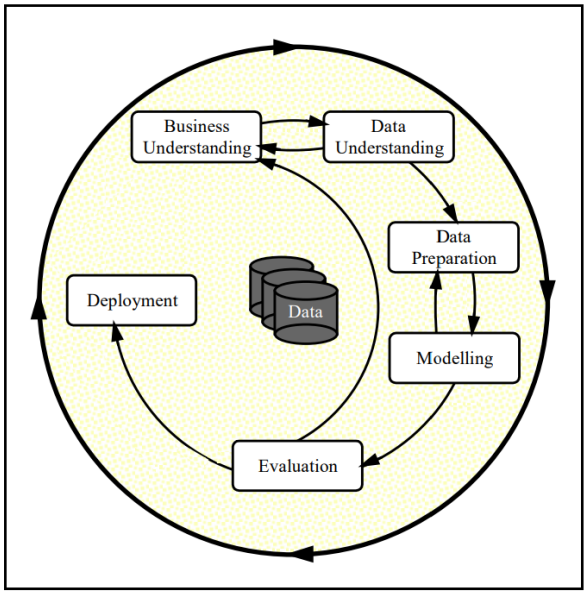

Figure 1: CRISP-DM

2 Business Understanding

The goal of this study is to build a reliable, scalable, and maintainable data architecture which is capable of ingesting big amounts of raw firewall logs in order to transform, enrich and finally analyze them. Dangerous internet traffic patterns should be identified and visualized. This is necessary to understand and improve the protection of sensitive data from various attacks on the web. To achieve the objective of continuous enhancement of the firewall performance, it is crucial to regularly review the logs provided by the software. This ensures that installed blocking rules can be updated to match the current threat environment (Taylor 2001). The data used for the analysis was provided by the company 'Frachtwerk GmbH' which uses an open-source firewall solution with the name 'pfsense' to secure their servers from incoming IP-requests. The supplied dataset consists of over 15 million data points, each representing a blocked request that was logged by the firewall (Figure 2) retrieved between September and December 2020. Each row contains multiple

columns of information about the blocked request which are explained in detail in the following sections.

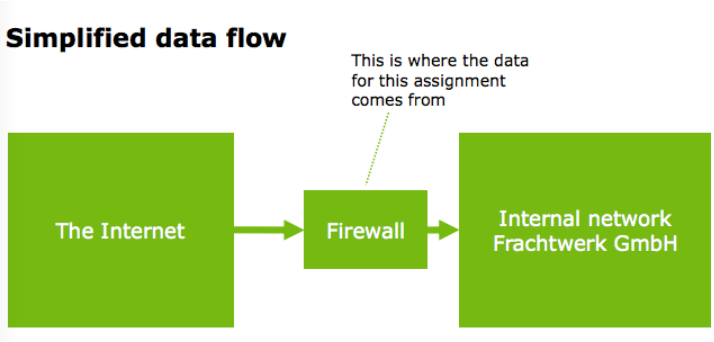

Figure 2: Simplified Data Flow (Renner 2020)

3 Data Understanding

Frachtwerk's firewall log file (pfsense.log) was retrieved on the 13th of December 2020 and takes up 2,81 GB of disk space. An initial analysis showed that the downloaded pfsense.log file contains 15.320.937 normal entries, which were logged by the firewall because of a configured rule or a default deny rule (pfSense 2021). Additionally, the file holds a small number of debug entries, which were later discarded as they were identified as not relevant for the context of this analysis. An example of a debug and normal entry can be seen in Figure 4.

```
 7 11:45:47 monitoring rsyslogd:  [origin software="rsyslogd" swVersion="8.1901.0"
rsyslog.com"] start
 7 11:45:50 gw01.extranet.frachtwerk.de filterlog: 200,,,1599469328,vtnet3,match,b
l72.23.0.2,23.54.112.189,51636,443,0,R,3551626076,,0,,
```

Figure 1: Debug entry in red and normal entry in blue

The most common fields, for a normal entry, are Action, Time, Interface, Rule, Source, Destination, Protocol as seen in Table 1.

Table 1: Most common fields of pfsense.log file

Name	Definition
Action	Shows the rule that generated the log entry (e.g. pass or block)
Time	The time that the packet arrived.
Interface	Where the packet entered the firewall (e.g. em0).
Rule Number	The firewall rule ID number which generated the log entry, if available.
Source	The source IP address and port.
Destination	The destination IP address and port.
Protocol	The protocol of the packet, e.g. ICMP, TCP, UDP, etc.

A full list of the fields is available online on Netgate's website: https://docs.netgate.com/pfsense/en/latest/monitoring/logs/raw-filter-format.html. The datatypes and approximate number of unique entries for the most common fields were retrieved as displayed in Table 2.

Table 2: Most common fields of pfsense.log file

Field name	Approx. unique entries	Data type
Source IP	309240	string
Source port	65551	int
Destination IP	633	string
Destination port	65514	int
Protocol	9	string

Furthermore, based on the described Time field, it was possible to determine that the logging took place between 07.09 11:45:50 and 07.12 09:22:02, which is approximately three months.

4 IT Landscape

The following section is going to revolve around the IT-Landscape, both software and hardware, that was chosen in order to support the extraction, transformation, loading and analysis of huge datasets in the context of firewall logs. Before building the architecture, it is fundamental to fully understand the business case of the operation, as well as the requirements of the dataset and the final analysis. Hence, the sections 'Business Understanding' and 'Data Understanding' preface the this one and can be seen as mandatory steps on the way to build a meaningful big data architecture. All of the components for the IT-landscape were carefully selected with the intention to build a reliable, scalable and maintainable infrastructure that can be easily ported to a different machine or cluster.

Reliability

The application is design in a way that it performs well and as expected under the envisioned data volume and processing load, without crashing in the case of unforeseen use or input. Furthermore, the software is configured in a way that approves access only to authorized entities. (Kleppmann 2017)

Scalability

Especially in the domain of big data it is crucial that an application can tolerate an increase in load, both data size and processing intensity. It should be designed in way so it can

dynamically adapt to the underlying hardware without the necessity to change a big chunk of the code base. Especially the ability to scale in a horizontal fashion, distributed over many nodes in a cluster, is a key feature of modern big data applications. (Kleppmann 2017)

Maintainability

Modern software is and should be an ever evolving and transforming product that is continuously enriched with features, adapted to new use cases, or ported to new environments. Because of this fact it is indispensable to make maintainability a priority when designing applications that are supposed to last. (Kleppmann 2017)

4.1 Software

4.1.1 Containerization

A common problem in software development projects is the dependency on the environment. From incompatible package versions, to differences in operation systems and discrepancies in the way floating point arithmetic are evaluated. All of these factors can not only make the software unable to run, but also change values in scientific calculations, which can have disastrous outcomes in production deployment. (Boettiger 2015)

A solution that gained significant traction in recent years, is to containerize applications in their own environment with all dependencies already installed. This enables high portability and reproducibility for projects. Another advantage of containerized software is the possibility to deploy atomic parts of the project as microservices. “A microservice is a cohesive, independent process interacting via messages” (Dragoni et al. 2017) and increases the reliability of an architecture by introducing fault tolerance in the case of failure of independent parts. (Dragoni et al. 2017)

Docker is an open-source framework and the de-facto standard for the development, shipping, and running containerized applications isolated from the local infrastructure and was used for this project. (Docker 2021)

4.1.2 Data Exploration and Analysis

In order to understand the data and its use case it is important to first explore its properties and features. The data architecture cannot be built on assumptions but needs a well-documented bases of facts for it to function like intended. For this task the Jupyter open-source ecosystem was chosen. “Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages” (Jupyter 2021) and enables the exploration and visualization of datasets in a flexible notebook environment. In particular the ‘JupyterLab’ and ‘JupyterNotebook’

development environment was used, which supports the execution on distributed clusters of machines, for an optimal use of scalable resources. (Jupyter 2021)

As the Jupyter ecosystem supports many different programming languages, one had to be chosen according to the requirements of the project. Python is the de facto language for Data Science and equally popular in the domain of Data Engineering, supports the majority of frameworks and packages for both fields and delivers satisfying performance for the processing load in the given project. This is why Python is used, not only for the data exploration and analysis in this project, but also for the preprocessing and transformation.

After the data processing pipeline has been built, once again Jupyter was used to deepen the first exploratory insights and analyze the data in depth. Packages that were used for these steps include the popular data analysis library 'Pandas' as well as visualization modules like 'Matplotlib' and 'Seaborn'.

4.1.3 Data Processing

Meaningful data analysis and visualization requires the data set to be in a cleaned, schematized and easily accessible format. A standard procedure in this context is the Extract-Transform-Load-Process (ETL) which collects heterogenous data form multiple sources and integrates it into the desired output configuration. A framework that is widely adopted for the transformation of big data sets is 'Apache Spark'. (Sun and Lan 2012 - 2012)

Spark is a cluster computing framework which can be regarded as the successor of 'MapReduce' with a performance advantage of up to 10-fold, while providing the same reliability and scalability. Spark is written in Scala und supports Python, Java and R through APIs. 'PySpark' was used to build a performant ETL-pipeline that can easily be deployed on a local machine, or in cluster mode, with the change of the spark submit parameter. (Zaharia and Franklin 2010)

4.1.4 Data Storage

While researching for the best data storage option for this project, the main focus was to settle for a system that can utilize the distributed nature of a cluster, while retaining the advantages of traditional relational database systems (RDBMS). 'PostgreSQL' is arguably one of the most popular and advanced open-source database systems which has been used for decades by an uncountable number of companies. The biggest drawback however, is the lack of support for distribution, which is why PostgreSQL by itself is not suited for the requirements of this project. Fortunately, it can be updated with an extension called 'Citus'

to scale out CPU, memory and storage in order to support parallelized operation across multiple nodes in a cluster. (Citusdata 2021)

4.1.5 Data Visualization

The information in the pfSense firewall log is a series of data points ordered by the time of creating, which qualifies it as a timeseries. One of the most popular analytics platforms for timeseries data is 'Grafana'. Grafana is open-source, highly customizable and intuitive to use, perfectly suitable for the task at hand. (Grafana Labs 2021)

4.2 Hardware

The hardware requirements for this programming task are that "the software must be able to run on a Debian or Ubuntu based GNU/Linux on top of hardware resources as of a typical standard laptop" (Renner 2020). With the purpose of complying with this requirement, all of the development was done on personal laptops, which led to some limitations, especially in terms of execution times. Nevertheless, the whole data-pipelineö. including all analysis can be run on commodity machines with as little as 8 GB of RAM. In an ideal scenario the whole pipeline would be deployed on a cluster with a master node and multiple workers in order to leverage the full potential of the scalability options.

4.3 DataPipeline

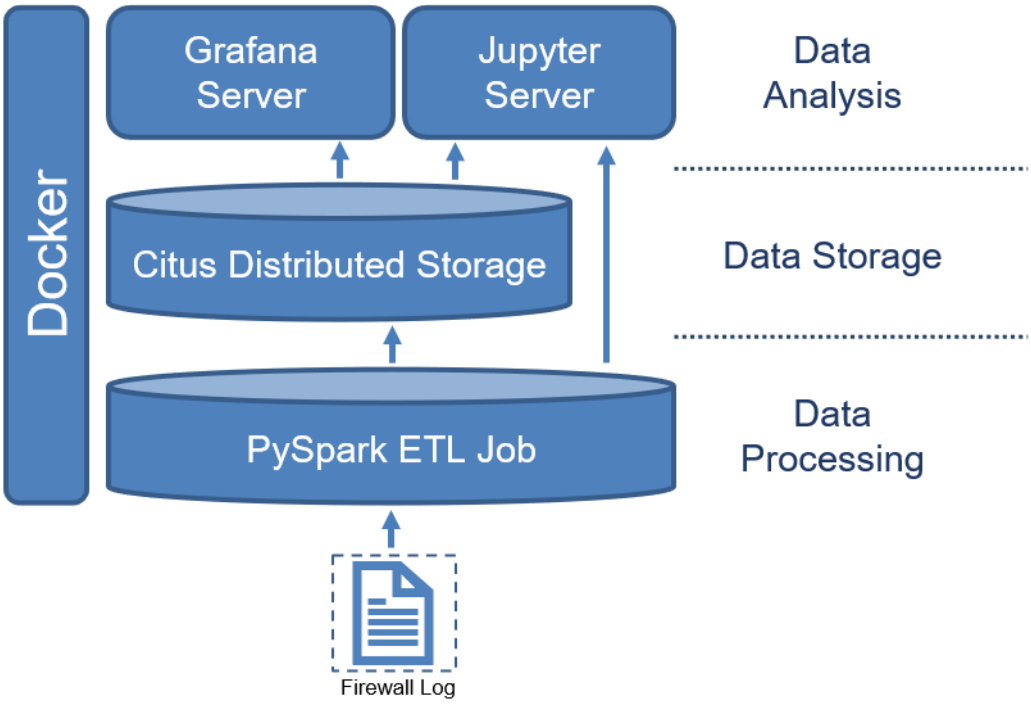

Figure 3: Graphical representation of the data pipeline

All of the previously mentioned individual parts have been orchestrated into a single data pipeline which firstly ingests the raw files into the PySpark ETL job. The same script

performs multiple cleaning, transformation and enrichment tasks to bring the log files into a structured and 'queryable' schema. Depending on the analysis method, the final dataset can now be saved in the columnar dataformat 'Parquet' or loaded into the Citus distributed storage. The last step of the pipeline is the data analysis and visualization, which is performed on a Grafana and Jupyter Server respectively. Every component of the architecture has been containerized as this enables the pipeline to be reliable against failure and ensures high portability. The architecture is depicted in Figure 3.

5 Data Preparation

The following section will give the reader a clear understanding of the data processing steps that are performed in the PySpark ETL job. The section is split into four parts which preserves the overview and the allows for the description of individual steps in more detail.

5.1 Extraction

The extraction process ingests the raw firewall log and performs light pre-processing steps to enable the transformation in the next step. The data rows are separated from logging rows and the dataframe is split into IP4 and IP6 entries, as they inherit different columns. In order to enable join queries later on a monotonically increasing ID is added to every entry.

5.2 Transformation

The transformation encompasses all methods that are used to clean and format the dataframes. The continuous strings, which make up the dataframe at this point, are split into distinct columns. Afterwards all columns that only contain null values are dropped and the time column is formatted as a timestamp. Finally, the columns are reordered and the dataframes are returned. As the dataframe for IP6 contains different columns than IP4, the transformation methods are also adapted for this peculiarity.

5.3 Enrichment

The Enrichment step incorporates three tasks. Firstly, the observations are scanned for threats in the context of exceptional high band width over a short period of time. The same holds true for the scan of IP's that show up in an unusual high frequency. Both of these cases can be signs for 'distributed-denial-of-service' (DDOS) attacks. The corresponding observations are flagged accordingly. The last enrichment step is to augment the IP information with geo specific details like country code.

5.4 Loading

The last step in the ETL job is the loading of the data. Up to this point the raw data has been cleaned, schematized, enriched and separated into three dataframes. One for all common columns between IP4 and IP6 and two for IP related information. The data can now be written into the Citus distributed storage or exported as Parquet files.

6 Data Analysis

The data analysis was split into 5 different sections: 1) time, 2) source and destination IP, 3) ports, 4) protocols and 5) bandwidth. This enabled the team to delegate tasks and look at each feature individually before forming synergies between them in 6) additional features and Grafana. The whole process is documented in Jupyter Notebooks and can be found in the appendix.

6.1 Time

Figure 4 shows that the firewall log entries are usually only seconds apart.

```
0      Sep  7 11:45:50 gw01.extranet.frachtwerk.de
1      Sep  7 11:45:52 gw01.extranet.frachtwerk.de
2      Sep  7 11:46:00 gw01.extranet.frachtwerk.de
3      Sep  7 11:46:00 gw01.extranet.frachtwerk.de
4      Sep  7 11:46:12 gw01.extranet.frachtwerk.de
```

Figure 4: Logs entries with time column

Therefore, an aggregation per minute was necessary to identify spikes of logs, which could hint to security threats like denial-of-service attacks. DoS attacks refer to large traffic volumes that target the internal network by consuming the available bandwidth (Keromytis, 2011). To identify high logs per minute a baseline is needed (Figure 5).

```
count    130593.000000
mean        117.871310
std         256.359653
```

Figure 5: Logs per minute baseline data

With as high as 30080 logs per minute, as displayed in Figure 6, the top log counts per minute are much higher than the baseline mean of 117 (Figure 5).

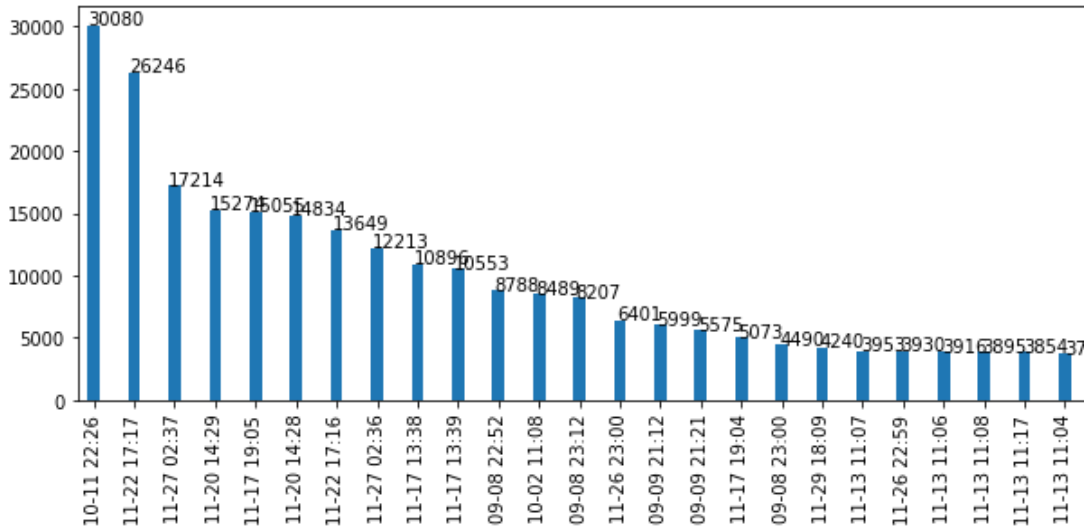

Figure 6: Logs per minute desc

Figure 7 indicates that the highest spike (10-11 22:26) was caused by the source IP 213.59.4.26.

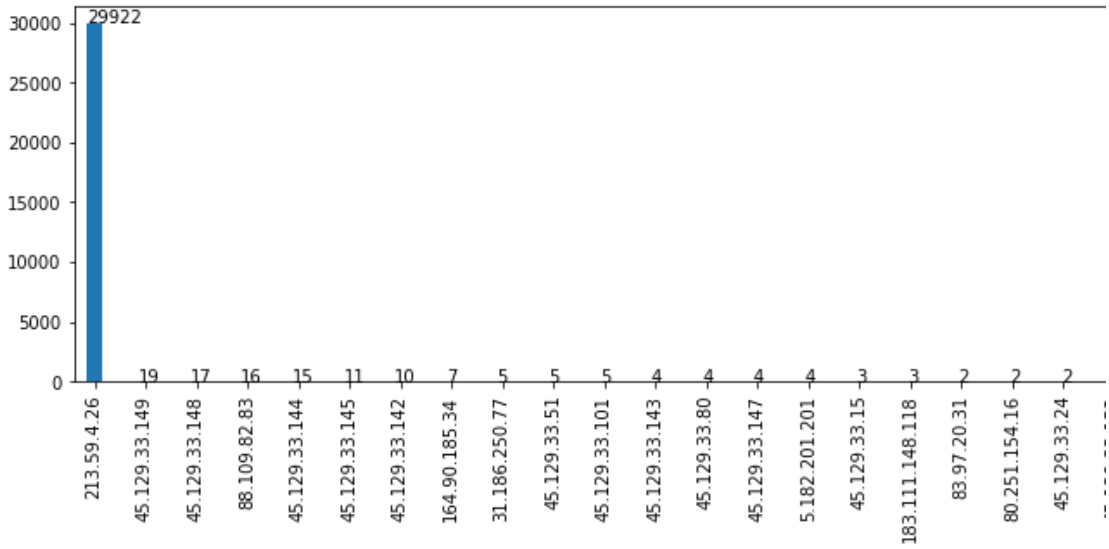

Figure 7: Logs per minute desc

Apart from the conclusion that this IP address is malicious and should be blocked, a lookup on shodan.io showed the following information.

Table 3: Shodan.io for 213.59.4.26

Country	Organization	ASN	ISP	Last Update
Russia	Rostelecom	AS8342	Rostelecom	2020-12-25T15:23:02.040649

To automize this pattern recognition processes a pyspark function was implemented. The python function tags all entries with a per minute log frequency of bigger than 117. Overall, 6168543 entries were tagged which includes 1124 distinct source IPs.

6.2 Source and Destination IP

A direct look at the log frequency of each source IP address shows that some appear unusually often (Figure 8).

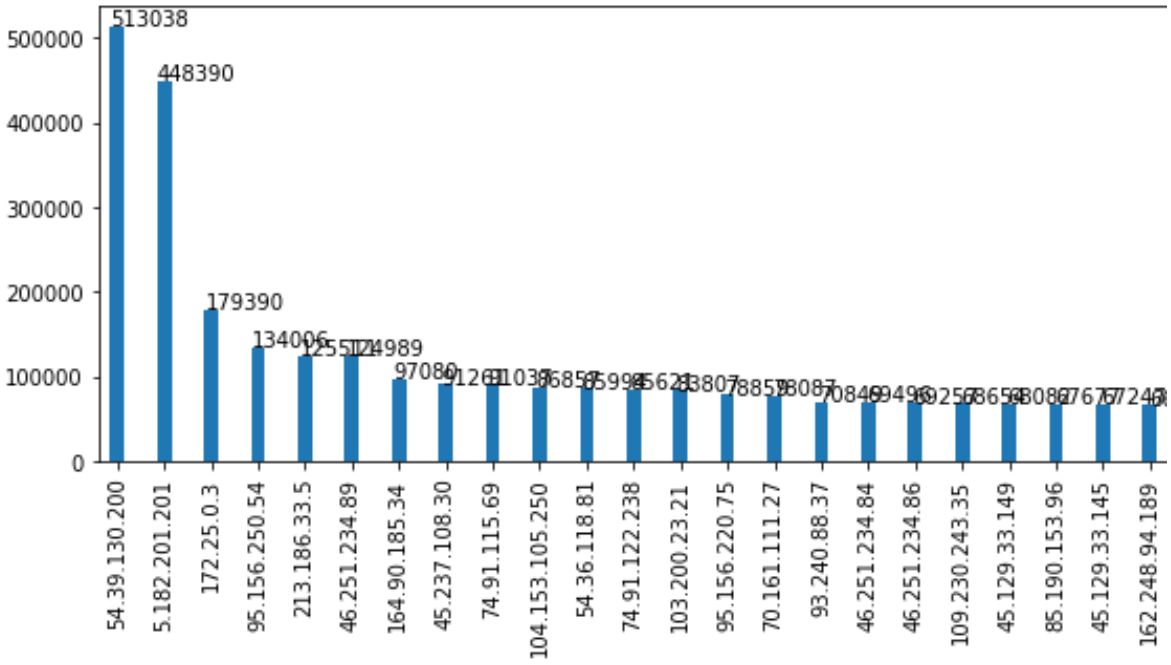

Figure 8: Source IP frequency desc

The source IP address 54.39.130.200 could be part of an external DoS attack, whereas the internal source IP address 172.25.0.3 could indicate that someone was able to hijack a server on the company's network. Unfortunately, the project team did not have access to the internal network of Frachtwerk, which means that no further analysis could be done. However, even though the firewall blocked the request, we recommend to check the IP address internally. Figure 9 reveals the log frequency of each destination IP address in descending order.

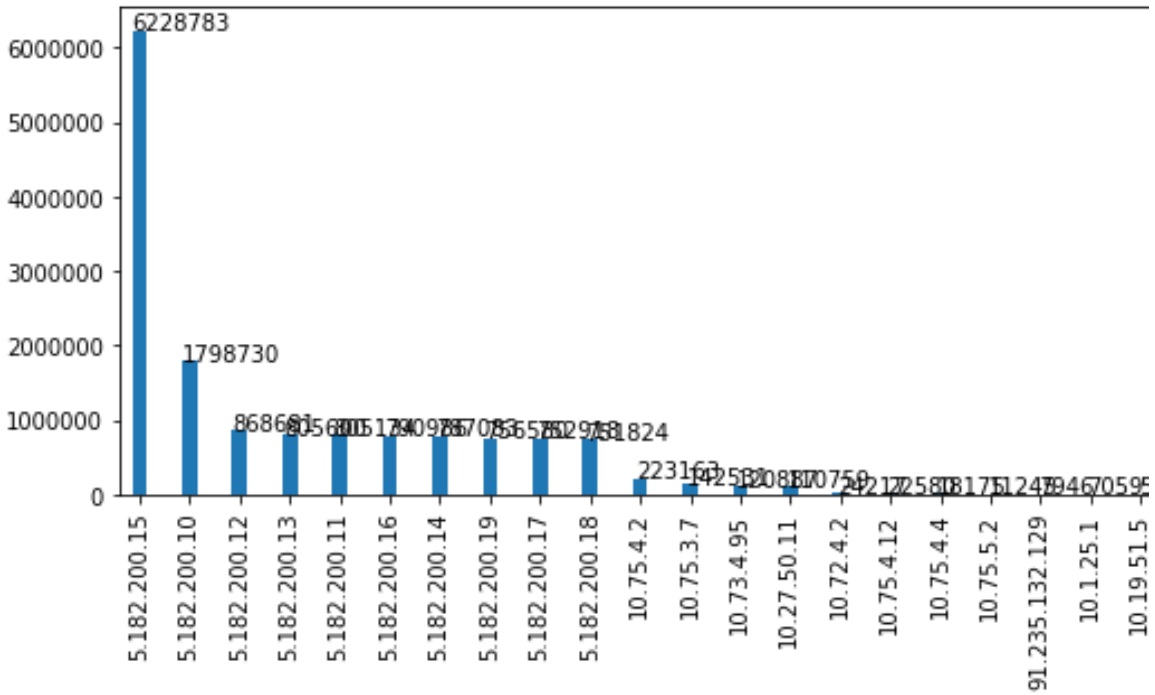

Figure 9: Destination IP frequency desc

It becomes clear that the destination IP addresses starting with 5.182.200 show unusual behavior. Again, the project team had no access to the internal services, but we also recommend to analyze the destination IPs internally.

6.3 Ports

To identify noticeable ports, the ports have been broken down by type and ranked by times requested as show in Figure 10.

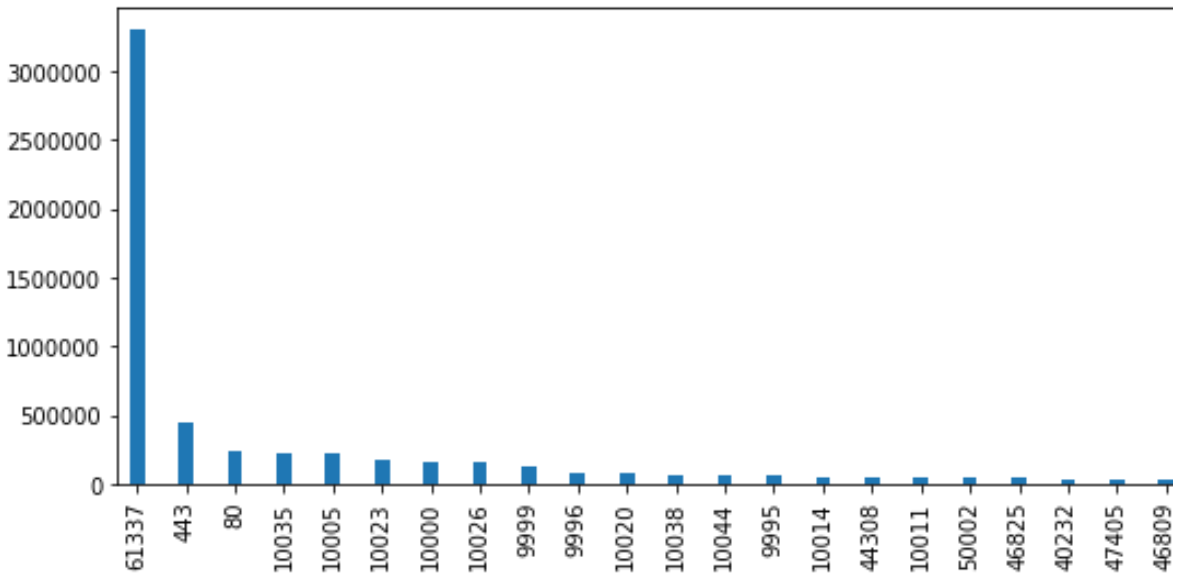

Figure 10: Source port frequency desc

Table 4 gives information on the most frequent ports and shows that the requests using port 61337 and 10005 are most likely malicious.

Table 4: Information on most frequent source ports

Port	Usage pattern
61337	Mostly used by trojans (AdminSub 2021)/61337
443	HTTPS (AdminSub 2021)/443
80	HTTP (AdminSub 2021)/80
10035	Unassigned (AdminSub 2021)/10035
10005	Ipcserver Mac OS or trojan (AdminSub 2021)/10005

A deep dive into the source IPs using port 61337 reveals that most entries were generated by 54.39.130.200 (Figure 11).

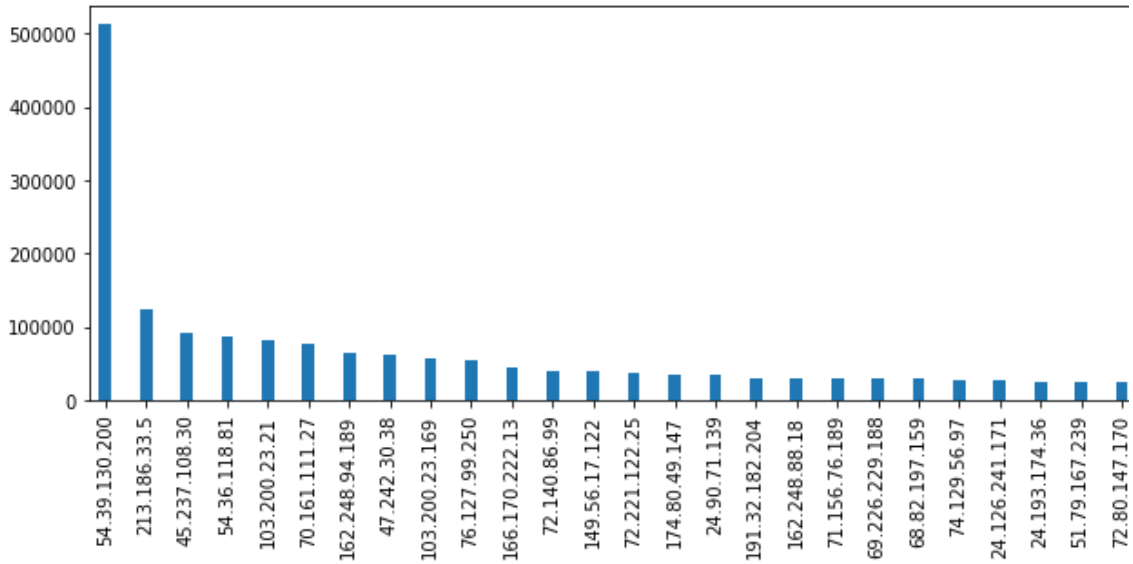

Figure 11: Information on most frequent source ports desc

A lookup on shodan.io shows following information, which indicates that the underlying server might be infected by a trojan (Table 5). Trojans are malicious files that claim to be something desirable, like a discount on shopping websites, but are in fact malicious. In this case the trojan was most likely used to connect the underlying server to a botnet to use its resources to perform DoS attacks (Broadcom 2019).

Table 5: Shodan.io for 54.39.130.200

City	Organisation	ISP	Hostname	Last Update
Victoria CA	OVH SAS	OVH SAS	firewallroozser-vers.bairesrp.net	2021-01-25T03:33:58.088311

Figure 12 reveals the frequency of the destination ports in the dataset in descending order.

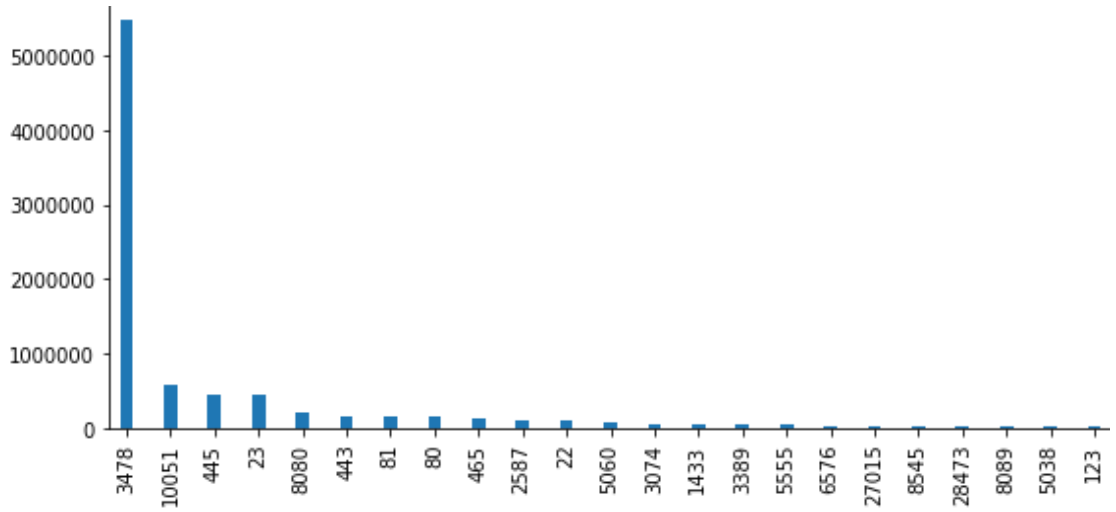

Figure 12: Destination port frequency desc

Again, the following Table 6 shows the most frequent ones and their usage pattern.

Table 6: Information on most frequent destination ports

Port	**Usage pattern**
3478	VoIP StUN behavior (AdminSub 2021)/3478
10051	Zabbix trapper (AdminSub 2021)/10051
445	Microsoft DNS or threat (AdminSub 2021)/445
23	Port protocol can be a malware known as TruvaAtl and is to be handled with caution or telnet, which is one of the oldest Internet protocols and has numerous security vulnerabilities (AdminSub 2021)/23
8080	A common alternative HTTP port used for web traffic (AdminSub 2021)/8080

In the case of destination IPs with port 3478 the same IP, which was referenced in the previous section (Source and Destination IP), is on top of the list (Figure 13).

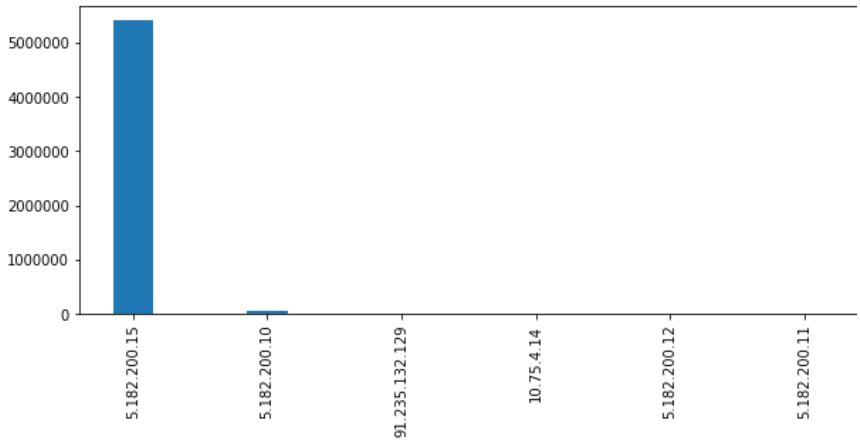

Figure 13: Information on most frequent destination ports desc

Even though, we can't evaluate anything inside of the internal network, the high frequency could indicate that the IP address belongs to the server which handles VoIP for Frachtwerk (Table 6).

6.4 Protocols

Figure 14 shows the most used protocols in descending order.

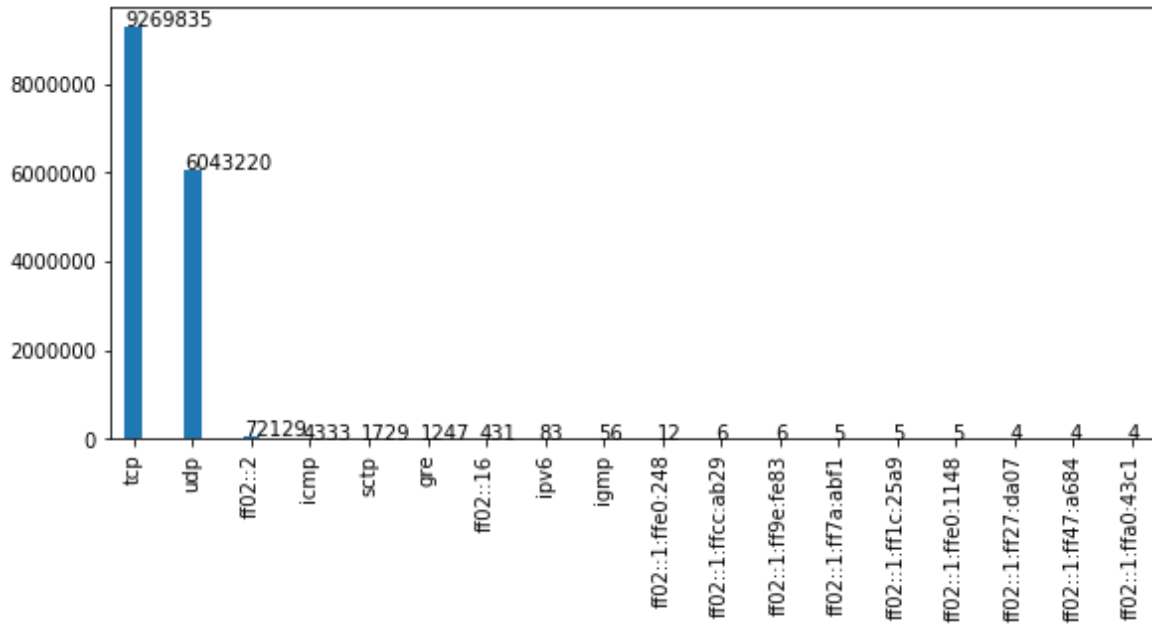

Figure 14: Protocol frequency desc

Table 7 reveals the usage pattern for these protocols.

Table 7: Protocol description

Protocol	Usage pattern
tcp	Transmission Control Protocol. Reliable, ordered but heavyweight handling. Mostly used for HTTP, FTP, SMTP (IETF 2021)/rfc793
udp	User Datagram Protocol. Unreliable, not ordered but lightweight handling. Mostly used for video and audio traffic (IETF 2021)/rfc768
icmp	Internet Control Message Protocol. Typically used or diagnostic or control purposes, also error handling (IETF 2021)/rfc792
sctp	Stream Control Transmission Protocol. Improvement over TCP and UDP, more secure, better reliability (IETF 2021)/rfc4960
gre	Generic Routing Encapsulation. Tunneling protocol. Connect one intranet to the other for example (IETF 2021)/rfc2784

Whereas, tcp and udp are very commonly used, a closer look at log entries using icmp protocol reveals that most of the traffic was generated internally (Figure 15).

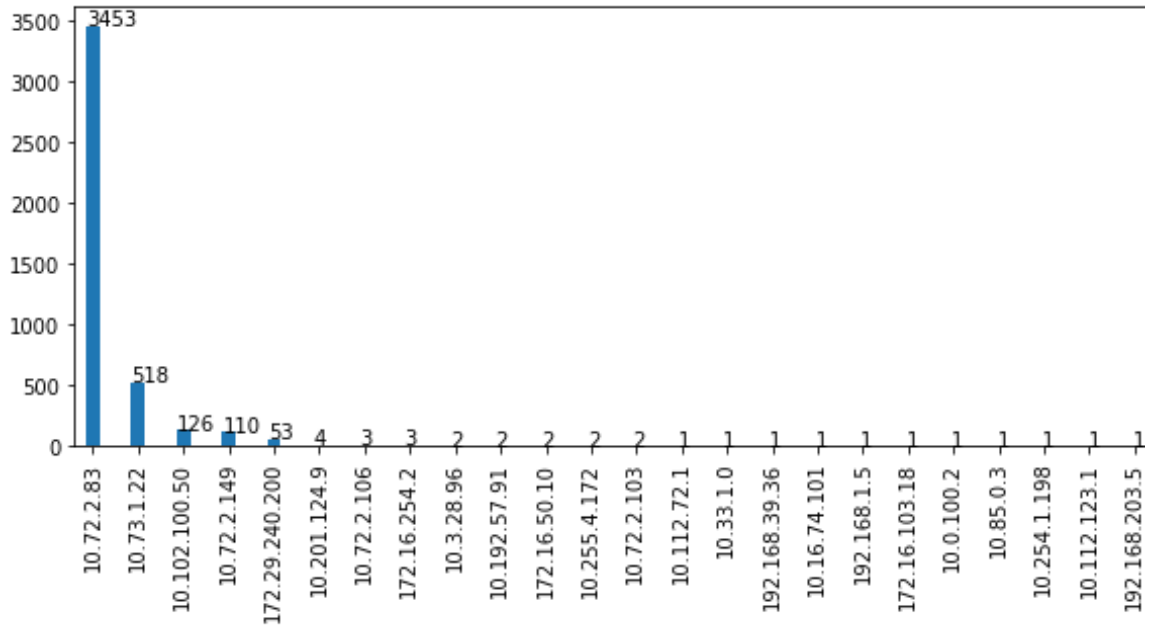

Figure 15: Source IP addresses using icmp protocol desc

The sctp protocol tells a different story as the source IP address 83.97.20.249 is trying to access the service from the outside (Figure 16).

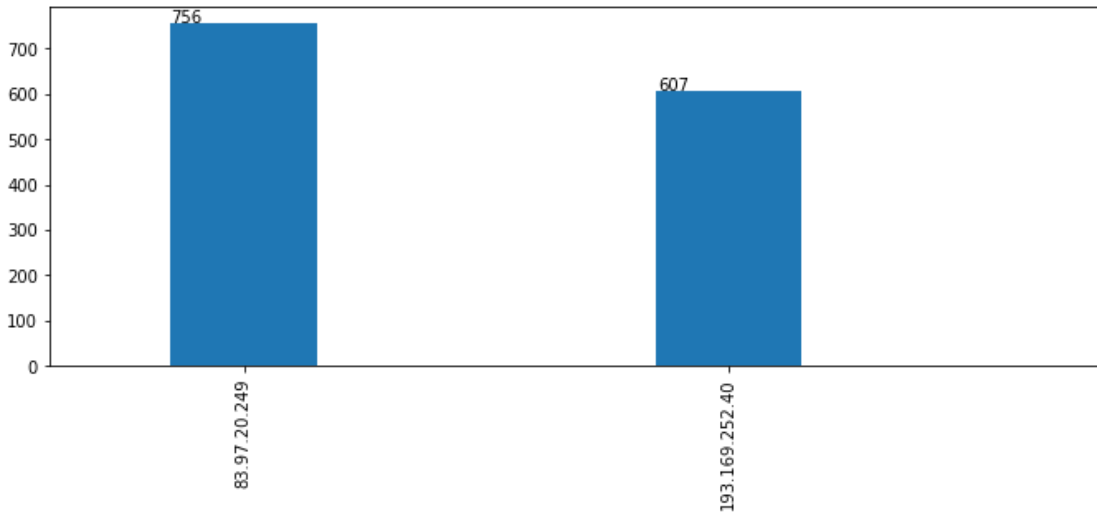

Figure 16: Source IP addresses using sctp protocol desc

Table 8 shows the ipinfo.io report of the IP address.

Table 8: ipinfo.io report for 83.97.20.249 (ipinfo.io 2021)

Country	Organisation	ISP	ASN	Last Update
Romania	OvO Systems Ltd	ripe	AS9009 M247 Ltd	2021-01-25T03:33:58.088311

6.5 Bandwidth

Bandwidth is defined as the amount of data transferred within a network given as bits per second (Comer, 2008) and important to understand the quality and speed of a network, as well as internet traffic. Therefore, for this analysis, the log entries were aggregated by minute and data length to identify spikes bandwidth spikes (Figure 17).

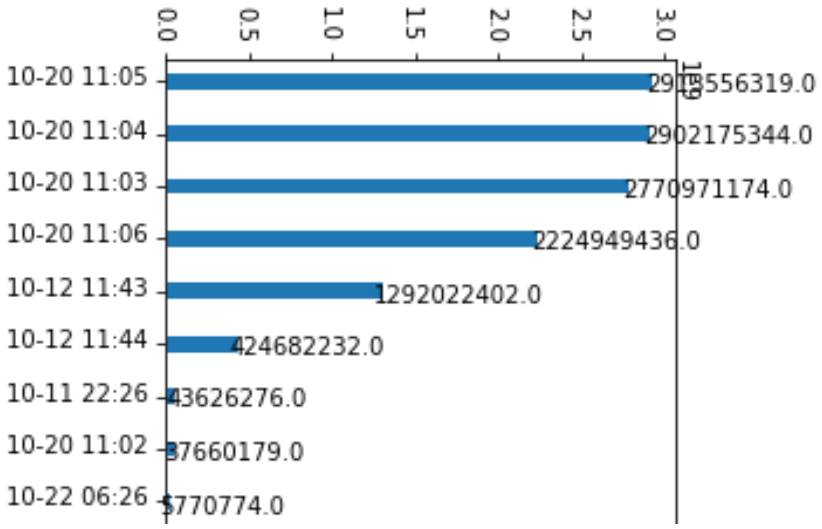

Figure 17: Source data length per minute desc

The baseline of data length per minute is displayed in Figure 18.

```
count    1.305930e+05
mean     9.897071e+04
std      1.550425e+07
```

Figure 18 Data length baseline data

The spikes, which are much higher than the mean of 98971, show a high correlation between them as the attacks were performed over the course of several minutes (Table 9).

Table 9: Bandwitdh time correlation pattern (top 8)

Nr	Time	Pattern
1	10-20 11:05	1 & 2 & 3 & 4 & 8 Possible denial of service attack
2	10-20 11:04	1 & 2 & 3 & 4 & 8 Possible denial of service attack

3	10-20 11:03	1 & 2 & 3 & 4 & 8 Possible denial of service attack
4	10-20 11:06	1 & 2 & 3 & 4 & 8 Possible denial of service attack
5	10-12 11:43	5 & 6 Possible denial of service attack
6	10-12 11:44	5 & 6 Possible denial of service attack
7	10-11 22:26	Possible denial of service attack
8	10-20 11:02	1 & 2 & 3 & 4 & 8 Possible denial of service attack

A deep dive into the pattern 1 & 2 & 3 & 4 & 8 shows that the internal source IP address 10.72.2.83 was responsible for most of the traffic (Figure 19).

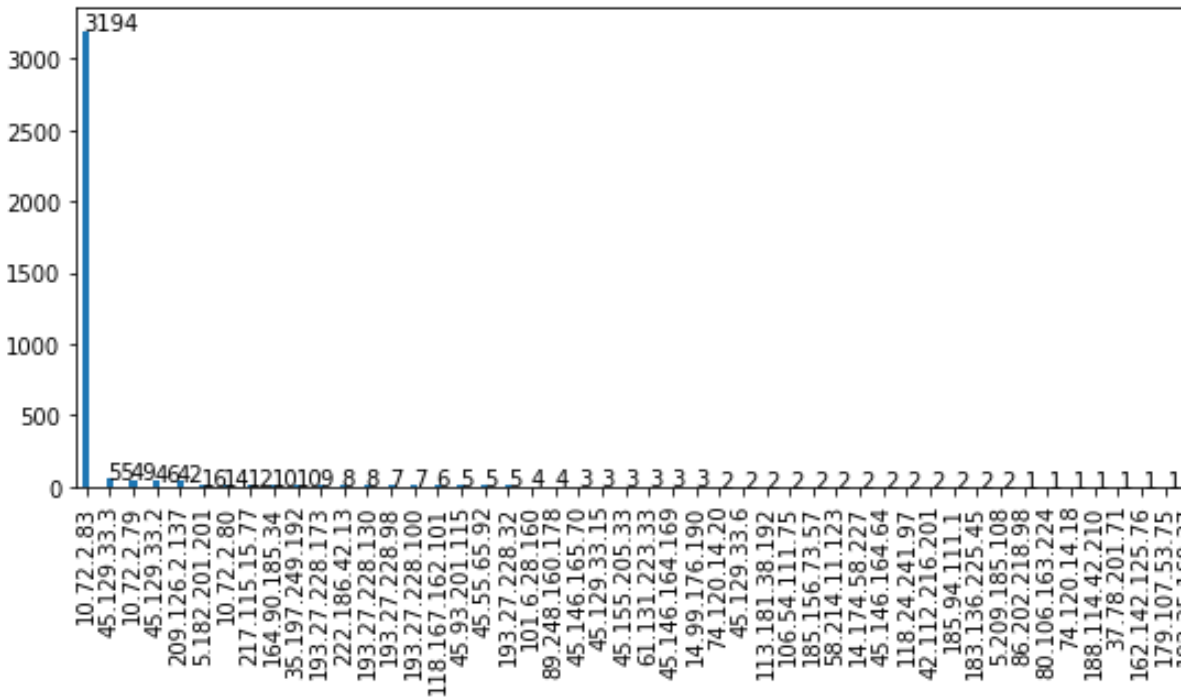

Figure 19: Source IP addresses for pattern 1 & 2 & 3 & 4 & 8 desc

In this case a trojan could have infected the internal underlying machine to target the network from the inside. Pattern 5 & 6 shows a similar pattern as source IP address 10.73.1.22 send the most packets (Figure 20).

To automize this pattern recognition processes a pyspark function was implemented. The python function tags all entries with a per minute data length of bigger than 98971. Overall, 166807 entries were tagged which includes 56 distinct source IPs.

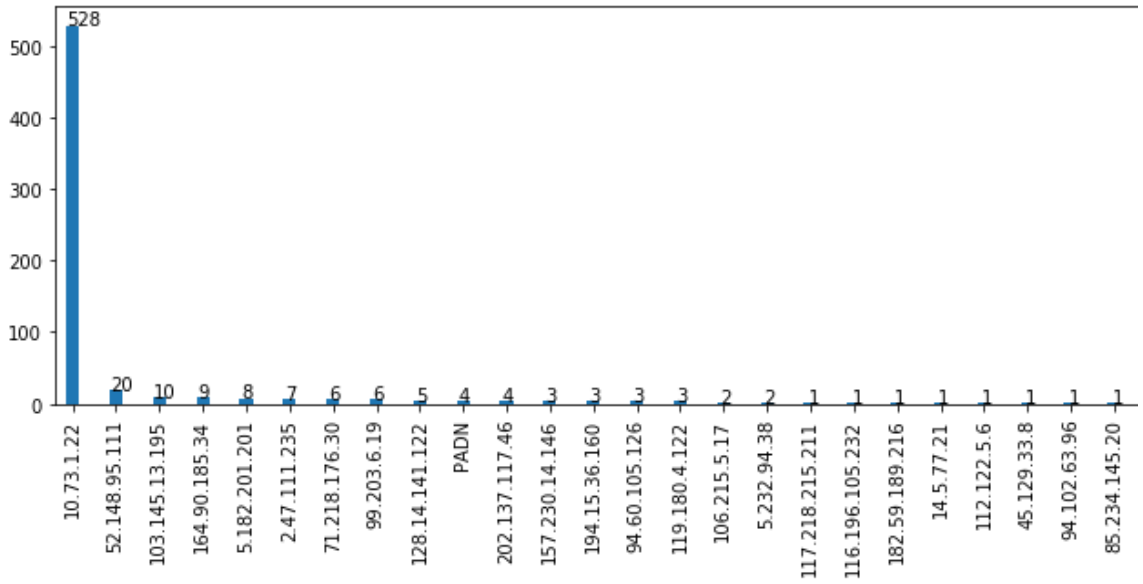

Figure 20: Source IP addresses for pattern 5 & 6 desc

6.6 Additional Features and Grafana

To enhance the dataset, country information was added for each entry by utilizing the open source ip.sb geoip API, as it doesn't have a rate limit, like the proposed Maxmind geoip API for example. Additionally, several IP blocklist were retrieved, like the Fedo Tracker Botnet (Fedo Tracker Botnet 2021) to identify malicious IPs. Unfortunately, none of these matched with the IPs in the dataset.

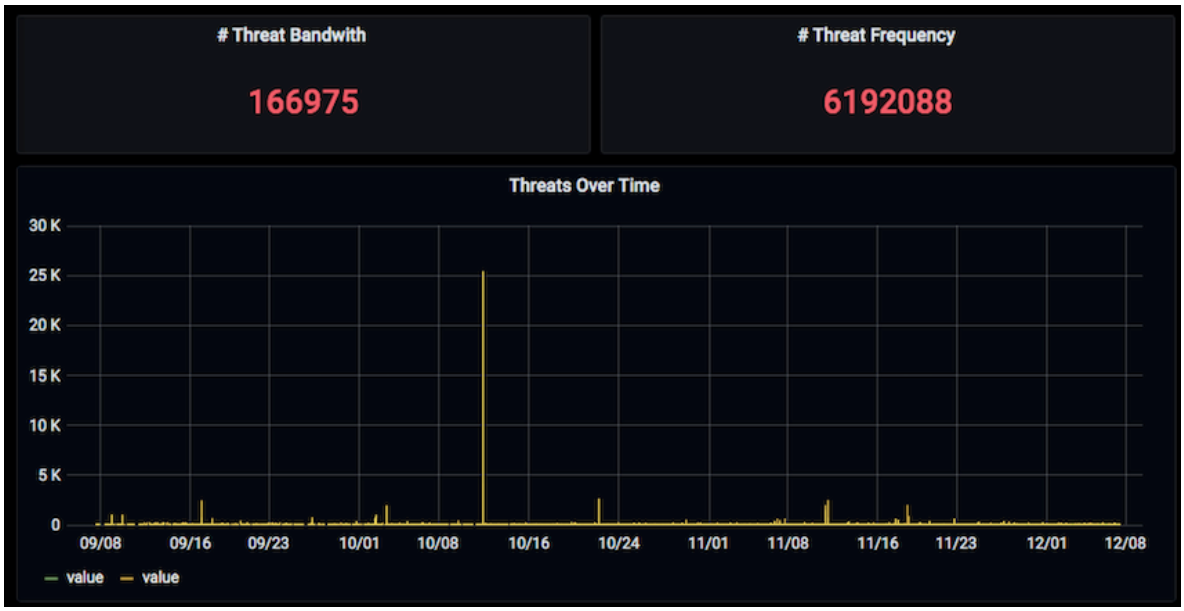

Figure 21: Total threats bandwith and total threats frequency.

Figure 22: Total logs per country

Figure 23: Total logs over time

Figure 24: Full dashboard in Grafana

The final result of the data analysis and added features can be seen in Grafana (Figure 21 - 24). Due to the high volume of data, Grafana seems to approximate numbers, as the values come close, but do not line up with the exact results from the SQL queries on the deployed

Citus DB. Therefore, making post threat identification easier for management and developers.

7 Deployment

The whole data pipeline, including the spark processing, the Citus distributed storage and the Jupyter and Grafana servers can be initialized with one single ‘docker-compose’ command. The individual services are configured in a way, so they wait for the completion of dependency tasks. A network has been created to connect the services in order to guarantee a seamless flow of data between the docker instances.

8 Evaluation

The analysis can be used to improve firewall activities and better understand high-risk web traffic, if applicable. The noticeable patterns that came to light during the analysis and the possible attacks that were identified show that the firewall has served its purpose and that its use is a sensible and important component of Frachtwerk's security concept.

Publication bibliography

AdminSub (2021): TCP/UDP Port Finder. Edited by AdminSub. Available online at https://www.adminsub.net/tcp-udp-port-finder/, checked on 1/28/2021.

Boettiger, Carl (2015): An introduction to Docker for reproducible research. In *SIGOPS Oper. Syst. Rev.* 49 (1), pp. 71–79. DOI: 10.1145/2723872.2723882.

Broadcom (2019): Difference between viruses, worms, and trojans. Available online at https://knowledge.broadcom.com/external/article?legacyId=tech98539, checked on 1/28/2021.

Citusdata (2021): Citus. Citusdata. Available online at https://www.citusdata.com/, checked on 1/28/2021.

Docker (2021): Get Started Overview. Docker. Available online at https://docs.docker.com/get-started/overview/, checked on 1/28/2021.

Dragoni, Nicola; Giallorenzo, Saverio; Lafuente, Alberto Lluch; Mazzara, Manuel; Montesi, Fabrizio; Mustafin, Ruslan; Safina, Larisa (2017): Microservices: Yesterday, Today, and Tomorrow. In Manuel Mazzara, Bertrand Meyer (Eds.): Present and Ulterior Software Engineering, vol. 27. Cham: Springer International Publishing, pp. 195–216.

Fedo Tracker Botnet (2021): https://feodotracker.abuse.ch/downloads/ipblocklist.txt. Available online at https://feodotracker.abuse.ch/downloads/ipblocklist.txt, checked on 1/28/2021.

Grafana Labs (2021): Grafana. Grafana Labs. Available online at https://grafana.com/, checked on 1/28/2021.

IETF (2021): IETF Documents. Available online at https://tools.ietf.org/html/, checked on 1/28/2021.

ipinfo.io (2021): The Trusted Source for IP Address Data. ipinfo.io. Available online at https://ipinfo.io/, checked on 1/28/2021.

Jupyter (2021): Jupyter. Jupyter. Available online at https://jupyter.org/, checked on 1/28/2021.

Kleppmann, Martin (2017): Designing data-intensive applications. The big ideas behind reliable, scalable, and maintainable systems. Sebastopol, CA: O'Reilly Media. Available online at http://proquest.tech.safaribooksonline.de/9781491903063.

pfSense (2021): Viewing the Firewall Log. pfSense. Available online at https://docs.netgate.com/pfsense/en/latest/monitoring/logs/firewall.html, checked on 1/28/2021.

Sun, Kunjian; Lan, Yuqing (2012 - 2012): SETL: A scalable and high performance ETL system. In : 2012 3rd International Conference on System Science, Engineering Design and Manufacturing Informatization. 2012 3rd International Conference on System Science, Engineering Design and Manufacturing Informatization (ICSEM). Chengdu, China, 20.10.2012 - 21.10.2012: IEEE, pp. 6–9.

Taylor, Laura (2001): Read your firewall logs. ZDNet. Available online at https://www.zdnet.com/article/read-your-firewall-logs-5000298230/, checked on 1/28/2021.

Wirt, Rüdiger; Hipp, Jochen (2000): CRISP-DM: Towards a Standard Process Model for Data Mining. Available online at http://www.cs.unibo.it/~danilo.montesi/CBD/Beatriz/10.1.1.198.5133.pdf, checked on 1/28/2021.

Zaharia, Matei; Franklin, Michael (2010): Spark: Cluster Computing with Working Sets. Available online at https://www.semanticscholar.org/paper/Spark%3A-Cluster-Computing-with-Working-Sets-Zaharia-Chowdhury/a0858dd960c635c3e6294908b794fae52f66e42f, checked on 1/28/2021.