TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technologies

Pasqaule Polverino 201680ICVM

NetFlow Based Anomalies Detector Using Prophet Forecasting Model

Master thesis

Supervisor: Kieren Niĉolas Lovell

LCDR Royal Navy

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogiate kool

Pasqaule Polverino 201680ICVM

NetFlow-põhine Anomaaliate Detektor, Kasutades Prophet Prognoosimismudelit

Magistritöö

Juhendaja: Kieren Niĉolas Lovell

LCDR Royal Navy

Tallinn 2022

Author's declaration of originality

I hereby certify that I am the sole author of this thesis and this thesis has not been presented for examination or submitted for defence anywhere else. All used materials, references to the litterature of others have been cited.

Author: Pasqaule Polverino .....

(signature)

Date: 16.05.2022

Abstract

During the last years, the sophistication and complexity of cyber threats have been gaining the capability of bypassing traditional monitoring solutions and compromising enterprise systems silently. The result is that traditional network security solutions cannot detect and prevent these types of threats. Hence, there is growing interest in building tools relying on machine learning techniques to detect abnormal network traffic. This research will present a novel method that uses the Prophet forecasting algorithm on NetFlow data to detect abnormal network traffic. The proposed method managed to achieve a classification error of 0.0107 and an accuracy of 0.99 in the detection of anomalies.

This thesis is written in English and is 65 pages long, including 8 chapters, 38 figures and 2 tables.

List of abbreviations and terms

ASA	Adaptive Security Appliance
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
ESP	Encapsulating Security Payload
IDS	Intrusion Detection System
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
RMSE	Root Mean Square Error
SSH	Secure Shell
TCP	Transmission Control Protocol
TOR	Top of Rack
TOR	Top of Rack
UDP	User Datagram Protocol

Table of Contents

List of figures

List of tables

1 Introduction

The following study focuses on the problem of detecting abnormal behaviour in the network traffic from the NetFlow data. A network anomaly can have many causes. For example, anomalies could result from port scans, DoS and DDoS attacks, faulty hardware, or some misconfigurations. The problem is of significant importance for cybersecurity experts dealing with the setup of network monitoring systems.

To understand the importance of the problem, one should consider that in 2022 new types of threats are built with the capability of bypassing traditional monitoring solutions and silently compromising enterprise systems [1]. The result is that the traditional network perimeter and signature-based security solutions might not be enough to protect enterprise systems from these new types of cyber threats because of the fact that attackers have tools that are able to bypass these types of security solutions [1]. The result is that increasingly more tools try to detect intruders by analyzing the network traffic behaviour to identify anomalies and generate alerts on their occurrence. There are already some commercial tools available that use this approach to detect intrusions. For example, the NetFlow Traffic Analyzer commercialised by SolarWinds [2] allows configuring different types of alerts by using some thresholds on the NetFlow traffic characteristics, like input or output traffic. However, the process of manually setting alerts can become a bit error-prone when managing large environments. The result is that the tool will generate many false positives and becomes difficult to scale.

For these kinds of reasons, it can become convenient to use artificial intelligence and machine learning methods. The idea is that a machine learning model is able to extract the network traffic behaviour and its pattern to create some baseline. When there is some traffic that does not conform to these patterns, the model will identify it as an anomaly and will generate an alert.

However, these types of security solutions are not meant to be a substitute for the traditional security solutions described before, but a better performance can be achieved when combining both security tools [1].

1.1 Scope and Goal

The main goal of this research is answering to the following question: does the Prophet forecasting model can be used to detect network traffic anomalies in the NetFlow data?

The Prophet forecasting model is a time-series model, which will be introduced in a more detailed way in Section 2. The result of this research will be a prototype of a tool that automatically detects some traffic anomalies on the time series generated from the NetFlow data. However, since it is a prototype, it is not a tool that is ready and meant to be used in a real production environment since it has some limitations which will be covered in more detail in Section 8.

1.2 Assumptions

The main assumption is that the network data on which the model will be tested comes from a real-world infrastructure. The infrastructure used is one of the data centres of the company where I work. Hence, this process might introduce some biases since the network traffic highly depends on the type of service deployed in the data centre and on their load during the collection of the data. However, I do not have many other means available to obtain the data needed to test the proposed solution.

The other element to consider is that the performance of the implementation of the solution depends on the traffic volume where it has been tested. Hence, there is no guarantee that the implementation will work in other production environments, and it will be able to handle their traffic volume since it has not been tested in different environments. However, the idea of this study is to present a framework, so the developed tool is just a prototype that should be extended before being used in real production environments.

1.3 Ethical Considerations

The main ethical concern comes from the fact that the implementation has been done on the infrastructure provided by the company where I am currently employed. Hence, it would be unethical from my side to share all the implementation details, all the exact configuration lines used to set up the NetFlow collection and the exact network architecture of the environment used to collect data. However, I will try to provide this information at a higher level during the study. For instance, there will be no explicit code during the presentation of this study, but it will be substituted by pseudocode. Also, I will report only the most important configuration lines and the network diagrams will not contain all the details.

1.4 Novelty

The main novelty of this study consists of the use of the Prophet forecasting algorithm to detect anomalies in the network traffic by using NetFlow data. In fact, as the literature in Section 2 will show, there are already existing studies using time series forecasting techniques on NetFlow data to detect network anomalies. While most of the literature focuses on using techniques such as Holt-Winters, there is no literature regarding the use of the Prophet forecasting algorithm on NetFlow data yet.

Another small novelty is that the proposed framework will try to separate and build a distinct time series for each of the services which generated the network traffic. The service identification will be made through the use of layer 4 protocol number and port number for UDP and TCP. The main reason is that this method allows some better visibility of which service is the cause of the anomaly. It is considered a lesser novelty because the literature does not make this separation but builds a single time series from all the NetFlow data collected.

1.5 Research Methods

The research will use quasi-experimental methods since the data used to build the proposed data will not be artificially generated but will be collected from a real-world environment. Hence, it is not possible to create a truly experimental setup since this process will not give any control over the type and volume of traffic, which might introduce some biases, as already discussed in Subsection 1.2.

In order to validate the proposed strategy, the study will provide some details for the implementation of the presented framework. The proposed implementation will be tested on two weeks of data to assess its performance.

The performance will be assessed by first computing the number of true/false positives and true/false negatives. Then, these numbers will be used to compute the actual accuracy and error rate for the implementation of the model over the two weeks of data used for testing. Section 7 will try to provide more details about the results and the methods used to validate the proposed method to detect anomalous network traffic.

1.6 Structure of this Report

Section 2 will try to provide an introduction of the concepts required to understand the methods used in the rest of the work. Section 3 will try to present the environment setup to collect the raw NetFlow data and turn it into the time-series data. Section 4 will try to collect some insight into the time series generated with the techniques described in Section 3 through some data visualization. Section 5 will try to provide some model visualisations used during the development process to make some decisions easier. Section 6 will try to provide some details about the implementation of the prototype of the alerting system. Section 7 will try to give an evaluation of the model presented throughout the whole study. Finally, Section 8 concludes the report.

2 Background

This section gives an introduction to the background concepts needed for the understanding of the methods used in the rest of the work. It gives an introduction to the NetFlow protocol and its uses. Then, there is a discussion of the approaches that other related works use to solve the same problem. Finally, it gives an introduction to the Prophet forecasting model, which is the model used to detect anomalies from the NetFlow data.

2.1 The NetFlow Protocol

One of the first concepts to introduce is the features of the NetFlow protocol. The NetFlow protocol can be used to provide several services, such as accounting, usage-based network billing, network planning, network security analysis, and traffic engineering [3, 4]. The following study [5] analyses and provides a critique of some of the most common usages of better known flow protocols, including the NetFlow protocol. One of the usages analysed in [5] is network anomaly detection, also known as anomaly-based IDS. Network anomaly detection consists of finding patterns in the network traffic that are not expected user's behavior [5]. This work will focus on using the NetFlow for anomaly-based IDS [5].

The two main components of NetFlow are flow caching and the NetFlow reporting collection [4]. The flow caching is responsible for the analysis and the collection of the IP data flows from the networking device and prepares them for export [4]. The NetFlow reporting collection is responsible for the aggregation of the data exported from the networking devices based on customer-defined policies [4]. From the NetFlow packet format defined in the RFC 3954 [6] and the Cisco documentation [7], it is possible to understand the data sent from the NetFlow protocol. The most important data exported consists of the source IP address, the source port, the destination IP address, the destination port, the transport protocol, and the number of bytes.

Then, it is important to understand how to deploy a NetFlow monitoring system. The following study [8] gives an in-depth analysis of all stages needed for the setup of a flow monitoring system. According to [8], the main steps needed to set up a flow monitoring system are the following:

1. *Packet Observation*. The stage in which the packets are captured [8].
2. *Flow Metering & Export*. The stage in which the packets are aggregated into flows

which are placed into the datagram of the deployed flow-export protocol [8].

3. *Data Collection*. The stage in which the flow data from the previous stages is received, stored, and pre-processed [8].
4. *Data Analysis*. The stage in which common analysis functions like correlation or aggregation are applied to flow data [8].

In particular, [8] shows many open source NetFlow collector implementations. A practical implementation of a NetFlow monitoring and visualisation system for an enterprise network is covered in the following study [9].

In addition, another important factor in setting up a flow monitoring system is the sampling of the collected data. As the following study [10] shows, the introduction of sampling on the flow data can drastically degrade the accuracy of the used methods to perform traffic classification. However, the removal of sampling introduces the problem of flow data storage, which on a large-scale infrastructure can be quite challenging because of the number of flows. The following study [11] presents a possible solution to mitigate this problem by using Adaptive NetFlow that dynamically adapts the sampling rate. Regarding the characteristics of the network traffic on large-scale infrastructure, the following study [12] gives an in-depth look of the type of traffic in the data centres, providing interesting statistics on traffic between the networking devices in different network architectures. Also, the following study [13] proposes a methodology to profile the Internet traffic, and the results show many characteristics of the Internet traffic, like volume and duration of the flows based on the protocols.

There are many applications of the NetFlow protocol for monitoring and understanding the network traffic behaviour. For example, the following study [14] presents an algorithm that uses NetFlow data to estimate the RTT and estimate network performance. There are also applications of NetFlow data to detect network attacks. For example, the following study [15] presents a framework that combines the score resulting from the combination of the exponentially weighted moving average (EWMA) based anomaly detection, the algorithm proposed by [16], the clustering-based anomaly detection, the algorithm proposed by [17] and the sketch-based anomaly detection proposed by [15] to detect anomalies on end-user machines.

2.2 Related Work

There are different classes of methods that can be used to detect anomalies in some network traffic. Based on the following study [18], the main classes of methods for network anomaly detection are the following:

- *Classification*. It is based on a profile built from the normal network traffic that creates a knowledge base, and any event different from the baseline is classified as anomalous [18].
- *Statistical*. It is based on a profile built on the normal network traffic, and any event with a large deviation from the profile is considered anomalous[18]. For example, the following article [19] provides an in-depth explanation of some statistical methods to detect anomalies. Also, there is a focus from companies like Netflix on statistical methods for anomaly detection, as the following article [20] shows. The article [20] presents the RAD algorithm, which is a method to detect anomalies in big data based on PCA. Also, the method that will be described in this study belongs to this class of methods.
- *Clustering*. It is based on a clustering algorithm that does not need labelled data to group events that are similar, and events are considered anomalous based on the compactness and the distance of the built clusters [18]. For example, the following study [21] shows different clustering algorithms to perform network traffic classification.
- *Information Theory*. It is based on the use of information theory measures like the entropy [18]. For example, the following study [22] proposes a method to detect different types of cyber-attacks that relies on the Shannon entropy measure.

Some studies use supervised machine learning to accomplish the task of detecting attacks from flow data. For example, in this paper [23], it is shown the use of different supervised machine learning algorithms on NetFlow data to detect the behaviour and the presence of bots on a network. In [23], the implementation using Random Forests is the one that gives the best accuracy. Also, the following study [24] proposes an approach using Random Forests to detect DDoS attacks that manage to achieve an average accuracy of more than 99% and a false-positive rate lower than 0.5%. There are also studies using SVM to accomplish the task of finding anomalies in the network traffic. For example, the following study [25] presents an approach using the C-support vector classification algorithm

that leads to an accuracy of 99.67%, a detection rate of 99.26%, and a classification error of 0.33% and a false alarm rate of 0.08%. Some studies provide machine learning solutions to detect traffic anomalies and provide human-readable labels that allow an interpretation of the results, like the algorithm presented in the following paper [26]. The already mentioned analysis [5] also reports some of the accuracies achieved in several studies with the use of machine learning in detecting network anomalies.

There are also some studies using unsupervised machine learning to accomplish the task of detecting attacks from flow data. For example, the following study [27] focuses on the use of unsupervised machine learning techniques, namely the K-Means algorithm and the Gaussian Mixture Models on NetFlow data, to detect anomalies in the WAN daily network traffic. In [27], the use of the Gaussian Mixture Models leads to better results over the K-Means algorithm.

There are also some studies using deep learning to accomplish the the task of detecting attacks from flow data. For example, the following paper [28] focuses on the use of a Recurrent Neural Network on NetFlow data to achieve an attack detection with 98% accuracy. Also, the following study [29] compares deep neural networks, recurrent neural networks, and long short-term memory in detecting cyber-attacks from NetFlow data. The results from [29] show that the long short-term memory and the recurrent neural network approaches lead to better results compared to the deep neural network approach.

There are also some studies that focus on the use of statics to detect anomalies in the flow data. For example, the already mentioned before study [9] also includes two statistic-based intrusion detection algorithms: one is based on variance similarity, and another is based on Euclidean distance. Also, the following study [30] gives a small analysis of the performance in anomaly detection of some statistical methods such as the Poisson processes or the Kolmogorov-Smirnov test.

The idea of treating the NetFlow data as time series data is also not new. For example, the already presented study [15] includes the use of the exponentially weighted moving average (EWMA) based anomaly detection algorithm proposed by [16] in the presented framework. The following study [31] presents a kernel function model to find anomalies in NetFlow data treated as a time series. However, as stated in [30], the presented method has some problems, such as the fact that the classification does not occur online. Also, the following study [32] treats the NetFlow data as a time series and presents a method

to find anomalies in the network traffic by generating mini-clusters without any previous knowledge of the data. There are also studies using time-series techniques on NetFlow data different from the ones described so far. For example, the following study [33] presents a framework using a multiplicative decomposition model to generate forecasts from NetFlow data. Based on the framework presented in [33], a data point at time t can be predicted using the following equation

$$y(t) = g(t) \times s(t) \times \varepsilon_t$$

where $g(t)$ is the trend component, $s(t)$ is a seasonality component and ε_t is a random error. According with the following study [34], the use of the Holt-Winters forecasting algorithm can also achieve a low level of error measured with MAE, MAPE and RMSE when forecasting NetFlow data. Then, the following study [35] shows how changing the aggregation level of flow data impacts the predictions generated from the Holt-Winters algorithm. The result is that the Holt-Winters with small flows aggregation levels achieve the best performance for short-term predictions, which are suitable for anomaly detection [35]. Also, the following study [36] manages to improve the accuracy of the traditional Holt-Winters algorithm through the introduction of a slight modification. The modification involves the use of time intervals equal to a seasonal cycle instead of the ones immediately before the current ones in the baseline and level equations [36].

2.3 The Prophet Forecasting Model

The Prophet forecasting model [37] uses a decomposable time series model [38]. In the Prophet forecasting model, a prediction at time t is given by the following equation

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t$$

In the above equation, the main components are the following

- $g(t)$ represents the trend component. The purpose of the tend component is modelling the non-periodic changes [37]. The two models to fit the trend component are the saturating growth model and the piecewise linear model [37]. The saturating growth model is used when there is a nonlinear growth, and it is modelled with the

following equation [37]

$$g(t) = \frac{C}{1 + \exp(-k(t-m))}$$

where k is the growth rate, m is an offset parameter and C represents the carrying capacity [37]. The carrying capacity is a term that does not have to be constant, but it can change over time [37]. When there is no saturating growth in the time series data, the piecewise linear model is used and it is given by the following equation [37]

$$g(t) = (k + \mathbf{a}(t)^{\intercal}\delta)t + (m + \mathbf{a}(t)^{\intercal}\gamma)$$

where k is the growth rate and m is an offset parameter[37]. Changepoints are points in the time series when there are changes in the trend growth rate [37]. Then, $\mathbf{a}(t)$ is a binary vector with $\mathbf{a}(t) \in \{0,1\}^S$ with S being the number of changepoints at times s_j in the time series [37]. The elements of vector $\mathbf{a}(t)$ are defined as follows [37]

$$\mathbf{a}_j(t) = \begin{cases} 1, & \text{if } t \geq s_j \\ 0, & \text{otherwise} \end{cases}$$

Then, δ is a vector $\delta \in \mathbb{R}^S$ with δ_j being the rate adjustment happening at time s_j [37]. The last parameter γ is a vector $\gamma \in \mathbb{R}^S$ with $\gamma_j = -s_j\delta_j$ to make the $g(t)$ function continuous.

- $s(t)$ represents the seasonality component. The seasonality component models the periodic changes in the time series data [37]. These periodic changes are modelled through a Fourier series [37, 39]. The main equation to model this component is the following

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right)$$

where the term P represents the period and the term N represents the order which regulates how quickly changes in the time series are fitted [37]. Hence, fitting the seasonality component consists into estimating a vector of size $2N$ [37].

- $h(t)$ represents the holiday component. The holiday component models all the changes that are caused by holidays [37]. Each holiday i is modelled by adding

a parameter κ_i, which introduces a change in the forecast [37]. This parameter κ_i is introduced based on the output of an indicator function which checks if time t is during holiday i.

- ε_t represents the error component. The error component models any change in the time series that is not captured by the model.

In general, as stated in [37], compared to other time-series forecasting methods, the main advantages of the Prophet forecasting model are the following:

- It is more flexible in capturing seasonality with more periods [37].
- Unlike models like ARIMA, There is no requirement to have regularly spaced data with no missing data-points [37].
- The reduced fitting time compared to other forecasting models [37].
- The model has parameters with an easier interpretation compared to other forecasting models like RNN [37].

The main reason why I considered using the Prophet forecasting model for this study is the reduced fitting times compared to other time-series forecasting methods. The need for reduced fitting times is because the proposed solution tries several parameters during the training phase before selecting the definitive model that is used to detect anomalies. The proposed solution tries to treat the traffic generated by each service separately. Hence, for each of the time-series data generated by each service, the proposed solution will perform a grid search to find the best performance parameters. Other methods might have longer training times for each of the parameters tested, so testing various models before picking a definitive one for each service becomes a bit unfeasible. In addition, the NetFlow data is exported every 5 minutes, so the amount of data exported is quite large. Hence, it becomes quite convenient to have reduced fitting times, especially when updating the model with new observations.

The second reason I decided to consider the Prophet forecasting model is that some services generate traffic patterns affected by multiple seasonality. For example, the volume of traffic to some services is affected by holidays, business working hours or business working days. In particular, the Prophet forecasting model allows capturing weekly and daily seasonality, so it is expected to perform well on these kinds of services. However, I will not focus on treating holiday effects and yearly seasonality since the time frame in which I collected the data was not enough to cover one year.

3 Data Collection

This section covers the problem of collecting the raw NetFlow data and turning it into the time-series data that will be used in the following sections of the study. It gives an overview of the relevant steps used to set up the environment from where the NetFlow data has been collected. Then, it gives a description of how the time-series data were obtained from the raw NetFlow data collected.

3.1 Data Collection Environment Setup

The data used in implementing the alerting system has been collected from a data centre. The data centre uses the Top-of-Rack topology [40]. Computes are connected to TOR switches which are interconnected through aggregation switches. The figure 1 shows the high-level architecture of the environment used to collect data. The study focuses on the

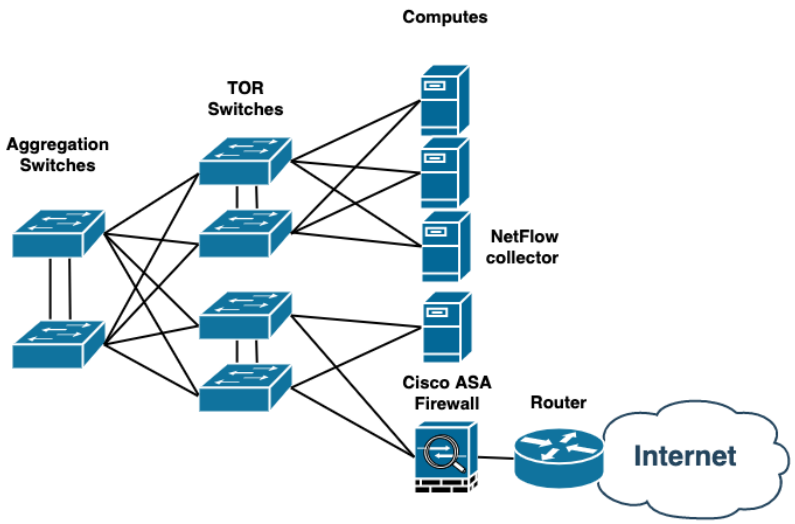

Figure 1. Topology of the flows' collection environment.

flows exported from the Cisco ASA firewall. The first step is configuring the destination where to send the NetFlow data. For example, the configuration lines shown in figure 2 export the NetFlow data to a collector at address 192.168.3.2 on port UDP 9555 which is reachable through interface `netflow`. Then, the class of traffic to export and the actions

```
(config)# flow-export destination netflow 192.168.3.2 9555
```

Figure 2. Cisco ASA firewall configuration lines to export the NetFlow data to a collector at address 192.168.3.2 on port UDP 9555 which is reachable through interface `netflow`.

to take on such traffic need to be defined. In the study, every type of traffic needs to be

analysed. Hence, the configuration shown in figure 3 would export all kinds of traffic to the destination example from the previous configuration lines.

```
(config)# class-map NETFLOW
(config-cmap)# match any

(config)# policy-map global_policy
(config-pmap)# class NETFLOW
(config-pmap-c)# flow-export event-type all
                  destination 192.168.3.2

(config)# service-policy global_policy global
```

Figure 3. Cisco ASA firewall configuration lines to export all kinds of traffic to the destination example from the previous configuration lines.

Once the exporter is configured, the next step is adding some NetFlow collector. For the study, the collector software used is NfSen [41]. After installing NfSen, the most crucial element in the configuration file is defining the sources exporting NetFlow data. Continuing the example above, the configuration lines to configure the sources in NfSen will be the ones shown in figure 4.

```
%sources = (
  'ASA_test'  => {
    'port' => '9555',
    'col' => '#cccccc',
    'type' => 'netflow'
  }
);
```

Figure 4. NfSen configuration lines to configure the sources of netflow data.

3.2 Time Series Data

The next step consists of converting the NetFlow flows from the networking devices into time-series data.

In this study, the data points of the generated time series are separated by regular intervals of five minutes. One of the reasons behind the choice of using five minutes spaced time series is related to the speed by which the misbehaviour of some services is detected. Choosing a long interval does not allow for detecting anomalies in a time that is quick enough to prevent some possible incidents. For example, an interval of one hour or one day have the problem of detecting some anomalous traffic when the effects of such an event might be already evident. On the other hand, choosing a really short interval can

have problems related to the storage and the processing of the collected data. For instance, an interval of one minute has the main issue of generating too many data points that require some extra disk space available for storing, resulting in problems related to the scalability of the solution. In addition, when the new measurements are available, it might not be guaranteed that the processing related to the previous measurements has completely finished. Hence, during the study, five minutes intervals have shown to be a good compromise between the quick detection of anomalies and the storage and processing overhead.

The idea of this study is to treat the traffic generated by each service in a different way to have better visibility of which service is misbehaving and generating some anomalous traffic. For this purpose, in this study, the NetFlow data will be aggregated based on the port and the protocol resulting in a different time series for each of the services running into the environment used for the data collection. For instance, flows generated from an SSH session on port TCP 22, a DNS query and response on port UDP 53, or an ESP session will generate data points belonging to three different time series. This study will use only some fixed services running in the environment used for data collection to test the proposed solution. The decision came from the long training times. However, it is possible removing such filtering to have a time series for the traffic generated by each service running in the monitored environment.

After aggregating the flows, there is the problem of generating time series data points from the raw NetFlow data. The possible options considered were the following.

- Using the number of flows recorded in the five minutes interval as a data point in the final time series.
- Using the amount of data exchanged in the five minutes interval as a data point in the final time series.
- Computing a ratio between the amount of data exchanged and the number of flows recorded in the five minutes interval as a data point in the final time series.

During the study, it was decided to generate the time series data point from the number of flows recorded for simplicity. However, it might be worth exploring the other two approaches in some future studies.

Hence, the figure 5 shows a high-level pseudo-code of the procedure that was used to turn the raw NetFlow data into time-series data.

```
Function aggregateFlows(flows)
  // let m be a new hash map from string to time series
  // let services be a set of services to monitor

  For flow in flows
    key = flow.protocol
    If flow.protocol is UDP or flow.protocol is TCP
      key = flow.destination_port + '/' + flow.protocol
    End If

    If key not in services
      Next
    End If

    time = flow.time
    time.milliseconds = 0
    time.seconds = 0
    time.minutes = time.minutes - (time.minutes mod 5)

    If key not in m
      m[key][time] = 0
    End If

    m[key][time] += 1
  End For

  Return m
End Function
```

Figure 5. High level pseudo-code to build time series based on the service traffic.

4 Data Visualisation

This section tries to provide some visualization of the time series data obtained to have some better insights into the data used to train the anomalies detection model used during the study. For such purpose, the ports TCP 443 and TCP 25 were selected since they are the two main services running into the environment where the data has been collected. Then, also the traffic on port TCP 22 was chosen since it has traffic size and patterns quite unpredictable.

4.1 Time Plots

The time series plots are the most obvious plots to start with to gain some insights into the collected data. For the sake of clarity of the plots, two weeks of data were used to generate the time plots in this section. The figure 6 shows the time plot of the time series built using the number of flows recorded during the time interval for port TCP 443. In

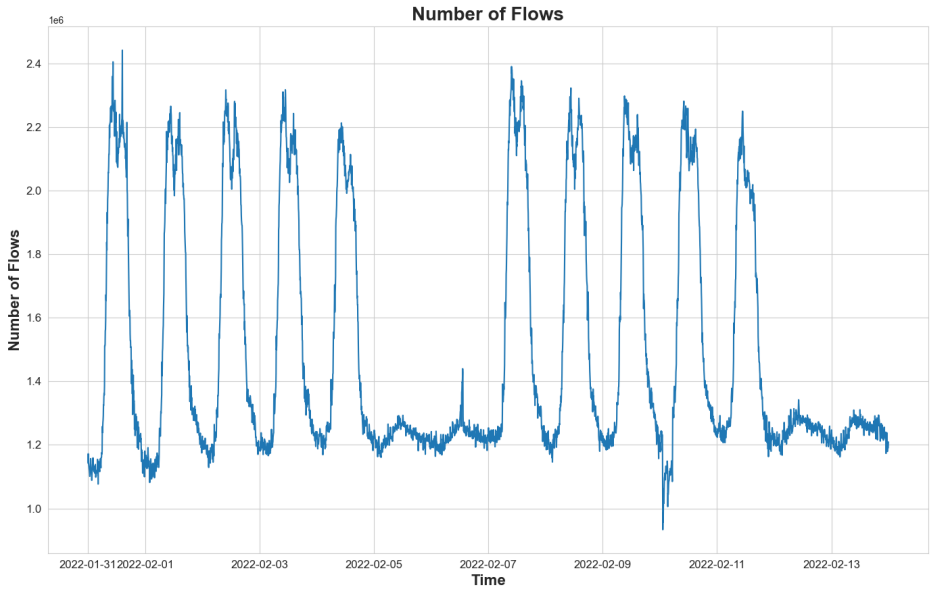

Figure 6. The time plot of the time series built using the number of flows recorded during the time interval for port TCP 443.

this case, the time plot shown in figure 6 immediately reveals some weekly and daily seasonality reflecting the business working hours. In fact, there is a clear drop in the number of flows during the night and during the weekends.

The figure 7 shows the time plot of the time series built using the number of flows

recorded during the time interval for port TCP 25. As in the previous case, the time plot

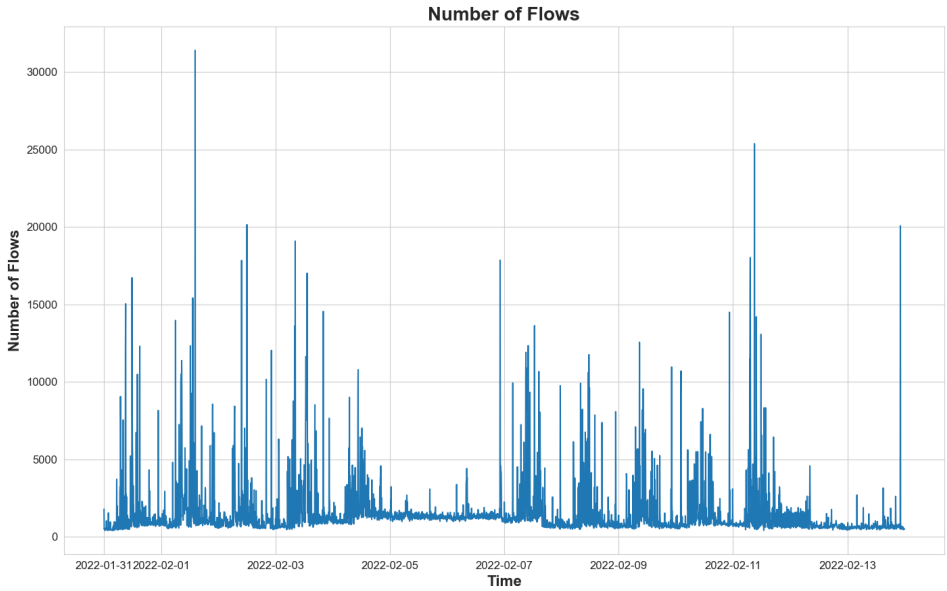

Figure 7. The time plot of the time series built using the number of flows recorded during the time interval for port TCP 25.

is shown in figure 7 above also shows some weekly and daily seasonality which reflects the business working hours. However, compared to the case of port TCP 443, there is a higher level of noise in the data and the drops at the weekends and during the nights are less evident.

The figure 8 shows the time plot of the time series built using the number of flows recorded during the time interval for port TCP 22. In the case of figure 8, the data seems quite noisy and does not show any particular pattern. There is also a spike that could be identified as an anomaly on 04/02/2022 at around 12 pm.

4.2 Seasonal Plots

Some of the time plots have revealed some daily and weekly seasonality, so it is worth investigating them and highlighting them through some seasonality plots. The following sections will analyse the seasonality plots generated with a period of one week first and a period of one day after.

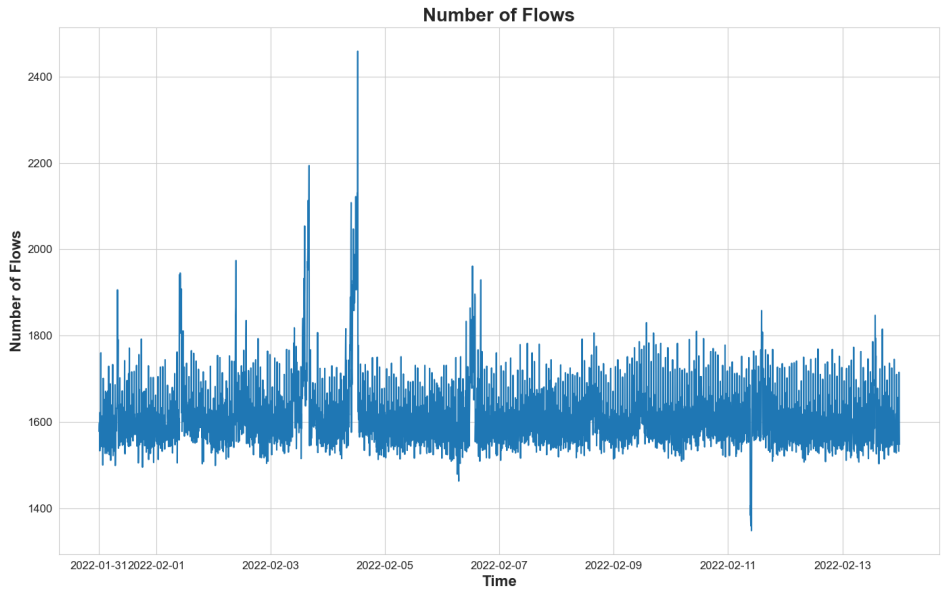

Figure 8. The time plot of the time series built using the number of flows recorded during the time interval for port TCP 22.

4.3 Weekly Seasonal Plots

The seasonal plots with seasonality of seven days will try to make the weekly seasonality more evident. To not have too crowded plots resulting in less readable plots, four weeks of data were used to generate the time plots in this section.

The figure 9 shows the weekly seasonal plot of the time series built using the number of flows recorded during the time interval for port TCP 443. As discussed previously, the plot shown in figure 9 reveals a strong weekly seasonality reflecting the 5-days working week. Hence, it is clear that the number of flows is lower during the weekends compared to the working weekdays.

The figure 10 shows the weekly seasonal plot of the time series built using the number of flows recorded during the time interval for port TCP 25. Also, in this case, the figure 10 shows some weekly seasonality which reflects the 5-days working week. However, in this case, the difference in the number of flows between the weekends and the working weekdays is less pronounced compared to the traffic on port TCP 443.

The figure 11 shows the weekly seasonal plot of the time series built using the number of flows recorded during the time interval for port TCP 22. As noted before, in the case of the data shown in figure 11 the number of flows does not show any particular pattern

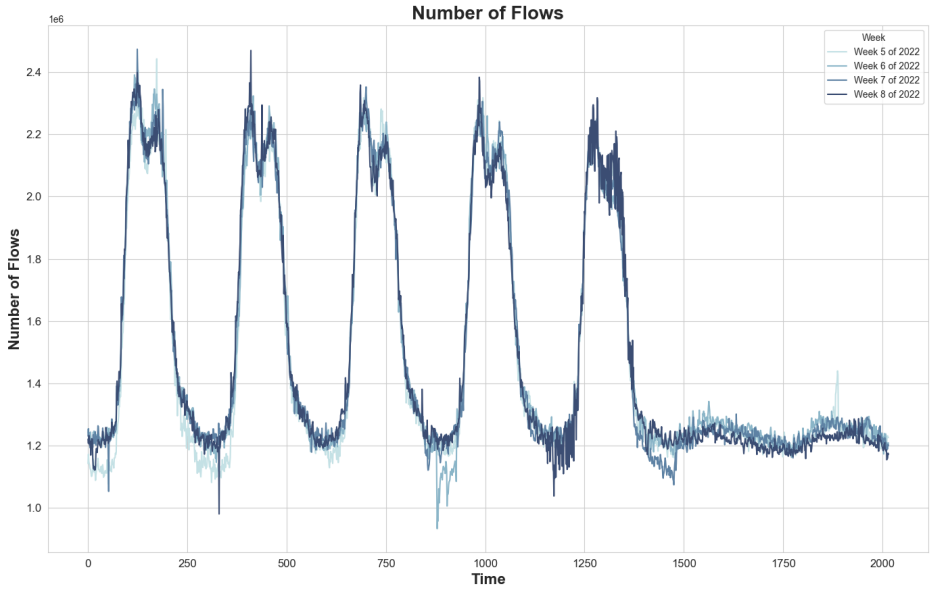

Figure 9. The weekly seasonal plot of the time series built using the number of flows recorded during the time interval for port TCP 443.

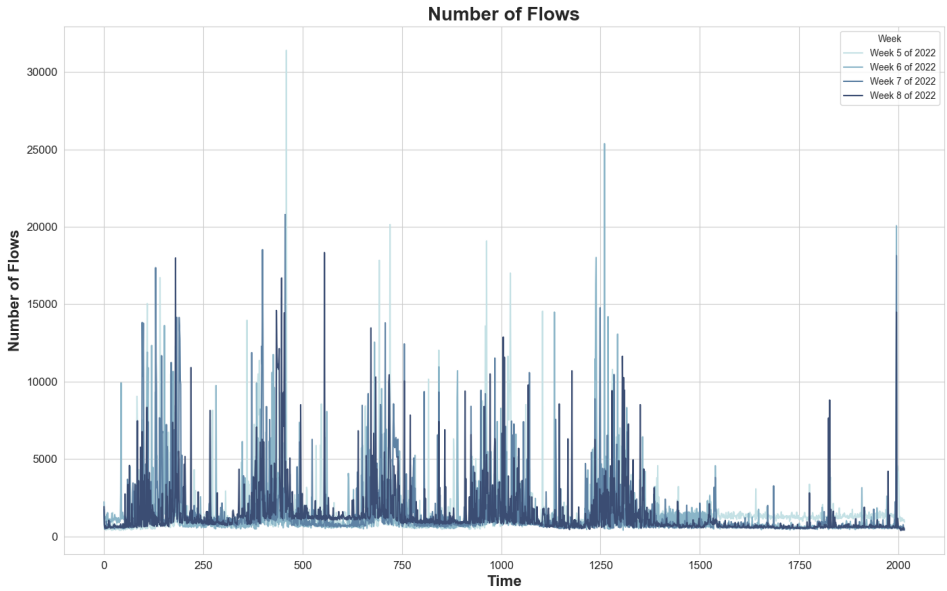

Figure 10. The weekly seasonal plot of the time series built using the number of flows recorded during the time interval for port TCP 25.

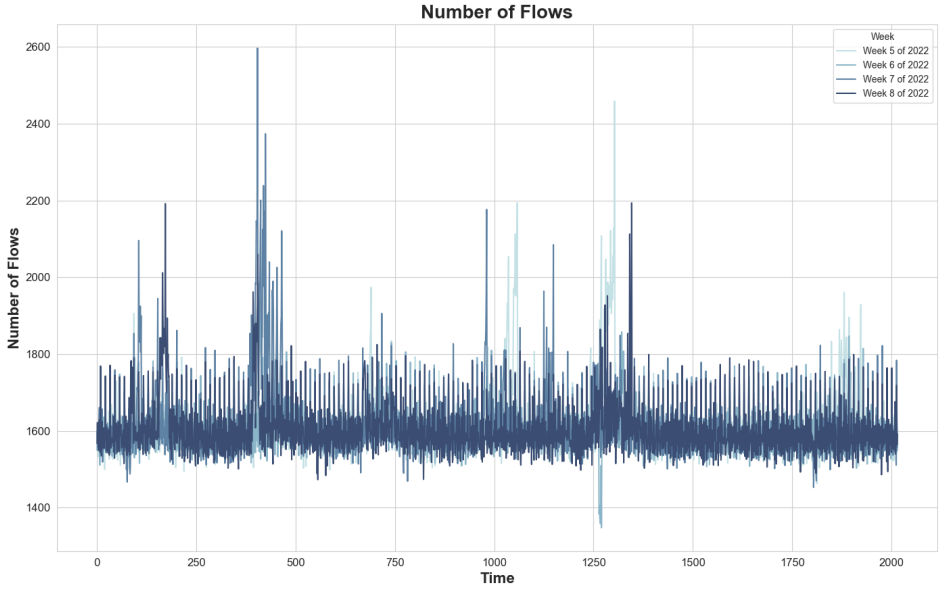

Figure 11. The weekly seasonal plot of the time series built using the number of flows recorded during the time interval for port TCP 22.

and seems constant through time.

4.4 Daily Seasonal Plots

The seasonal plots with seasonality of one day will try to make the daily seasonality more evident. To not have too crowded plots resulting in less readable plots, two weeks of data were used to generate the time plots in this section. Also, when the daily seasonality has a different pattern between weekend and workday, two plots were generated to have more understandable readings which are not affected by the scaling due to a lower number of flows during the weekend.

In the case of the traffic on port TCP 443, the difference between weekend and weekday number of flow was already evident from the previous plots, so two different plots for weekday and weekend were produced. The figure 12 shows the daily seasonal plot for the weekdays of the time series built using the number of flows recorded during the time interval for port TCP 443. The figure 13 shows the daily seasonal plot for the weekend of the time series built using the number of flows recorded during the time interval for port TCP 443. The two plots in figure 12 and figure 13 show the daily seasonality of the data, which respects the working hours during the day. For instance, the number of flows during the day is higher than the ones during the night. Also, from the second plot, it is

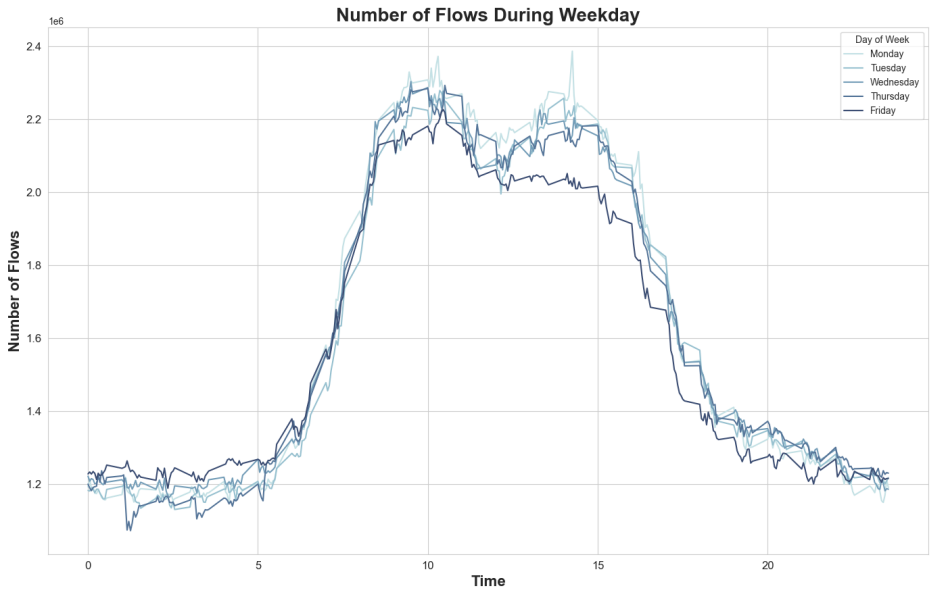

Figure 12. The daily seasonal plot for the weekdays of the time series built using the number of flows recorded during the time interval for port TCP 443.

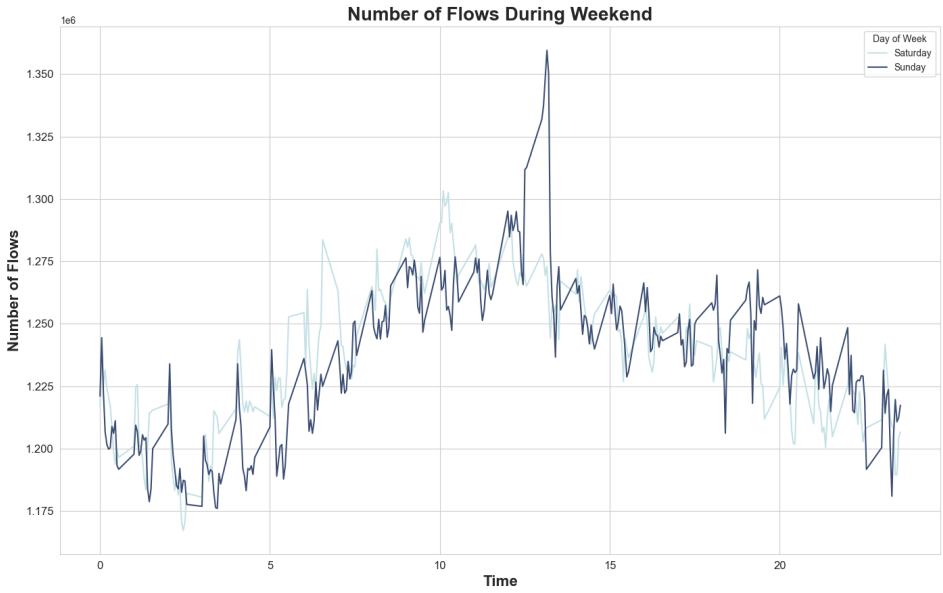

Figure 13. The daily seasonal plot for the weekend of the time series built using the number of flows recorded during the time interval for port TCP 443.

evident that the time series built with the number of flows recorded during the weekend follows a different profile compared to the weekdays since there is a lower number of flows.

The figure 14 shows the daily seasonal plot for the weekdays of the time series built using the number of flows recorded during the time interval for port TCP 25. The figure 15

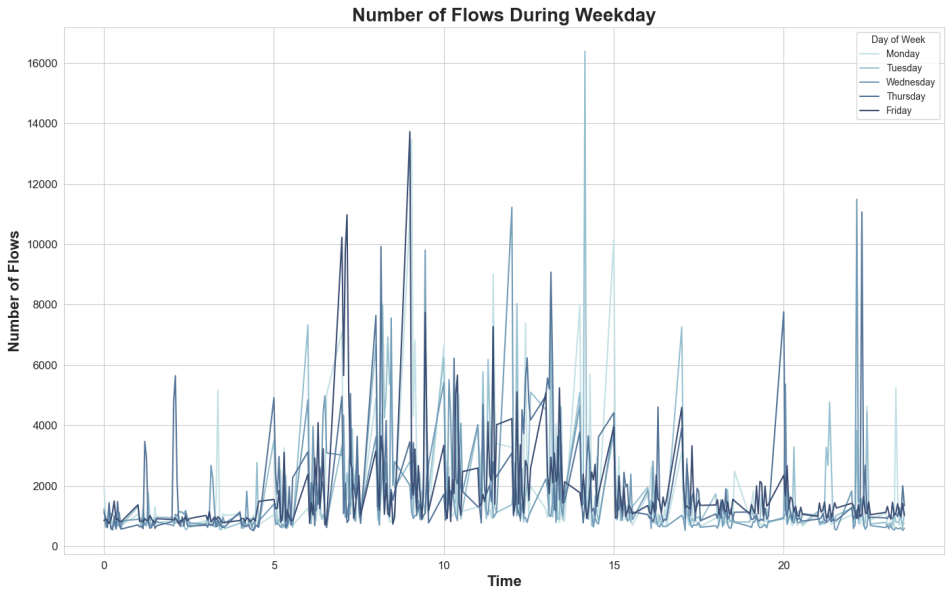

Figure 14. The daily seasonal plot for the weekdays of the time series built using the number of flows recorded during the time interval for port TCP 25.

shows the daily seasonal plot for the weekend of the time series built using the number of flows recorded during the time interval for port TCP 25. In this case, the two plots in figure 14 and figure 15 show a daily seasonality as in the case of the plots generated for port TCP 443. However, such seasonality is less evident since there is a lower number of flows, and there is more noise in the collected data.

The figure 16 shows the daily seasonal plot of the time series built using the number of flows recorded during the time interval for port TCP 22. Also, in this case, the figure 16 shows that the number of flows is not affected much by the working hours during the day. For instance, the number of flows during the day is slightly higher compared to the ones during the night.

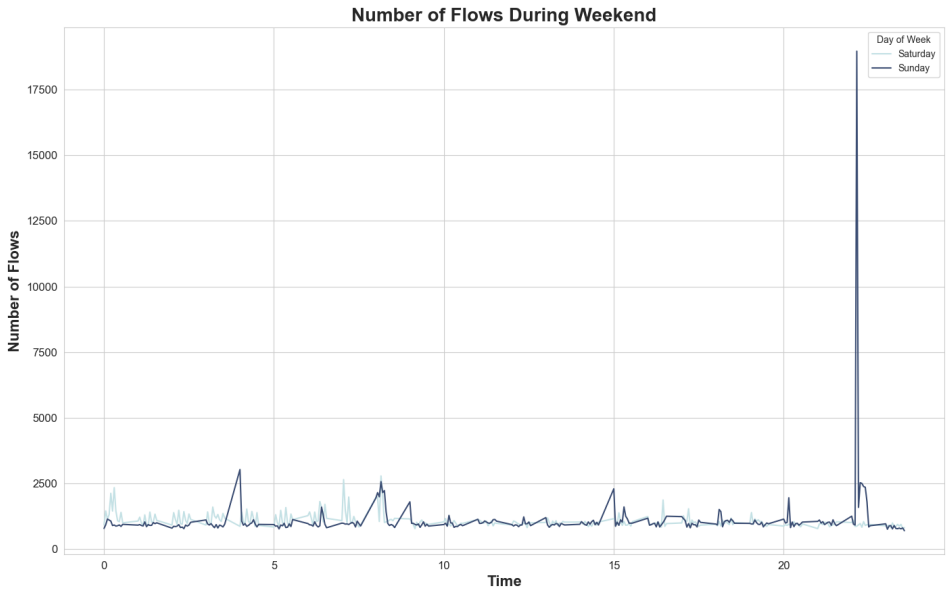

Figure 15. The daily seasonal plot for the weekend of the time series built using the number of flows recorded during the time interval for port TCP 25.

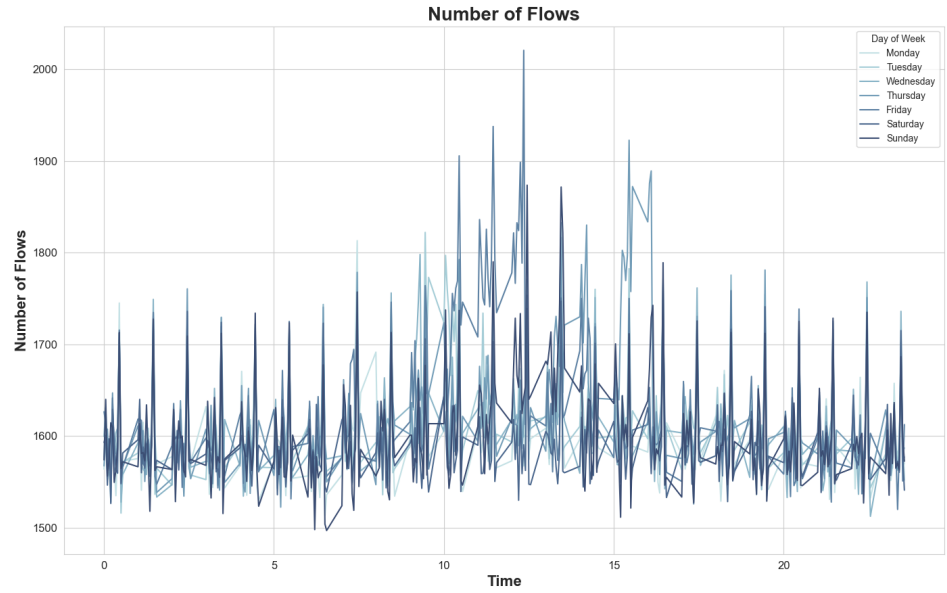

Figure 16. The daily seasonal plot of the time series built using the number of flows recorded during the time interval for port TCP 22.

4.5 ETS Decomposition

It is useful to separate out the components of the time series to build a more detailed understanding of the general behaviour insights of the time series analysed during the study. For such purpose, the ETS decomposition was used. The figure 17 shows the ETS decomposition plots of the time series built using the number of flows recorded during the time interval for port TCP 443. From the ETS decomposition in figure 17, the time series

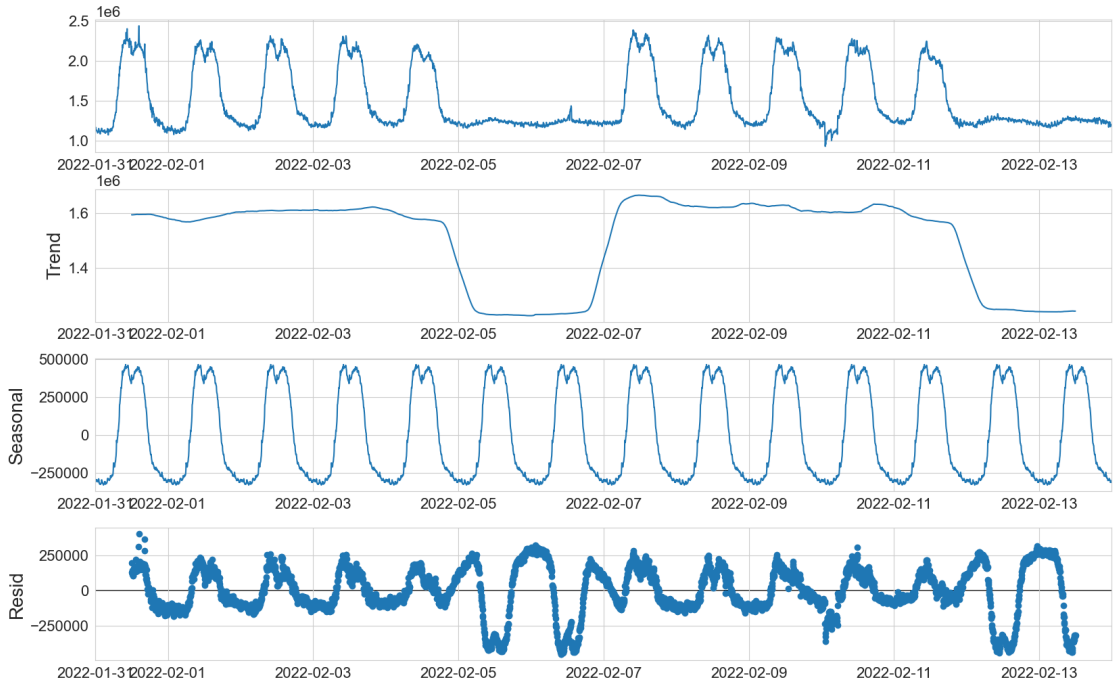

Figure 17. The ETS decomposition plots of the time series built using the number of flows recorded during the time interval for port TCP 443.

has a clear seasonality. Also, it shows a constant trend with some sudden drops during the weekends.

The figure 18 shows the ETS decomposition plots of the time series built using the number of flows recorded during the time interval for port TCP 25. In this case, the ETS decomposition in figure 18 shows a slightly less evident seasonality of the time series. Also, there are no clear trend components in the time series analysed.

The figure 19 shows the ETS decomposition plots of the time series built using the number of flows recorded during the time interval for port TCP 22. As in the previous case, the ETS decomposition in figure 19 shows some seasonality with no clear trend pattern.

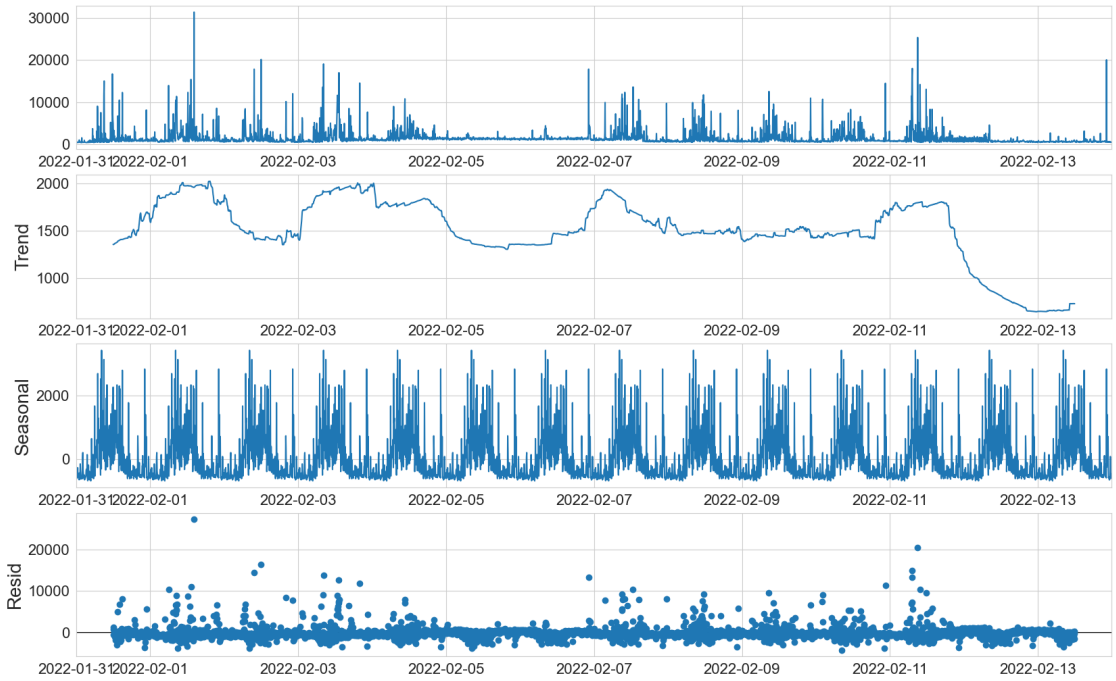

Figure 18. The ETS decomposition plots of the time series built using the number of flows recorded during the time interval for port TCP 25.

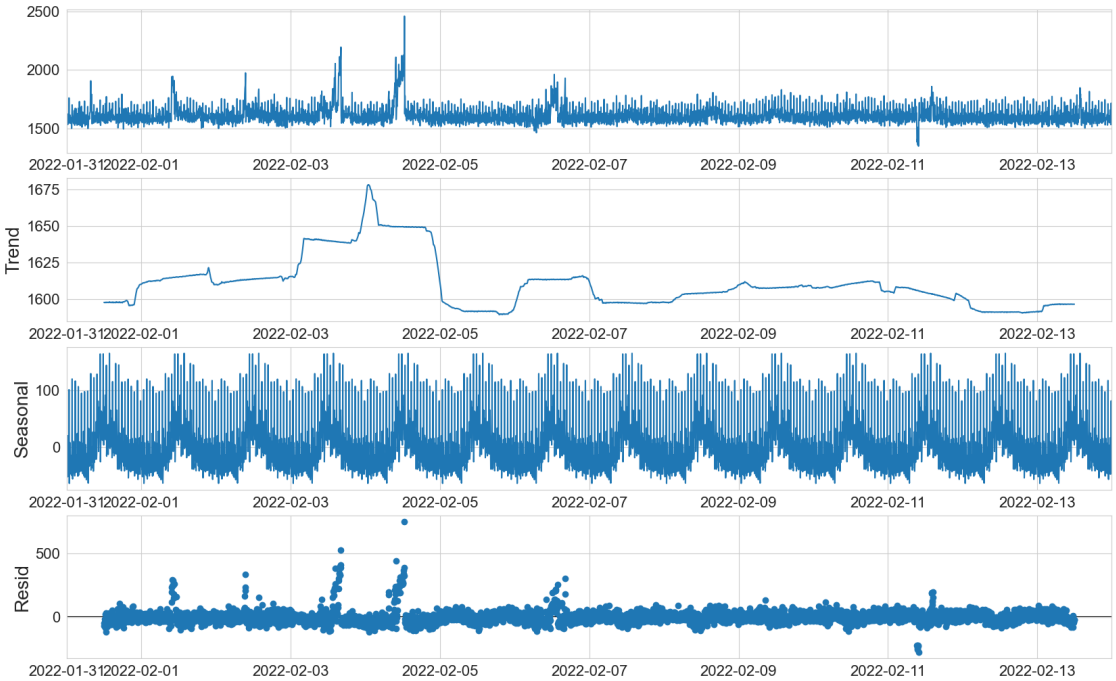

Figure 19. The ETS decomposition plots of the time series built using the number of flows recorded during the time interval for port TCP 22.

4.6 Autocorrelation Plots

Plotting the autocorrelation function for the time series can give more insights into the time series. Namely, it allows identifying how much the older values impact the current value in the time series due to the presence of periodic patterns. The plots were generated with a confidence level of 95% represented by the blue shaded area on the plots. The figure 20 shows the ACF plot of the time series built using the number of flows recorded during the time interval for port TCP 443. The plots in figure 20 show a gradual decline

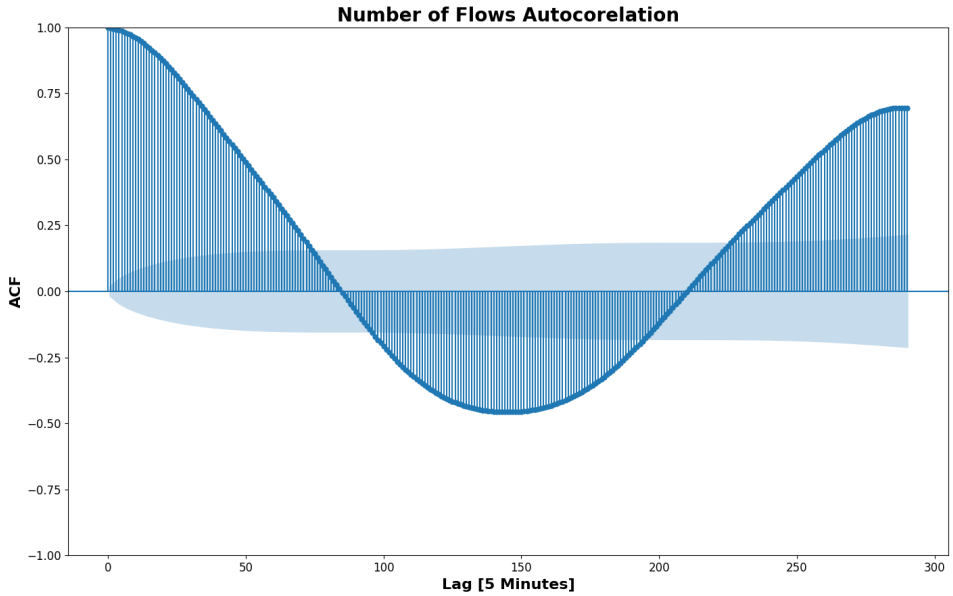

Figure 20. The ACF plot of the time series built using the number of flows recorded during the time interval for port TCP 443.

meaning that newer values are slowly less impacted by previous values. From the plot, it is also possible noticing the seasonality of the data by the fact that autocorrelation increases after a certain amount of shifts which corresponds approximately to one day. In addition, it is clear the presence of a cycle of the length of approximately 145, which corresponds to a half-day cycle. Also, it is worth noticing the presence of a negative value, which implies that newer numbers of flows recorded will be more likely below average if the previous numbers of flows recorded were above average and the other way around.

The figure 21 shows the ACF plot of the time series built using the number of flows recorded during the time interval for port TCP 25. Similarly, for the case of port TCP 443, the plot in figure 21 shows a gradual decline which is an indicator of the fact that newer

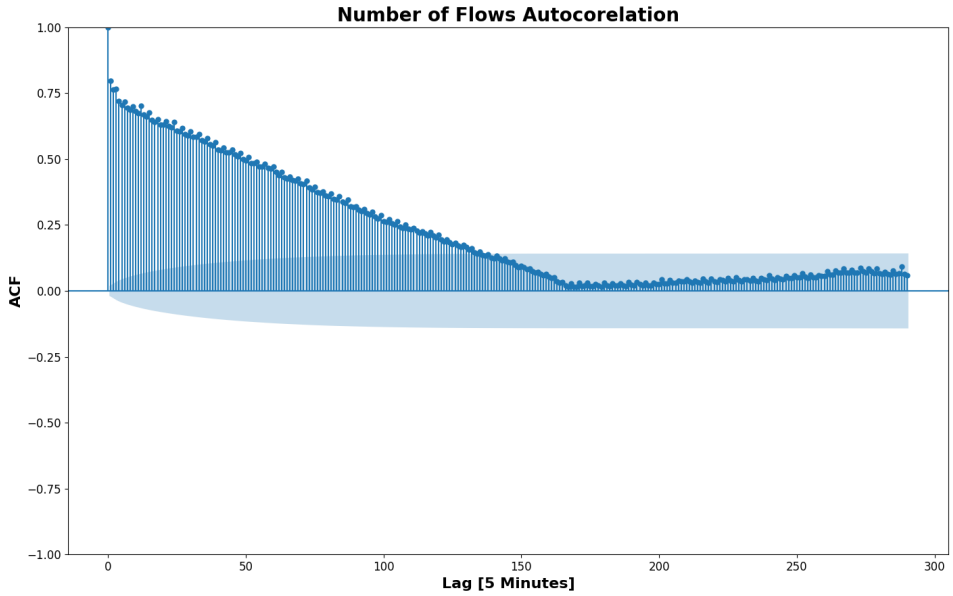

Figure 21. The ACF plot of the time series built using the number of flows recorded during the time interval for port TCP 25.

values are slowly less impacted by previous ones.

The figure 22 shows the ACF plot of the time series built using the number of flows recorded during the time interval for port TCP 22. Differently from the previous plots,

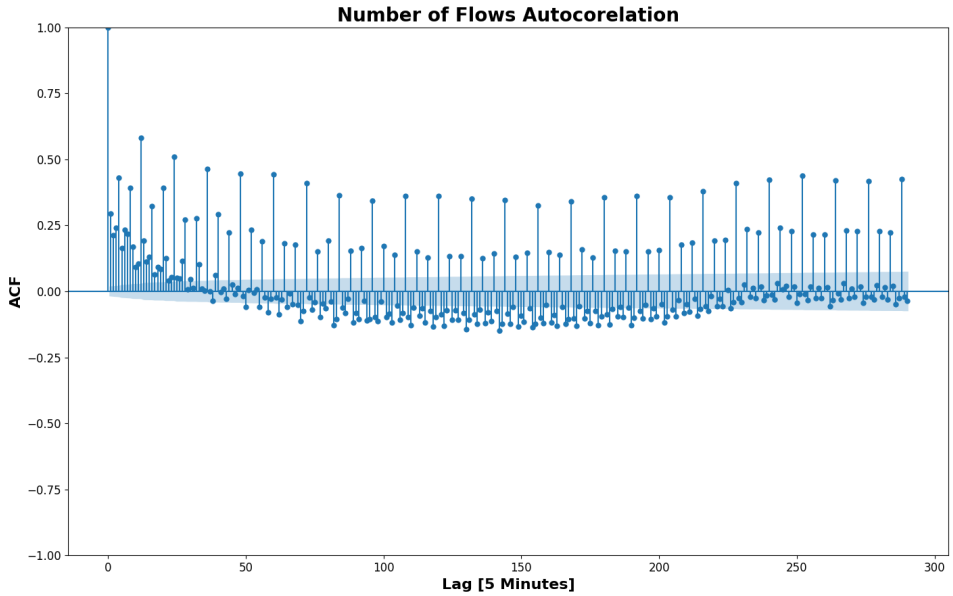

Figure 22. The ACF plot of the time series built using the number of flows recorded during the time interval for port TCP 22.

the plot in figure 22 shows a sharp drop off, meaning that newer values are more quickly less impacted by older values.

5 Software Modelling

This section will try to provide some of the models used during the development process of the alerting system prototype to visualise how the project works and facilitate decision making. It first provides the diagrams created to model the scenario in which a malicious user causes some traffic anomalies by generating some traffic over the network. Then, it provides a model for the functioning of the anomaly detection system.

Malicious User Caused Traffic Anomaly Modelling

The first element modelled was an attack scenario where a malicious user creates a traffic anomaly. The decision of modelling a typical attack scenario is to visualise a use case of the proposed solution and the processes allowing that allow the detection of an attack. For this purpose, the scenario was modelled using the security risk-oriented BPMN modelling language discussed in [42].

The first model will visualise the generic environment where it is possible to deploy the NetFlow based anomaly detection system. The anomaly detection system should be running on the infrastructure with some network traffic. In general, the infrastructure hosts some application that generates traffic over the network by connecting to some infrastructure services it relies on for its functioning. Hence, some application component generates some data which needs to be transferred over the network to some infrastructure service that will be responsible for processing it. From a security perspective, it is essential ensuring the availability of the application data and their processing from the service running on the infrastructure for the correct functioning of the application component. The figure 23 shows the model containing the asset analysis and identification just described.

Then, the second model tries to analyse the security risk scenario where there is an anomaly in the network traffic generated by the action of some malicious user. In this attack scenario, the malicious actor will try to generate some network traffic to overflow the network. The attacker tries to exploit the limited network bandwidth and use all of it to overflow the network. The impact of this action is that there will be that some of the traffic will start being dropped, resulting in reduced network latency and high packet loss. It means that the network will also start dropping some legit traffic between the application components and the infrastructure services they try to connect. The result is

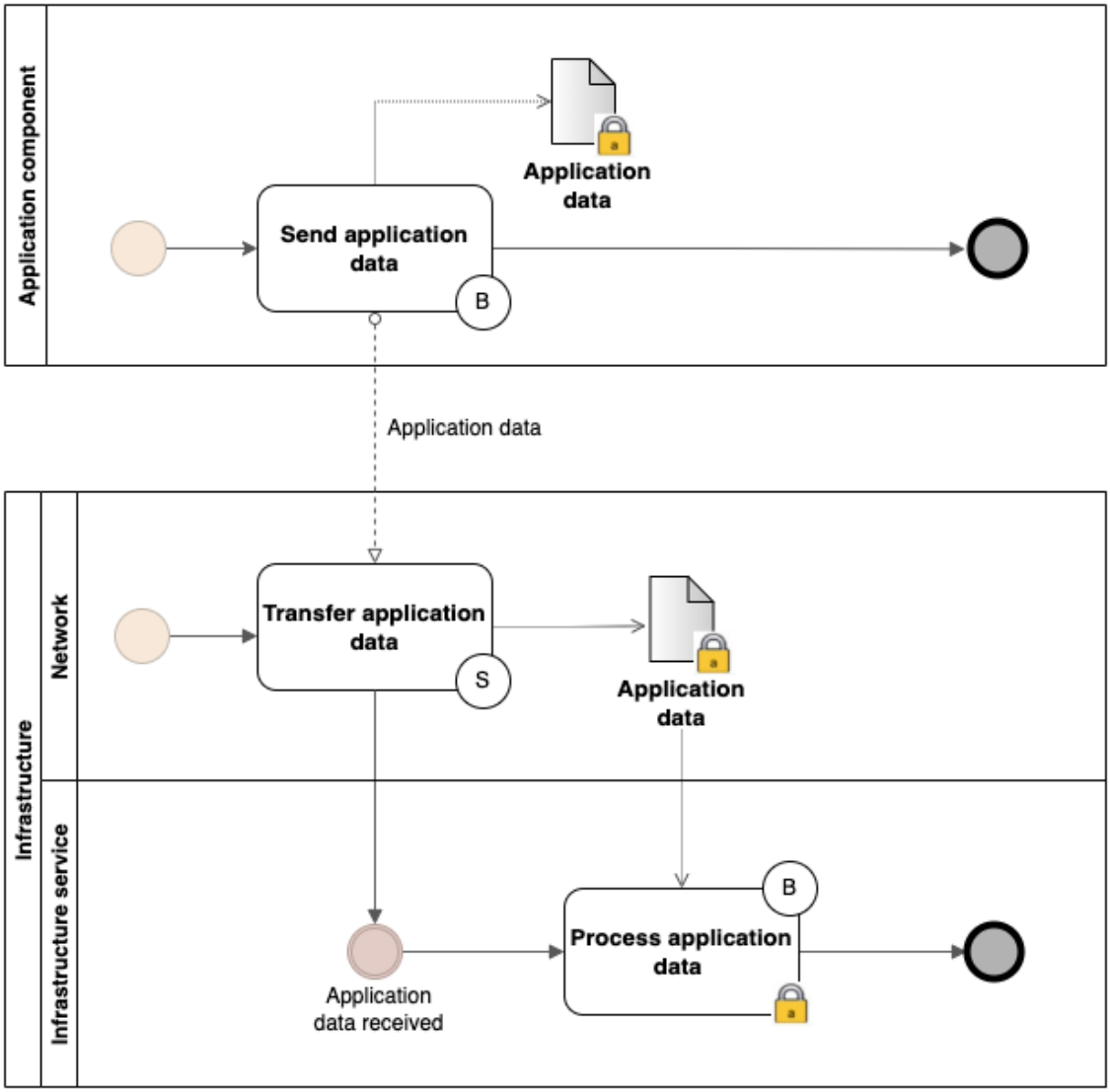

Figure 23. Asset analysis with security risk-oriented BPMN.

that the infrastructure services will not be available to the application components which rely on them for their functioning. The figure 24 shows the model containing an analysis of the security risk scenario just described.

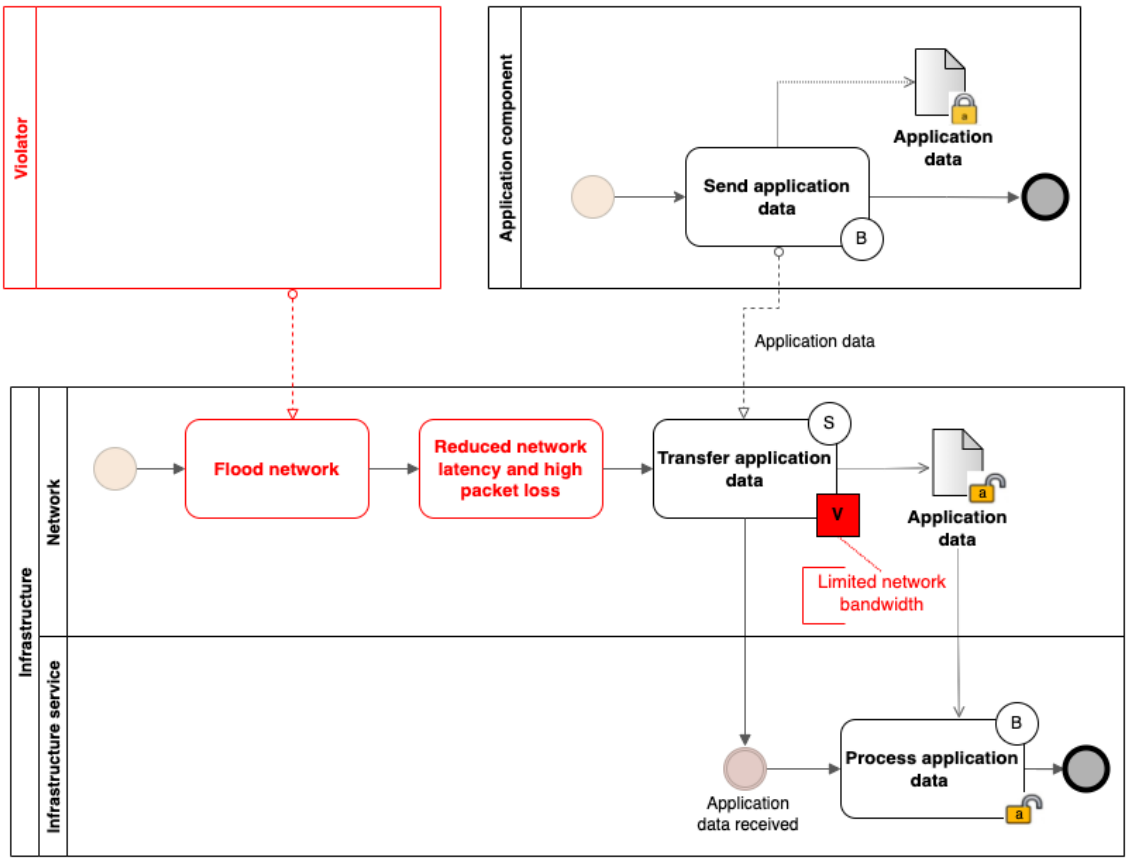

Figure 24. Model of the security risk scenario of an anomaly in the network traffic generated by the action of some malicious user with security risk-aware BPMN.

Finally, the third model tries to analyse how the use of a NetFlow based anomaly detection system allows detecting the actions of the malicious user, providing treatment to the security risk identified in the model in figure 24. Every five minutes, the networking devices will export the NetFlow data, which the NetFlow anomaly detector will receive for processing. The NetFlow anomalies detector compares the received NetFlow data against some baseline. If there is a large deviation from the baseline, it is considered an anomaly and the NetFlow IDS will generate an alert. The figure 25 shows the model containing an analysis of how the NetFlow based anomaly detection system provides a security risk treatment to the security risk scenario identified in the model in figure 24.

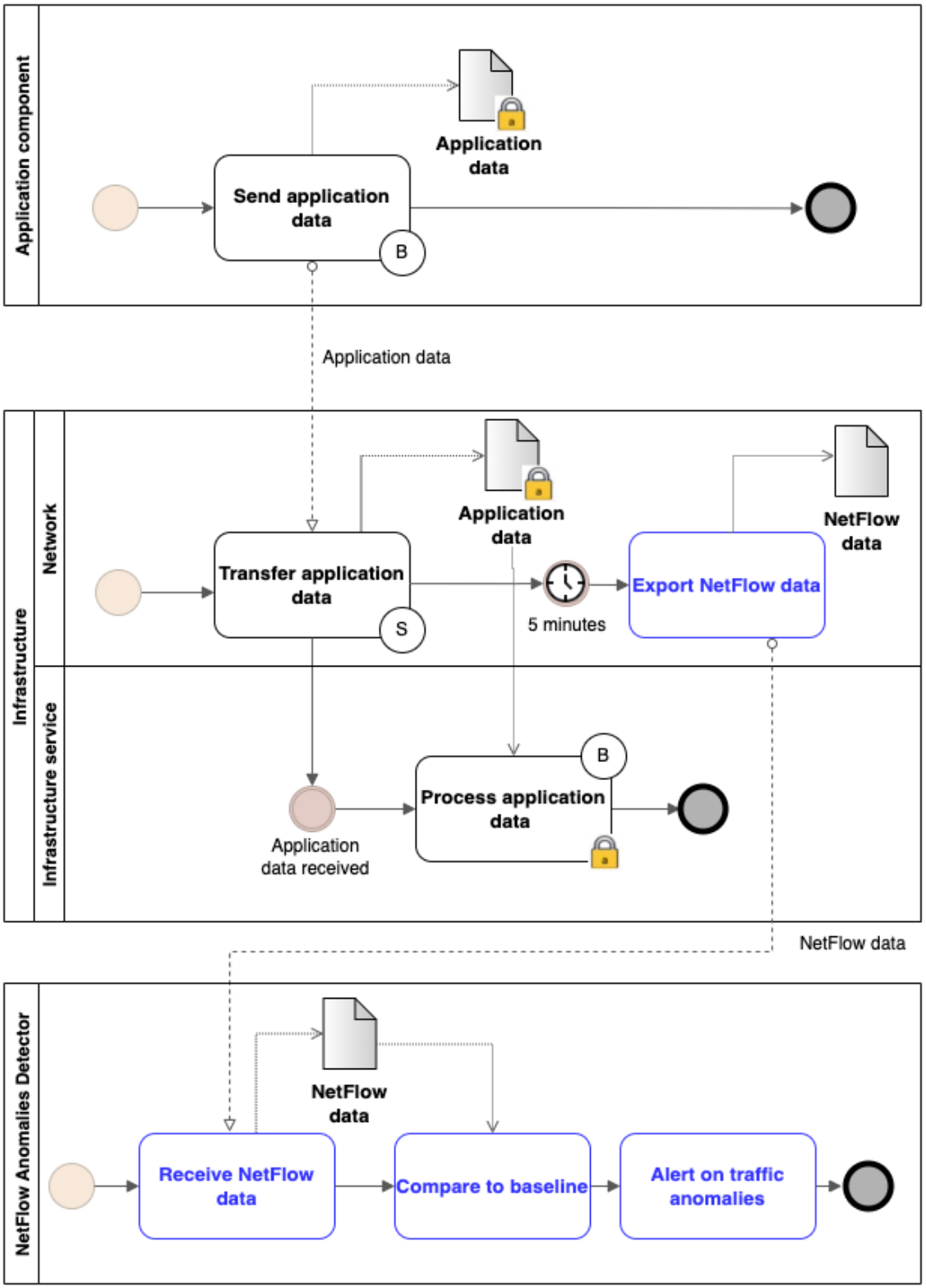

Figure 25. Security risk treatment provided by the NetFlow based anomaly detection system with security risk-oriented BPMN.

NetFlow Anomalies Detector Modelling

The functioning of the anomalies detection system was modelled using the activity diagram modelling language discussed in [43].

The first model produced tries to show the functioning of the tool from a high level. Once the tool is started, it reads the configuration file that contains all the parameters, such as the set of parameters that should be tried during the training phase. Then, parses the arguments given to the program. One of the arguments is used to decide if the tool will run in training mode or not. If it runs in training mode, it will train the model for each of the services configured in the configuration file. If it does not run in training mode, it will read the best parameters from the training mode and the data. Then, it will train the model with the best performance and will use it to detect anomalies. The figure 26 shows the activity diagram for the high-level functioning of the NetFlow anomalies detector.

Then, the second model produced tries to show how the anomaly detection cycle for one service works. The model makes a forecast and computes the uncertainty interval on that. The way the uncertainty interval is computed will be discussed in Section 6. When the NetFlow data is available from the NetFlow collector, the NetFlow anomalies detector parses the NetFlow data and compares it to the prediction by checking if it is outside of the uncertainty interval. If the data point is outside of the confidence interval, the model will generate an alert; otherwise, nothing will happen. Finally, the model will be updated with the new data point received. The figure 27 shows how the anomaly detection cycle for one service works. Lastly, the last model produced tries to show how the training procedure for one service works.

The model splits the available data into train, validation and test data. Then, it loads the training parameters and starts iterating over them. If there are still training parameters, it fetches the next parameters to test and trains the model using the training data and the fetched parameters. Finally, it assesses the performance of the model on the validation data. If there are no training parameters left, it uses the parameters which gave the best performance on the validation data to train the model on the training and validation data. Then, it assesses the model performance on the test data and saves the best parameters to make them available for when the tool runs in anomaly detection mode. The model performance on the test data is used to compute the uncertainty interval, but Section 6 will provide mode details about the formula used.

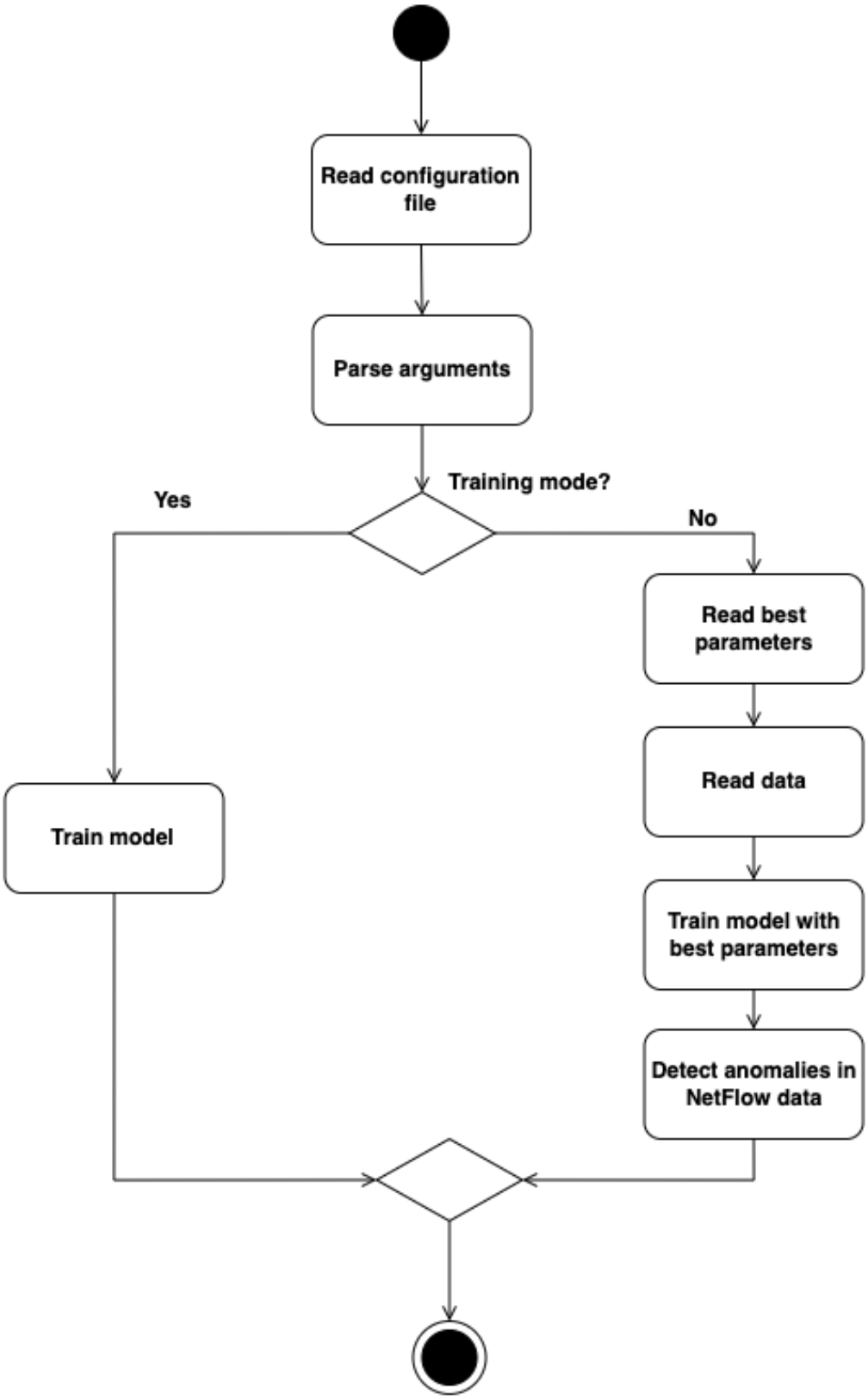

Figure 26. The activity diagram for the high-level functioning of the NetFlow anomalies detector.

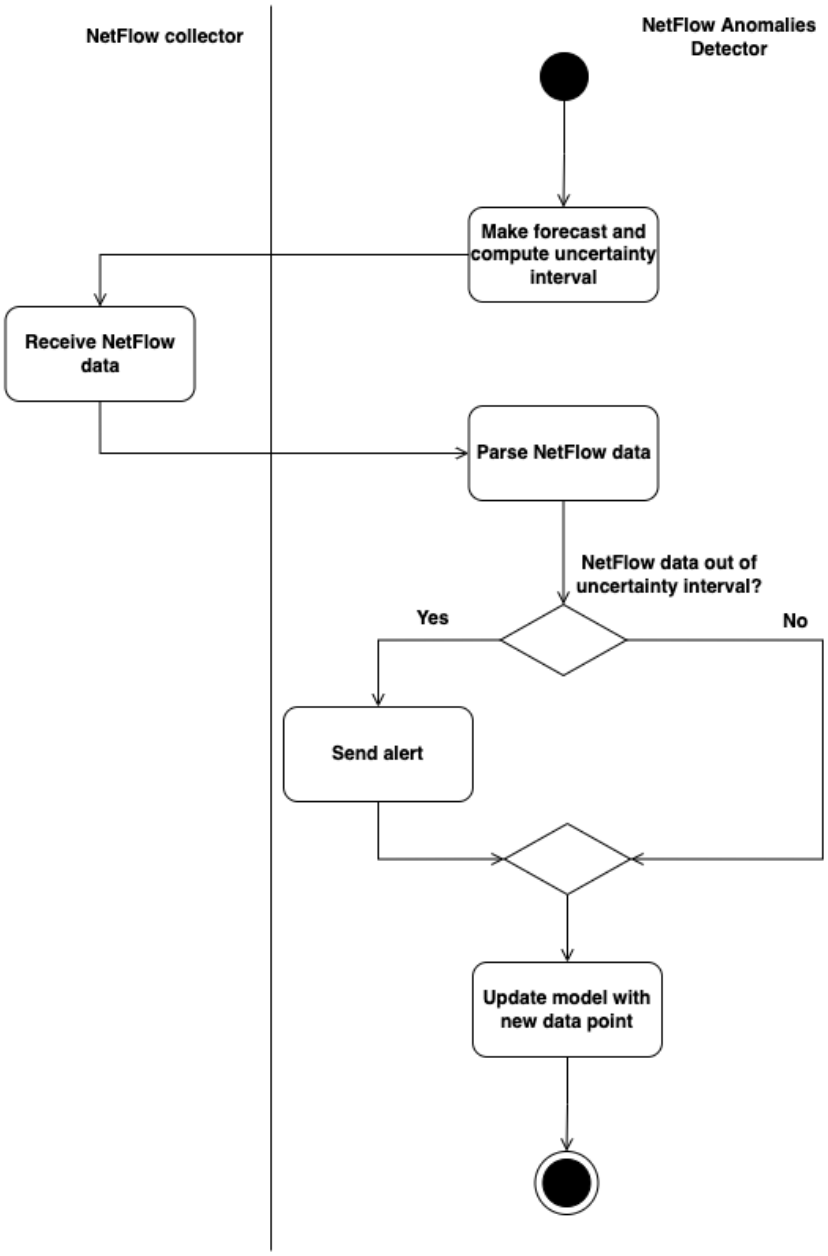

Figure 27. The activity diagram showing how the anomaly detection cycle for one service works.

The figure 28 shows how the training procedure for one service works.

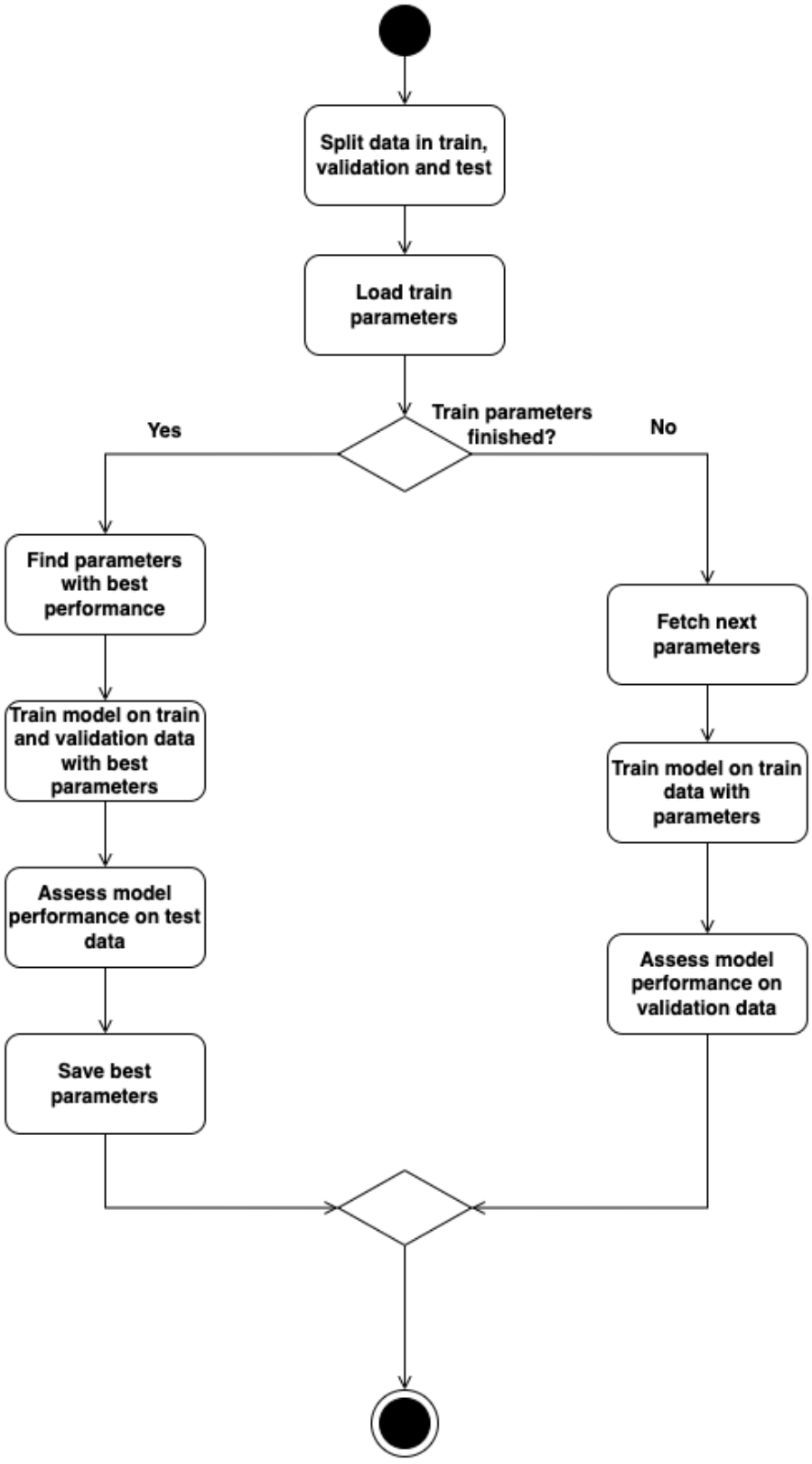

Figure 28. The activity diagram showing how the training procedure for one service works.

6 Implementation

This section tries to provide some of the details about the implementation of the NetFlow based anomalies detection system. I will explain which components of the Prophet forecasting model was used. Then, it will give an overview of the condition used to decide if an observation is anomalous or not. Finally, it will try to show how model parameters were selected and which parameters were tuned.

6.1 Model Specifics

The specifics of the model to find anomalies in the NetFlow data were derived from the insights gained from the data visualisation presented in Section 4. The model implementation is in Python, so the Python API of the Facebook Prophet library [44] was used.

The first element taken into consideration is the seasonality of the data. In general, the data was showing a weekly seasonality which was simply modelled with a Fourier series with period 7 and order 3, as described in [37]. The data was also generally showing a daily seasonality. However, the daily seasonality was a bit more challenging to model since, in most cases, the weekdays have a different profile from the weekends. To solve this problem, the Prophet model offers a conditional seasonality which is a seasonality that affects only some of the dates in the training set and the predictions.

Once the model has generated a forecast, the next important element to consider is the criteria used to classify an observation as an anomaly or not. To do this task, the model checks if the observation is outside or inside some uncertainty interval. Given a prediction $\hat{y}$, the upper bound of such interval is defined as

$$\hat{y}_{up} = \hat{y} + k \cdot \beta$$

and the lower bound of such interval is defined as

$$\hat{y}_{low} = \hat{y} - k \cdot \beta$$

In the above equations, k is a configurable constant parameter indicating how noisy should the alerting be. In the concrete implementation of the model during the study, this parameter has been set to $k = 3.5$. The other parameter β is a parameter indicating how well the

model is able to make forecasts. In the concrete implementation of the model during the study, this parameter has been set to the RMSE computed on the test data after the training procedure. Hence, given an observation it is simply needed to verify the following predicate

$$p(y) : y > \hat{y}_{up} \vee y < \hat{y}_{low}$$

The other crucial element to discuss is the training of the model. The training phase aims to find the best parameters for each of the time series from each monitored service. For this purpose, the implementation uses a grid search where the performance of the model is evaluated by using the RMSE between the prediction and the observed data. Figure 29 shows a high level pseudo-code of the procedure used to grid search.

The main parameters tuned during the training are the following:

- `changepoint_prior_scale`: it is a parameter that regulates how much the trend component changes at the changepoints [45]. This parameter was selected since it was revealed to be one of the most impactful in reducing the error in the predictions.
- `seasonality_prior_scale`: it is a parameter which regulates how much the seasonality component changes [45]. This parameter was tuned for both weekly and daily seasonality.

```
Function train(data, params)
  // Split data into train, validation, test

  best_e = +Inf
  best_param = nil

  For param in params
    // let model be the model fitted on train
    // with parameter param

    // predicting validation.length data points in
    // future with the fitted model
    predictions = model.predict(validation.length)

    // computing the RMSE between the predictions
    // and the validation data
    e = rmse(predictions, validation)

    If e < best_e
      best_e = e
      best_param = param
    End
  End


  // let model be the model fitted on train and
  // validation with parameters best_param


  // predicting test.length data points in
  // future with the fitted model
  predictions = model.predict(test.length)

  // computing the RMSE between the predictions
  // and the test data and save it in
  // best_param['beta'] which is used for
  // computing uncertainty interval
  best_param['beta'] = rmse(predictions, test)

  Return best_param
End Function
```

Figure 29. High level pseudo-code of the grid search used to find the parameters for the model.

7 Evaluation

This section tries to provide some evaluation of the resulting solution. It will try to show some of the time plots used to evaluate the model. Finally, it will provide the error rate and accuracy classification metrics used to assess the model.

Predicted Time Series Plots

In order to validate the performance of the model implementation, the model was used to predict two weeks of data on some of the services running in the data centre where the data collection happened. Since the time-series data points are evenly spaced at 5 minutes, the total number of predicted points is 4032. The services on which the model was tested run on ports TCP 443, TCP 25, TCP 22, TCP 3306, TCP 12280 and TCP 8000. Only some services were selected because of the long training times due to the parameter grid search discussed in Section 6 which is one of the limitations of the initial model implementation. The model limitations will be discussed more in Section 8. During the two weeks used to evaluate the solution, there were no recorded attacks, so the data will not contain any anomalies generated by an attack. However, an attack is not the only cause for a network traffic anomaly, but also events such as a misconfiguration of the service or some fault deployment of the application using the given service might generate anomalies in the network traffic. Hence, the cause of some of the anomalies identified can be traced back to these kinds of events, which were present in the two weeks used for testing.

After running the model, the time series plots of the predicted data against the original, the uncertainty interval, and the anomalies detected were generated to find true/false positives and true/false negatives. The anomalies identified by the model were classified as true or false positive based on the incident records and the identified cause of the incident from the company owning the environment used for testing. According to the incident causes reported in the records, there were no attack traces. Still, there were some faulty deployments or configuration changes around the time when the model alerted to some anomalies.

Figure 30 shows the time plot of the predicted data against the original data and the uncertainty interval and the anomalies detected for port TCP 443. For the case of port TCP 443, the model was able to identify 1 anomaly. However, by looking at the incident records and the plot in figure 30, it is clear that the identified anomaly is a false positive

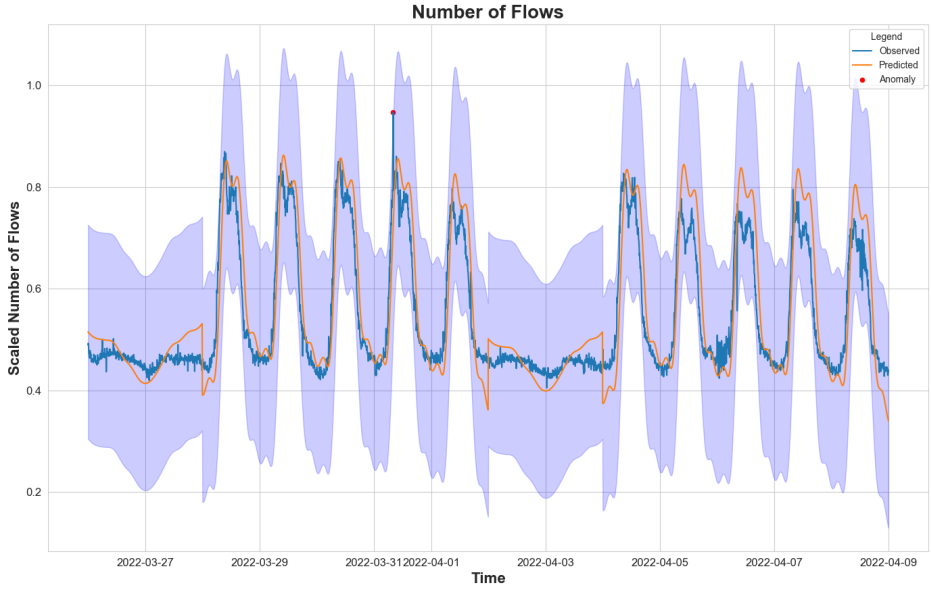

Figure 30. Time plot of the predicted data against the original and the uncertainty interval and the anomalies detected for port TCP 443.

since there were no events in the incident records affecting the service listening on port TCP 443.

Figure 31 shows the time plot of the predicted data against the original and the uncertainty interval and the anomalies detected for port TCP 25. For the case of port TCP

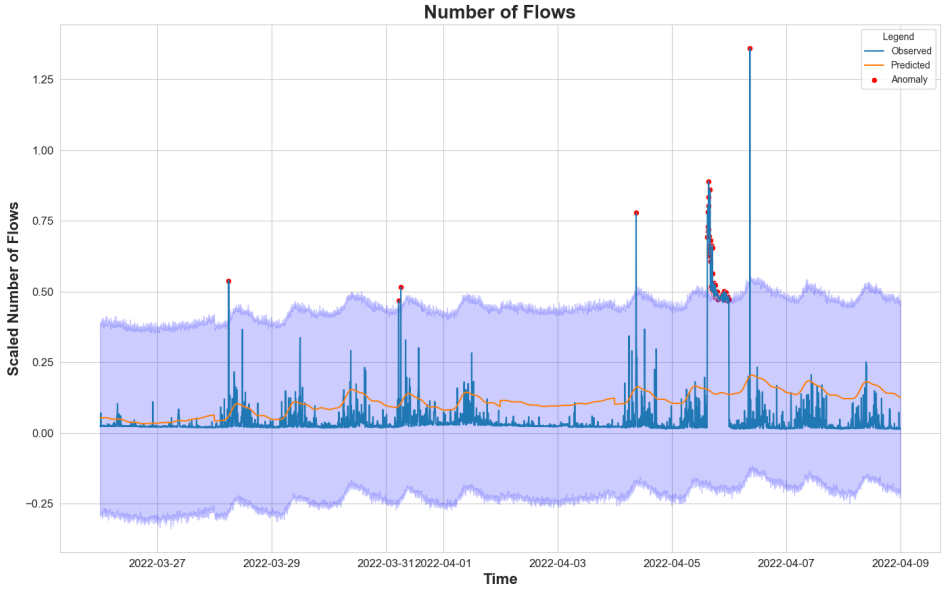

Figure 31. Time plot of the predicted data against the original and the uncertainty interval and the anomalies detected for port TCP 25.

25, the model was able to identify 17 anomalies. By looking at the incident records and the plot in figure 31, it is possible identifying that only some of the identified anomalies are false positives. For instance, the anomalies identified on 28/03/2022, 31/03/2022 and 04/04/2022 can be considered false positives since the incident records do not report any event that affected the service running on port TCP 25. Hence, by looking at figure 31, it is clear that the total number of false positives is around 4. However, on 05/04/2022, a faulty deployment resulted in the application abusing the service running on port TCP 25. Therefore, the identified anomalies are legit, and the number of true positives is around 13.

Figure 32 shows the time plot of the predicted data against the original and the uncertainty interval and the anomalies detected for port TCP 22. For the case of port TCP 22,

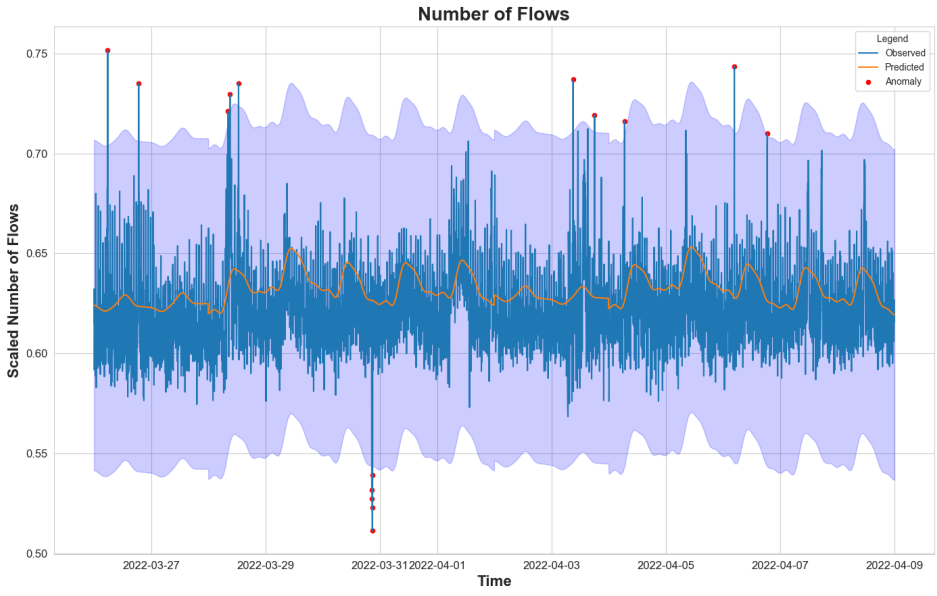

Figure 32. Time plot of the predicted data against the original and the uncertainty interval and the anomalies detected for port TCP 22.

the model was able to identify 15 anomalies. However, by looking at the incident records and the plot in figure 32, it is clear that the identified anomaly is a false positive since there were no events in the incident records affecting the service listening on port TCP 22.

Figure 33 shows the time plot of the predicted data against the original and the uncertainty interval and the anomalies detected for port TCP 3306. For the case of port TCP 3306, the model did not identify any anomaly. From the incident records, it is clear that

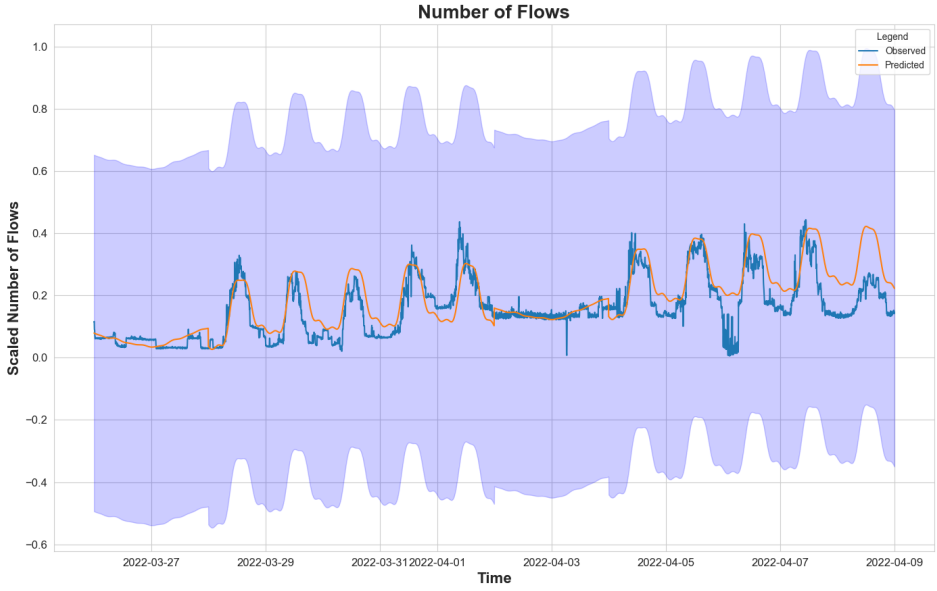

Figure 33. Time plot of the predicted data against the original and the uncertainty interval and the anomalies detected for port TCP 3306.

there were no events that affected the service listening on port TCP 3306. By looking at the plot in figure 33, the model performed pretty well since it did not produce any false positives or false negatives.

Figure 34 shows the time plot of the predicted data against the original and the uncertainty interval and the anomalies detected for port TCP 12280. For the case of port TCP 12280, the model was able to identify 229 anomalies. However, by looking at the incident records and the graphic in figure 34, it is clear that the identified anomaly is a false positive since there were no events in the incident records affecting the service listening on port TCP 12280.

Figure 35 shows the time plot of the predicted data against the original and the uncertainty interval and the anomalies detected for port TCP 8000. For the case of port TCP 8000, the model was able to identify 12 anomalies. By looking at the plot in figure 35, it is possible identifying that only some of the identified anomalies are false positives. For instance, the anomalies identified between 28/03/2022 and 06/04/2022 can be considered false positives since the incident records did not report any event that affected the service running on port TCP 25. Hence, by looking at figure 35, it is clear that the total number of false positives is around 10. However, on 08/04/2022 afternoon, some engineers tested some configuration changes for the service running on port TCP 8000 to mitigate an inci-

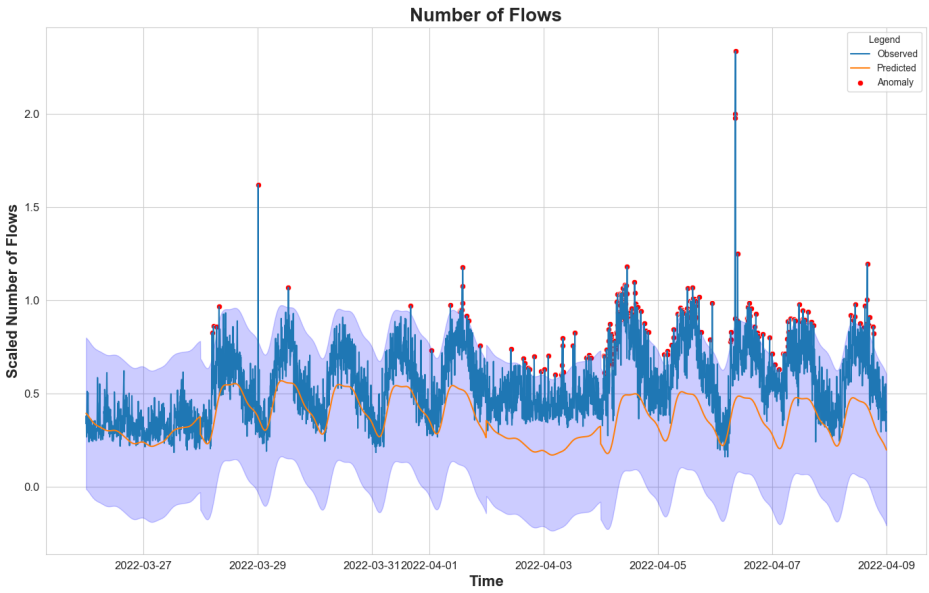

Figure 34. Time plot of the predicted data against the original and the uncertainty interval and the anomalies detected for port TCP 12280.

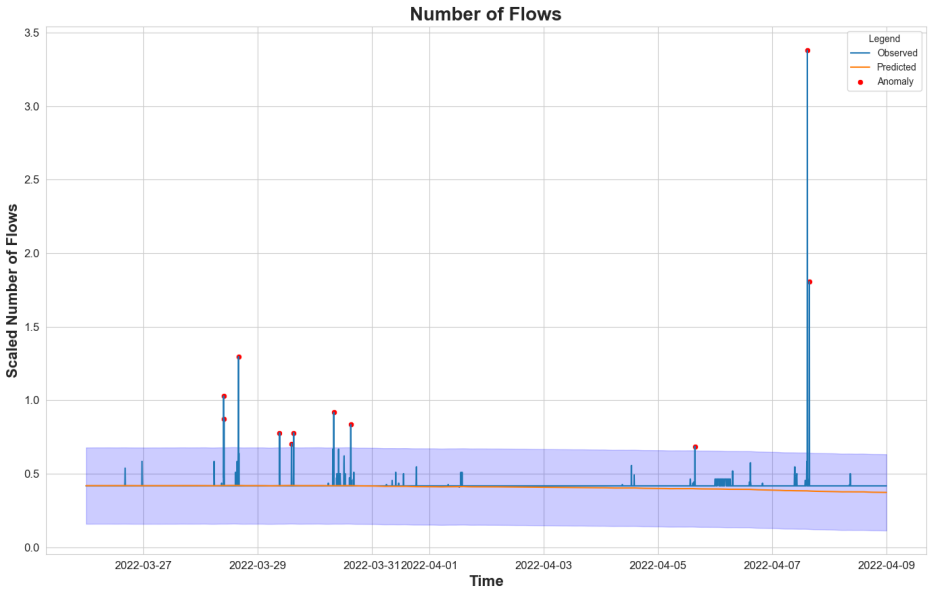

Figure 35. Time plot of the predicted data against the original and the uncertainty interval and the anomalies detected for port TCP 8000.

Service	Anomalies found	False positives	True positives	False negatives	True negatives
TCP 443	1	1	0	0	4031
TCP 25	17	4	13	0	4015
TCP 22	15	15	0	0	4017
TCP 3306	0	0	0	0	4032
TCP 12280	229	229	0	0	3803
TCP 8000	12	10	2	0	4020

Table 1. Summary of the number of anomalies found, true/false positives and true/false negatives.

dent. Therefore, the identified anomalies can be considered legit, and the number of true positives is around 2.

7.1 Classification Metrics

The problem of finding anomalous NetFlow time series can be reduced to a binary classification problem where each data point belongs to either an anomalous or a not anomalous class. Hence, it makes sense to use classification metrics to evaluate the approach proposed in this study, like the accuracy and the error rate. The accuracy of a model is given by the following formula

$$\text{Accuracy} = \frac{TP + TN}{N}$$

where TP is the number of true positives, TN is the number of true negatives and N is the number of data points. The error rate of the model is given by the following formula

$$\text{Error rate} = \frac{FP + FN}{N}$$

where FP is the number of false positives, FN is the number of false negatives and N is the number of data points.

Hence, before computing such metrics, it is helpful to find the number of true/false positives and true/false negatives. For this purpose, table 1 presents a summary of the number anomalies found, true/false positives and true/false negatives. The values reported in table 1 have been computed based on the analysis from Subsection 7. It is worth noticing that table 1 does not report any false negatives. It is possible to identify the main reason for the lack of false negatives in the absence of no attack traces during the weeks used to test the solution. In addition, the incident records do not show any other event

Service	Accuracy	Error rate
TCP 443	0.99	0.0002
TCP 25	0.99	0.0007
TCP 22	0.99	0.0037
TCP 3306	1	0
TCP 12280	0.94	0.0568
TCP 8000	0.99	0.0024
Total	0.99	0.0107

Table 2. Accuracy and error rate for the model.

that affected the services used for testing in that environment other than the ones already discussed before. From the data in table 1 it becomes easy to compute the accuracy and error rate for the model described during the study.

Hence, table 2 reports the accuracy and error rate for the model. As we can see, table 2 shows that the model has quite some good performance on the data. However, there are some limitations that will be discussed more in Section 8.

8 Conclusion

During this study, we have been able to take on the traffic anomalies detection problem using the NetFlow data. We have developed a novel method to detect traffic anomalies that rely on the use of the Prophet forecasting model. During a training phase, the model tries to first find which are the best parameters to fit the time series data. The time-series data is generated from the number of flows collected during a five minutes interval. After fitting the data with the parameters found in the training mode, it starts generating alerts. It generates predictions and an uncertainty interval around them. If the observed value is outside of the predicted uncertainty interval, the model will generate an alert to notify about the anomaly detected.

This model managed to achieve an overall accuracy of 0.99 and an error rate of 0.0107, which is a quite satisfying result. However, the model has some limitations which leave room for improvements and could be the topic for some future studies.

8.1 Model Limitations

The proposed approach has two major limitations, which are the following:

- The training of the model is an operation that is quite inefficient. The grid search works well enough when the number of parameters and the number of time series fitted are not excessive. However, the problem rises when there are many parameters to tune or many services to monitor since the training times become quite long. For this reason, the simple implementation of the proposed model might not work in a real-world environment where there are many services to monitor.
- The model does not fit well spikes in the traffic. The problem is particularly evident in Section 7 with the results obtained by fitting data from ports TCP 12280 or TCP 25. In fact, in this case, the traffic is quite noisy. The result is that the model does not achieve good performance on this data.

8.2 Future Work

Since one of the model's limitations relates to the slow training times, one possible future research could focus on trying to find a more efficient alternative for finding the best parameters. For example, it could be interesting to test if some heuristics allow optimising the grid search by excluding some parameters without fitting the model on the training

data.

Another room for further development, analysis and possible improvement consists of trying to improve the model performance when fitting spiky data. For example, the additive Holt-Winters method has also been tested in addition to the Prophet model. The method used to compute the uncertainty interval used to classify an observation as an anomaly or not is the same as the one described in Section 6. The only difference is in using the Holt-Winters method to compute the predictions instead of the Prophet model. The Holt-Winters method has been tested on traffic generated from services running on ports TCP 25, TCP 22, and TCP 8000. These services were selected because they did not present any regular pattern when visualised.

Figure 36 shows the time plot of the predicted data using additive Holt-Winters against the original data and the uncertainty interval and the anomalies detected for port TCP 22.

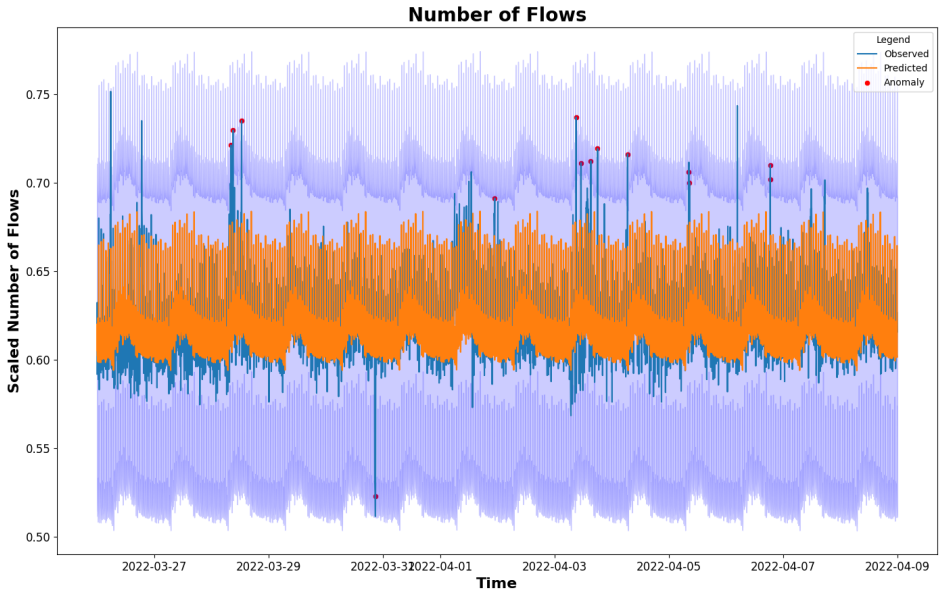

Figure 36. Time plot of the predicted data using additive Holt-Winters against the original and the uncertainty interval and the anomalies detected for port TCP 22.

Figure 37 shows the time plot of the predicted data using additive Holt-Winters against the original data and the uncertainty interval and the anomalies detected for port TCP 25.

Figure 38 shows the time plot of the predicted data using additive Holt-Winters against the original data and the uncertainty interval and the anomalies detected for port TCP

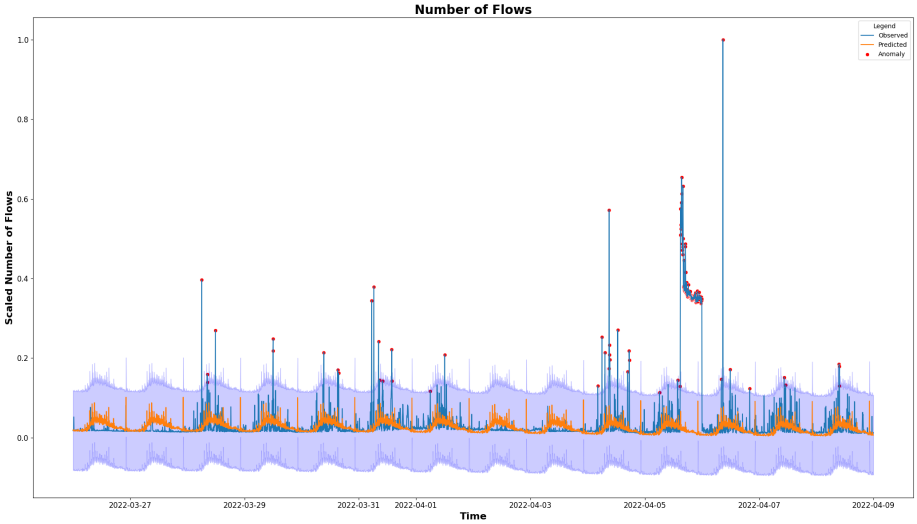

Figure 37. Time plot of the predicted data using additive Holt-Winters against the original and the uncertainty interval and the anomalies detected for port TCP 25.

8000. In all three cases shown in figure 36, figure 37 and figure 38, the Holt-Winters

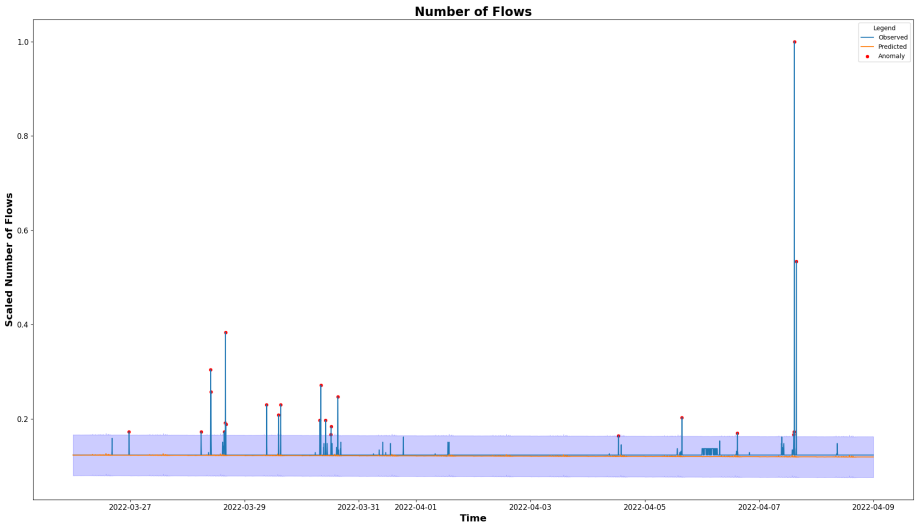

Figure 38. Time plot of the predicted data using additive Holt-Winters against the original and the uncertainty interval and the anomalies detected for port TCP 8000.

managed to provide a better fit of the curve compared to the Prophet model on the spiky data, even though it found more anomalies, so a higher number of false positives since the period used is the same as the one discussed in Section 7. However, the cause of the higher number of false positives might have its roots in the method used to find the uncertainty interval. Hence, an area of further development could be related to testing other ways

of computing the uncertainty interval compared to the one proposed in this study. In addition, the Holt-Winters method was also more expensive in computing resources than the Prophet model, so it is possible exploring some ways of improving its performance. For instance, the Prophet and the Holt-Winters methods were both tested on a machine with two cores and 2 GB of RAM. The machine where the Holt-Winters method ran was constantly alerting about CPU usage in contrast with the machine where the Prophet method was running. Hence, future studies could also focus on measuring and improving the computational cost of the method used to generate predictions. However, there is a lot of room for improvement to find a way of fitting data with spiky behaviour. For example, a future study could try testing the method proposed in [46] to detect anomalies in the NetFlow data.

Finally, it could be interesting to try the proposed approach on time series built differently from the approach used in this study. For instance, during this study, the time series was built by using the number of flows collected in the time frame of 5 minutes. However, there are other possible options that could be considered, such as the amount of data exchanged or the ratio between the amount of data exchanged and the number of flows collected.

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis

I Pasqaule Polverino

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis ”NetFlow Based Anomalies Detector Using Prophet Forecasting Model”, supervised by Kieren Niĉolas Lovell
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

References

[1] H. Ashtari, "What is network behavior anomaly detection? definition, importance, and best practices for 2022." https://www.toolbox.com/tech/networking/articles/network-behavior-anomaly-detection/. Accessed: 2022-04-10.

[2] solarwinds, "Configure flow alerts." https://documentation.solarwinds.com/en/success_center/nta/content/nta-configuring-flow-alerts.htm. Accessed: 2022-04-10.

[3] Cisco, "Cisco ios netflow." https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html. Accessed: 2020-10-22.

[4] Cisco, "Cisco ios flexible netflow." https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/flexible-netflow/product_data_sheet0900aecd804b590b.html. Accessed: 2020-10-22.

[5] B. Li, J. Springer, G. Bebis, and M. H. Gunes, "A survey of network flow applications," *Journal of Network and Computer Applications*, 2013.

[6] Cisco, "Cisco systems netflow services export v9." https://www.ietf.org/rfc/rfc3954.txt. Accessed: 2020-10-22.

[7] Cisco, "Netflow version 9 flow-record format." https://www.cisco.com/en/US/technologies/tk648/tk362/technologies_white_paper09186a00800a3db9.html. Accessed: 2020-10-22.

[8] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix," *IEEE communication survays & tutorials*, 2014.

[9] L. Bin, L. Chuang, Q. Jian, H. Jianping, and P. Ungsunan, "A netflow based flow analysis and monitoring system in enterprise networks," *Computer Networks*, 2008.

[10] V. Carela-Espanol, P. Barlet-Ros, A. Cabellos-Aparicio, and J. Solé-Pareta, "Analysis of the impact of sampling on netflow traffic classification," *Computer Networks*, 2011.

[11] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a better netflow," *ACM SIGCOMM 2004: Conference on Computer Communication*, 2004.

[12] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement 2010*, 2010.

[13] K. C. Claffy, H.-W. Braun, and G. C. Polyzos, "A parameterizable methodology for internet traffic flow profiling," *IEEE Journal on Selected Areas in Communications*, 1995.

[14] G. Cheng and H. Wu, "A netflow v9 measurement system with network performance function," *Springer-Verlag Berlin Heidelberg*, 2012.

[15] R. Vaarandi and M. Pihelgas, "Netflow based framework for identifying anomalous end user nodes," *International Conference on Cyber Warfare and Security*, 2020.

[16] R. Hofstede, V. Bartoš, A. Sperotto, and A. Pras, "Towards real-time intrusion detection for netflow and ipfix," *Proceedings of the 2013 International Conference on Network and Service Management*, 2013.

[17] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in largespatial databaseswith noise," *Proceedings of the 1996 International Conference on Knowledge Discovery and Data Mining*, 1996.

[18] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, 2015.

[19] P. J. Rousseeuw and M. Hubert, "Anomaly detection by robust statistics," *WIREs Data Mining Knowl Discov*, 2018.

[20] J. Wong, C. Colburn, E. Meeks, and S. Vedaraman, "Rad - outlier detection on big data." `https://netflixtechblog.com/rad-outlier-detection-on-big-data-d6b0494371cc`. Accessed: 2020-10-22.

[21] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data*, 2006.

[22] T. Komazec and S. Gajin, "Analysis of flow-based anomaly detection using shannon's entropy," *27th Telecommunications forum TELFOR*, 2019.

[23] R. Faek, M. A. F. Reh, and M. A. Fayoumi, "Exposing bot attacks using machine learning and flow level analysis," *Association for Computing Machinery*, 2021.

[24] J. Hou, P. Fu, Z. Cao, and A. Xu, "Machine learning based ddos detection through netflow analysis," *Proceedings of the 2018 IEEE Military Communications Conference*, 2018.

[25] T. Jafarian, M. Masdari, A. Ghaffari, and K. Majidzadeh, "Sadm-sdnc: security anomaly detection and mitigation in software-defined networking using c-support vector classification," *Springer-Verlag GmbH Austria*, 2021.

[26] P. Krishnamurthy, F. Khorrami, S. Schmidt, and K. Wright, "Machine learning for netflow anomaly detection with human-readable annotations," *IEEE Transactions on Network and Service Management*, 2021.

[27] S. Campbell, M. Kiran, and F. B. Wala, "Unsupervised anomaly detection in daily wan traffic patterns," *Springer Science and Business Media Deutschland GmbH*, 2021.

[28] C.-T. Yang, J. Liu, E. Kristiani, M. Liu, I. You, and G. Pau, "Netflow monitoring and cyberattack detection using deep learning with ceph," *Institute of Electrical and Electronics Engineer*, 2020.

[29] J. Liu, C. Yang, Y. Chan, E. Kristiani, and W.-J. Jiang, "Cyberattack detection model using deep learning in a network log system with data visualization," *The Journal of Supercomputing*, 2021.

[30] J. B. D. Cabrerat, B. Ravichandran, and R. K. Mehra, "Statical traffic modeling for network intrusion detection," *Proceedings 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2002.

[31] C. Wagner and T. Engel, "Detecting anomalies in netflow record time series by using a kernel function," *2012 6th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security*, 2012.

[32] K. Flanagan, E. Fallon, A. Awad, and P. Connolly, "Self-configuring netflow anomaly detection using cluster density analysis," *19th International Conference on Advanced Communications Technology*, 2017.

[33] M. V. O. de Assis, A. M. Zacaron, and M. L. Proença, "Time series forecasting methods for creating digital signature of network segments using flow analysis," *31st International Conference of the Chilean Computer Science Society*, 2012.

[34] J. Ekberg, J. Ylinen, and P. Loula, "Network behaviour anomaly detection using holt-winters algorithm," *6th International Conference on Internet Technology and Secured Transactions*, 2011.

[35] H. Brügner, "Holt-winters traffic prediction on aggregated flow data," *Seminar Innovative Internet Technologies and Mobile Communications SS2017*, 2017.

[36] S. Sarmadi, M. Li, and S. Chellappan, "A statistical framework to forecast duration and volume of internet usage based on pervasive monitoring of netflow logs," *IEEE 32nd International Conference on Advanced Information Networking and Applications*, 2018.

[37] S. J. Taylor and B. Letham, "Forecasting at scale," 2017.

[38] A. Harvey and S. Peters, "Estimation procedures for structural time series models," *Journal of Forecasting*, no. 9, 1990.

[39] A. C. Harvey and N. Shephard, "Structural time series models," *Handbook of Statistics*, 1993.

[40] W. Odom, *CCNA 200-301 Official Cert Guide, Volume 2*. Cisco Press, 29th ed., 2020.

[41] P. Haag, "nfdump." `https://github.com/phaag/nfdump`. Accessed: 2021-04-10.

[42] R. Matulevičius, *Fundamentals of Secure System Modelling*. Springer Nature, 1st ed., 2017.

[43] M. Fowler, *UML Distilled*. 3rd ed.

[44] Facebook, "Quick start." `https://facebook.github.io/prophet/docs/quick_start.html`. Accessed: 2022-04-10.

[45] Facebook, "Diagnostics." `https://facebook.github.io/prophet/docs/diagnostics.html#hyperparameter-tuning`. Accessed: 2022-04-10.

[46] D. Qi and A. J. Majda, "Using machine learning to predict extreme events in complex systems," *Proceedings of the National Academy of Sciences*, 2020.