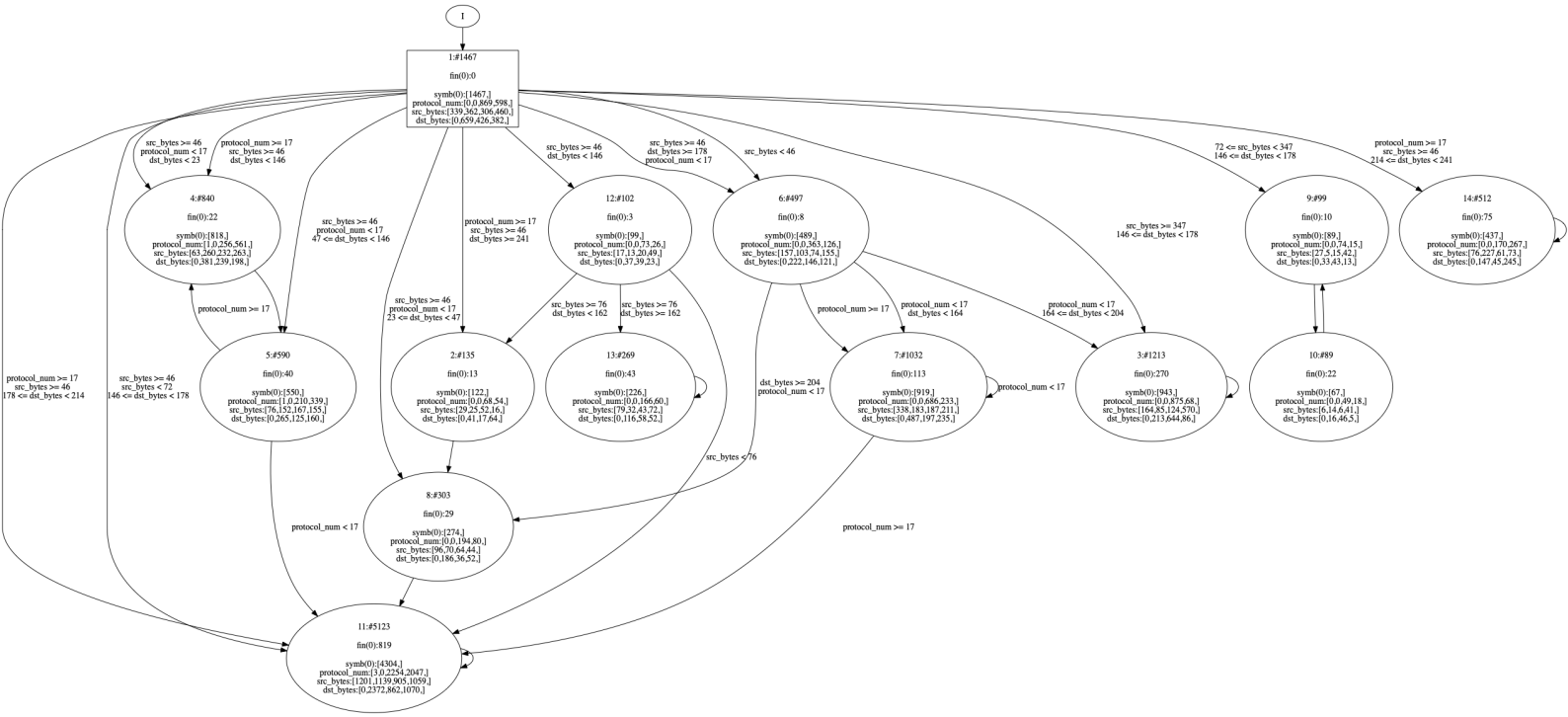
Anomaly Detection in Network Traffic using Multivariate State Machines

Vasileios Serentellos



TU Delft

Anomaly Detection in Network Traffic using Multivariate State Machines

by

Vasileios Serentellos

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday August 25, 2020 at 2:00 PM.

Student number:	4843789	
Project duration:	November 28, 2019 – August 25, 2020	
Thesis committee:	Dr. ir. S. Verwer,	TU Delft, supervisor
	Prof. dr. ir. R. L. Lagendijk,	TU Delft
	Dr. A. Panichella,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

Computer networks have nowadays assumed an increasingly important role in the expression of modern human activity through the ongoing rapid development in the field of Information and Communication Technologies (ICT). More and more individual users and businesses around the world are gaining access to networks online, while the range of services offered by these networks span multiple domains of human life, leading them to grow in terms of both size and complexity, and in parallel handle a constantly growing volume of user-generated data. As with every important aspect of human life, computer networks need to be protected from malicious adversaries aiming to degrade the quality of the offered services and acquire unauthorised access to them, so as for their intended functionality to be uneventfully maintained. The broad success of Machine Learning (ML) based techniques in applications originated from a wide range of fields has led to the wide adoption of such techniques in the premises of automated network traffic analysis systems aiming to detect malicious activity within computer networks, with a notable portion of these systems employing solutions inspired from the field of anomaly detection. Such an automated system for anomaly detection in network traffic, attempting to address as many of the major shortcomings of earlier relevant works as possible, constitutes the content of this thesis. In particular, the proposed system is designed to offer fine-grained analysis of the recorded traffic, by leveraging powerful sequential learning models, like multivariate state machines, equipped with well known anomaly detection algorithms in their structure, towards the extraction of benign behavioral profiles from NetFlow traces of aggregated network entities, like hosts or connections, so as to use these profiles towards the identification of any behavior not conforming to them as anomalous. Three publicly available Netflow-based datasets, incorporating a diverse set of cyber attacks, are utilized to evaluate the detection potential of the proposed methodology. First, the effectiveness of multiple different settings of the designed detection system is quantified, so that the configurations with the most promising detection potential can be identified. Subsequently, the proposed system is compared with various easily developed baseline detection methodologies for the extent of the impact of its inherent complexity to be evaluated. Finally, the designed system is examined in comparison to a state-of-the-art detection technique operating on one of the three datasets used in this thesis, achieving higher or similar detection performance on all the scenarios considered.

Acknowledgements

I am grateful to many people that contributed with one way or another to the completion of this thesis.

First of all, I would like to thank my supervisor, Dr. ir. Sicco Verwer, who provide me with the opportunity to engage a highly interesting topic in the premises of this work. Throughout these nine months, he was constantly available for consultation and willing to offer his advice and suggestions when needed. His inspiring ideas contributed a lot into shaping both the goal and the content of this thesis.

Second, I would like to thank Prof. dr. ir. R. (Inald) L. Lagendijk, and Dr. Annibale Panichella for agreeing to be part of my Thesis Examination Committee, as well as their cooperation in organizing and carrying out my Thesis Defence.

Third, I would like to thank everyone from my supervisor's group for the weekly meetings, through which we could share our ideas and concerns for successfully undertaking the final milestone of MSc studies, the Thesis project.

Last but not least, I would like to express my gratitude to my family for their nonstop support throughout my studies, especially during the difficulties that I faced over the past two years, as well as my friends, many of which were also my classmates, for being there for me throughout this course.

Vasileios Serentellos
Delft, August 2020

Contents

List of Figures

List of Tables

1 Introduction

The rapidly increasing rate of development in the field of Information and Communication Technologies (ICT) recorded over the last years has rendered computer networks an indispensable part of modern human activity. Nowadays, more and more individual users and businesses around the world are gaining access to networks online, while the pool of services offered by such networks is being constantly enriched. Computer networks provide important assistance in various domains of modern human life, ranging from simple web search to business intelligence and medical foresight, rendering the operation and prosperity of organisms, and human societies in general, inherently dependent to the proper functioning of these networks. On top of that, the remarkable development observed in the fields of Artificial Intelligence (AI) and Machine Learning (ML), along with the notable ongoing success of projects and applications related to these fields, have contributed significantly to the tremendous growth on both the size and the complexity of computer networks. This progress is inherently linked to the enormous increase in the volume of data stored on public or privately owned cloud structures, and transferred within and across networks online, while the number and variety of smart devices connected to the internet and communicating with other such smart devices online are continuously rising too.

As with every important aspect of human life, the proper functioning of computer networks needs to be protected and secured by malicious adversaries aiming to degrade the quality of the provided services and acquire unauthorised access to such networks. Of course, the degree of difficulty associated with securing businesses and individuals against cyber attacks grows proportionally to the number of interconnected devices, services offered and transactions conducted online, since the pool of potential attack targets is constantly widening. On top of that, the inherent dependence of multiple human activities on computer networks renders the consequences of successfully performed cyber attacks quite severe. In fact, according to Cybersecurity Ventures 2019 Official Annual Cybercrime Report[1], global cybercrime damages are predicted to cost up to 6 trillion dollars annually by 2021, while the research conducted by the AV-TEST IT-Security Institute[2] suggests that 144.91 million new malware samples were recorded in 2019, with 57.8 million new samples having been recorded already in 2020. These figures, along with the importance regarding the management of computer networks illustrated above, indicate the eminent need for designing and developing automated systems capable of accurately detecting malicious traffic activity among the vast amount of available traffic information.

The broad success of machine learning techniques in applications originating from a wide range of fields has led to the increasing deployment of such techniques as pivotal components of automated network traffic analysis systems aiming to detect malicious activity within networks. A significant portion of such techniques are inspired from the field of anomaly detection, since identified anomalies in the traffic of the network under examination could indicate the presence of malicious behaviour in the premises of that network. The core functionality of such techniques can be roughly summarised as the development of a learning model, extracted from a set of observed data points, capable of distinguishing between the benign and malicious data points being present in the traffic of the monitored network. Of course, such systems need to meet certain requirements regarding both the type of data used as the input of the system, and the way that such data are

[1] `https://www.herjavecgroup.com/the-2019-official-annual-cybercrime-report/` - Date accessed: 12/06/20
[2] `https://www.av-test.org/en/statistics/malware/` - Date accessed: 12/06/20

processed, while delivering high detection and low false alarm rates. Such an automated system for anomaly detection in network traffic constitutes the content of this thesis. In particular, the proposed system aims to provide fine-grained analysis on the recorded traffic, by leveraging powerful sequential learning models, like multivariate state machines, towards the extraction of benign behavioral profiles from the recorded traffic. The motivation behind the designed system, as well as a brief summary of its structure and characteristics, along with the problems that it aims to address, are presented in the sections to follow.

1.1. Motivation

As mentioned above, the increasing complexity in the structure of computer networks, as well as the massive loads of data transported both within and between networks online, have significantly complicated the task of properly securing such networks, as well as the endpoint devices connected to them, from malicious activity. The effective identification of malicious behaviour in computer networks is of pivotal importance for the network administrator in order to ensure that such behavior cannot significantly impact the quality of services offered to end users. It can be easily understood that the manual inspection of network traffic towards the goal of detection constitutes an impossible task, especially in the current era of big data. As a result, there is the need of developing automated systems able to identify anomalies in the recorded traffic of a given network, so that possible threats can be detected and mitigated. To that end, various automated approaches have been proposed over the years, with each of them associated with various advantages and disadvantages. The purpose of the current thesis is to develop a fully automated end-to-end detection system able to address as many of the major shortcomings of earlier works as possible, while providing high detection performance.

The earliest attempts made towards the development of such systems were mostly based on the inspection of the payload of the packets transmitted across the monitored network, under the assumption that anomalies in the payload could indicate the existence of malicious traffic associated with the examined packets. Despite the fact that such methods initially provided high detection rates, the great volume of data exchanged in modern networks, along with the enforcement of strict data privacy regulations, like the General Data Protection Regulation (GDPR)[3], and the increasing attempt made by malicious adversaries to evade detection using techniques such as code obfuscation, encryption, and polymorphic code, have rendered the adoption of such approaches unrealistic. A way of addressing such issues, that has received increasing attention during the last years, regards the analysis of high-level aggregated network communication information, like NetFlows. The collection of such data is considered relatively easy and cheap, while a system based on the analysis of such data is able to both respect any existing privacy requirements due to the utilization of solely high-level features of the recorded inter- and intra-network communication, and separate its detection algorithm from the content of the examined packets' payload. Yet, extracting meaningful information towards detection solely from high-level features of the monitored traffic constitutes a quite challenging task.

The ongoing success recorded in the field of Machine Learning has led many network management and intrusion detection systems to employ ML techniques towards the goal of identifying malicious behaviour in a network. Most of such approaches perceive the whole network traffic as a dataset on which a classification algorithm needs to be fitted, so that benign data samples can be distinguished from malicious ones. Such approaches, though, may commonly entail some of the following drawbacks. First, deriving a detection model from the entire recorded network traffic does not provide a fine-grained representation of the communication behavior in the network under examination. Instead, a detection system that could associate specific hosts or connections with malicious activity seems a more meaningful choice for a network administrator operating on the traffic of the monitored network. In addition, the robust operation of many machine learning algorithms is founded on the existence of a training set of representative data samples from all classes of the problem. It can be easily understood that, especially when dealing with malicious activity, having access to malware is not always the case. To that end, various unsupervised approaches have been developed in the field of anomaly detection for identifying outlying data in the provided dataset, and have been leveraged in malware detection systems. Yet, many of such approaches are of black-box nature, meaning that the detection procedure cannot be easily interpreted. Such a characteristic is significantly undesired, especially in an era that privacy concerns are constantly raised. Finally, capturing any underlying temporal relation between data points is crucial when dealing with timed data like NetFlow traces, thus it would be beneficial for a detection system to consider such information.

In the anomaly and malware detection literature there have been various noteworthy works [33, 34, 68, 79] trying to address the aforementioned issues, that are associated with impressive results, yet the proposed ap-

[3] `https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679` - Date accessed: 12/06/20

proaches were based on the development of detection models learnt from malicious traffic. However, as explained before, acquiring representative samples of malware is not always easy, while the diversity of malware and the constant attempts of malicious adversaries to adapt themselves on the detection mechanisms, so as to structure their attacks in such a way that they would demonstrate a behaviour similar to normal traffic, render the applicability of such techniques quite restrictive. Learning from benign traffic, which a network produces during its normal execution, constitutes a more flexible solution that could be associated with higher possibilities of generalising across different kinds of malware. Of course, since network traffic analysis constitutes a domain, in which normal behavior keeps evolving, meaning that the current notion of normal behavior might not be adequately representative in the future, it would be beneficial for an automated detection system to make use of as little prior information as possible. All things considered, the detection system proposed in the premises of this thesis aims to address the aforementioned issues by utilizing powerful sequential models, like multivariate state machines, equipped with well known anomaly detection algorithms in their structure, in order to extract benign behavioral profiles from NetFlow traces of aggregated network entities, like hosts or connections, and use these profiles towards the identification of any behavior not conforming to them as anomalous.

1.2. Problem Statement

The problem faced in the premises of the current thesis can be typically formulated as an anomaly detection task, since, given a set of both normal and anomalous data points, a model capable of distinguishing between these two categories shall be developed, achieving both high detection performance and a relatively low false alarm rate. In the context of malware detection, NetFlow-based data capturing basic traffic summary statistics about the connections between endpoint devices in the monitored network are frequently utilized as the data points on which the detection system will operate, with the normal set of points representing the benign traffic of the network, and abnormal points referring to malicious activity. Apart from the ability to identify malicious activity, the proposed system should be able to associate the detection results with specific aggregated network entities, like hosts or connections, characterizing in that way their behavior and offering a greater level of analysis to the network administrator. It can be easily understood that, if the detection system is capable of identifying malicious hosts or connections, the task of identifying the origin of a possible threat and securing the monitored network against it can be effectively simplified. Furthermore, the generalization potential of the modelling procedure should be evaluated, since the proposed method should be able to identify different types of malware activity. Towards that end, three different datasets of NetFlow traces incorporating a diverse set of cyber attacks are used to evaluate the performance of the proposed detection system. More information on these datasets can be found in Chapter 4 of this work. On top of that, the impact of the inherent complexity of the proposed detection model needs to be examined, thus the proposed system is compared with various easily developed baseline detection methodologies across all the NetFlow datasets taken into account in this thesis. Finally, it is considered beneficial to compare the detection performance of the proposed system with the one attained by a state-of-the-art detection technique, so as to better comprehend the contribution of the work conducted in this thesis in the field of network-oriented anomaly detection. This comparison is conducted against a recently published (2020) method operating on one of the three NetFlow based datasets used in this thesis, and is presented in Chapter 7.

1.3. Proposed Solution

The functionality of the proposed detection system is founded on the assumption that, if models are extracted from the normal operation of a monitored system, then any behavior rejected from these models is considered an anomaly, thus it could be potentially associated to malicious activity. In more detail, the proposed architecture builds upon this assumption, while attempting to address the issues mentioned in the previous sections in the following way: State-of-the-art automata learning algorithms are used to extract multivariate state machines from the benign NetFlow traces of certain network entities, like hosts and connections. As a result, depending on the preferred level of analysis (host or connection level), each state machine can be perceived as a communication profile capturing the benign behavior of the corresponding network entity. Each state of the derived state machines can be further perceived as a temporal cluster of the general behavior of the corresponding network entity, so that underlying temporal relations can be captured by the model. Subsequently, the benign traces can be replayed on each corresponding state machine, and an anomaly detection model can be learnt in each state. As a result, each state can be used as a detection predictor when unseen NetFlow traces are replayed on the associated state machine. At this point, it should be mentioned

that the proposed system utilizes multivariate state machines, meaning that there is no need for the creation of a symbolic alphabet out of the raw NetFlow features, which as a matter of fact allows a straightforward learning process on each state of the extracted automata. Finally, each fitted multivariate state machine can be utilized as a benign communication profile providing predictions on unseen sets of NetFlow traces, with the benignity of each set depending on the extent of its match to the extracted benign profiles.

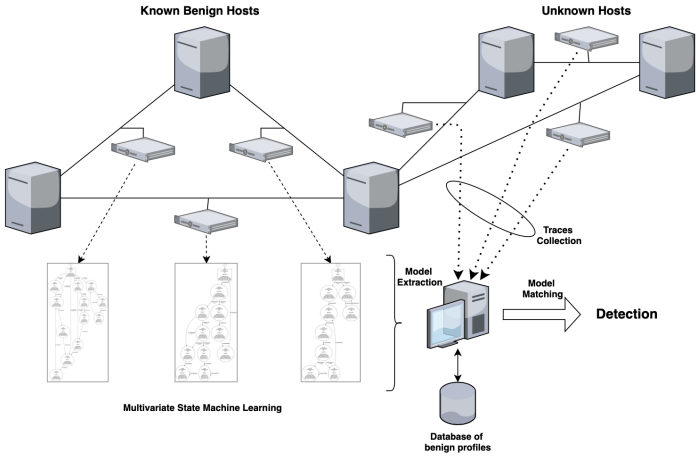


Figure 1.1: High level illustration of the detection process on a virtual network topology

A high level illustration of the detection process described above can be seen in Figure 1.1. In brief, the NetFlow traffic of a set of benign hosts is collected through some monitoring devices and used to extract multivariate state machines capable of capturing the temporal relations in the communication traffic of a specific benign network entity (host or connection). Subsequently, the benign set of flows (traces) associated with each network entity are replayed on the corresponding state machine, and a outlier detection algorithm is fitted on the flows captured in each state of that machine. In the premises of this thesis, three well-known outlier detection techniques, namely Local Outlier Factor (LOF), Isolation Forest, and a Gaussian Kernel Density Estimation (KDE) based method, are evaluated on the designed detection system. After fitting the detection algorithm on each state of every multivariate state machine inferred from the considered benign network entities, these state machines can act as behavioral communication profiles of the corresponding entities, composing in that way a "database" of benign profiles learnt from the benign traffic recorded in the network under examination. Finally, when NetFlow traces are collected from unknown hosts or connections, these traces can be replayed on each benign model in the database and a local prediction can be made from the detection algorithm fitted in the states of these models. The aggregated predictions from all states of a model define the extent to which a set of traces from an unknown network entity match the benign behavior represented by that model. If there is no confident match between the traces of the unknown network entity and any benign model, then this entity is considered as anomalous and an alarm is raised. As it was mentioned earlier, the implemented detection process is briefly illustrated through Figure 1.1 for the reader to acquire an initial understanding of the proposed methodology. The detailed presentation of the proposed solution can be found in Chapter 6.

1.4. Research Questions

The primary research questions, that are addressed in this work, are the following:

1. How effective can multivariate behavioral communication profiles extracted solely from basic NetFlow features of benign data be proved for the purpose of malware detection in network traffic?

2. What type of malicious behaviour can be successfully identified from the proposed detection system?

3. What is the impact from the incorporation of outlier detection models in the structure of multivariate state machines towards the goal of detection?

1.5. Contributions

The most prominent contributions of this thesis can be summarized as follows:

- A fully functional pipeline for identifying malicious behavior in NetFlow data, with the use of behavioral communication profiles extracted from multivariate sequential models, that has not been implemented before, to the best of our knowledge
- An anomaly detection system that does not require any complex feature preprocessing steps, since the NetFlow features are directly and transparently used in the detection process
- An anomaly detection system with great generalization potential, originating from the fact that this system is capable of operating on basic NetFlow features available in all types of captures
- A dynamic window extraction technique capable of adjusting the window length and stride according to the temporal information incorporated in the examined data sequences
- The introduction of a multivariate approach for learning sequential models, without the need of a symbolic alphabet, and the loss of information that it entails
- An empirical evaluation on effectively identifying malicious activity on NetFlow data based on multiple publicly available datasets, the promising results of which illustrate the high detection potential of the proposed pipeline
- A well-founded comparison between the proposed system and a state-of-the-art detection technique found in the literature on a common dataset, in which the designed methodology attains competitive detection performance to that of the latter technique

1.6. Thesis Structure

In this section the structure of the current thesis is discussed and explained. In Chapter 2, the required background knowledge, for the reader to properly comprehend the main terms and concepts used throughout this thesis, is presented. In Chapter 3, previous works on anomaly detection in network traffic, as well as state machine learning, are listed and discussed. In particular, this chapter includes the works from which inspiration towards the design of the developed system was derived, as well as those that gave birth to the main research questions of this work. Chapter 4 contains an extensive presentation on the NetFlow datasets used to train and test the effectiveness of the proposed system, along with some exploratory analysis conducted on these datasets towards the achievement of a better understanding about the nature of the data. In addition, in this chapter the main choices regarding the preprocessing are highlighted and explained.

Chapter 5 constitutes an in-depth presentation of the sequential model used as the primary building block of the designed pipeline. The framework used for learning the sequential model from NetFlow traces is introduced, while the learning algorithm, as well as the additional features taken into consideration in the learning process, are explained and discussed. In chapter 6, the main contribution of this thesis is presented. The pipeline developed in the premises of this thesis is analysed, and design choices are explained and justified. In more detail, this chapter touches upon the different levels of abstraction, in which the NetFlow data are processed, the trace extraction process, needed to transform raw NetFlow data into their sequential representation compatible with the input format of the learning algorithm, the different outlier detection algorithms applied on each state of the extracted models, as well as the way in which these models were used towards the goal of anomaly detection. Chapter 7 contains a presentation of the experimental procedure followed, along with the results obtained through the undergone experiments, expressed in terms of accuracy, precision, and recall achieved. Finally, Chapter 8 concludes on the results of the conducted research, while introducing its limitations and some possible future directions.

2
Background

The goal of this chapter is to introduce the reader to the theoretical background needed to properly comprehend the main terms and concepts used throughout this thesis, thus it does not constitute by any means a full analysis on the presented topics. More detailed information of these topics can be addressed through the references provided in each section. This chapter starts with an introduction to the simplest and most generic, yet important, concepts, and gradually arrives to the presentation of more complex notions and ideas upon which the proposed detection methodology is built.

2.1. Network Traffic

Computer networks constitute the primary components of the digital infrastructure that supports interpersonal communication, services and resources sharing, as well as the unprecedented data propagation recorded across the world, as the field of Information and Communications Technology (ICT) keeps evolving. Data transmission, both between computer networks and between nodes of the same network, constitutes the primary procedure on which the functionality of all the services, and applications supported nowadays online, is based. The data moving across a network at a given point of time comprise the traffic of this network, and are mostly encapsulated in network packets. Accordingly, these packets consist of two main kinds of data: control information and user data (payload). The former type of data is used for regulating the correct transmission of the data, while the latter constitutes the data, on which the functionality of the network is structured. As a result, it can be easily seen that disruptions in the data traffic, intentional (cyber attacks) or not (critical endpoints malfunction), can lead to unwanted behaviour within a computer network that eventually may hinder its operation. To avoid such unpleasant situations, large organisations tend to invest substantial resources for protecting and monitoring their enterprise networks, consisting of all connected devices and communications infrastructure within the organisation. Proper capturing and analysis of network traffic provides the organization the ability to identify such anomalies in the transmitted data and develop protection and recovery mechanisms from any anomalous network states.

2.1.1. Flow Monitoring and NetFlow Data

As it was mentioned above, network monitoring has been proven to be one of the most important network management processes, since it can provide significantly valuable information to a network administrator regarding the nature of the traffic recorded within the network under examination. As a result, various network monitoring approaches have been developed throughout the years. These approaches can be divided into two main categories, namely the active and the passive ones [39]. Active approaches interfere with the examined network by injecting traffic towards the end of performing different types of measurements in the network, while passive approaches, an example of which this work utilizes, solely observe the existing traffic as it passes by a measurement point (usually a router), on which the data exporter has been attached. In the past, passive network monitoring was primarily based on packet capture, since deep packet inspection provides significantly insightful information on the network traffic. Nevertheless, there are two main problems concerning such approaches. Firstly, when used in high-speed networks, as most networks are nowadays, packet capture requires expensive hardware and infrastructure for storage and analysis of the massive amount of data recorded. Secondly, inspecting the payload of packets could reveal information, that should

not be disclosed to a third party, raising multiple privacy concerns over the procedure. To deal with such issues, passive monitoring approaches based on flow export have been developed. In such approaches, flows constitute the basic traffic recording unit, which, according to [17], can be defined as "a set of IP packets passing an observation point in the network during a certain time interval, such that all packets belonging to a particular flow have a set of common properties". Thus, the main functionality of such approaches can be summarized as the aggregation of packets into flows so that they can exported for further processing.

NetFlows constitute a flow export protocol, incorporating network flow event summaries collected typically from routers within the examined network. In more detail, NetFlow records contain information about connections between endpoint devices either with both of them residing in the premises of the examined network, or with one of them originating outside of the studied network. Such flow-based data include extensive meta information about the recorded connections, and typically encompass a *Timestamp* with the date and time that a connection has been initiated, the *Source* and *Destination IP addresses* of the endpoints included in the connection, the *Source* and *Destination Ports* used, the *Protocol* used for the communication between the included devices, the *Duration* of the connection, as well as the number of *Packets* and *Bytes* transferred while the connection was maintained. Of course, even more descriptive features might be included in a NetFlow record, like the *TCP-flags* used or the number of *Missed Bytes* in the connection, nonetheless, the attributes mentioned above constitute the ones captured in the majority of cases. An example of a set of two NetFlow records can be seen in Figure 2.1. In this example, apart from the aforementioned features, the symbol -> is included, denoting the direction of each flow. Such information is incorporated in bidirectional NetFlow captures, so that the direction, in which the transmission of bytes and packets occurred in each connection, can be described. At this point, it should be mentioned that the scope of this section is to introduce the reader to the concept and the functionality of NetFlows, since such type of data were used as the input of the designed pipeline. For a more in depth presentation of the concept, as well as the process of flow export, the reader is directed to [39].

Date flow start	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Packets	Bytes
2010-09-01 00:00:00.459	0.000	UDP	127.0.0.1:24920	->	192.168.0.1:22126	1	46
2010-09-01 00:00:00.363	0.000	UDP	192.168.0.1:22126	->	127.0.0.1:24920	1	80

Figure 2.1: A sample bidirectional NetFlow recording

As it was explained earlier, flow export demonstrates important advantages comparing to packet capturing and inspection, especially in terms of scalability and privacy preservation, yet deep packet inspection provides more insight into the network traffic, which as a matter of fact could render flow-based techniques less suited for security analysis and threat detection tasks. In practice, this is not the case, since multiple types of cyber attacks, like DoS attacks, network scans, and botnet communication, affect metrics that can be directly derived from flow records, such as the volume of packets and bytes transferred, the number of active flows in a specific time interval, or suspicious port numbers and blacklisted destination hosts [39]. Thus, the fact that flow-based data, like NetFlows, can be leveraged in threat detection, combined with the increasing need for scalable and privacy-preserving solutions in the network monitoring process of an organization or an enterprise, renders methods working with such data highly beneficial. In fact, various anomaly and threat detection systems operating on flow-based data have been developed, as it will be presented in great extent in Chapter 3. Finally, given that, as mentioned above, NetFlows include a timestamp among their features, it can be said that such data demonstrate a sequential nature. Therefore, some basic understanding on the nature of sequential data and the ways in which such data can be handled and processed in the premises of a learning pipeline is needed, in order for the reader to be able to comprehend the data processing steps followed in this work.

2.2. Sequential Data

Data incorporating temporal information among their features are characterized as sequential. This temporal information can range from the association of time values with each data instance, as in the case of time series, to simply the consideration of the relative position of a data instance within the series of available instances, as in the case of text. The common characteristic in these examples is the presence of a notion of ordering between the data instances. In a more formal way, the general classification framework composed by a set of data instances (or samples) $\{(x_i, y_i)\}_{i=1}^{N}$, where N is the number of data instances, x_i is the attribute vector of instance i, and y_i is the class label of instance i, can be transformed to fit the notion of sequential learning by representing each data instance by a temporal sequence of its attributes, as

$x_i = \langle x_{i,1}, x_{i,2}, \ldots, x_{i,T_i} \rangle$ and $y_i = \langle y_{i,1}, y_{i,2}, \ldots, y_{i,T_i} \rangle$, where T_i denotes the temporal index in the sequence of the *i-th* data instance. Typical machine learning algorithms tend to perform well when the data samples x, as well as the class labels y, in which they belong, are drawn independently and identically (iid) from some joint distribution $P(x, y)$, which, as a matter of fact, is not the case with sequential data. As a result, when such kind of data are provided, the use of standard machine learning techniques might lead to the creation of a system that neglects the underlying temporal information, as well as the structural dependencies between subsequent instances. Such dependencies are quite important when modelling the behaviour of sequential systems, thus the learning algorithm to be applied should take into account such information in order for a model of high quality to be developed. Apart from that, modelling methods that consider the sequential structure of the input data can provide significant assistance in better understanding and analysing the system under learning, since the sequential dependencies can be depicted in the inferred model too. All things considered, in order to leverage the sequential nature of NetFlow data in the developed learning system, a model capable of capturing sequential patterns should be incorporated. Towards the fulfillment of that need, state machines are utilized, since they have been proven quite effective in modelling such data. More information on state machines in general, and specifically about the models used in the premises of this work, can be found both later in this chapter and in Chapter 5.

Before feeding the data into the learning algorithm, though, significant decisions on the way in which such data will be preprocessed need to be taken. The way, in which the sequential data will be processed, can significantly impact the quality of the produced model. If the sequential patterns and dependencies present in the data are not appropriately captured, misleading representations of the underlying temporal behaviour may be extracted, which as a matter of fact will eventually degrade the learning procedure. There are many choices to be made when preprocessing sequential data, with the primary ones involving the aggregation of the data into meaningful sequential clusters, as well as the way in which the attributes associated with each data instance are represented within an extracted sequence. These choices, and their effects on the quality of the produced models, are discussed in the following section.

2.2.1. Data Aggregation & Window Extraction

Data aggregation, in the context of sequential learning, can be primarily expressed as the clustering of data according to their temporal values. Yet it should be noted that, apart from any temporal features, attributes capable of uniquely characterising the clusters to be extracted can be leveraged in the data aggregation procedure too. For instance, network connections can be grouped according to both their timestamps and their source-destination IPs pairs. This sequence-aware clustering is usually conducted through the use of time windows, with different techniques having been developed around this strategy. Some of the most frequently used time-windowing methods are the following:

- **Time slicing**: This strategy suggests the grouping of data instances according to fixed non-overlapping time periods. Thus, each temporal cluster is composed of the data instances the temporal index of which lies within the time period associated with that cluster.
- **Context-aware time slicing**: The rationale of this approach is the same as that of the previous one, with the main difference being that the clustering of the data is also dependent on a subset of features that can uniquely identify each data instance. For instance, if NetFlows were to be considered, as it is the case in the current thesis, the flow could be clustered both in terms of their Timestamps and their source IP address.
- **Static sliding windows**: In the case of static sliding windows clustering is again conducted on a temporal level, as in the previous two strategies, with the main difference being that there can be an overlap between consecutive time bins. In particular, a window of fixed length is "slid" upon the data sequences under a predetermined sliding step (stride). It can be easily seen that if the length of the stride equals the length of the time window, the sliding windows strategy is equivalent to time slicing. An interesting point regarding the implementation of this strategy arise by the fact that the window, as well as the stride used, can be of either a timed or a numerical nature. In the first case, fixed time periods are used to define the window and the stride of the windowing process, leading to clusters with a varying number of data instances, while, in the second case, a fixed number of data instances is used instead, meaning that each cluster contains the same number of instances.
- **Dynamic sliding windows**: The adoption of static sliding windows has been proven to be quite effective when applied to data measured at a fixed temporal rate. Nevertheless, when timings are not fixed,

which, as a matter of fact, is usually the case, static methods tend to neglect sequential dependencies that fall out of the fixed length history taken into account by each window. Methods residing in this category adopt a more sophisticated windowing version, aiming to deal with the aforementioned issue. In more detail, instead of specifying the length of the time window and the stride to some constant values, these values are dynamically inferred by the sliding procedure, based on some user-defined metrics regarding the nature of the captured data. In fact, the windowing method implemented in the main pipeline of this work utilizes dynamic windows, thus a better understanding of such approaches can be achieved after reading Chapter 6.

One of the main advantages of time slicing/windowing approaches can be addressed by the fact that they facilitate the application of standard machine learning algorithms on the data captured in each slice or window. In particular, a given sequence X can be truncated in windows and a standard machine learning predictor can be applied to each window. Apart from that, distance metrics, either standard, like the Euclidean distance, or temporal ones, like Dynamic Time Warping (DTW) [7], can be utilized to measure the (dis)similarity between patterns captured by different windows, with such information used as an additional classification rule for the data in each window. Subsequently, the predicted labels $\hat{y}_t$ of each window are concatenated according to their temporal index and the final predicted sequence $\hat{Y}$ is formed. In addition, the extraction of temporally local correlations between data samples achieved through a temporal clustering procedure, can benefit pure sequential learning methods, like state machines, and hidden Markov models (HMMs). Nevertheless, as it was mentioned above, the parameters associated with the structure of the adopted windowing strategy (length of the window, stride, etc) shall be chosen with caution, since improper choices could lead to falsely extracted sequential dependencies, and, as a result, to a sequential model of poor quality.

After grouping the data into meaningful temporal clusters, the strategy of aggregating the features of the clustered is sometimes adopted as a way of lowering the computational complexity that long data sequences entail. In particular, especially when extremely long data sequences (in the order of millions of data samples) are considered, it is common practice to reduce the number of data instances captured in each window either by fully aggregating the attributes of the windowed samples through the use of some statistical quantities (median, mean, standard deviation, mode, etc) to construct an aggregated view over the window, or by constructing smaller aggregation windows within the premises of the initial windows, with the aforementioned aggregation happening on these smaller windows [28]. Attribute aggregation can produce both positive and negative results on the learning process. On the one hand, significant reduction in the computational complexity of the learning process can be attained, while, the additional information originating from the incorporation of attributes statistics could improve the generalization ability of the model. On the other hand, the aggregation of consecutive instances could lead to significant temporal information loss, since the learning process would be applied to a "compressed" version of the data.

2.2.2. Discretization

In most cases discretization is employed as another measure to reduce the computational complexity in sequential learning tasks through dimensionality reduction. To this end, the features associated with each data instance, except from the temporal index, are combined into a symbolic representation, leading to a sequence of symbolic events derived from a finite alphabet [51]. In the current thesis, such approaches are not adopted, since the mapping of the data attributes into symbolic events degrades the interpretability of the developed model. Instead, a multivariate approach is followed, so that in case of detected anomalies, the actual attributes of the potentially anomalous flows can be inspected. Such approaches can be significantly helpful when the root cause of a detected anomaly is investigated. To this end, it can be concluded that in the premises of this work NetFlows are perceived as multivariate sequences, rather than discrete event sequences.

2.3. Finite State Automata

As it will be explained in greater depth in the chapters to follow, the developed methodology is based on the extraction of models able to capture any underlying patterns in a given series of network flows. The models used in the premises of this thesis fall into the category of Finite State Automata (FSA), or Finite State Machines (FSM), or simply State Machines. Before elaborating further on the nature and characteristics of these models, it should be pointed out that the current section constitutes a brief, yet insightful, introduction to automata theory, aiming to communicate the basic notions used in the premises of this project to a reader unfamiliar with this specific field. More information on the field can be found in [40, 76, 78].

FSMs are mathematical models of computation used to formally describe the behaviour of systems that perform a predetermined sequence of actions depending on the sequence of events received as input. In more detail, FSMs can be in exactly one of a finite number of states at any given point in time, and they can "transit" from one state to another as a response to a received input [88]. As a result, a FSM can be defined by the list of its states, its initial state, as well as the inputs that fire each transition from one state to another. FSMs can be classified either as deterministic or as non-deterministic, with the main distinction between those two categories being the fact that in the case of deterministic state machines each transition can be uniquely specified by the source or origin state and the input symbol observed. In the current thesis, modelling is based solely on deterministic state machines. The general structure and characteristics of such models will be briefly explained in this section, while more information about the specific nature of the models that are eventually used in the proposed methodology will be presented in Chapters 5 and 6.

2.3.1. Deterministic Finite State Automata

A Deterministic Finite Automaton (DFA), or Deterministic Finite State Machine (DFSM), M can be formally defined as a quintuple $\langle Q, \Sigma, \delta, q_0, F \rangle$, with each term denoting the following:

- Q : a finite set of states
- Σ : a finite set of symbols, also referred as the alphabet of the state machine
- δ : a transition function $Q \times \Sigma \rightarrow Q$, mapping each state $q \in Q$ and an input symbol $\sigma \in \Sigma$ to the next state $q' \in Q$
- q_0 : an initial or starting state, where $q_0 \in Q$
- F : a set of accepting or final states, where $F \subseteq Q$

A given string $s = \sigma_1\sigma_2...\sigma_n$ of n symbols over the finite alphabet Σ (which can be also called as word) is considered as accepted by the automaton M, if a sequence of states $q_0, q_1, ..., q_n$ exists in Q meeting the following conditions:

- q_0 is the starting state of M
- $q_{i+1} = \delta(q_i, \sigma_{i+1})$, for $i \in [0, n-1]$, meaning that each state in the sequence is produced by feeding the transition function δ with the previous state and the corresponding input symbol from s
- $q_n \in F$, meaning that q_n is an accepting state

In any other occasion, it can be said that M rejects the string. Strings, or words, that are accepted by M, are considered as positive words over its alphabet, while those that are rejected as negative words over its alphabet. The set of all positive words of M constitutes the language recognized by M, denoted as $L(M)$.

DFAs can be visualized as directed graphs, with nodes representing states and edges representing transitions. Transitions (or edges) are labelled with the corresponding input symbols capable of firing them, while, typically, accepting states are denoted with a double circle. Finally, the starting state is indicated by an incoming sourceless arrow. A sample DFA is presented in Figure 2.2, so that its structure and characteristics can be better illustrated. This DFA consists of 3 states, namely q_0, q_1, q_2, with the starting state q_0 denoted by the sourceless arrow, as explained earlier, and an alphabet $\Sigma = \{0, 1\}$ of two symbols, while q_2 constitutes the only accepting state and is indicated through a second circle, as described above. It can be easily seen that the accepted language of the considered DFA can be expressed as $0^*1(0|1)^*$, where the superscript * can be interpreted as "zero or more occurrences" of the preceding symbol.

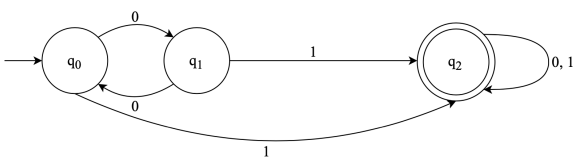


Figure 2.2: Sample DFA with 3 states, 2 symbols, and an accepted language of $0^*1(0|1)^*$

After presenting the main structure and the primary features of a simple DFA, some more complex DFA variants, the characteristics and the inference procedure of which resemble more to the models used in the core of the main pipeline of the detection system developed in the premises of this thesis, should be discussed. In particular, the nature of Probabilistic Deterministic Finite Automata (PDFA) [70], Real-Time Automata (RTA) [21], as well as their probabilistic version, Probabilistic Deterministic Real-Time Automata (PDRTA) [85], will be introduced, in an attempt to provide the needed ground knowledge upon which the multivariate models presented in Chapter 5 build. These "special" types of automata incorporate more information in their structure, like probabilities and time, which as a matter of fact renders them capable of modelling systems or processes with a more complex nature comparing to that of the systems modelled sufficiently by simple DFAs.

2.3.2. Probabilistic Deterministic Finite Automata

A Probabilistic DFA can be formally described as a quintuple $\langle Q, \Sigma, \delta, q_0, \pi \rangle$, where the first 4 terms have the same meaning as that in the case of simple DFAs, and the final term $\pi : Q \times \Sigma \rightarrow [0,1]$ denotes the *next symbol probability function*, which returns the probability of observing an event (or symbol) $\sigma \in \Sigma$ in state $q \in Q$. The probability function needs to satisfy the two following constraints: (1) $\pi(q,\sigma) = 0 \; if \; \delta(q,\sigma) = \varnothing$, where $q \in Q$ and $\sigma \in \Sigma$, and (2) $\sum_{\sigma \in \Sigma} \pi(q,\sigma) = 1, \forall q \in Q$. In simple words, these constrains state that (1) the probability of observing the symbol σ at state q is equal to 0, if the transition from state q is undeclared, or unseen, and (2) the discrete probabilities of observing each symbol of the alphabet Σ in a state q sum up to 1. Furthermore, similarly to the functionality of a simple DFA, a PDFA starts in the start state q_0 and generates strings by traversing transitions and drawing events using π. As a result, given the discrete probability distribution of all the symbols, that can be drawn from the alphabet Σ, in each state of a PDFA M, the probability $P_M(s)$ of generating a string $s = \sigma_1\sigma_2...\sigma_n$ can be expressed as

$$P_M(s) = \pi(q_0, \sigma_1) \prod_{i=1}^{n-1} \pi\big(\delta(q_{i-1}, \sigma_i), \sigma_{i+1}\big) \tag{2.1}$$

, where $\sigma_i \in \Sigma$ and $q_i \in Q$. The incorporation of probabilities related to the labelled transitions between states has rendered PDFAs quite powerful in modelling real-life systems, since the data extracted from most systems in practice are noisy and lack the formal syntax that simple DFA modelling implies.

2.3.3. Real-Time Automata

In real-time systems, the occurrence of an event is not only associated with the state in which the system was previously residing, but also with a relative time value denoting the time period that the system remained in that state before reacting to the observed event. A series of such time-dependent events can be expressed in the form of a sequence $\tau = (\sigma_1, t_1)(\sigma_2, t_2)...(\sigma_n, t_n)$ of symbols $\sigma_i \in \Sigma$ coupled with time values $t_i \in \mathbb{N}$, named as *timed string*. The time values can be modelled using natural numbers, since in practice real-time systems adhere to a finite precision of time, like milliseconds. Each time value t_i in a timed string, therefore, represents the time delay between the occurrences of the events σ_i and σ_{i-1}. It can be easily seen that (P)DFAs are not capable of modelling such timed strings since they only take into account the sequence in which events occur, rather than their exact timing.

In automaton theory, the computational model that captures such timed events is called Timed Automaton (TA) [2]. In this model, the timing conditions associated with transitions, are expressed through the use of a finite number of clocks, accompanied by a finite set of clock guards and resets on each labelled transition. Given the fact that, it has been shown that Deterministic Timed Automata cannot be identified efficiently in the limit from labeled data [86], this section will focus on a special type of TAs, that demonstrate the latter characteristic [84], named Real-Time Automata (RTA). RTAs possess solely one clock capturing the time delay between two consecutive events observed in the system, with clock guards, again, representing constraints on the time delays capable of firing a transition. Similar to the case of Finite Automata, an RTA is considered deterministic, if it does not contain two transitions with the same symbol, the same source state, and overlapping delay guards. In a more formal way, a Deterministic Real-Time Automaton (DRTA) can be defined as a quintuple $\langle Q, \Sigma, \Delta, q_0, F \rangle$, where each term has the same meaning as its corresponding one in the case of non-timed DFAs, while Δ denotes a finite set of timed transitions. A timed transition $\delta \in \Delta$ is a quadruple $\langle q, q', \sigma, [n, n'] \rangle$, where $q, q' \in Q$ are the source and destination states, $\sigma \in \Sigma$ a symbol, and $[n, n']$ with $n, n' \in \mathbb{N}$ a clock (delay) guard, and can be interpreted as follows: whenever the automaton resides in state q, observing a timed symbol (σ, t) such that $t \in [n, n']$, then the DRTA will move to the next state q'.

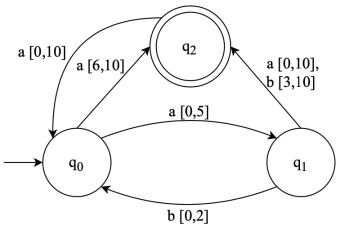


Figure 2.3: Sample DRTA with 3 states, 2 symbols, and 6 delay guards

A sample DRTA can be seen in Figure 2.3. This DRTA has 3 states, with the leftmost state being the start state and the topmost state being the final state, following the visualization conventions introduced in the case of simple DFAs. In addition, the alphabet of the depicted DRTA consists of two symbols a and b, with six delay guards included in the transitions. It is worth noting that missing transitions lead to a garbage rejecting state, not depicted in the visualized example. This DRTA accepts and rejects sequences based both on their event symbols, and on their time values. For instance, it accepts $(a,5)(b,3)$, while it rejects $(a,5)(b,2)$. Finally, it should be stated that, even though (R)TAs are not directly used in the modelling part of the developed pipeline, their inference procedure constitutes the basis of the inference procedure used in the premises of this work, thus a brief presentation of their structure is considered essential.

2.3.4. Probabilistic Deterministic Real-Time Automata

Probabilistic DRTAs introduce a probability distribution over the structure of simple DRTAs by incorporating the probability of observing a particular timed event (σ, t) given the current state q of the automaton, i.e. $P\big(O = (\sigma, t)|q\big)$. The probability distribution of the random variable O can be determined through the combination of the probability distributions of the two variables associated to each state of the PDRTA, meaning the probability distributions of the symbols and the time values, or more formally the values $P(S = \sigma|q)$ and $P(T = t|q)$. The first distribution can be determined in the same way as in the case of PDFAs, while the latter can be estimated using histograms [85], which seem like a reasonable and flexible way of estimating distributions of random variables drawn from a domain with great variance (as it is the set of possible time values). A final decision to be made when modelling the symbol and time distributions concerns the possibility of making these distributions dependent or not. As mentioned in [85], modelling these distributions in a dependent fashion would lead to a polynomial increase in the size of the model, resulting to a proportional increase in the number of data needed to sufficiently learn a model, thus an independent modelling is considered preferable.

Formally, a PDRTA can be described as a quadruple $\langle A', H, S, T\rangle$, where $A' = \langle Q, \Sigma, \Delta, q_0\rangle$ is a DRTA without final states, H is a finite set of bins (or time intervals) $[u, u']$, $u, u' \in \mathbb{N}$ comprising the aforementioned histogram, S is a finite set of symbol probability distributions, with $S_q = \big\{P(S = \sigma \mid q) \mid \sigma \in \Sigma, q \in Q\big\}$, and T is a finite set of time-bin probability distributions $T_q = \big\{P(T \in h \mid q) \mid h \in H, q \in Q\big\}$. Given the fact that the probability distribution over symbols follows the same rules as those presented in the case of PDFAs, more attention will be paid on the time-bin probability distribution. In particular, the probability that the next time value equals t given that the current state is q can be defined as

$$P(T = t \mid q) = \frac{P(T \in h \mid q)}{u' - u + 1} \tag{2.2}$$

, where $h = [u, u'] \in H$ constitutes the time bin containing the time value t, with the probabilities of the individual time points being modeled uniformly in each time bin. As a result, the probability of observing the timed event (σ, t) in state q can be calculated as $P(O = (\sigma, t) \mid q) = P(S = \sigma \mid q) \times P(T = t \mid q)$, meaning that the probability of moving from state q to state q' can be defined as

$$P(X = q' \mid q) = \sum_{\langle q, q', \sigma, [u,u']\rangle \in \Delta} \sum_{t \in [u,u']} P\big(O = (\sigma, t) \mid q\big) \tag{2.3}$$

Finally, as in the case of PDFAs, the probability of a PDRTA M generating a timed string $\tau = (\sigma_1, t_1)(\sigma_2, t_2)...(\sigma_n, t_n)$ can be defined as,

$$P_M(\tau) = \prod_{i=1}^{n} P\big(O = (\sigma_i, t_i) \mid q_{i-1},\ H,\ S,\ T\big) \tag{2.4}$$

It should be noted that the incorporation of probabilities, both in the cases of timed and non-timed automata, is mostly associated with the need to infer models able to learn only from a set of approved event sequences S^+ (positive words) originating from the system under learning (SUL), which, as a matter of fact, is much easier to collect, since such sequences constitute the normal behaviour of the system, comparing to the set of rejecting (negative words) ones S^-. More information on the identification process of a Finite Automaton can be found in the section to follow.

2.3.5. DFA Inference

Various approaches have been developed throughout the years regarding the identification of a model capable of generalizing over the observed behaviour of a system. In the case of Deterministic Finite Automata, this process is called DFA inference, with the developed approaches falling within two categories, namely *passive* and *active* learning [32]. The distinction made between these categories is founded on the different kind of interaction between the learning algorithm and the examined system during the identification process. More information on the nature and the main differences of the methods originating from these two categories can be found as follows.

Passive Learning

In passive learning a finite set of observations, produced during the execution of the system under learning, is gathered, and provided as input to the learning algorithm. These observations can consist either from both a set of positive words $S^+ \subset \Sigma^*$ and one of negative words $S^- \subset \Sigma^*$ or from solely the former one, with the goal of the learning algorithm being the derivation of the minimal model consistent to these data. In the context of DFA, "minimal" means that the inferred model should be as simple as possible in terms of the number of states and transitions, so that its ability of generalizing on unseen data would be high, while "consistent" means that the produced model should accept every positive word and reject any negative word (if any) present in the input data. In practice, labelled execution traces are not always available, especially when it comes down to "negative" ones, since they do not represent the normal behaviour of the examined system. To meet such requirements, probabilistic approaches are deployed, the inference of which has the same target as that in the case of simple DFAs, meaning the extraction of the minimal consistent model. In the case of probabilistic approaches, though, the term "minimal" addresses the need of few probabilistic parameters, while the term "consistent" refers to the creation of a model with small distance from the input sample distribution. It has been proved that the problem of identifying a DFA with the minimal number of states, using passive learning methods, is NP-complete [29]. Nevertheless, multiple passive learning approaches have developed throughout the years, ranging from the *TB* [5] and *Traxbar* [47] algorithms, to the more recent *RPNI* [66] and *Blue-Fringe evidence-driven state-merging* [48] algorithms. In fact, the latter algorithm constitutes the basis for most modern passive learning approaches, like the one used in the premises of this thesis, with analytical information on this approach provided in Chapter 5.

Active Learning

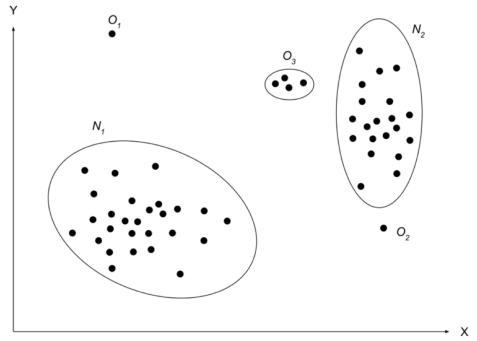
In active learning the inference process may also start from a given set of system-generated observations, yet the learning algorithm will continue collecting evidence on the behaviour of the system by interacting with it. In particular, the notion of an oracle or an expert is introduced, the purpose of which is to answer queries provided by the learning algorithm regarding its current hypothesis on the structure of the system under learning, and in that way guide the learning procedure accordingly. In the field of automata learning the most well-known algorithm operating under the concept of active learning is Angluin's L^* algorithm [4], which introduced the MAT (Minimally Adequate Teacher) framework to infer models with a number of queries polynomial to the number of states of the state machine to be learnt. In this framework, there are two classes of queries to be asked to the teacher, the Membership and the Equivalence queries. Through Membership queries, the learner can ask the teacher whether an input word w belongs to the language L of the automaton and receive a binary answer (yes or no) on that question. Through Equivalence queries, the learner presents a hypothesis automaton H (the automaton that in its belief models sufficiently the system) and the teacher confirms or disproves the hypothesis that L is the language of H. In case of confirmation the algorithm terminates, while in the opposite case a counterexample is provided to the learner as evidence that L is not the language of H, and the learning process continues.

2.4. Graphs

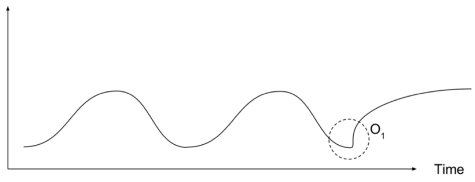
As it was seen earlier, finite automata can be represented in a straightforward way as graphs, if their states are perceived as nodes and their transitions as edges. In particular, the graphs capable of capturing the nature and structure of finite automata are called Directed Labelled graphs, meaning that each edge is directed from a source to a destination node, like transitions are directed from a source to a target state, and possesses a label describing the conditions under which the edge can be crossed. More formally, a Directed Labelled Graph is a triple $\langle N, L, A \rangle$, where N is the set of nodes, L is the set of labels, and $E \subseteq N \times N \times L$ is a set of edges. It can be easily seen that generating a string from a Finite Automaton is equivalent to following a path in a directed graph. The labels on the edges crossed along the path correspond to the symbols comprising the generated string. Similarly, many operations on finite automata can be easily mapped to equivalent operations on directed graphs, which as a matter of fact proves the usefulness of the graphical representation of automata.

2.5. Anomaly Detection

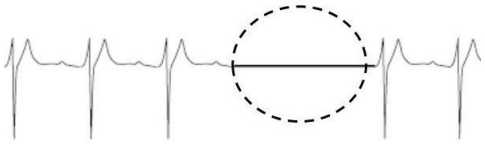
In a group of data, those data points that deviate to such an extent from other points, so as to inflame suspicion that their generation mechanism differs from that of the majority of the data [36], can be characterized as anomalies, or outliers. In other words, the anomalous instances of a dataset tend to display patterns and characteristics significantly different to other data instances, that are considered normal, or benign. In the anomaly detection literature [15], three main categories of anomalies are addressed: (1) *point anomalies*, which refer to data points that can be observed as anomalous against other data points, (2) *contextual anomalies*, which concern data points that, when perceived from a global point of view on the dataset seem as normal, yet in some specific context are identified as anomalous, and (3) *collective anomalies*, which refer to data points that can be considered anomalous only when they occur in a collection. The last two categories are applicable mostly to data with an implicit or explicit notion of ordering (spatial, sequential, etc). In such data the context can be easily defined trough the notion of ordering to which they adhere (time, coordinates, etc.), while a collection of data can be defined as a sequence of neighbouring data points with respect to the implied notion of ordering. A visual illustration of all three categories can be found in Figure 2.4.



(a) Illustration of point anomalies (denoted with O) in a 2-D dataset



(b) A contextual anomaly example (denoted again with O)



(c) A collective anomaly example (denoted by the dotted circle)

Figure 2.4: Illustration of the different categories of anomalies

The detection of anomalies is considered of high relevance in the provision of critical actionable information in a wide range of application domains. For instance, the detection of an unusual computer network traffic pattern could indicate a cyber attack or an event of unauthorized access, while anomalous behaviours in credit card transactions could be related to fraudulent credit card use. As a result, the development of robust anomaly detection systems characterized by a high detection rate, and, in parallel, a significantly low number of false alarms, so that the manual inspection of the identified anomalies will remain feasible, con-

stitutes an inherent requirement for the orderly operation of such systems. At first glance, the detection of anomalies, potentially existing in a dataset, might seem like a simple task, yet, in practice, it can be quite challenging. First of all, defining or modelling normal regions in the space of the data is a particularly difficult task, since, in many cases, the boundaries between anomalies and normal data are blurred. On top of that, if the anomalous instances in a dataset are considered of malicious origin, as in the cases mentioned above, the attackers tend to adapt the instances that they generate so that they seem more similar to the benign ones. Finally, in domains that the characteristics of the data change dynamically, the notion of normality constantly changes, which as a matter of fact renders the development of a robust anomaly detection system a complex undertaking.

Anomaly detection can be perceived as a learning problem, thus the approaches, aiming to tackle it, can be classified as the conventional problems in the field of Machine Learning into three categories, namely supervised, semi-supervised, and unsupervised. In the premises of this work, an unsupervised approach based on the normal data points of the facing problem is developed. In particular, the selected approach involves the construction of profiles of the normal instances of the given dataset, and the utilization of these profiles to identify instances that deviate significantly from them. Of course, such approaches are accompanied by both advantages and disadvantages. The main advantages of methods following this direction can be addressed by the fact that, during modelling, they require solely examples of the normal behaviour of the system, the retrieval of which is considered, in most cases, an easy task. In addition, the process of learning from normal data in order to recognize abnormal ones fits the natural procedure that most humans adopt to identify odd or rare phenomenons in real life. Nonetheless, a major drawback can be encountered in such approaches. The anomaly detectors developed are optimized to identify normal instances, rather than actually detecting anomalies, which could lead to either highly sensitive models (too many false alarms), in case the normal data samples on which the model was learnt are not representative enough, or excessively generic models, with low detection rates.

Another interesting direction in the field of anomaly detection is encountered within the semi-supervised context, according to which the abnormal data instances are leveraged during training. The methods, residing within this category, follow the tactic of creating behavioural profiles for each class of anomalous instances known in the premises of the facing problem. These anomalous profiles are then compared against the data instances, and in case of a match an anomaly is identified. The obvious shortcoming of such approaches is that they need sufficiently representative malicious samples for the anomalous profiles to be created, which in most cases are not available. In addition, if a type of anomaly, not existing in the training process of the model, appears, then it is highly likely that no anomalous profile will match it, thus leading to that anomaly being undetected. In a dynamic environment, where anomalies are the product of malicious behaviour, such behaviour is significantly unwanted, since it would result to low detection rates. On the other hand, models extracted from malicious instances tend to produce quite low false alarms, since they are optimized to detect specific types of anomalies. In case that the majority of anomalies found in the data resemble to the ones used during the training procedure, this low false alarm rate can be accompanied by a proportionally high detection rate.

In the next two sections, the main approaches included on the designed detection pipeline are discussed. At this point it should be noted that, approaches belonging to different learning paradigms were evaluated in this work, so that the anomaly identification problem could be tackled from multiple directions. Briefly, Local Outlier Factor (LOF) [13] can be characterized as a density-based approach, Isolation Forest [54] as a classifier-based one, while the Multivariate Gaussian approach as a statistical one. More information on the functionality and the characteristics of these methods can be found as follows.

2.5.1. Local Outlier Factor

Local Outlier Factor builds upon the idea of assigning a *"degree of outlierness"* to each object in a given dataset by considering how isolated that object is with respect to its surrounding neighborhood. The strategy followed for quantifying the "outlierness" of an object can be perceived as a variant of density-based clustering. In more detail, the calculation of the LOF of an object is based on the number of nearest neighbours used when defining its local neighbourhood. The notion of the local neighbourhood of each object is leveraged to deal with objects that are outlying relative to their close neighbours, with their outlying nature not being easily identifiable in a global scale. In fact, both the cases of contextual and collective anomalies, presented earlier in this work, can be considered special cases of local outliers. An example of a dataset with both global and local outliers can be seen in Figure 2.5.

Given the data distribution presented above, it is clear that there are two clusters of data, a dense one

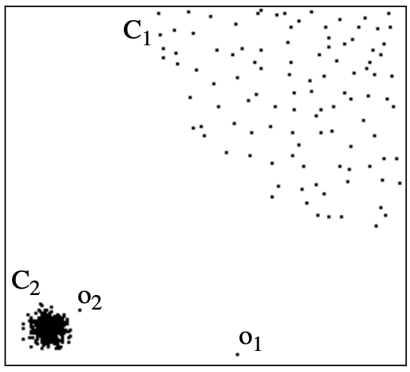


Figure 2.5: Sample 2-D data scatter plot with both global (o_1) and local (o_2) outliers

(C_2), and a sparser one (C_1), as well as two more data samples, o_1 and o_2. In the notion of local outliers, both objects o_1 and o_2 should be identified as outliers. Nevertheless, when viewing the data from a global point of view solely object o_1 would be considered as an outlying point due to the sparse nature of cluster C_1. To deal with such shortcomings, two main terms, namely the *k-distance* and the *reachability distance* of an object p, are introduced in the LOF algorithm. In more detail, the *k-distance* of an object p is defined as the distance between this object and one of each neighbouring objects, such that there are at most $k-1$ objects with lower and at least k objects with lower or equal distance to object p. As far as the *reachability distance* of an object p with respect to an object o is concerned, it can be expressed as the maximum of the actual distance between these objects and the *k-distance* of o. Intuitively, the former distance metric is used to introduce the notion of the *k-distance* neighbourhood of an object, while the latter one is used to assign the same value to the distance metric between objects residing in the same neighbourhood. Furthermore, as it was mentioned above, LOF constitutes a variant of density-based clustering, thus a notion of local density shall be defined. In particular, the local reachability density (*lrd*) of an object p can be formally defined as follows:

$$lrd_{MinPts}(p) = 1/\left(\frac{\sum_{o\in N_{MinPts}(p)} reach\_dist_{MinPts}(p,o)}{|N_{MinPts}(p)|}\right) \tag{2.5}$$

, where $MinPts$ denotes the minimum number of points used to define the neighbourhood of an object, and $N_{MinPts}(p)$ the *MinPts-neighbourhood* of object p. Practically, the local reachability density of an object p constitutes the inverse of the mean reachability distance based on the *MinPts-neighbourhood* of p. All things considered, the outlier score of an object p in the context of LOF can be calculated as

$$LOF_{MinPts}(p) = \frac{\sum_{o\in N_{MinPts}(p)} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPts}(p)|} \tag{2.6}$$

Given equation 2.6, the outlier factor of an object p is expressed as the mean ratio of the local reachability density of p and those of its *MinPts-nearest* neighbors. Intuitively, this means that LOF measures the local deviation of density of a given object p with respect to its neighbors. It can be easily seen that the lower local reachability density of p is, and the higher the local reachability densities of its *MinPts-nearest* neighbors are, the higher is the outlier score assigned to p.

2.5.2. Isolation Forest

Isolation Forest constitutes a tree-based method for anomaly (or outlier) detection, aiming to isolate anomalous points in a dataset, rather than profiling the normal ones. The intuition behind Isolation Forest is based on two quantitative properties of anomalous instances: (1) they constitute the minority class in a given dataset, and (2) they demonstrate attributes with significantly different values from those of normal instances. As a result, these "few and different" anomalous instances can be isolated more easily than normal data points. To that end, a tree structure is leveraged to isolate every single point in the dataset, with the expectation that anomalies will be isolated closer to the root of the tree, while normal points will be isolated much deeper inside the tree. Such trees are called Isolation Trees, and in the context of this approach an ensemble of Isolation Trees is built for a given dataset, with instances with short average path lengths being identified as anomalies.

The core functionality of the Isolation Forest algorithm is based on the recursive random partitioning of data instances until all instances are isolated. In more detail, partitions are generated by randomly selecting an attribute of the data and a split value between the maximum and minimum values of that selected attribute. Since the recursive partitioning undertaken can be represented by a tree structure, the number of partitions needed to isolate an instance can be considered equivalent to the path length from the root to the leaf of the tree, in which this instance resides. Given the fact that, the partitions are generated in a random fashion, each Isolation tree is constructed under a different set of partitions. Thus, averaging the path length of each instance seems the natural choice to calculate the expected isolation score of that instance. The lower that score is, the higher the possibility of the examined data sample being an anomaly is.

The parameters that primarily have an effect on the quality of the results produced from this model are the number of isolation trees to be used as base estimators in the ensemble, and the number of samples to draw from the input data to feed each tree. Since each isolation tree acts as an estimator of the anomaly score of an instance, the higher the number of estimators, the more accurate the expected path length of each instance will be. In practice, the estimated path lengths converge after some specific number of trees is reached. In fact, the authors of the algorithm suggested a number of trees equal to 100 as a value that produces stable results [54]. Before addressing the second parameter, the value of subsampling in the premises of this algorithm should be highlighted. Subsampling is used as a measure against swamping and masking, which constitute significant problems in the field of anomaly detection. The first term refers to the incorrect identification of normal instances as anomalies due to close proximity between normal and anomalous instances, while the second term refers to the concealment of the presence of anomalies due to their high cardinality in the input data. Since both issues originate from the inclusion of too many data in the anomaly detection task, subsampling is used to both regulate the data size and enable the creation of tree "specializing" on different set of anomalies.

2.5.3. Multivariate Gaussian Kernel Density Estimation

The rationale behind the Multivariate Gaussian Kernel Density Estimation approach is based on the assumption that the distribution of the provided data instances can be estimated through the application of Gaussian kernels (or windows) on those instances. After estimating the probability density function (pdf) of the data, a threshold value e can be set through experimentation, so that, if the pdf value of a data instance is lower than e, this instance is considered as an anomaly. Kernel-density (KDE) constitutes a non-parametric approach to estimate the probability density function of a random variable, and can be perceived as an improved generalisation of histogram density estimation. Originally, such approaches were utilized in the case of univariate data [67, 71], and later were developed to also address the case of multivariate data. Intuitively, the goal of density estimation is, given a finite sample of data points, to infer the underlying probability density function for any point on the attributes' space of the data, including regions where no data were observed. The contribution of each data point is smoothed out into a region of space surrounding it, defined by the use of a window. By aggregating the smoothed contributions of each data point an overall estimate of the structure of the data and its density function can be achieved.

In a more formal way, given a set of N d-variate data instances $x_1, x_2, ..., x_N$ drawn from some distribution described by the density function f, the kernel density estimate of that function can be calculated as:

$$\hat{f}_H(x) = \frac{1}{N}\sum_{i=1}^{N} K_H(x - x_i) \tag{2.7}$$

, where H is a $d \times d$ positive semi-definite matrix, called the bandwidth, K is a symmetric multivariate density, called the kernel function, with $K_H(x) = |H|^{-1/2}K(H^{-1/2}x)$. When the multivariate Gaussian kernel function is used, the last relation obtains the form $K_H(x) = (2\pi)^{-d/2}|H|^{-1/2}e^{-\frac{1}{2}x^T H^{-1}x}$, where the bandwidth H acts as the covariance matrix. The choice of the bandwidth H influences the estimate obtained from the KDE to a great extent, thus its selection shall be carefully undertaken. In the premises of this work, Scott's rule of thumb [74] is utilized towards that end, meaning that $H_{ij} = 0,\ if\ i \neq j$, and $\sqrt{H_{ii}} = N^{-1/(d+4)}\sigma_i$, with σ_i denoting the standard deviation of the *i-th* attribute.

2.6. Evaluation Metrics

The facing problem conforms under the general principle of machine learning problems, in the essence that the main goal is to achieve the objective of classifying a number of objects (network flows), given as an input, into two categories (malicious and benign). Therefore, the most frequently used machine learning evaluation

metrics, meaning *accuracy*, *precision*, and *recall*, are leveraged to evaluate the performance of the designed system. In order to better understand the meaning of each metric, the notion of the *confusion matrix* and its content need to be firstly introduced. A confusion matrix, also known as error matrix, is a specific type of table allowing the visualization of the performance of a machine learning algorithm, with each row representing the instances in an actual class, and each column the instances in a predicted class (or vice versa). In the case of binary classification, where the facing problem belongs, there are two classes, the positive and the negative one. Thus, the instances that are correctly classified as positive are called *True Positives*, while the ones that are wrongly classified as positive (meaning that their real label is negative) are called *False Positives*. Correspondingly, the instances, which are correctly classified as negative are called *True Negatives* and the wrongly classified as negative (meaning that their real label is positive) are called *False Negatives*. With respect to these terms, Figure 2.6 depicts the visualization of a confusion matrix.

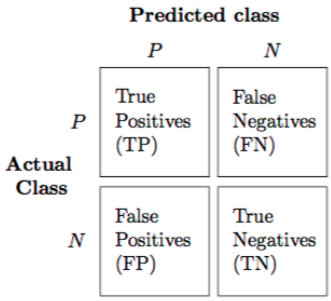


Figure 2.6: Sample confusion matrix

Building upon the insight provided by the confusion matrix, the formulas of the evaluation metrics utilized in the premises of this thesis can be seen as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.8}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.9}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.10}$$

In more detail, *accuracy* measures the fraction of correctly classified (predicted) instances, with *precision* denoting the fraction of actual positive instances out of the predicted positive instances, and *recall* measuring the fraction of positive instances identified correctly. The latter two metrics are considered of great importance in the context of anomaly detection, since they provide significant insight on the ability of the designed systems to not only identify the anomalies, but also to raise as few false alarms as possible. In particular, if anomalies are considered the positive class in an anomaly detection task, and given the fact that anomalies constitute typically the minority class among the available instances, accuracy could attain high values even if few (or none) anomalies (positive instances) were identified, which as a matter of fact explains the usefulness of recall in system evaluation. In the opposite scenario, that an anomaly detection system identifies many instances as anomalous (despite their actual nature) and the anomalies represent a considerable portion of the dataset, both accuracy and recall would attain high values, while in practice there would be many false alarms. Of course the development of a system so sensitive to anomalies would render the manual examination of the alarms an infeasible task. Such cases highlight the importance of high precision as one of the core characteristics of a well-designed anomaly detection system. All things considered, the desired performance of anomaly detection systems is reflected by the achievement of high values in all these metrics.

3

Related Work

In this chapter various approaches relevant to the two primary fields that this thesis touches upon, namely anomaly detection and state machine inference, are presented and discussed. The primary purpose of this discussion is to help the reader familiarize with the research relevant to the development of the current work, as well as present in a concise, yet enlightening way the undergone progress in the aforementioned two fields throughout the years. Firstly, works spanning the field of anomaly detection are briefly presented and discussed. In particular, a more generic view of anomaly detection methods is initially presented, followed by works purely drawn from the field of network traffic, with the focus being shed upon flow-based methods, since, such an approach has been also adopted in the premises of this thesis. These techniques are distributed in different sections according to the data analysis concept lying in their core. Finally, important works in the field of state machine inference are presented, so that the broad learning capabilities of such approaches can be highlighted.

3.1. Standard Anomaly Detection Techniques

Throughout the years, researchers have adopted concepts from multiple disciplines, such as classification, nearest neighbour, clustering, statistics, and spectral theory, to deal with anomaly detection tasks [15]. Of course, this section will not demonstrate the full extent of such approaches, since such an undertaking lies beyond the scope of this thesis, rather it will briefly illustrate the primary key assumptions, which the techniques of each category utilize to identify anomalies, and provide a representative sample of some influential approaches. Thus, most of the approaches presented in this section might not be considered currently as the state-of-the-art, yet they laid the foundation for the majority of anomaly detection systems used nowadays.

3.1.1. Classification Techniques

This category includes techniques mostly operating under the typical supervised machine learning assumption that, given a representative set of samples of both normal and abnormal instances into a defined feature space, a classifier can be learnt to differentiate between samples originating from different categories. Apart from the typical binary setting, these approaches have been extended to include one-class and multi-class data samples. Anomaly detection techniques of the first category assume primarily that only normal instances are included in the training phase, and try to learn discriminative boundaries around these normal instances. A typical example of such technique can be found in [3], where two modified one-class SVM [73] based approaches are introduced to increase the robustness of one-class SVMs and render them less sensitive to outliers by regulating the extent in which outlying samples can contribute to the decision boundary. As far as the second category is concerned, multi-class classification based anomaly detection techniques, which typically include also the binary classification methods, operate under the assumption that the training data contain instances from multiple normal classes, thus the classifier needs to distinguish between each normal class against the other classes. Similarly to the case of one-class SVM, a modified version of SVM, called Robust Support Vector Machines (RSVM) [77], and aiming to solve the over-fitting problem produced by the presence of outliers in the training dataset, has been used in [41] to differentiate between normal usage and intrusive profiles of computer programs.

Of course, the works belonging in this category span also various other machine learning concepts, rang-

ing from complex neural networks to simpler rule based approaches. In [58], stacked Long Short Term Memory (LSTM) networks were utilized for anomaly detection in time series. In particular, the LSTM network is trained on normal data, with the produced prediction errors over a number of time steps being modeled by a multivariate Gaussian distribution. Subsequently, the probability density function of this fitted error distribution is used to assess the likelihood of anomalous behaviour. As far as rule-based approaches are concerned, the main intuition lies on the fact that rules can be learnt about the normal behaviour of the examined data, thus data instances not conforming to those rules can be identified as anomalies. This idea might seem simplistic, yet Isolation Forest, an algorithm originating from this category, constitutes, as explained earlier in this thesis, one of the most well-known anomaly detection techniques delivering great results in various settings.

3.1.2. Nearest Neighbour Techniques

The main assumption in such techniques lies on the fact that normal instances tend to occur in dense data neighbourhoods, while anomalous instances are placed far from their closest neighbours. Nearest Neighbour based approaches typically require a distance (dissimilarity) or a similarity measure defined on a pair of instances, and can be broadly classified into methods that use such metrics to quantify the (dis)similarity between a data instance and its k-th nearest neighbours as the anomaly score, and those that utilize the relative density of the neighborhood of each data instance to assign such an anomaly score. In fact, LOF, one of the outlier detection algorithms presented in Chapter 2, constitutes one prominent example of density based techniques. As far as the k-th nearest neighbours approaches are concerned, in [50], a k-NN classifier was used to identify intrusions in computer system calls, by comparing the average distance of each data instance to its k nearest neighbours and identifying as anomalies those instances with an average distance below a user-defined threshold.

3.1.3. Clustering Techniques

Techniques from this category are mainly founded on one of the following three assumptions regarding the behaviour of normal and anomalous data:

1. Normal data instances can be grouped in clusters, while this is not the case for anomalous instances.
2. Normal data instances tend to lie close to their nearest cluster centroid, while anomalies lie far from their closest centroid.
3. Normal data instances tend to form large dense clusters, while anomalous instances are mostly encountered in small or sparse clusters.

Before presenting a sample of works belonging in this category, the distinction between clustering and nearest neighbour techniques shall be discussed. At first, it seems hard to differentiate between approaches belonging to one of these categories, yet there is a key difference between them. This difference can be expressed by the fact that clustering techniques consider typically each data instance with respect to the cluster in which it resides, while nearest neighbour techniques evaluate each instance within the context of its local neighbourhood. Having addressed the former, the first clustering category includes clustering techniques that do not assign all available data instances to a specific cluster, rather they treat such non-clustered instances as outliers. DBSCAN [24], as well as its improved hierarchical version, HDBSCAN [61], constitute two representative approaches meeting the previous condition. In the second category, approaches involving the use of k-means clustering and Self-Organizing Maps (SOMs) [44] to first cluster the provided data and then use the distance of each data point to its closest centroid to measure its degree of normality are mostly considered. The main disadvantage of the approaches belonging in the first two categories may arise in case that the anomalies in a dataset form clusters by themselves. In such scenarios, approaches from the third category can be adopted. Cluster-Based Local Outlier Factor (CBLOF) [37] constitutes a paradigm of such techniques, incorporating the density-related outlier detection capabilities of LOF into a clustering mechanism.

3.1.4. Statistical Techniques

The main intuition supporting the foundation of techniques in this category lies on the fact that, if a stochastic model is used to capture the behaviour of normal instances in a dataset, then anomalies should occur in low probability regions of the fitted model. Typically, such approaches fit a statistical model on the normal instances of a dataset, and use some statistical inference test to identify data instances not belonging to the

inferred model as anomalous. As in standard machine learning tasks, statistical approaches include both parametric and non-parametric methods, with the main distinction between them lying on the fact that the first ones assume the knowledge of the underlying distribution of the observed data.

A typical and simple anomaly detection approach followed in the parametric setting assumes that the observed data are generated from a Gaussian distribution with the parameters of that distribution been estimated using Maximum Likelihood Estimates (MLE) [15]. After estimating these parameters, measures like the distance from the distribution mean and the Inter Quantile Range (IQR) are used to determine if a given data instance is anomalous or not. In a similar manner, complex statistical tests have been applied towards the goal of anomaly detection [90], while in case that a temporal component is associated to the provided data, regression models, like ARIMA (Autoregressive Integrated Moving Average) [11] and ARMA (Autoregressive Moving Average) [35], are employed. The intuition, on which regressive anomaly detection approaches are founded, can be roughly expressed as follows: Given a model fitted on a set of past data values, the feature data values can be predicted through the regression technique applied. Then, the data points that produce a regression error above some user-defined threshold can be identified as anomalies. Finally, the mixture of parametric distributions is also employed to model either the normal and anomalous instances as different parametric distributions, or solely the normal instances, and use the learnt distributions towards the identification of anomalies.

In the non-parametric setting, histogram-based approaches and kernel density estimation constitute the most frequently used concepts in the development of anomaly detection pipelines. Histogram-based approaches attempt to firstly model the distributions of the data features through the use of binning methods, and then assign an anomaly score according to the bin heights in which the features of a data instance fall. An important consideration in the case of histogram based approaches regards the size of the bins, since too small bins could lead to an overfitting model with a high false alarm rate, while too broad bins could lead to an underfitting model falsely labelling anomalous behavior as benign. As far as kernel density estimation is concerned, as it was discussed in Chapter 2, Parzen windows estimation [67] constitutes one of the most well known methods falling under this category, while the intuition behind such techniques is similar to that of parametric methods, with the main difference identified in the way that probability density estimation is conducted.

3.1.5. Spectral Techniques

Spectral anomaly detection techniques are founded on the assumption that, if the provided data instances are embedded into a lower dimensional subspace, normal and abnormal instances can obtain a distinctively different representation. As a result, such techniques search for such subspace mappings (projections, embeddings, etc.), which can render the identification of anomalous instances more profound than working with the data in the original feature space. The subspace method is most frequently implemented with the application of Principal Component Analysis (PCA) [23] on the data. The general setting of this approach can be explained as follows: Given a set of data samples with p features, it is assumed that there is a underlying correlation between these features, meaning that the typical variation of the entire feature set can be expressed as a linear combination of less than p variables. By applying PCA on the dataset, a partition of the feature space into two orthogonal subspaces, namely the principal-component (normal) subspace, and the residual subspace, can be achieved. Given the fact that the normal subspace is assumed to describe the normal variation of the data, if each data instance is decomposed into its normal and residual components, unusually large values in the residual components could act as indicators of the degree to which that particular instance is anomalous. Such an approach was utilized in [46] towards the goal of anomaly detection in large-scale network flow captures, by applying the subspace method on the entropy of the distributions of simple traffic features, like IP addresses and port numbers.

3.2. Anomaly Detection in Network Traffic

In the domain of computer networks, anomaly detection methods can be broadly classified into two complementary categories: signature-based vs. profile-based techniques [27]. The methods of the first category base their functionality on the use of prior knowledge about the characteristics associated with each kind of anomaly, so that potential incidents can be matched to there previously known patterns. The profile-based systems, on the other hand, aim in the development of a behavioural profile representative of the normal traffic and identify as anomalies the events not conforming to such profile. Despite the fact that the latter approaches tend to produce more false alarms, since it is difficult to model efficiently every aspect of

the normal traffic behaviour, such systems are considered more promising, since they can be used towards the detection of previously unknown anomalies. In the sections to follow, a taxonomy of various interesting network-related anomaly detection approaches is presented. The main categories identified are the following:

- **Randomized approaches**: Systems residing in this category incorporate some notion of randomness in the detection procedure, mostly by utilizing random aggregation of the network samples available.
- **Clustering approaches**: This category is populated by approaches that attempt to cluster the examined network data in groups with similar underlying network behaviour, and use these clusters to detect and group anomalies.
- **Combined Machine Learning approaches**: This category is comprised of the systems that employ various ML techniques (sometimes clustering techniques are included) in a combined fashion to detect anomalies. Most approaches of this category need labelled data of both classes in order to operate properly.
- **State Machine Learning approaches**: Finally, the systems of this category attempt to leverage the underlying temporal patterns of the behaviour of the examined network samples by inferring state machines, so that sequential models of this behaviour can be extracted.

Systems originating from all these categories are discussed in brief in the following sections.

3.2.1. Randomized Approaches

In [49], a technique, based on the use of random aggregations (sketches) of IP flows collected from two backbone networks, is developed as a solution towards the identification of anomalies on a flow level. This technique, called *Defeat*, is founded on the insight that sketching the global traffic preserves the normal variation of the flow traffic, as well as most of the residual subspace. To this end, *Defeat*, initially, constructs multiple sketches of the entropy of the empirical distribution of various flow-related features (source and destination IP addresses and port numbers), and, subsequently, applies the subspace method to these sketches, aiming to identify time points at which unusual traffic distributions are detected and in parallel identify the "infected" IP flows. The main intuition behind this approach is based on the fact that, by using random projections, anomalous patterns hidden under some aggregations can be exposed under others, which is not the case when fixed mappings, like the origin-destination flow aggregation, are utilized. On top of that, the authors suggest that, since anomalies are shuffled randomly across different sketch entries, approximate agreement among sketches can be used for robust anomaly detection, by developing a voting scheme to be applied on the results of each sketch to reduce the false alarms.

In a similar manner, the researchers in [20] combine sketches with multiscale non-Gaussian marginal distribution modeling to develop a profile-based anomaly detection approach towards the real-time detection of both short-lived and long-lasting low-intensity anomalies in a series of network packet captures. In more detail, the proposed method is based on the extraction of a statistical profile capable of characterizing anomalies, by firstly dividing the traffic data into sketches using the source or destination IP address as a hashing key, and secondly by retrieving the shape parameter of the marginal distributions of the traffic for each sketch at multiple scale levels through non-Gaussian modelling. Subsequently, the Mahalanobis distance is leveraged to draw comparisons between the evaluated sketch distributions and some reference distributions derived by average behaviors and typical variabilities estimated on the traffic. Despite the use of packet captures, deep inspection is not conducted, since solely the source and destination IP addresses, the port numbers, and the packet arrival time are used during modelling.

Another interesting approach was presented in [12] through the introduction of association rules theory in the context of anomaly detection. Firstly, the authors used multiple histogram-based detectors applied on different flow-related features (e.g. source and destination IP addresses and port numbers, as well as the number of transmitted packets) to acquire a divergent view of the examined network traffic and extract various candidate sets of anomalous flows. The flows were processed in fixed time intervals, and for each interval multiple randomized histogram clones are used to estimate the distribution of each feature across time. Subsequently, the Kullback-Leibler (KL) distance [45] is utilized to detect significant differences between the distribution estimated by each clone of the current interval and the corresponding distribution from the previous interval. In that way, histogram bins with significant difference with the reference bins are detected and the feature values residing in them are extracted as potentially anomalous. In order to avoid

high false alarm rates, due to the randomized placement of feature values in each bin, only the intersection of the feature values extracted by each clone is taken into account. Finally, a modified version of the Apriori algorithm [1], one of the most influential algorithms in the field of association rule mining, is used to create the general profiles of the anomalous flows. This work shows high potential since it combines the detection of anomalous flows from a real backbone network, and some partial interpretability on the results, yet the evaluation procedure seems a bit unnatural since the identification of anomalies is conducted on an interval level.

3.2.2. Clustering Approaches

The researchers in [6] proposed a clustering approach to identify and group malware samples with similar behavioural profiles, focusing, according to their statements, in the scalability of their system. Their system operates on traces of system calls generated by the execution of some malicious binary, rather than network flows, which as a matter of fact sets their research apart from the majority of works presented in this chapter, yet the idea of extracting behavioural profiles from a set of data points, and clustering these profiles to identify similar behaviours in the data, is used extensively in the anomaly detection literature. Briefly, behavioural profiles are created for each executed program by abstracting its associated system calls through the use of a dynamic malware analysis system of their development, called *ANUBIS*, and Locality Sensitive Hashing (LSH) [42] is employed to cluster efficiently these behavioural profiles in such a way that samples exhibiting similar behavior are combined in the same cluster.

Under a similar principle, the authors of [69] introduced a network-level behavioural clustering approach, based on single-linkage hierarchical clustering, and operating on the malicious HTTP traffic generated by various malware samples. Initially, some simple statistical features (like the total number of HTTP requests the malware generated, the number of GET and POST requests, the average length of the URLs, etc.) are extracted from the HTTP traffic and coarse-grained clusters of malware samples are created from these feature representations. Subsequently, these coarse-grained clusters are split into fine-grained ones by examining the structural similarity of the HTTP queries used by the malware samples in each cluster. Finally, given the set of fine-grained malware clusters created, the cluster centroids of each one of them is calculated and used as network signatures that summarize the HTTP traffic generated by the samples in the cluster. In the end, fine-grained clusters with similar centroids are merged for the final partitioning of the original malware set to be obtained.

In [10], one of the most recent anomaly detection approaches operating on NetFlow records is presented. The authors of that work introduce a bot detection technique, named *BotFP*, which aims to characterize hosts' behavior in a given network using attribute frequency distribution signatures extracted from each host. As in many anomaly detection approaches, the operation of the technique proposed in that work is divided between a training and a testing phase, with the available NetFlow records in both phases grouped according to their source IP address, so that host level detection can be performed. After retrieving the flows associated with each host, a set of basic NetFlow attributes, like the source and destination port numbers and the destination IP address, are used to extract a frequency distribution signature for each host. In particular, the flows linked with each host are further separated according to the communication protocol used, and the concatenation of the normalized frequency distributions of the aforementioned attributes of each protocol-based category of flows is used to construct each host's signature. Subsequently, the well-known clustering algorithm DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [24] is used to cluster the hosts composing the training set according to their signatures, and a label of benignity is assigned to each cluster based on the existence of at least one malicious host in its premises. Finally, each host in the testing phase is classified according to the label of the closest signature-based cluster derived during training. It should be mentioned that this technique is evaluated on one of the datasets used also in the premises of this thesis. As a result, it was considered beneficial to compare the results attained in [10] with the ones achieved in the current work over this common dataset, with this comparison presented in Chapter 7.

3.2.3. Combined Machine Learning Approaches

The authors of [9] present a botnet detection system, named *DISCLOSURE*, aiming to distinguish C&C channels from benign client-server communication using features extracted from NetFlow records. In particular, features related to the flow sizes, client access patterns, and the temporal behaviour of flows are taken into consideration. Furthermore, the authors address the false alarm problem, commonly encountered in anomaly detection techniques operating on NetFlow data, by incorporating a number of external reputation scores into the detection process. Their system is evaluated on NetFlow data collected from a university

network, and a Tier 1 ISP, where sampling was used to deal with the massive number of flows. In brief, the proposed system architecture was split in two phases, the detection models' generation and the real-time detection. The first phase constituted the training phase of the system, in which features from the aforementioned categories are extracted using some basic NetFlow attributes, like the source and destination IP addresses and port numbers, the start and finish timestamps, and the bytes and packets transferred. This feature representation of the NetFlow data is used as input to a group of supervised machine learning algorithms (decision tree, support vector, and random forest classifiers) for the detection models to be developed. Finally, in the detection phase, a FP reduction module is implemented, leveraging the information obtained by three public services, providing reports about a wide range of malicious activities on the Internet, to create a reputation-based set of weights for each server encountered in the test set.

The system presented in [79] is bot-oriented too. The functionality of this approach is founded on the observation that C&C connections associated with a certain bot family tend to demonstrate specific regular patterns, thus the proposed system is trained on groups of NetFlow traffic produced by a set of bot families, so that a detection model from each bot activity can be extracted. In more detail, during the training phase, the system extracts the statistical properties able to characterize the C&C traffic of different bot families, and leverages these properties to build models capable of identifying similar traffic. The examined flows are aggregated in chronologically ordered sequences of connections between two IP addresses on a specific destination port, and five statistical features, namely the average time between the start times of two subsequent flows in each connection, the average duration of a connection, the average number of bytes transferred to and from the source IP, and a Fourier Transformation over the flow start times in each connection, are extracted. These statistical feature values are individually clustered according to the CLUES (CLUstEring based on local Shrinking) algorithm [89], so that a model out of a set of five clusters can be derived for each bot family. Finally, the identification part is conducted through the application of feature matching between the connections of the test set and the modelled bot clusters.

Remaining in the field of botnet detection, the researchers in [28] developed two methods, both operating on NetFlow data, towards that goal. The first one, named *CAMNEP* (Cooperative Adaptive Mechanism for NEtwork Protection), constitutes a Network Behaviour Analysis system performing detection in three abstraction layers. Initially, eight different anomaly detectors, drawn from the relevant literature, are applied on different sets of NetFlow features, in an attempt to search for an anomaly from different perspectives towards a two-fold purpose. These detectors aim to both extract meaningful features associated with each NetFlow and assign an anomaly score to each of the examined NetFlows. Subsequently, their output is aggregated into events with the use of different statistical functions and passed to the trust models layer. The role of the trust models is to cluster Netflows with similar behavioural patterns, as they are extracted by the anomaly detectors. Each cluster centroid acts as the trustee for each model and is used to measure the trustfulness of each NetFlow residing in its vicinity. The trust value of each centroid is updated by new artificial flows added in the first layer through the introduction of malicious and legitimate challenges attempting to "fool" the aggregators, which are the output layer of the system. This layer produces a combined output integrating the separate "opinions" of the anomaly detectors as they are expressed by the trust models. This combined output constitutes the final anomaly score of each NetFlow.

The second method introduced in [28], called *BClus*, can be characterized as a behavioural-based botnet detection approach, and resembles primarily a clustering mechanism. The purpose of this approach is to cluster the Netflow traffic sent by each IP address in the dataset and identify the clusters that have similar behaviour to the botnet traffic. Firstly, the flows are separated in time windows, and in each time window a number of new aggregation windows are defined. In these aggregation windows the flows are aggregated by their source IP address and six NetFlow-related features are extracted for each source IP address. The feature set comprises of the number of unique source and destination ports used, the number of unique destination IP addresses contacted, the number of flows used, as well as the number of packets and bytes transmitted by each source IP address. Subsequently, the Expectation-Maximization (EM) algorithm [63] is utilized to cluster the aforementioned aggregated representations. Each cluster is represented by a new feature set consisting of the mean and standard deviation of the previously derived features. In addition, labels are assigned to each cluster according the percentage of botnet NetFlows belonging to the cluster. Finally, the RIPPER (Repeated Incremental Pruning to Produce Error Reduction) classification algorithm [18] is fitted on the labelled clustered groups. In the detection phase the flows residing on the test set are processed in a similar way, with the fitted classification algorithm used for the prediction of the cluster labels.

An interesting anomaly detection approach, operating on network samples of lower granularity comparing to NetFlows, yet offering the inclusion of both classification and clustering-based methods in the detec-

tion process, is presented in [62]. The proposed system, named *AMAL*, consists mainly of two subsystems, *AutoMal* and *MaLabel*, with the first one aiding in the extraction of behavioural artifacts through the execution of malware samples in virtualized environments, and the latter one leveraging the information associated with these artifacts to classify and cluster malware samples into families of similar characteristics. In more detail, *AutoMal* uses memory and file system forensics, as well as network activity logging, and registry monitoring to create profiles able to summarize the behaviour malware samples. Accordingly, *MaLabel* is employed by offering the options of binary classification and clustering. In the case of classification, multiple well known ML algorithms are employed, including SVM, decision trees, linear regression, and k-NN, among others, and models are trained on some manually labelled training samples, while, in the case of clustering, hierarchical clustering is applied on the data, aiming to group samples with similar low-granularity behaviours.

Another work that introduces two approaches towards anomaly detection in network traffic is presented in [27]. Both approaches aim, firstly, in developing a traffic profile, called Digital Signature of Network Segment using Flow analysis (DSNSF), able to capture the normal network behaviour, and, secondly, in utilizing these profiles to identify anomalous events by comparing them with the real network traffic by means of a modified version of the Dynamic Time Warping (DTW) metric [72]. The two methods used for extracting the aforementioned behavioural profile of the benign traffic are based on Principal Component Analysis and a modified version of the Ant Colony Optimization metaheuristic [22]. Both methods operate on a daily aggregation level of the network flows, and use historical data recorded on multiple days of a month to extract a digital signature for each day. According to the authors, the main intuition behind the utilization of such level of temporal aggregation lies on the fact that network-wide behaviors are severely affected by the working hours and the workdays period of people using the examined network. As far as the two flow characterization procedures are concerned, the first method aims to characterize the normal traffic by identifying time intervals with a medium variance among the provided historical data, with the intuition that such intervals would be associated with normal behaviour, while the second method constitutes a clustering mechanism attempting to achieve near-optimal solutions in clustering the data through the use of self-organised agents. Again, as seen in other NetFlow-based works, some relatively simple IP flow features were used in the modelling procedure, namely source and destination IP addresses and port number, as well as bytes, packets, and number of flows transmitted per second.

In [25], the authors mainly focus on the development of a statistical approach for predicting the future behaviour of a device conditioned on a features vector extracted from observed historic NetFlow records associated with that device. The analysis is conducted on host level, in the sense that the behaviour of the individual devices in a network is examined. To do so, the NetFlow data associated with each device are grouped into separate time intervals by examining whether a given NetFlow record crosses temporally a given time interval. The authors suggest that the behaviour of each device can be represented by the NetFlow records assigned to each time bin. Subsequently, for each time bin 31 features regarding the timing characteristics of the associated NetFlows (start time, end time, working hours indicator, etc.), the corresponding event types (number of records starting or ending in a bin, etc.), some NetFlow-specific characteristics (median number of packets and bytes transmitted, median duration, etc), and the nature of the captured connections (the protocols used, the entropy of the ports, etc.), are extracted and assigned to it. The correlation between the features extracted for each time bin is evaluated through the calculation of the Pearson's correlation coefficient, and features with high correlation are either discarded manually or through the application of PCA on the binned data. Finally, the dataset is split temporally in two groups of days, and a regression tree is fitted on the data retrieved for each IP on the temporally first group of days, with the predictions evaluated on the data extracted during the remaining days. This approach does not directly relates to an anomaly detection mechanism, yet the utilization of regression models as a means of capturing the temporal behaviour of a host points towards a promising direction, which is highly leveraged by more complex sequential models, like state machines.

A comparative work incorporating approaches from different aspects of Machine Learning is presented in [31]. This work constitutes an attempt of investigating the existence of features extracted from user network traffic able to differentiate between infected and uninfected user behaviours, by examining various supervised and unsupervised approaches. In more detail, 36 features are extracted from network traffic, ranging from the number of unique source and destination IP addresses and port numbers to the mean time difference between session start times, with PCA applied on them so that the 13 principal components explaining the greatest percentage of variance in the dataset can be retrieved. In the supervised environment, ten well-known ML techniques are utilized, while in the unsupervised setting k-means clustering is employed. Algo-

rithms from both settings were evaluated on both the entire feature set and the PCA-preprocessed one. The main contribution of this work can be identified in the wide variety of ML algorithms evaluated towards the goal of anomaly detection in network traffic, since a rough understanding about the effectiveness of different ML techniques in similar tasks can be acquired.

3.2.4. State Machine Learning Approaches

In [33], the authors focus on the extraction of behavioural communication profiles for each host in the examined network, so that these hosts can be classified according to their traffic summary statistics. To do so, they design a system based on the use of complex sequential models, derived by finite state machines, towards the extraction of fine grained communication profiles from the NetFlow traffic of each host. In the premises of this work, the communication profile of a host is practically a PDFA inferred from its IP flows. In more detail, the IP flows of each host are encoded into a symbolic string by concatenating five NetFlow features, namely the protocol used, the duration of the flow, the number of packets exchanged, and the number of incoming and outgoing bytes transferred. Of course, in order for this symbolic encoding to be concise the ids of the discretization bins, in which numerical values reside, are utilized instead of their actual values. After obtaining this symbolic representation, flows are aggregated into a fixed time period by sliding a window over all flows, while incrementing the start of the window one flow at a time. The main motivation behind this process is to capture the short-term temporal behaviour of a host and use it to learn an representative communication profile over it. Finally, after the extraction of the communication profiles for each host is completed, the identification of a malicious host is conducted by feeding its communication profile with a set of windows collected from an evaluation set, and calculating the ratio of the accepted versus the rejected windows.

A continuation of this work is presented in [34], where an online method for identifying change points in the recorded traffic of network flows is proposed. Again communication profiles are learnt for each host, yet since each host can get involved in multiple activities, meaning that it can be associated with multiple divergent behaviours, the profiling process is conducted on the groups of flows residing between the identified change points. The main motivation is to cluster the continuous flow traffic originating from a particular host into segments according to the different associated activities over time, so that a new model can be learnt on each segment, and a more fine-grained representation of the communication behavior of a host over time can be extracted. As in [33], aggregated features are used for modelling, and PDFAs are extracted from these aggregated views as the communication profiles of the hosts. Finally, the identification of change points is achieved through the introduction of the freshness metric. This metric is expressed as the ratio between the number of newly added transitions to the APTA under development, while processing a new word w, and the total number of transitions included in that APTA. Under a clustering perspective, a period of monotone falling freshness with low values would indicate the lack of new behaviour, while extreme spikes could relate to the identification of new behaviour in the data. Thus, this metric is calculated on multiple segments of a host's flows and a PDFA is learnt between every two consecutive minima.

Another interesting approach leveraging the capabilities of state machines is presented in [68]. In fact, this work constitutes the main inspiration for the system designed in the premises of this thesis. The authors introduce a behavioural fingerprinting system, named *BASTA* (Behavioral Analytics System using Timed Automata), which uses PDRTAs to extract identity fingerprints of hosts from their corresponding Netflow traces. As in the rest state machine based works presented above, a host based approach is adopted, with each host ranked according to an indicator of suspiciousness derived from its outgoing and incoming flows. This indicator is calculated by comparing the overlap in communication behavior between a candidate and a known infected host, with a low indicator denoting that there is small overlap between the behavioural patterns extracted from the flows of the candidate host and those of the malicious one. To learn PDRTAs on NetFlow data, consecutive flows are, first, mapped into timed events by keeping the associated timestamp and combining the NetFlow features (protocol, direction, duration, bytes, packets) into a symbolic representation, and, secondly, grouped into small subsequences at regular and predefined time intervals. Finally, two methods for identifying PDRTA fingerprints in previously unseen Netflow data are proposed, with both of them founded on the detection of suspicious pairs of states and events (infection symptoms) obtained from the models of infected hosts. The first approach, called error-based, compares the occurrence counts of similar infection symptoms between a candidate and known malicious hosts, while the second approach, called fingerprint-based, utilizes a configuration dataset of known benign hosts to identify symptoms that never occur to these hosts, and leverages this information to identify malicious symptoms in new candidate hosts.

3.3. State Machine Inference

In the previous sections some interesting applications of state machine modelling in the field of anomaly detection are presented. Yet, the learning capabilities associated with state machine learning and the theory of automata have been leveraged in various other fields. Especially when there is a sense of sequentiality in the examined data, state machines can be proven to be quite powerful in capturing the underlying sequential patterns. These modelling capabilities are further highlighted in this section through the presentation of various state machine related works and applications. In [55], the effectiveness of incorporating data flow information in finite state automata inference on execution traces retrieved from software systems is evaluated. In particular, the authors attempt to question whether the introduction of data specific parameters along with the observed event sequences produced by a given software system can lead to the inference of state machines able to represent more accurately complex program behaviours. To do so, they compared the performance of two simple passive learning algorithms, namely kBehavior [60] and kTail [8], against two more advanced learning methods capable of inferring extended FSMs, namely KLFA [59] and gkTail [56], when learning on execution traces with different levels of sparseness produced by different software systems.

Furthermore, in [14], state machine inference is used as the core component in a project regarding the validation and testing of security properties of services and web applications. In particular, the so called *SPaCIoS* project includes two state machine based approaches to reverse engineer models from their implementations. The first approach utilizes an EFSM (Extended Finite State Machine), with parameterized transitions, guards, and a function for each output parameter, to infer models capable of capturing the behavioural patterns present in remote HTTP connections of web applications in black-box mode. The second approach, on the contrary, operates in white box mode, by inferring an EFSM for each servlet of the system under examination. Extended FSMs has been also used in [65] as the building block of a passive learning system, called *Perfume*, able to infer behavioral resource-aware models of software systems from logs of their executions, so that a better understanding about the system's behaviour and resource use can be acquired. According to the authors, the main distinction between the proposed system and typical FSM inference approaches can be identified on the fact that it manages to differentiate behaviorally similar execution traces associated with different resource consumption. To evaluate the appropriateness of their modelling technique, *Perfume's* inference ability is compared with two other FSM inference approaches found in the literature, on two case studies drawn from a set of TCP communication traces, and a web log retrieved from a real estate website.

In [26], a reverse engineering method for security protocols, named *SPREA* (Security Protocols Reverse Engineering Approach), is proposed, aiming to infer protocol state machines based on network traces features, like the protocol invariable and variable fields, the ciphertext fields and their length. To do so, a 4-stage approach is followed. First, the captured network packets are clustered according to a set of statistical features (packet length, offset, and direction). Subsequently, sequences of protocol constants, like protocol name, version, and control codes, are extracted from the packets in each cluster as ordered keywords using sequential pattern mining. These keywords constitute the protocol invariable fields, while the bytes between two keywords are considered as variable fields, with byte sample entropy used to locate ciphertext fields within these invariable fields. Finally, the Exbar algorithm [19] is leveraged to identify the minimal DFA able to represent the protocol behaviour associated with the clustered packets.

State machine learning has been also applied on mobile applications, as it can be seen in [81]. This work introduces a testing methodology involving the application of state machine learning on mobile Android applications towards the goal of vulnerability detection. To this end, an active learning approach is adopted, and two well known active learning algorithms, namely the L^* [4] and TTT [43] algorithms, are employed for the model inference part, while two methods, namely the straightforward random walk approach, and the W-method [16], are considered for the equivalence checking part of the models produced during the inference procedure. The state machines were learnt on an alphabet extracted from internal actions of the Android applications, under the intuition that, since the state machine model would include the sequences of actions allowed within the application, a better understanding on the cohesion between logical components could be achieved. Finally, the detection part was performed through the identification of behavioural patterns conforming to a publicly available list of crucial vulnerability classes in the field of mobile security.

In [52], the learning capabilities of timed automata are leveraged towards anomaly detection in Industrial Control Systems (ICS). In particular, a graphical model-based approach, named *TABOR* (Time Automata and Bayesian netwORk), is proposed for profiling the normal operational behaviour of the sensors and actuators used in an automated water distribution system. This profiling procedure is conducted through the application of timed automata inference on the sensors signals, and the use of bayesian networks to capture the dependencies between the sensors and actuators. In that way, the extracted timed automata can act as

one-class classifiers capable of recognizing abnormal behavioral patterns and dependencies. Of course, as in most state machine learning applications, a symbolic representation of the sensors signals is extracted, before the timed automata can be inferred. The class of timed automata used is PDRTAs and the learning algorithm utilized is RTI+ [85]. More information about this algorithm can be found in Chapter 5, since it is also used in the premises of the current work. Finally, the authors argue that, through the inclusion of timed automata, the proposed detection system provides both interpretable and localized results, in the sense that a graph-based model can be easily visualized, while modelling the behaviour of each sensor can significantly aid in the identification of the exact malfunctioning components of the system under examination.

On similar grounds, PDRTAs were used in [53] to model car-following behaviours. The goal of this work was to effectively model the changes in individual drivers' behaviour throughout the data collection period associated with the examined dataset. The functionality of the proposed system, named *MOHA* (Multi-mOde Hybrid Automaton model), can be briefly described as follows: Initially, k-means clustering is utilized to obtain temporal symbolic representations (timed strings) from the time series representing the values of multiple vehicle trajectory features, like the speed, and the relative distance and speed, over time. Subsequently, as in [52], the RTI+ algorithm is employed, so that PDRTAs able to capture frequent temporal patterns in the observed timed events can be produced. After the PDRTA inference is complete, a mapping between the original timed strings and the individual states in the extracted model is constructed, so that these mapped state subsequences can be clustered according to their associated timed events into different driving behaviour modes. Finally, the learnt models equipped with the driving behaviour mode information can be used for online predictions on the longitudinal acceleration associated to an incoming stream of vehicle data.

4

Data Exploration

Before designing the detection methodology and experimenting on the provided network traffic, it is highly important to acquire an insightful view regarding the nature of that traffic. The development of the detection module, as well as its detection performance, are highly dependent on the acquisition of a clear understanding about the data on which the system is expected to operate. This understanding can be attained by inspecting the provided data and visualising useful summary statistics regarding their nature and the correlations between the available data features. This data exploration procedure can significantly aid the decision making process in several steps of a machine learning pipeline, like the preprocessing of the input data, the feature selection, the training and test set split of the input data, as well as the development of the core learning module of the pipeline. This chapter includes an overview of the datasets used to evaluate the performance of the proposed detection system, along with the analysis performed on the nature of the provided data.

4.1. Datasets Overview

As it was mentioned earlier, the designed system should be able to operate on high-level network traffic summary statistics, without having access to the payload of the packets transmitted within the examined network. As a result, only NetFlow datasets are taken into account for the evaluation of the proposed detection methodology. Apart from this datatype specification, the datasets to be selected for the evaluation process should fulfill three additional requirements, namely availability, reliability, and diversity. The first requirement refers to the need of publicly available datasets, since in that way the data could be directly acquired, while the public availability of data enhances the reproducibility of the conducted work. The second of these requirements reflects the confidence around the quality of a given dataset. When working with publicly available datasets, trusting the procedures followed by the creators of the dataset regarding the collection and labelling of the data constitutes a necessity. To address this requirement, it was considered beneficial for the selected datasets to have been cited in multiple malware/intrusion detection works. Finally, the diversity requirement refers to the range of attacks and malicious behaviors included in the selected datasets. As it was stated in Chapter 1, the designed system should be able to detect as many types of attacks as possible, thus it should be evaluated on datasets containing a diverse set of malicious behavior. As it will be better illustrated in the sections to follow, the selected datasets cover a wide range of malicious network activity, fulfilling in that way this final dataset selection condition. Taking all these requirements into consideration, the following three publicly available NetFlow datasets were used:

- **CTU-13**: This dataset was first presented in [28], containing botnet traffic captured in the CTU University, Czech Republic, in 2011.

- **UNSW-NB15**: This dataset was introduced in [64], and was generated by the IXIA PerfectStorm tool[4] in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) in 2015 with the goal of creating a NetFlow-based dataset that would incorporate various synthetic attack behaviours.

[4] `https://www.ixiacom.com/products/perfectstorm`

- **CICIDS2017**: This dataset was initially presented in [75], and was created by researchers of the Canadian Institute for Cybersecurity (CIC), based at the University of New Brunswick (UNB), Canada, in 2017, as an attempt to address issues in earlier intrusion detection datasets.

Deeper insight on these datasets is provided in the sections to follow, yet, if a detailed presentation of the datasets is sought, the aforementioned cited papers should be considered.

4.1.1. The CTU-13 Dataset

As it was mentioned above, the CTU-13 dataset includes a large capture of real botnet traffic, along with normal (benign) and background traffic. In more detail, the CTU-13 dataset is composed of thirteen captures (called scenarios by the creators of the dataset) of various botnet samples. The topology used for the production of the dataset consisted of a set of virtualized computers running on a Linux host being connected to the CTU University network, as it can be seen in Figure 4.1, which is retrieved from the original work presented in [28]. The traffic included in the dataset was captured both on the malicious Linux host and on one of the university routers. The traffic originating from the Linux host was purely malicious, while the traffic associated with known and controlled computers in the network was labelled as normal. The rest traffic was attributed to background communication, meaning that the benignity of that traffic cannot be safely assumed. Each scenario is associated with a specific malware performing a set of different malicious actions, ranging from sending SPAM, and doing Click-Fraud, to port scan, and DDoS attacks. A detailed presentation of the main characteristics of the scenarios, along with the corresponding Botnet behavior can be sought in [28].

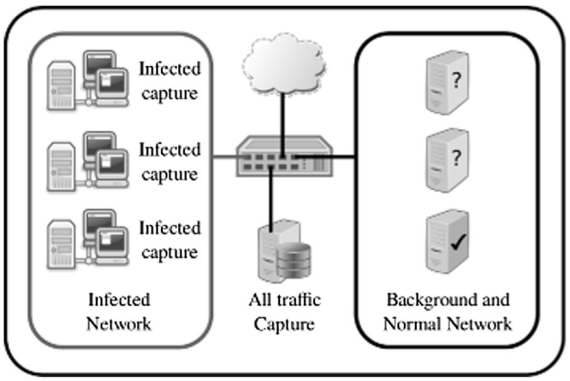


Figure 4.1: Network topology of the CTU-13 dataset (retrieved from [28])

After capturing the packets transmitted across the network, the dataset was preprocessed and converted into the NetFlow file standard, consisting of both a unidirectional and a bidirectional set of NetFlows. In the premises of the current work, the bidirectional set was used, since, as the creators of the dataset advise, this capture offers a more detailed representation of the network communication comparing to that offered by the unidirectional flows. Each bidirectional flow is composed of the following fields: start time, duration, protocol, source IP address, source port, direction, destination IP address, destination port, state, source Type of Service (sToS), destination Type of Service (dToS), total packets and bytes transmitted, and source bytes. Apart from the aforementioned attributes, a label was assigned to each flow, denoting the class of the host that this flow either originated from or was directed to. The initial distribution of labels in each scenario can be found online[5] on the Stratosphere Lab repository, with the majority of the recorded flows assigned to the background class. Nevertheless, in the greatest part of this work, solely the flows associated with the normal and botnet traffic are taken into consideration, since the primary goal of this thesis is to evaluate the detection performance of the proposed methodology, and the lack of a ground rule regarding the benignity of the background traffic would question the reliability of the prediction results. Thus, as it is suggested by the creators of the dataset in their online blog[5], only the flows originating from either the malicious hosts or the acknowledged normal computers are retained in the datasets comprising each scenario. The distribution of the labels in the retained flows can be seen in Table 4.1 below.

[5] `https://www.stratosphereips.org/datasets-ctu13`

Scenario	Total Flows	Normal Flows	Botnet Flows	Number of Benign Hosts	Number of Bots	Type of Bots
1	71219	30258	40961	6	1	Neris
2	30023	9082	20941	5	1	Neris
3	143125	116303	26822	6	1	Rbot
4	27775	25195	2580	6	1	Rbot
5	5561	4660	901	6	1	Virut
6	12101	7471	4630	6	1	Menti
7	1732	1669	63	5	1	Sogou
8	78766	72639	6127	7	1	Murlo
9	214880	29893	184987	7	10	Neris
10	122157	15805	106352	7	10	Rbot
11	10873	2709	8164	6	3	Rbot
12	9783	7615	2168	6	3	NSIS.ay
13	71782	31779	40003	6	1	Virut

Table 4.1: Distribution of records in each scenario of the CTU-13 dataset

Apart from the distribution of the labels among the included flows, Table 4.1 presents also the number of benign and malicious hosts, along with the type of bots used to generate the malicious traffic in each scenario. It should be mentioned that the traffic associated with each host in the CTU-13 dataset originates strictly from one class, meaning that each host can be either strictly benign or strictly malicious. Finally, it can be seen that scenario 3 contains the highest number of benign flows, with scenarios 9, and 10 containing the highest number of botnet flows. Despite the obviousness of this observation, it is pointed out since the distribution of scenarios between the training and testing phases is based on that observation, as it will become apparent in Chapter 7.

4.1.2. The UNSW-NB15 Dataset

As stated by the creators of this dataset in [64], the motivation behind its generation can be identified in the attempt to address the issues of the previous Network Intrusion Detection System benchmark datasets found in the literature through the incorporation of various up-to-date synthetic attack behaviors in a flow-based dataset. As mentioned earlier, the network traffic included in this dataset was generated synthetically by the IXIA PerfectStorm tool[4] in an attempt to create a mixture of the contemporary (at the time of course) normal and abnormal network traffic. In particular, this dataset includes nine attack families, namely Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms, simulated by the aforementioned tool based on its inherent attack set that is continuously updated by a dictionary of publicly known information security vulnerabilities and exposures. Finally, the network topology used for the creation of this dataset can be seen in Figure 4.2, which is retrieved from the original work presented in [64].

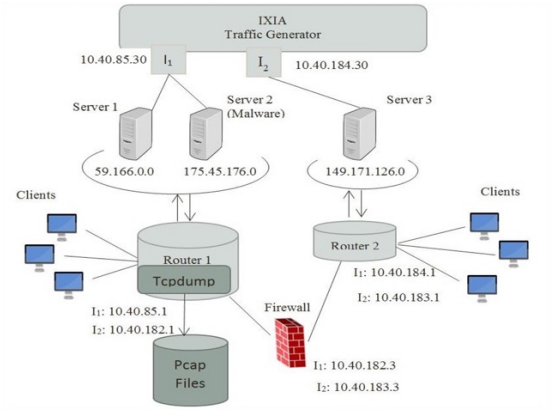


Figure 4.2: Network topology of the UNSW-NB15 dataset
(retrieved from [64])

As it can be seen in Figure 4.2, the traffic generator was configured with three virtual servers, two of which (servers 1 and 3) were responsible for generating the normal traffic, and the last server (server 2) was injecting the malicious activities in the network traffic. These servers were connected through two routers to the hosts of the network, with each router being connected to the firewall of the network too. Apart from routing traffic within the network, one of these routers (router 1) was used for capturing the network traffic, originating from the traffic generator, that was dispersed among the network nodes. After capturing the transmitted packets within the network, a flow-based monitoring system and a network traffic analyser are used to extract features from the raw network traffic, resulting to a set of 47 flow and packet based features. Of course, since the packet-based features include information about the payload of the packets, they are disregarded within the scope of the current thesis, and solely basic flow-based features are used in the analysis and the experiments conducted. Yet, in case the reader desires to inspect the full feature set of the dataset, this can be found in [64], where this dataset was introduced. Finally, the derived dataset was split in a timely fashion into four chunks of data, as it can be seen in Table 4.2.

Scenario	Total Flows	Normal Flows	Malicious Flows	Number of Benign Hosts	Number of Malicious Hosts	Type of Attack
1	700000	677785	22215	36	4	Multiple
2	700000	647251	52749	36	4	Multiple
3	700000	542575	157425	36	4	Multiple
4	440043	351149	88894	34	4	Multiple

Table 4.2: Distribution of records in each scenario of the UNSW-NB15 dataset

As it can be easily seen from the table above, the first three chunks contain an equal number of flows, with the last chunk containing the remaining flows, while there are 40 hosts (source IP addresses) in the dataset, with four of them being considered as malicious. At this point, it should be mentioned that these four malicious hosts do not demonstrate purely malicious behaviour, meaning that only a part of the flows associated with them are labelled as malicious. Nevertheless, in the premises of this thesis it is assumed that if a host is linked with at least one malicious flow, then it should be considered as malicious. Finally, it shall be pointed out that Table 4.2 does not contain an analytical overview of the attacks associated with each split of the dataset, since each split contains flows associated with all types of attacks injected in the network traffic.

4.1.3. The CICIDS2017 Dataset

This dataset is characterised by its creators as an Intrusion Detection Evaluation dataset containing malicious traffic resembling a wide variety of the most commonly encountered attacks. The traffic capture is organised across five weekdays, and was performed on a network topology consisting of two separate networks, namely the Attack-network and the Victim-Network, in an attempt to create a topology resembling to the greatest extent a real-life setting. Towards that end, the Victim-network constitutes a complete network topology, including switches, routers, a modem, and a firewall, along with a group of different benign machines operating on various operating systems, while the Attack-network is composed by a number of machines with public IP addresses responsible for executing the designed attacks, along with a switch and a router used for bridging this network to the Victim-network. This topology is better illustrated in Figure 4.3, retrieved from the original work presented in [75]. Finally, it should be mentioned that the benign traffic is generated by an agent responsible for profiling the abstract behavior of human interactions through the use of machine learning and statistical analysis techniques.

After capturing the packets transmitted within the presented network topology, the CICFlowMeter tool[6] is used to extract a bidirectional flow representation of the recorded traffic, incorporating more than 80 flow-level statistical features. Of course, as it was mentioned earlier in this work, the designed system should be able to work on universally available features, thus solely some basic NetFlow attributes are utilized out of the extended feature set provided in this dataset. Yet, in case the reader wants to inspect the full feature set provided by this dataset, this can be accessed on the online CIC repository[7]. As mentioned above, these flows are captured within a period of five consecutive days, including a wide range of attacks. The first day (Monday) contains only benign traffic, while each of the following days is associated with different types of attacks. The distribution of these attacks for each day is presented in Table 4.3, along with the number of

[6] `https://github.com/ahlashkari/CICFlowMeter`
[7] `http://www.netflowmeter.ca/netflowmeter.html`

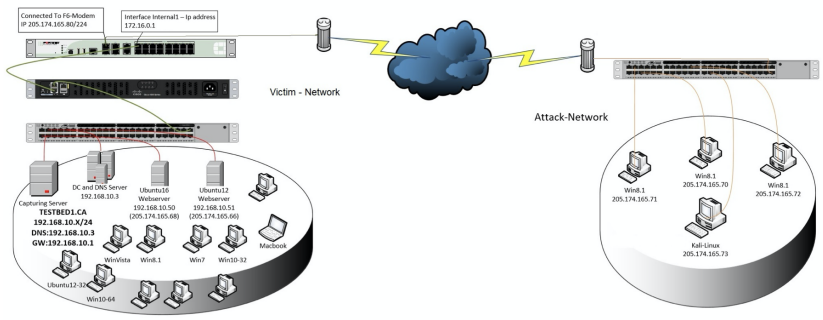


Figure 4.3: Network topology of the CICIDS2017 dataset (retrieved from [75])

distinct benign and malicious hosts in the network. In the premises of this dataset, the condition of hosts demonstrating solely one class of behaviors is not satisfied. Thus, for the purpose of host level analysis it was assumed that only hosts associated entirely with normal flows are considered as benign. Finally, it can be easily seen that the number of benign hosts is significantly greater than the handful of machines comprising the Victim-network, meaning that there is also some unknown background traffic captured within the designed network topology. In the premises of this thesis, solely the benign hosts associated with a significant amount of traffic are utilized in the modelling and detection procedure.

Scenario	Total Flows	Normal Flows	Malicious Flows	Number of Benign Hosts	Number of Malicious Hosts	Type of Attack
Monday	529918	529918	0	8239	0	-
Tuesday	445909	432074	13835	7191	1	Brute Force
Wednesday	692703	440031	252672	7688	1	DoS
Thursday (morn.)	179366	168186	2180	4202	1	Web Attack
Thursday (aft.)	288602	288566	36	5100	1	Infiltration
Friday (morn.)	191033	189067	1966	4461	3	Bot
Friday (aft. - DDoS)	225745	97718	128027	2066	1	DDoS
Friday (aft. - port scan)	286467	127537	158930	3666	1	Port scan

Table 4.3: Distribution of records in each day of the CICIDS2017 dataset (morn. and aft. refer to morning and afternoon respectively)

4.2. Data Preprocessing

After acquiring the datasets on which the proposed detection system would be trained and evaluated, an initial preprocessing of the raw data should be performed in order for them to be converted into a format that the designed system is able to process. Since the provided captures consisted of a series of timestamped flows stored in *.txt* or *.csv* formats, they should be "translated" into a format that would enable the machine to understand the type of each feature, while handling the case of missing or uncommonly formatted values. Towards that end, the following preprocessing steps were adopted:

1. The timestamp fields were parsed according to the timing format specified in each dataset, so that the input data could be properly handled as multivariate time series.

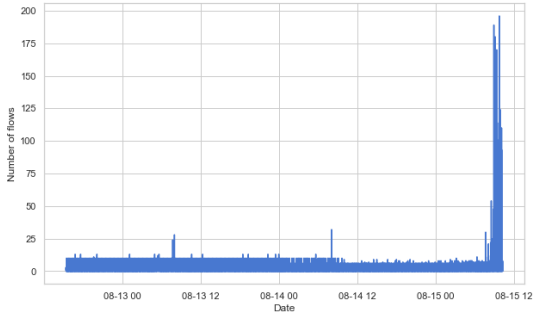
2. The missing values in categorical features, like the protocol used, were filled with the keyword "missing", while, in the case of numerical features an invalid value, like -1, was used instead.

3. There was the case that the port numbers, apart from the expected Integer representation, would be given in a hexademical form. Thus, in order to maintain a universal format for the port attributes, the conversion of any hexademical values to their decimal representation was performed.

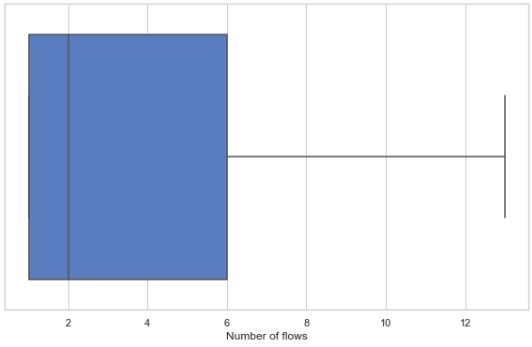
4. Each categorical feature to be processed by the detection system, like the protocol attribute, was encoded into a numerical representation, by assigning a unique number to each distinct value of that attribute.

5. As a final preprocessing step, a datatype was assigned to each numerical attribute, as an indicator for the way that they should be handled by the machine.

After this initial preprocessing was completed, an analysis of the distribution of flows in each dataset was undertaken, so that a better understanding on the way that the data should be processed, before entering the designed detection system, could be acquired. As it was mentioned earlier, and will be further explained in Chapter 6, the proposed system is based on a multivariate sequential model, which operates on traces extracted in a timely fashion from the raw input data, while aggregating these data according to a user-specified level of analysis (host or connection level). As a result, by visualising the distribution of flows both across time and within each analysis level would be beneficial on the decisions made regarding the trace extraction and data aggregation procedures. These visualization can be seen in Figures 4.4, 4.5, and 4.6 for each dataset examined. In particular, Figures 4.4a, 4.5a, and 4.6a depict the number of flows recorded every second in each dataset. These figures illustrate the problems that could arise, in case a static windowing technique would be used for the trace extraction process. It can be easily seen that there are multiple data spikes in all datasets, while in the case of the CICIDS2017 dataset the problem of time regions with low to zero flow recordings is encountered too. Such issues led to the adoption of a dynamic windowing technique, the functionality and effectiveness of which are analysed in Chapter 6.

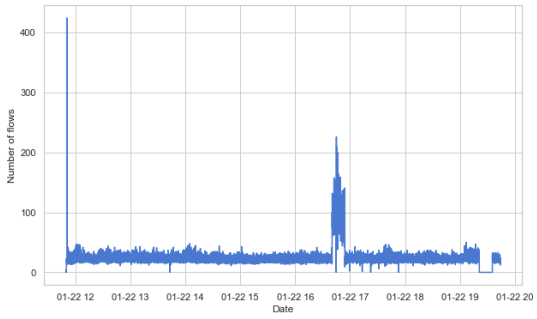


(a) Number of flows per second

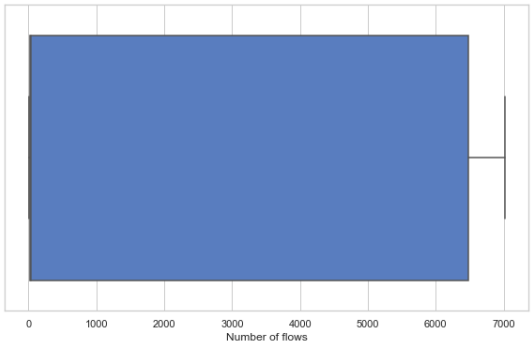


(b) Number of flows per connection

Figure 4.4: Flow distribution plots in CTU dataset



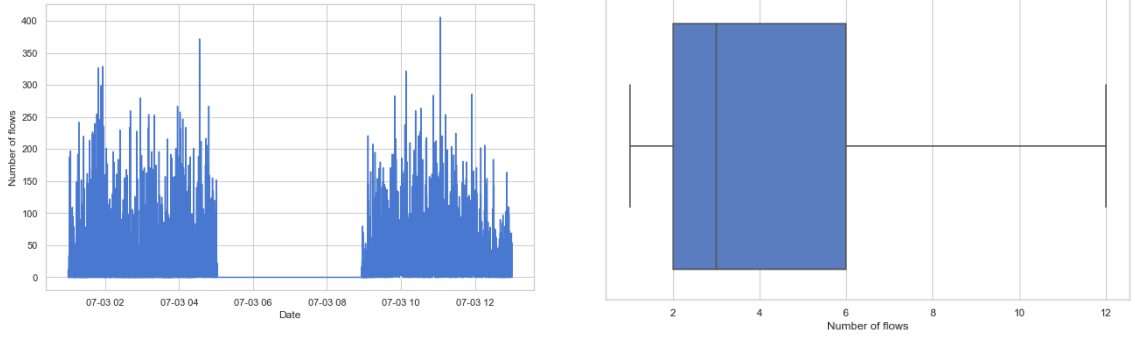
(a) Number of flows per second



(b) Number of flows per connection

Figure 4.5: Flow distribution plots in UNSW-NB15 dataset

Figures 4.4b, 4.5b, and 4.6b constitute box plots visualising the number of flows belonging to each unique connection for the majority of the recorded connections in each of the examined datasets. It can be easily seen that, in two of the three datasets used (CTU-13 and CICIDS2017), most connections are short-lived, consisting of a handful of flows, with solely few major connections present in the dataset (these connections are not depicted in these figures, since they constitute outlying points in the given context and have been excluded for visualization purposes). As long as the UNSW-NB15 dataset is concerned, the range of the number of flows in each distinct connection is significantly greater comparing to the rest datasets, meaning that a connection level analysis could be meaningful for this dataset. Nevertheless, if Figure 4.5b is inspected with



(a) Number of flows per second

(b) Number of flows per connection

Figure 4.6: Flow distribution plots in CICIDS2017 dataset

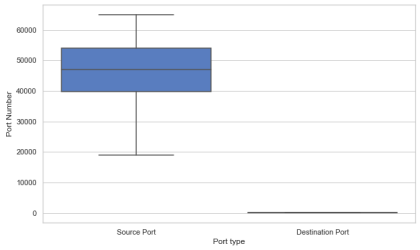
greater attention, it can be seen that the line representing the median of the number of flows per connection is close to the left part of the visualised box plot, meaning that despite the fact that there is a notable number of connections with a significant number of flows contributing to the visualized range, the majority of connections consist of few flows.

It can be easily understood that connections with such a low communication load cannot be used to derive robust behavioral models in the training phase of the proposed system, which as a matter of fact would lead to the creation of too few models solely from the major connections present in the dataset. To avoid such scenarios, a host level analysis was selected in the premises of this thesis. Yet, in the case of the UNSW-NB15 a connection level analysis could be applicable, and possibly beneficial, for the major hosts of the dataset. This possibility is not examined in the premises of this thesis, yet the discussion about the quality of the results regarding this dataset refers to this possibility in Chapter 7. Finally, it should be mentioned that the figures presented above are derived from benign captures of the datasets. This decision was made to maintain the integrity of the conducted analysis, since one of the desired characteristics of the developed detection system was its ability to learn solely from benign data.

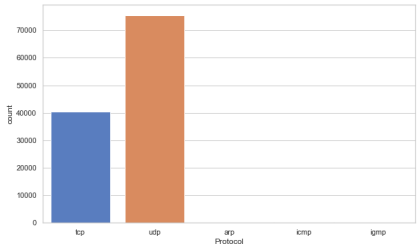
4.3. Feature Exploration and Selection

As it became apparent from the discussion regarding the overview of the datasets presented in Section 4.1, each dataset offers a different set of flow-based features, with the CTU dataset providing the most limited set, and the rest datasets offering significantly extended ones. Yet, as it was discussed earlier in this thesis, the proposed system should be capable of operating in a way as universal as possible, meaning that its detection performance should not depend on highly sophisticated features that are not existent in the majority of NetFlow datasets. On top of that, in order to fairly evaluate the performance of the designed system across the three datasets included in this work, it is considered necessary to come up with a unified set of features that are available in all datasets. As a result, the following basic NetFlow features were taken into consideration in this work: the timestamp of each flow, the source and destination IP addresses and port numbers, the protocol of communication, the packets transmitted, as well as the bytes originating from the source and destination hosts. Not all of these features were used in the learning process though. Some of them were eventually discarded, while others were only used implicitly. In particular, the timestamps of the flows were used for the extraction of traces from the flows of the dataset, while the source IP address was used to group the data in a meaningful manner for a host level analysis to be conducted. In case a connection level analysis had been considered, then both the source and the destination IP addresses would have been used in the grouping procedure. The number of transmitted packets was another feature that was not taken into account eventually, since this feature is directly related to the number of transmitted bytes, thus its incorporation was considered redundant. The rest features were directly used in the benign communication profiles' learning process.

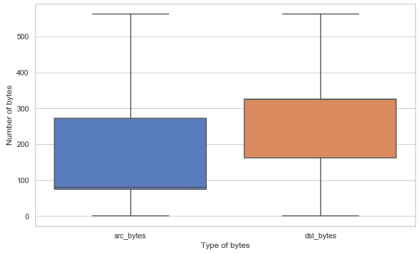
A visualised overview of these features in each dataset is presented in Figures 4.7, 4.8 and 4.9. Of course, as it was mentioned in Section 4.2, the figures presented above are derived from benign captures of the datasets. In more detail, in Figures 4.7a, 4.8a, and 4.9a the range of the source and destination port numbers recorded for the majority of benign flows of each dataset is depicted. It can be easily seen that in the case of CTU-13 and CICIDS2017 datasets the recorded destination port numbers attain significantly lower values comparing to



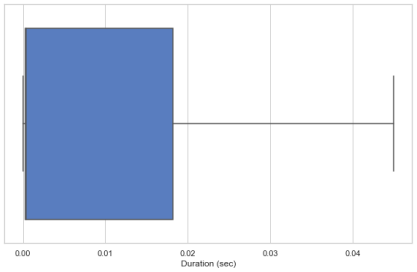
(a) Source and destination port range



(b) Number of flows per protocol type

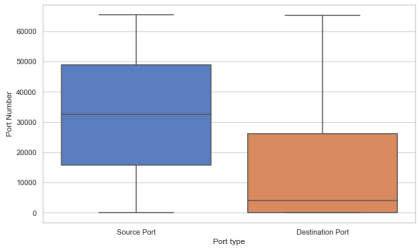


(c) Range of source and destination bytes

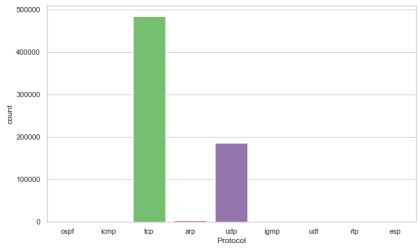


(d) Duration range

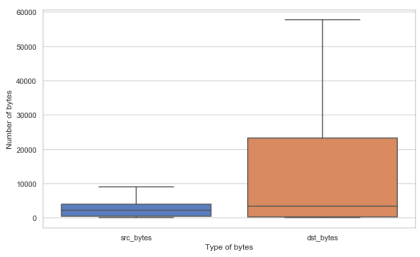
Figure 4.7: Visualization of basic NetFlow features for CTU dataset



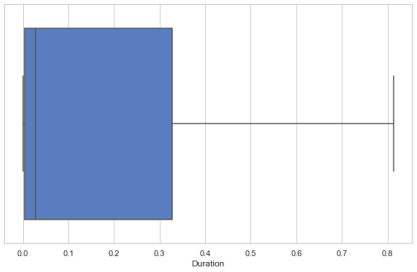
(a) Source and destination port range



(b) Number of flows per protocol type



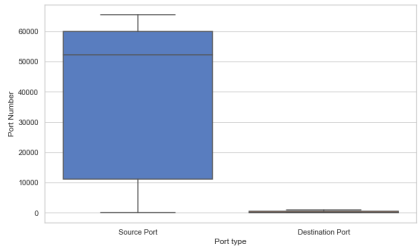
(c) Range of source and destination bytes



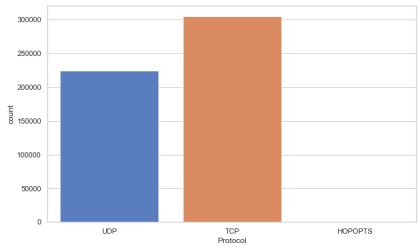
(d) Duration range

Figure 4.8: Visualization of basic NetFlow features for UNSW-NB15 dataset

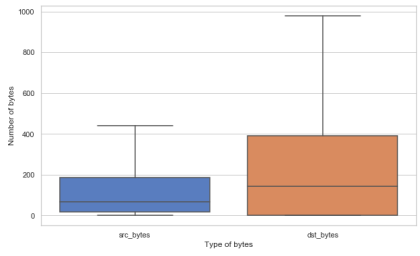
the source port numbers, which makes sense since the benign communication is directed to ports associated with specific services (like 53 for DNS, 80 for HTTP, and 443 for HTTPS), while, in most cases, the source ports are ephemeral ports allocated automatically from a predefined range by the IP stack software. Nevertheless, this does not seem to be the case for the UNSW-NB15 dataset, since a much higher range of destination port numbers is recorded. As far as the communication protocols are concerned, Figures 4.7b, 4.8b, and 4.9b, presenting the number of flows associated with each protocol type in each dataset, clearly illustrate the dominance of TCP and UDP protocols in the benign network traffic. Subsequently, Figures 4.7c, 4.8c, and 4.9c, illustrate the range of bytes sent (src_bytes) and received (dst_bytes) within benign flows in each dataset. It can be observed that, despite the fact that the ranges between source and destination bytes tend to differ, their median values lie close together. Finally, in Figures 4.7d, 4.8d, and 4.9d, the range of the duration of flows across each dataset is illustrated. It can be easily seen that the majority of flows in the CTU-13 and UNSW-NB15 datasets demonstrate a significantly low duration, while this is not the case for the CICIDS2017 dataset. Yet, since the measurement unit of the duration for this last dataset was not provided, there cannot



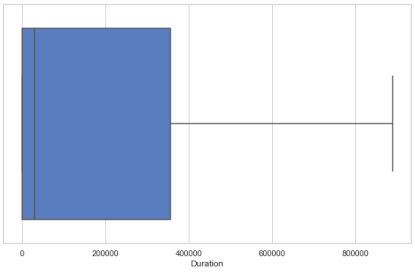
(a) Source and destination port range



(b) Number of flows per protocol type



(c) Range of source and destination bytes



(d) Duration range

Figure 4.9: Visualization of basic NetFlow features for CICIDS2017 dataset

be any safe assumptions regarding the actual extent of the duration values recorded.

4.4. Challenges

Developing an anomaly detection system able to operate on basic high-level statistics of network traffic, while learning multivariate sequential models only from benign network behavior constitutes a task inherently connected with various challenges, the primary of which can be seen as follows:

- **Representative benign traffic modelling**: Extracting benign traces representative enough to properly capture the behavioural patterns that encompass every possible normal sequence of NetFlows is considered a particularly difficult task. In many cases, the boundary between normal and anomalous behavior is quite unclear, which as a matter of fact renders the identification of anomalous observations lying close to the boundary significantly complex.

- **Selection of the windowing strategy**: In order to learn a sequential model from a series of NetFlows, a set of traces shall be extracted from this series. This extraction is based on a window applied on the series, so that partial subsequences can be retrieved and offered as input to the learning algorithm. The quality of this extraction procedure is highly correlated to the quality of the learning process. If the applied windows fail to capture the underlying temporal patterns, then the developed model will not reflect the intended traffic behaviour.

- **Noisy data**: Often, the benign network traffic contains noise, and in case this noise is incorporated into the training traces used to create the models of the benign behavioural patterns, it is highly likely that some models could relate to actual anomalies, rendering the identification of such anomalies impossible. Thus, the development of a system robust to noise is of high importance.

- **Sensitivity to deviations**: Elaborating further on the previously presented challenge, the phenomenon of deviating patterns in benign NetFlow data is quite common, given the fact that these data are extracted through a monitoring process of real-life network traffic. Thus, the designed system should be as insensitive as possible to such deviations, so that the extraction of overfitting models can be avoided.

5

Model Creation

Flexfringe [83], the tool used for learning behavioural models from NetFlow traffic, constitutes a passive automaton learning package with multiple functionalities and tunable parameters affecting the learning procedure. Both topics will be discussed in the current chapter, along with a high level description of the algorithms supporting the state machine learning process offered by flexfringe. First, such core algorithms, and the intuition behind them, will be presented in short. Subsequently, the attention will be focused on the multivariate version of this tool, since this approach was adopted in the modelling part of this thesis. Finally, the decisions made regarding the tunable parameters of the system will be discussed and justified.

5.1. Red-Blue State Merging Algorithm

Flexfringe bases its functionality on a well-known greedy state-merging method, the Evidence-Driven blue-fringe State-Merging algorithm (EDSM), introduced in [48]. This algorithm was the winning DFA inference algorithm of the Abbadingo One DFA Learning Competition and is based on the introduction of a red-blue framework for labelling and merging consistent states. EDSM starts by constructing an Augmented Prefix Tree Acceptor (APTA) consistent with the input data and attempts to reach a compact hypothesis by repeatedly merging compatible pairs of states. A state merge is considered compatible (or consistent), when both states to be merged are associated with the same operation (accepting or rejecting). During the merging procedure, the incoming and outgoing transitions of a newly merged state consist of the incoming and outgoing transitions of the corresponding pair of candidate states. On top of that, when a merge introduces a non-deterministic choice, in the form of a set of similarly labelled outgoing transitions to different states, these target states are merged as well. This procedure is called determinization, and is repeated until no non-deterministic choices are left. Of course, if at some point of this determinization process two inconsistent states are to be merged, the whole process aborts and the originating pair of states cannot be merged. Two conclusions can be easily drawn from such a procedure. First, EDSM constitutes a passive DFA learning algorithm based on the availability of both positive and negative training samples. Second, each state merge introduces new constraints for future merges, which could be proved wrong in case an incorrect merge is made. As a result, it is highly important for the algorithm to follow a well-founded merging strategy supported in the highest extent possible by the evidence provided through the input data, especially during the initial merging attempts, since a wrong merge could have an avalanche effect on the learning process.

In the red-blue framework, the merging procedure and the calculation of the evidence score are based on the maintenance of a core of red nodes and a fringe of blue nodes, aiming to effectively reduce the search space of the possible merge pairs by solely performing red-blue merges. The red nodes represent the identified parts of the automaton, while the blue ones are the children of red nodes acting as merge candidates. Given the aforementioned setting, three actions are eligible during the learning procedure: (1) compute the score for merging a red/blue pair, (2) color a blue node red if there is no possible merge with any red node, and (3) merge a blue with a red node. Of course, the aforementioned actions along with the red-blue state representation can lead to multiple algorithms of varying performance depending mainly on the way that the evidence score is calculated and the order in which the three actions presented above are performed. In the original work presented in [48], the difference in accepting and rejecting states before and after performing a potential merge was used as the evidence score associated with that merge, while, as far as the order

of actions is concerned, a highest priority was given to coloring the shallowest unmergeable blue node red, followed by the action of performing the highest scoring blue/red merge, in case no unmergeable blue node could be found. Of course the first node to be colored red is the root of the APTA, and the aforementioned procedure is repeated until there are only red nodes left.

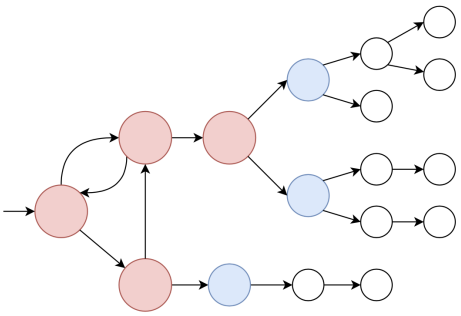


Figure 5.1: A snapshot of a sample DFA during the determinization process in the red-blue framework

A snapshot of a sample DFA during its determinization process in the red-blue framework can be seen in Figure 5.1. The red nodes are the red states representing identified parts of the automaton, while the blue nodes constitute the blue states acting as the current candidates for merging. Furthermore, there are some uncoloured nodes denoting the unexplored parts of the initially created APTA. Since the initial development of the algorithm, various proposals have been made regarding the search techniques employed in the merging process of EDSM. Flexfringe builds upon this algorithm, as well as some of the proposed search techniques, by adding numerous functionalities, so that the identification of Deterministic Real Time Automata (DRTA) can be achieved, as it will be seen in more detail in the following sections.

5.2. Learning from Labelled Data: The RTI Algorithm

The core functionality of flexfringe was initially based on an extension of the classic red-blue fringe EDSM approach for DFA identification to the field of Deterministic Real-Time Automata (DRTA). As it was explained earlier in this thesis, DRTAs provide better modelling capabilities for capturing the behaviour of real-world systems, the operation of which is affected greatly by the timing of their observed events. To deal with such real-time systems, the RTI (Real Time Identification) algorithm was proposed in [82]. This algorithm follows a similar merging process as the one described in the previous section with two primary differences: (1) unlike the case of simple DFAs, each transition is associated with a delay guard representing the timing condition connected with that transition, which as a matter of fact renders the introduction of a timed evidence value necessary, and (2) a new *SPLIT* operation is added in the merging process, along with the *MERGE* and *COLOR* operations, that are originally included in the EDSM algorithm. Through the addition of the split operation the authors of the algorithm aimed in aiding the resolution of inconsistencies introduced by the delay guards initially set in the APTA, as it is explained in greater depth below.

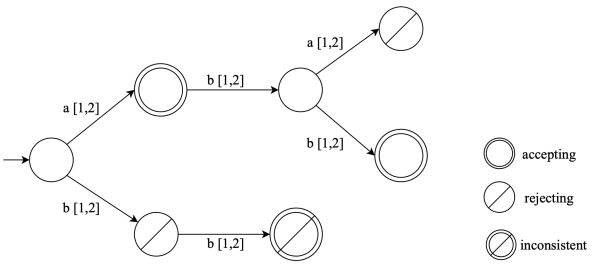


Figure 5.2: A real-time APTA generated from the timed sample with $S^+ = \{(a,1),\ (a,1)(b,3)(b,4),\ (b,2)(b,3)\}$ and $S^- = \{(a,1)(b,3)(a,4),\ (b,2),\ (b,1)(b,2)\}$

Similar to the procedure described in the previous section, initially a timed version of an APTA is constructed from the input data samples. At first, the APTA is derived in the same manner as in the case of plain DFAs by considering the untimed versions of the symbol sequences provided in the training set. Subse-

quently, each transition is enriched by a delay guard, with the starting values of the lower and upper bounds of such guards being set to the minimum and maximum delay values recorded in the training set. It can be easily seen that such an initialization could lead to the existence of inconsistent states in the APTA. Such an example is presented in Figure 5.2, where the lower right state of the depicted APTA is both accepting and rejecting. Such inconsistencies can be resolved by splitting the transition leading to inconsistent states according to its delay guard. In more detail, given a transition δ with a delay guard $g = [t_1, t_2]$, a split at time $t \in [t_1, t_2)$ would lead to the generation of two transitions with the same label as the original one, but with delay guards equal to $g' = [t_1, t]$ and $g'' = [t+1, t_2]$ respectively. Of course, in case of a split, apart from the inconsistent state split into two new states, the part of the APTA starting from the previously inconsistent state needs to be separated into two parts too. The first part is reached by timed strings that fire δ with a delay $t' \leq t$, and the second by those that fire δ with a delay $t' > t$. Finally, since RTI is based also on the red-blue framework, only transitions targeting blue states constitute split candidates.

The future of each state plays a pivotal role in the calculation of the evidence score used to evaluate the set of actions available at a given step of the learning process. In more detail, the derivation of the evidence value is driven by the intuition that suffixes (or tails) lying far away from each other are more likely to be separated by a future split operation. Thus, the evidence value is based on the amount of overlap among the tails of each blue node, with a high overlap being achieved when two tails share a common untimed representation and their probability of being pulled apart is low. Various heuristics attempting to express the aforementioned intuition into a measurable score has been proposed [87], ranging from naive untimed approaches, like the one used in the original EDSM setting, to more sophisticated ones taking into account the consistency value of both compatible (associated with same label) and incompatible (associated with different labels) pairs of tails. Every possible type of action is evaluated in every iteration of the algorithm, with the action recording the highest score being eventually performed. In case of equally scoring actions, preference is given first to a merge, then to a split, and finally to a color operation.

5.3. Learning from Positive Data: The RTI+ Algorithm

As it was discussed earlier in this thesis, in most real-life scenarios, when learning the behaviour of a system, traces from the normal execution of the system are mostly available. This is the case also in data extracted from network traffic, since it is much easier to collect NetFlow data from the benign operation of the examined system or network. It can be easily seen that in such cases the application of RTI for the purpose of extracting behavioural models of the system under examination is impossible. To deal with such issues, an extension of the original RTI algorithm, called RTI+, capable of learning solely from positive data, was introduced in [85]. This extended version models the input data with the use of probabilistic DRTAs (PDRTA), and constitutes the core of the multivariate approach used in the premises of this thesis to model the underlying behavioral patterns in sequences of NetFlow data. In the premises of this algorithm, the identification of PDRTAs able to effectively model the input data is dealt as the following two-fold problem: (1) the problem of identifying the correct DRTA structure, and (2) the problem of appropriately setting the probabilistic parameters associated with each transition of the model. Finally, a likelihood-ratio statistical test for evaluating the statistical difference between the tails of a pair of states is introduced as the new evidence value used to decide on the available actions during the determinization process.

Similarly to RTI, initially, an APTA consistent with the input data is generated. Of course, in the case of RTI+, the generated APTA will not include any accepting or rejecting states, since solely positive data are used for modelling. The rest of the learning procedure followed is identical to that of RTI, with the only difference identified on the evidence value used. As it was mentioned above, a likelihood-ratio test is used to determine the statistical difference between the suffixes of a pair of states, which as a matter of fact could provide insight on whether the examined states should be merged. Intuitively, a merge between two states with significantly different futures should be considered inconsistent, therefore it should not be performed. The likelihood-ratio statistical test used to test the null hypothesis that the suffixes of strings occurring after two different states follow the same PDRTA distribution is based on the concept of nested hypotheses. A hypothesis H is considered nested within another hypothesis H', if the possible distributions under H form a strict subset of the possible distributions under H'. In the context of a PDRTA identification process, these concepts can be mapped to the PDRTA distributions before and after a merge (or respectively after and before a split) is performed, since before the merge (respectively after the split) more parameters are present in the model (hypothesis H'), while after the merge (respectively before the split) a more compact model should be created (nested hypothesis H). Given these two hypotheses, and an input dataset S, the likelihood ratio test statistic

can be computed as

$$LR = \frac{likelihood(S,H)}{likelihood(S,H')} \tag{5.1}$$

, where the likelihood of each hypothesis in the context of RTI+ is calculated as the joint PDRTA distribution associated with each hypothesis, and is extracted through the use of the occurrence counts of the events in the dataset. Subsequently, the random variable $y = -2ln(LR)$ is compared to a χ^2 distribution and the p-value of this test is use to determine whether the null hypothesis should be rejected. Typically, if the p-value is lower than 0.05, the two hypotheses are considered statistically different with 95% confidence. This statistical evidence is used in RTI+ to decide on the possible set of operations in the following way: If there is a split resulting to a p-value lower than 0.05, the split with the lowest p-value is selected. If there is a merge resulting to a p-value greater than 0.05, the merge with the highest p-value is selected. If none of the previous actions is available, a colouring operation is selected. Finally, it should be mentioned that in the initial version of RTI+ only the most visited transition from a red state to a blue state was examined at each iteration, so that the run-time of the algorithm could be effectively reduced. As it will be seen in the final section of this chapter, such decisions can be tuned according to the conditions introduced by the facing problem.

5.4. The Multivariate Approach

In most cases, when modelling the behaviour of real-time systems, the extracted traces are composed of events characterized by various features. As a matter of fact, this is the case when dealing with NetFlow data, as it is better illustrated in Chapter 2. Each NetFlow record incorporates various features, ranging from the communication protocol used, and the duration of the flow, to the number of bytes and packets transferred between the source and destination IP addresses of the flow. Yet, state machines operate on a symbolic language, which drives most researchers using state machine learning approaches to develop a symbolic representation out of the original features by compressing the information provided by different feature values into a set of symbols [33, 34, 68]. It can be easily seen that when such symbolic representations are used, a great amount of attention should be paid on the mapping between the raw features and the symbolic alphabet, so that the produced symbolic sequence will be representative of the original multivariate time series. Yet, even if the obtained mappings are of high quality, there will be some information loss between the original traces and the symbolic ones, since each set of feature values is compressed into one symbol. To avoid such information loss, a model operating on the raw features associated with each record can be used.

The multivariate version of flexfringe offers such capability, by extending the process used for dealing with delay guards in the RTI+ algorithm to each numerical feature associated with the event types under examination. As in the case of timed symbolic event sequences, each feature of a timed event (a NetFlow record in the premises of this thesis) is dealt in the same way as the timing information in the original RTI+ algorithm, since, after all, time is just one numerical feature of the sequence. Again, solely positive data are used for modelling and the probability distribution of each feature is estimated independently through the use of histograms, as it was explained in Section 2.3.4. The automaton learning process is mostly identical to that of the RTI+ algorithm, with two primary differences: (1) there are multiple types of guards in each transition, with each one of them associated to a particular feature, and (2) there is no need for symbols, since each event is represented by the set of features associated with it. For completeness reasons, there is one symbol used in the learning process, yet its contribution to the identification procedure is null.

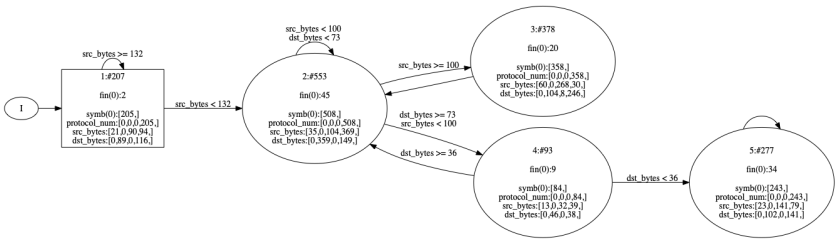


Figure 5.3: Example of a multivariate model extracted from NetFlows with 5 states and 3 attributes in each state

A sample multivariate model extracted from a sequence of Netflows is depicted in Figure 5.3. It can be easily seen, that three features of NetFlow records have been used in this model, namely the communication protocol, and the number of incoming and outgoing bytes. It should be noted that since the protocol is

a categorical feature (e.g. TCP, UDP, ICMP, etc.), a mapping to numerical values is used in the modelling process. Each transition in the depicted model is fired according to a set of feature guards, meaning that each of the conditions introduced by the guards should be met for the transition to be fired. For instance, in order to move from state 2 to state 4, the number of bytes sent by the source IP address should be less than 100, while the number of bytes received should be at least 73. On top of that, each node is accompanied by some additional information on the nature of records reaching the associated state, like the quantile distribution of each feature value (the quadruple next to each feature label), the number of traces ending in that state (the number next to the *fin* symbol), as well as the total number of records reaching the state (the number on the top of each node next to the # symbol). Another advantage of using such models, apart from the inclusion of the raw data in the learning process, can be identified on the interpretability originating from that exact inclusion of raw data features in the visualization of the extracted model. Finally, it should be pointed out that, as depicted above, the time parameter is not included in the model. The inclusion of time into the modelling procedure can be achieved with two ways, either explicitly by adding the difference between the timestamps of each consecutive pair of records as one of the features to be modelled or implicitly through the trace extraction process, which is the case in this thesis. More insight on the latter implicit inclusion of temporal information to the model is provided in Chapter 6.

5.5. Additional Constraints & Tunable Parameters

Flexfringe offers a great number of tunable parameters able to affect significantly the learning process. Most of these parameters have been developed as additional merge constraints that, given the nature of the problem, could enhance the learning process, or as a way to manually set the variables associated with the learning heuristic utilized. For instance, when the likelihood ratio test, mentioned before, is selected as the learning heuristic, the p-value used to decide on the action to be performed at each merging step of the learning process can be manually set according to the amount of confidence that is needed when making a decision. In this section the parameters primarily affecting the multivariate state machine modelling procedure are presented and discussed, so that the rationale behind the choices made can be transparently explained. These parameters can be seen as follows:

- **method**: This parameter sets the type of method to be used when selecting the action to perform at each inference step. The obvious choice would be to select the action providing the highest evidence value, yet, since each action accounts for local optimality, it could lead eventually to a suboptimal path. To deal with such issues, the random greedy selection technique was introduced in [38]. The basic functionality of this technique is the multiplication of the evidence value of each action by a value in $[0,1]$ drawn uniformly at random. Thus, this greedy procedure will sometimes try actions that do not seem locally optimal, under the rationale that these actions may lead to optimal paths. This technique has been selected also in the modelling procedure of the current thesis. It should be also mentioned that, when this random greedy technique is utilized, the number of times that it will be repeated to get more robust results can be also tuned.

- **largestblue**: Setting this parameter suggests that the learning algorithm should solely consider the most frequent candidate for a merge, rather than evaluating all possible candidates. Again this parameter can lead to significant improvements in the execution time, while also aiding the learning process since the most frequent candidate is associated with the greatest amount of information available at the given phase of the learning process.

- **blueblue**: This parameter can be used to enable the merge of a candidate (blue) state with another candidate (blue) state, instead of solely merging candidate states with identified red states. The use of such parameter alters the merging concept expressed in the red-blue framework, yet after some experimentation conducted on that parameter, it was observed that more concise models of the input NetFlow based traces were developed when this parameter was set.

- **extrapar**: Through this parameter, any variable, associated with the statistical tests performed by the learning heuristic used, can be manually set. In the premises of the current work, this parameter corresponds to the p-value of the likelihood ratio test, and was set to 0.05, as it is suggested in the work [85] introducing this likelihood heuristic.

6

Modelling Benign Flows

The goal of this thesis is to develop a system capable of extracting behavioral profiles from the benign network traffic of a computer network under examination, as well as leveraging these profiles towards the identification of malicious (or abnormal) traffic. The functionality of the developed detection system is based on the assumption that the behavior associated with malicious traffic should not conform to the profiles extracted from benign flows. The intention behind the design and deployment of the proposed pipeline was to create a system able to effectively identify malicious network behaviour, while operating on a privacy preserving abstraction of packet captures (NetFlows), and leveraging to the maximum extent both the temporal relations between sequences of flows and the multifaceted information accompanying NetFlow records. To do so, a sequential modelling approach was adopted through the use of multivariate state machines to capture the benign network behaviour, accompanied by the incorporation of three well-known anomaly detection algorithms in the structure of the derived state machines to further boost the detection process. In the sections to follow, the questions raised while developing this detection system are discussed, and the implementation details of the proposed pipeline, as well as the design choices made, are presented and explained.

6.1. Preliminaries

As it was mentioned in the introductory paragraph of this chapter, when designing the pipeline of the anomaly detection system developed in the premises of this thesis, multiple questions regarding the nature of the components comprising it, as well as their connection and integration into an end-to-end fully functional system were raised. Such questions should be transparently communicated to the reader for a better understanding on the design choices described in the following sections to be achieved. Thus, the most important questions needed to be answered during the system design and development process are enumerated as follows:

1. **What is the modelling entity?**

   This question refers to the way in which the data, the NetFlows in this case, should be abstracted, grouped, and analysed, so that a meaningful behavioral profile can be extracted for every group of data. Host and connection levels of analysis are the two obvious options in the case of NetFlow data, with each one associated with different advantages and disadvantages. In the current work both levels were initially examined, yet the host level of analysis was adopted in the experimental setting developed. More information about the characteristics, as well the advantages and disadvantages of each approach, are discussed in Section 6.2.1.

2. **In which way can NetFlow traces be extracted for each modelling entity, so that temporal relations existent in the data flows can be retained at the greatest extent?**

   After grouping the NetFlows according to the preferred level of abstraction, a set of temporal sequences (traces) shall be extracted and provided as input to the behavioral model extraction process. Since each flow is associated with a specific timestamp, a windowing technique can be used for extracting traces, yet, the choices concerning the implementation details of this technique are highly important towards the creation of meaningful behavioral models able to accurately capture the underlying behavioral patterns. For that reason, a sliding window technique along with a mechanism for dynamically adjusting

the window length and stride are employed, with more information on that technique being provided in Section 6.2.2.

3. **How will the behavioral profiles of the benign traffic be represented and extracted?**

   As it was explained earlier, multivariate state machines have been deployed as the sequential models attempting to capture the underlying behavioral patterns in the corresponding set of flow sequences of each modelling entity, while *flexfringe* [83] has been used to extract these multivariate models from the set of traces of each entity. Yet, this chapter will not focus on the learning algorithm, since such information was provided in Chapters 2 and 5.

4. **In which way can the extracted profiles be used to distinguish between benign and malicious traffic?**

   The final question to be answered refers to the training and testing phases of the developed system. After the extraction of the behavioral profiles from the benign flows comprising the training set, a mechanism shall be developed for the utilization of such models towards the goal of detection. In the current thesis, a replay approach involving both the training and testing NetFlow traces has been adopted as a part of the training and testing methodologies. In depth analysis of the training and test approaches followed can be found in Sections 6.2.3 and 6.2.4 respectively.

All the aforementioned questions are answered in the following section, where the designed pipeline is extensively presented and the implementation details are discussed and explained.

6.2. Main Pipeline Analysis

A high level presentation of the components comprising the pipeline of the proposed detection system can be seen in Figure 6.1. As it can be seen, the designed pipeline consists of two main sets of components. The first of these sets is associated with the training phase of the developed system and includes the components residing in the area defined by the blue dotted line in Figure 6.1, while the second set consists of the components associated with the testing phase and its members reside in the area defined by the red dotted line in Figure 6.1. In addition, there are two initial components used in both phases, which are mostly connected to the needed preprocessing of the input NetFlows, so that they can be transformed into a representation suitable for the learning process followed.

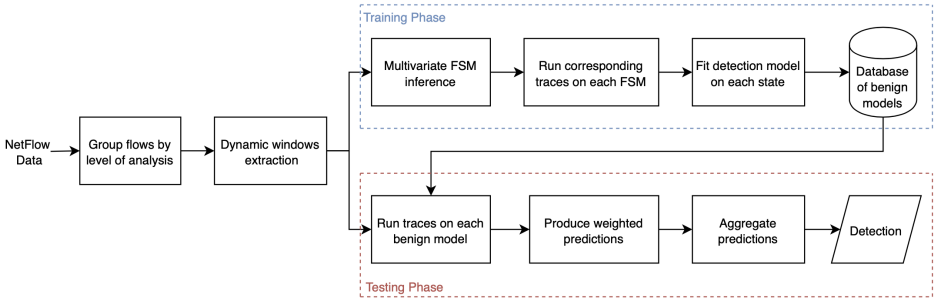


Figure 6.1: High level flowchart of the detection pipeline

Briefly, the procedure adopted during the training phase is the following: Initially, the input NetFlows of the training set are grouped either according to their source IP address in case of host level analysis or according to the pair of their source and destination IP addresses in case of connection level analysis. Subsequently, the dynamic windows extraction procedure is employed for each group of data, so that a set of traces from the NetFlows belonging to each group can be produced. These traces are fed into the multivariate version of *flexfringe*, so that the multivariate FSM capturing the behaviour of each group can be extracted. After the creation of the multivariate behavioral model, the corresponding traces are run (or replayed) on the directed graph representing the structure of the model. Through that process, each state of the derived multivariate FSM is associated with the subset of events leading to that state, and this subset of events is treated as a state-local training set on which three different models (LOF, Isolation Forest, Gaussian KDE) can be fitted. Finally, each multivariate FSM with its incorporated state-local detection models is stored as a reference benign behavioral profile.

The first two steps are the same also in the testing phase of the proposed system, as it can be clearly seen in Figure 6.1. After the extraction of the testing traces for each modelling entity (host or connection), these traces are replayed on all the behavioral models extracted during the training phase, so that state-local test sets can be assigned to each of their states in a similar manner as the state-local training sets are created. Subsequently, weighted predictions on the events included in the test set of each state are produced from the corresponding state-local model. After the collection of all predictions from a particular model, the ratio of benign to total predictions is used as an indicator of the benignity of the testing entity. Of course, all steps comprising both the training and testing phase of the developed pipeline will be discussed in greater depth in the sections to follow.

6.2.1. Levels of Abstraction

As it was mentioned earlier, when employing a detection solution that involves the creation of behavioral profiles of the NetFlow data, an abstraction level of analysis on that data needs to be applied. The obvious choices for such strategy constitute the host level and the connection levels of abstraction. The selection of one of these abstraction types is affected by multiple conditions, with the primary ones related to the nature of the detection purpose that the developed system aims to serve, as well as the available data. Of course, both levels of abstraction entail particular advantages and disadvantages, the significance of which should be properly weighted when selecting the type of analysis to adopt for the facing detection task. These exact advantages and disadvantages, as well as the choices made regarding the analysis level of the flows in the current thesis, are discussed in the following paragraphs.

The adoption of host level analysis means that the available NetFlows are grouped according to their source IP address, with the flows associated to each host used as input data to the behavioral model extraction process. Systems operating on such level of analysis perform also their detection process on hosts, rather than raw NetFlow data. For instance, such level of analysis would be appropriate for a detection system keeping track of possibly infected hosts through the retention of a blacklist, meaning that every time a host would be labelled as suspicious (or malicious), this host would be included in the blacklist and would be treated as infected by the system. Host level analysis offers a coarse-grained view on the flows, since the behavioral profile created by a group of flows incorporates multiple distinct connections of the host under modelling, including in that way multiple diverse individual behaviours of the host. In addition, in case that profiles of all modelling entities in a network are extracted and compared during the detection phase, as it is the case in the proposed system, modelling the hosts of the network offers a much more efficient modelling and detection procedure comparing to a connection level modelling approach, since the number of hosts is usually much smaller than the number of connections in a network. Finally, in order to create a behavioral profile out of a group of data, a sufficient amount of data points is needed for the created model to accurately capture the behavior of the modelling entity. It is evident that when hosts are considered, more flows can be provided as input to the learning algorithm comparing to the case of the connection level of analysis.

Of course there are also some disadvantages associated with the adoption of a host level of analysis. First, a behavioral model created from the set of flows associated with a host could incorporate significant amount of noise, since the different connections in which the host is included could demonstrate a quite diverse range of behaviours. In addition, a host-based modelling procedure suffers from the assumption that a host is either strictly benign or strictly malicious. Nevertheless, when a host is associated with both benign and malicious traffic, a host-based detection procedure might fail in identifying this partial behaviour. Especially, when the percentage of malicious traffic associated with a host is relatively low, such a scenario is highly probable, and of course undesired. In the case of the proposed system, there is an attempt of handling such situations by setting an adjustable decision threshold regarding the benignity of a host, when host level analysis is employed, with more information about this strategy provided in Section 6.2.4.

As far as the connection level of analysis is concerned, such an analysis involves grouping the available NetFlows by the distinct pairs of source and destination IP addresses, and creating a behavioral model for each of these pairs. Such a modelling procedure offers a fine-grained view on the data, since the behaviour associated to each connection is much less noisy comparing to that in the case of hosts. As a result, the extracted models offer specific views of the benign behavioral patterns existent in the examined NetFlows, possibly leading to the creation of models capable of identifying even abnormalities associated with a small portion of flows. Of course, not everything is ideal when adopting a connection level of analysis. As it was mentioned earlier, in order for the extracted multivariate models to adequately capture the behavioral patterns of a group of flows, without overfitting on that group of flows, a sufficient number of traces should be provided. When connections are taken into account, it is highly likely that most of these connections consist

of a relatively low number of flows, rendering the modelling procedure impossible. In fact, as it was illustrated in Chapter 4, in the majority of the datasets used in the premises of this thesis most connections are short-lived including less than 100 flows, with only few connections consisting of thousands of flows. In such cases, two approaches can be followed: either only the major connections will be used during training, which entails sufficient information loss, or even small connections are modelled, leading to many small, and possibly unreliable, models. The latter choice would also entail a significant burden in the detection module since a much higher number of models should be taken into account, comparing to the case of host based analysis. In the premises of this thesis, both levels of analysis were initially examined, yet due to the nature of the data the host level of analysis was eventually selected in the experimental procedure implemented.

6.2.2. Traces Extraction

After grouping the data flows according to the desired level of analysis, a set of traces consisting of subsequent flows grouped in a temporal manner should be extracted and provided as input to the FSM inference algorithm used for the production of the needed behavioral models. Such a trace extraction procedure involves two main considerations. The first one regards the selection of the NetFlow related features to be included in each event of each trace, while the second one concerns the mechanism used to temporally group the events into different traces. Both decisions affect significantly the quality of the modelling procedure, since the first decision affects the content of the multivariate sequences used to derive the behavioral model of each modelling entity, while the second decision affects the degree to which temporal patterns are properly mapped across the different set of traces extracted. For example, if traces included temporally unrelated flows, the creation of sequence models able to capture the actual underlying temporal behavior would be severely impeded.

As far as the first decision is concerned, different combinations of some basic NetFlow features, like the source and destination ports, the protocol used, the number of bytes sent and received, and the duration of a flow, were used, avoiding the incorporation of other highly sophisticated features, aiming in that way to create a system able to operate on features that are as universal as possible among flow representations. A deeper discussion on the different combinations of NetFlow features used in the modelling process can be found in Chapters 4 and 7, where the nature of the data and the experimental procedure respectively are presented and analysed. Regarding the temporal grouping of NetFlows, sliding window approaches are commonly employed in works [33, 68] related to similar tasks. In most cases, windows of static length and stride are rolled upon a timely ordered set of flows in order for the flows residing inside the same window to be concatenated into a sequence of timed events comprising each trace. Such a procedure is preferred mostly due to its simplicity, yet it can potentially lead to unfortunate results in the modelling process.

The main implication in using a static windowing technique rises from the fact that NetFlow records tend to appear in a non-periodic fashion, meaning that within a timely ordered set of flows, there could be temporal regions with high volume of traffic, as well as sparse regions with few dispersed flows. In such cases, if a windowing technique with a static time window was to be employed, traces with highly imbalanced "temporal load" would be extracted. In particular, there would be many traces with few timed multivariate events, corresponding to the sparse temporal regions, accompanied by few traces with a high number of records, corresponding to the dense temporal regions. On top of that, when sliding a static time window over a sparse temporal region, it is highly likely that multiple windows will include the same data points, since the stride of the window would not be able to immediately push the window into capturing new data. Traces of such nature are highly undesired, since the highly loaded traces would tend to dominate in the multivariate inference process leading to the creation of models ignoring a significant number of traces. An easy solution to such an unfortunate scenario could be expressed through the adoption of a static window grouping a predefined number of subsequent flows, so that all traces would be of the same length regardless of the temporal distance within the flows in each trace. Such a technique, though, suffers from the obvious drawback of disregarding the actual temporal information associated to each flow. It can be easily understood that such a technique could potentially group temporally distant flows, leading to noisy traces, and therefore noisy models.

As it was illustrated in Chapter 4, the datasets considered in the evaluation phase of this work demonstrate the aforementioned characteristics in their temporal nature. Thus, to deal with any of aforementioned shortcomings of the static windowing techniques, a mechanism for extracting traces with sliding windows able to dynamically adjust their length and stride according to the observed temporal contention is developed. The functionality of this mechanism is illustrated in Code Snippet 1, and is further explained as follows. The main two considerations taken into account when designing this dynamic windowing mechanism regarded the extraction of windows incorporating a sense of well-distributed traffic load, and in parallel respecting the tem-

poral distances between consecutive flows. To facilitate the first concern two manually set limits regarding the minimum and maximum number of flows allowed in a window were utilized, while the second concern was dealt with through the introduction of a high level split in non-overlapping windows and a dynamically adjustable striding technique adopted in sparse temporal regions. Thus, the developed mechanism starts by sorting in ascending time order and splitting the provided flows into non-overlapping high level windows according to the observed time difference of consecutive flows, before proceeding into a dynamic processing of each of these windows independently.

The intuition behind the aforementioned decision is based on the fact that flows lying temporally far from each other should be separated and not included in the same window. After temporally clustering the input flows, each cluster (or high level window) is being individually processed. Initially, the window length is set proportional to the median time differences of the flows residing in the cluster, while the stride was set as the 1/5 of the window length, so that overlapping traces could be extracted through the sliding window process. While sliding the window over the flows of the temporal cluster, three primary conditions are repeatedly evaluated:

1. **The new traces condition**: This condition is evaluated in lines 12-19 of Code Snippet 1 and aims in dealing with sparse temporal regions, in which the sliding process could move slowly leading to the extraction of consecutive traces with the exact same information captured within their content. To deal with such unfortunate occasions, a condition regarding the incorporation of unseen flows in the extracted trace at every phase of the trace extraction process is evaluated. If there is no new information included in the trace, the trace is discarded and the stride of the window is increased, so that a previously unseen flow can be reached faster. Such an approach runs the risk of skipping unseen flows in case the stride is increased excessively. To avoid such a situation, if the index of the first unseen flow encountered in the process is not equal to the highest index seen so far incremented by one, the beginning of the window is reset to match the flow with the aforementioned index. In that way, the integrity of the process is ensured.

2. **The maximum trace length condition**: This condition is evaluated after a window with new information has been identified and can be seen in lines 22-27 of Code Snippet 1. This condition aims in keeping the length of the window lower than a predefined number of flows, so that the extraction of extremely long traces in temporally dense areas can be avoided. To do so, as long as a trace, the length of which is higher than the maximum permitted length, is encountered, the window is decreased by a dynamically adjustable scale (initialized at 2, meaning that initially the window length is repeatedly decreased by half).

3. **The minimum trace length condition**: This condition is complementary to the aforementioned one and is evaluated in lines 28-33 of Code Snippet 1. It can be easily understood that this condition aims in maintaining the length of the window higher than a predefined number of flows, so that extremely small traces are not extracted. To do so, as long as a trace, the length of which is lower than the minimum permitted length, is encountered, the window is increased by a dynamically adjustable scale (initialized at 2, meaning that initially the window length is repeatedly doubled).

As it can be seen in lines 21-35 of Code Snippet 1, the last two conditions must be met in parallel, thus they have been encapsulated in a while-loop aiming to enforce both of them. Such an approach runs the risk of leading to an infinite loop, in which the maximum trace length condition decreases the window length too much, and in parallel the minimum trace length condition increases the decreased length proportionally much, not enabling the process to converge in an acceptable window length. To avoid such an unfortunate situation, the scale used to increase or decrease the window (initially set to 2) is also slightly decreased after each unsuccessful run of the outer while-loop (starting at line 21 of Code Snippet 1) until it reaches the value of 1, which would mean that the window is no longer changed. If that condition is met, meaning that an acceptable window was not found, the window associated with the closest length to the specified limits is selected. Finally, it should be pointed out that in all of the above conditions the stride of the window should be kept lower than the length, so that there is no possibility of flow loss in the extraction process.

As a final step of the trace extraction process, the developed mechanism offers the possibility of using another aggregation window inside the window identified by all the steps presented above. The application of aggregation windows is controlled through a flag provided in the trace extraction mechanism. In case this flag is set, non-overlapping aggregation windows of one second (or the length of the outer window in

Algorithm 1 Dynamic Windows Extraction

Input: A list with timestamped flows $flowsList$; The minimum permitted trace length $minTraceLen$; The maximum permitted trace length $maxTraceLen$; A flag for producing aggregation windows within each window $aggregationFlag$

Output: A list of traces $traceList$

Sort $flowsList$ in ascending time order
$highLevelWindows \leftarrow$ Extract high level non-overlapping time windows from $flowsList$
$traceList \leftarrow [\,]$
for $highLevelWindow : highLevelWindows$ **do**
    $medianDiff \leftarrow$ Calculate the median time difference between flows in $highLevelWindow$
    Initialize $windowLen$ and $windowStride$ as multiples of $medianDiff$
    $currTrace \leftarrow [\,]$
    $seen \leftarrow$ empty set
    $prevFlows \leftarrow null$
    while not all flows of $highLevelWindow$ in $seen$ **do**
        $currFlows \leftarrow$ Apply the time window of $windowLen$ on the flows of $highLevelWindow$
        while $currFlows == prevFlows$ **do**
            Increase $windowStride$
            $currFlows \leftarrow$ Apply the time window of $windowLen$ on the flows of $highLevelWindow$
            if $windowStride$ surpassed last seen flow **then**
                Set the start of the window at that flow
            end if
            Ensure that $windowLen \geq windowStride$
        end while
        $numWindowFlows \leftarrow count(currFlows)$
        while $numWindowFlows < minTraceLen$ or $numWindowFlows > maxTraceLen$ **do**
            while $numWindowFlows > maxTraceLen$ **do**
                Decrease $windowLen$
                $currFlows \leftarrow$ Apply the time window of $windowLen$ on the flows of $highLevelWindow$
                $numWindowFlows \leftarrow count(currFlows)$
                Ensure that $windowLen \geq windowStride$
            end while
            while $numWindowFlows < minTraceLen$ **do**
                Increase $windowLen$
                $currFlows \leftarrow$ Apply the time window of $windowLen$ on the flows of $highLevelWindow$
                $numWindowFlows \leftarrow count(currFlows)$
                Ensure that $windowLen \geq windowStride$
            end while
            Ensure that the loop is not infinite after an amount of unsuccessful attempts has been made
        end while
        if $aggregationFlag == True$ **then**
            Append $currFlows$ to $currTrace$
        else
            $aggFlows \leftarrow$ Create aggregated view of $currFlows$
            Append $aggFlows$ to $currTrace$
        end if
        $prevFlows \leftarrow currFlows$
        Slide time window
    end while
    Append $currTrace$ to $traceList$
end for
return $traceList$

case that length is less than one second) are applied on the flows of each time window, with the features of all flows inside each aggregation window being compressed to one value leading to an aggregated view of these flows. This aggregation is conducted by using the median values of numerical features and the mode of categorical ones. This functionality is inspired by the work conducted in [28] and aims in serving a two-fold purpose. First, by compressing the flows into aggregation windows the average length of the traces extracted is reduced, which as a matter of fact leads to a proportional reduction in the computational time needed for the multivariate model extraction process that follows the dynamic trace extraction phase. The second and most important reason justifying the adoption of aggregation windows lies on the fact that through this process robust estimates of the feature values observed in a sequence of flows are extracted, rendering the model extraction process less prone to outliers, and subsequently overfitting to the input traces. In fact, without the application of aggregation windows there was a high number of cases when *flexfringe* could not converge to a model able to represent the input sequential multivariate data, especially when significantly long traces (in the order of thousands) were considered.

After implementing this dynamic window extraction technique, it was considered beneficial to quantify its contribution to the learning and detection process by comparing the detection performance of the designed system using that technique to that achieved when the two aforementioned static window techniques are employed. To carry out this comparison, one of the datasets included in this thesis, the CTU-13 dataset, was utilized, and an experimental process similar to the one presented in Chapter 7 was followed. Yet, since the main purpose of these experiments was to quantify the appropriateness of the proposed dynamic window extraction technique, the experimental procedure adopted was simplified comparing to the one used for the rest of the experiments carried out in this thesis. In particular, a feature selection step was not incorporated in the evaluation process, meaning that all the NetFlow features identified through the feature exploration process (source and destination port numbers, protocol, duration, source and destination bytes) were included, while the baseline detection technique introduced in Chapter 7 was fitted in each state of the multivariate FSMs used by the designed system. It should be pointed out that the detailed information regarding the training and testing phases of the designed methodology, as well as the experimental configuration are later presented in this thesis. Yet, for reasons concerning the reading coherence of this thesis, it was decided to include the results of the window related experiments in the current section.

Scenario	Dynamic timed windows				Static non-timed windows			
	TP	TN	FP	FN	TP	TN	FP	FN
1	1	4	1	0	1	4	1	0
2	1	2	0	0	1	2	0	0
3	1	4	1	0	1	4	1	0
4	1	4	0	0	1	4	0	0
5	1	3	0	0	1	3	0	0
6	1	3	0	0	1	3	0	0
7	1	3	0	0	1	3	0	0
8	1	4	0	0	1	4	0	0
9	10	4	1	0	10	4	1	0
10	10	4	1	0	10	4	1	0
11	2	1	1	0	2	1	1	0
12	3	3	0	0	3	3	0	0
13	1	4	0	0	1	4	0	0
Total	34	43	5	0	34	43	5	0

Table 6.1: Comparative results on the CTU dataset only for major hosts when two different window extraction techniques are used

As far as these experiments are concerned, two major observations were made. First, the static timed window extraction technique was proven to be 1000 times slower than the dynamic extraction technique introduced in this section, which as a matter of fact is associated with the ability of the latter technique to better handle sparse temporal regions, providing in that way a significant boost in the computational efficiency of the entire detection methodology . In fact, the substantial overhead, that the static timed technique entailed, led to its removal from the rest of these experiments. Second, the rest two techniques produced the same detection results when applied to the designed methodology and evaluated on the CTU-13 dataset, as it can be seen in Table 6.1. This last observation justifies the adoption of the dynamic window extraction technique in the traces extraction process of the designed system, since it performed at least as well as the

static non-timed window technique on the evaluated dataset, while taking into account the temporal nature of NetFlows, a characteristic that could turn out to be valuable in the rest datasets evaluated in the premises of this work.

6.2.3. Model Extraction and Training

After extracting a set of traces for each modelling entity (host or connection), the multivariate version of *flexfringe* is employed to produce the sequential model representing the network behavior of each entity. The learning algorithm used to model the extracted multivariate traces was covered in depth in Chapter 5, thus there will not be a similarly extensive presentation of the process in this chapter too. Rather, the attention will be focused on the training methodology applied on the produced multivariate FSMs, so that behavioral profiles able to capture the underlying temporal patterns of the benign traffic associated with each modelling entity can be extracted. The primary motivation behind the design of the training, as well as the testing methodologies, lies on the perception of the multivariate state machines extracted trough *flexfringe* as temporal clustering mechanisms. In particular, each state of the multivariate state machine can be perceived as a temporal cluster capturing similar NetFlow activity. Thus, if the input traces used to derive a multivariate FSM are replayed onto the directed graph structure of the FSM, each state can be associated with the individual flows crossing it during the replay procedure, assigning in that way a local training set of flows to that state.

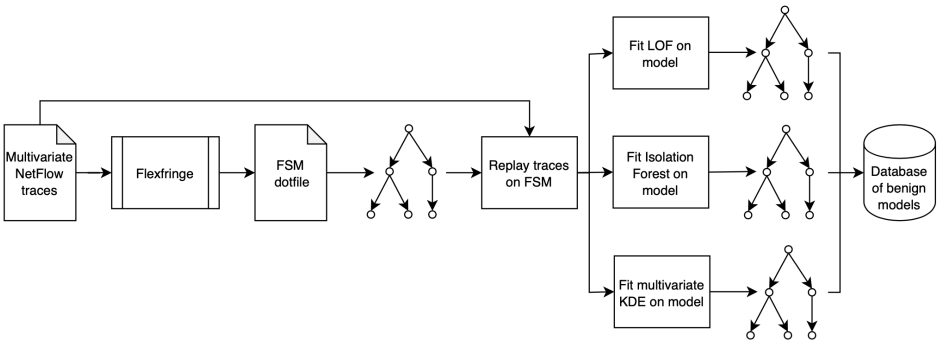


Figure 6.2: High level flowchart of the training pipeline

In Figure 6.2 a more detailed presentation of the designed training pipeline is depicted. It can be seen that after feeding *flexfringe* with the extracted multivariate traces, a dot[8] file of the inferred FSM structure is produced. This file is parsed and the obtained multivariate model is fed with its corresponding NetFlow traces. After replaying the multivariate traces associated with each modelling entity on the corresponding FSM, three well-known anomaly detection algorithms (LOF, Isolation Forest, Multivariate KDE) are fitted on each state-local training set of the FSM. In that way, each state can be converted into a temporally aware predictor trained on the state-specific benign behavior represented by the flows comprising the corresponding local dataset. At this point it should be mentioned that the functionality of these three models was analysed in depth in Chapter 2, thus it will not be covered again in this chapter. As it was mentioned earlier in this thesis, these three detection algorithms are selected, since they are extensively used in the anomaly detection literature, while originating from different machine learning disciplines. Thus, it was considered beneficial to evaluate the appropriateness of incorporating such techniques in the structure of a sequential model towards the goal of detection. Following this state-local detection model fitting process, all three versions of each multivariate FSM are stored in memory, so that they can be used for matching any unseen benign network traffic behaviour examined during the testing phase. Thus, the result of the training phase is a "database" of benign behavioral profiles, which is accessed during test time to identify any non-matching behaviour as potentially anomalous.

6.2.4. Model Testing

In the testing phase, a methodology similar to the one used during training is adopted. As in the training phase, the flows comprising the test set are first grouped according to the preferred level of analysis (host or

[8] `https://graphviz.gitlab.io/_pages/doc/info/lang.html`

connection), and then sorted in an ascending timely order. Subsequently, the flows belonging to each modelling entity are provided as input to the dynamic trace extraction process presented in Section 6.2.2, so that a set of multivariate testing traces can be extracted for each entity. At this point, the testing phase is decoupled from the training phase, since, instead of inferring a multivariate model out of each set of traces, each set is replayed on every multivariate behavioral model stored in the "database" created during the training phase. Similarly to the replay procedure followed in the training phase, the testing replay process aims in deriving state-local test sets for each modelling entity on every benign behavioral model extracted. Through this process, the detection model fitted in each state can be used to make predictions on its local test set, so that outlying points can be identified in each state. Finally, all these local predictions can be aggregated into a final detection result on the modelling entity under examination.

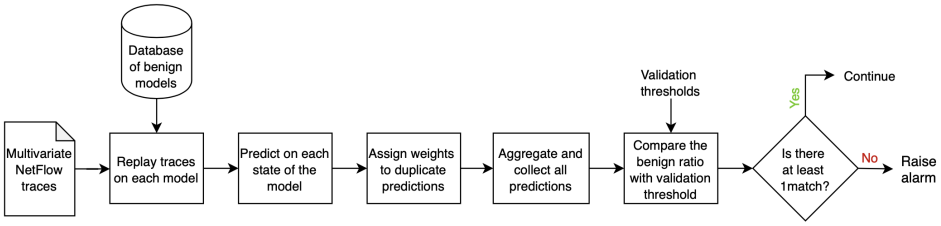


Figure 6.3: High level flowchart of the testing pipeline

The testing pipeline is better illustrated in Figure 6.3. It can be easily seen that, after each state-local model provides its predictions on the test set captured by the state on which it has been fitted, a weight is assigned to duplicate predictions. This step is included in the prediction process, since it is possible that the same flow has been captured by multiple states due to the sliding nature of the windowing technique used to extract the traces. As a result, if a flow has been included in the test sets of multiple states, multiple predictions on that flow will be produced. As a result, these predictions need to be combined into one final prediction for the flow under examination, especially if the predictions include labels from both classes (normal or outliers). A straightforward solution to that problem would be to assign the majority class of the individual predictions as the final prediction on each flow. Yet, such solution does not take into account the confidence behind each individual prediction, meaning that it would be possible that the majority predicted class was derived by state-local models of low detection confidence. For example, state-local models trained on a low number of flows provide less confident predictions comparing to models derived from a larger state-local dataset. To take into consideration such parameters, a weighted average approach was employed. Each state s_i of the behavioral model was associated with a weight equal to $w_{s_i} = \frac{num\_of\_training\_instances_{s_i}}{\max_{s_j \in S} num\_of\_training\_instances_{s_j}}$, where S is the set of all states in the model, and $num\_of\_training\_instances_{s_i}$ is the number of training instances in state s_i. This weight is associated with every prediction made on a state, and the final prediction for a flow f spanning a set of states S_f is calculated as the weighted average of the individual predictions on each state $s_f \in S_f$, as it can be seen in Equation 6.1 below.

$$prediction_f = \frac{\sum_{s_f \in S_f} w_{s_f} \times prediction_{s_f}}{\sum_{s_f \in S_f} w_{s_f}} \tag{6.1}$$

After the aggregated predictions of all flows comprising the testing behaviour of the entity (host or connection) under examination are calculated, a final prediction on the benignity of that entity shall be produced. To do so, the ratio of the flows identified as benign to the total number of flows associated with the examined entity is used as an indicator of the potential match between the behavior of the examined entity and the benign profile captured by the behavioral models learnt during training. A match between a testing entity and a benign model is considered to be successful if the predicted benign ratio is greater than a predefined decision threshold. This threshold is dependent on the detection algorithm used in each model (LOF, Isolation Forest, multivariate KDE) and can be either manually set according to the number of alarms that a system is able to handle or through a validation procedure on the training traces of each model. If the latter option is selected, then each inferred model is fed with the traces used during its training phase and the threshold for each model can be tuned according to the benign ratio of the predictions made on these training traces. More information on this tuning procedure is provided in Chapter 7. Finally, since the traces of each testing entity are replayed on every benign model extracted during the training phase, a decision rule

must be defined regarding the benignity of each entity. In the premises of this work, an *all-anomalous/any-benign* rule is adopted. Since each behavioral model is learnt on benign traffic, it is assumed that if at least one benign match is found during testing, then the examined entity is considered benign, otherwise an alarm is raised for that entity. This strategy implies that the detection system needs to be highly confident that an anomaly has been found to raise an alarm, and aims in reducing the number of false alarms, which is a common problem among anomaly detection systems trained on the normal data points, as it was explained in Chapter 2.

6.3. Some Thoughts on Complexity

Before elaborating further on the computational complexity associated with the operation of the designed detection system, it should be clarified that the work conducted in the premises of this thesis was not oriented primarily towards computational performance. As a result, multiple choices made during the design and the deployment process could be improved towards the pursuit of higher efficiency. In fact, Chapter 8 includes a brief discussion on such potential improvements. Of course, the aforementioned statement does not imply the fact that the designed system does not provide adequate computational performance. This section is utilized towards the provision of some rough insights regarding the complexity of each phase of the designed pipeline, with the attention focused on the trace extraction process, the model creation with the use of *flexfringe*, as well as the training and testing phases. Finally, it should be pointed out that big-O estimations are not included in this section, rather a rough overview of the complexity is presented.

Starting the discussion with the dynamic trace extraction procedure, it should be pointed out that such a process can easily turn into a significant performance bottleneck for the whole system, given the fact that it is applied before both the training and the testing phase, while the proposed system handles data points with irregular frequency of appearance. Yet, the trace extraction process is equipped with mechanisms for handling potential edge cases associated with steep changes in the way that flows appear, either by spanning fast through sparse temporal areas or by identifying and breaking potential infinite (or close to infinite) loops associated with the dynamic nature of the process. As a result, the trace extraction process adds minimal computational overhead to the pipeline. As far as the FSM inference process is concerned, it was observed that there was a significant overhead added when sets of long traces (in the order of thousands) were provided. As it was discussed in Section 6.2.2, this problem was overcome through the use of aggregation windows in the trace extraction procedure. Of course the inclusion of this technique entailed a computational overhead on the trace extraction process itself, yet this overhead was negligible comparing to that associated with the provision of long traces to the inference tool.

Following the model inference, the training process is computationally dominated by the three factors: (1) the detection model used, (2) the number of states in the FSM structure, and (3) the number of flows captured in each state. It is obvious that the more states and flows involved, the higher the computational overhead, which especially in the case of the LOF model could lead to some undesired results, since it constitutes a nearest neighbour technique operating on the set of pairwise distances between all instances of the training set. Lastly, the complexity of the testing phase is primarily associated with the number of benign models available for comparison, since every trained behavioral model is taken into account. As a result, the more benign models extracted in the training phase, the more comparisons are needed during the testing phase. This problem could be partially solved if solely a part of the benign models were evaluated for a potential match during testing, yet a robust mechanism regarding the selection of these models should be implemented.

7

Experiments

After designing the detection system presented in the previous chapters, the establishment of an evaluation procedure to quantify its detection performance is considered necessary. As it was explained in Chapter 4, three publicly available datasets containing NetFlow captures from real network traffic were utilized to evaluate the effectiveness of the detection pipeline. These datasets include a wide variety of cyber attacks, ranging from bots, and DDoS attacks, to port scans and SQL injections. The purpose of the evaluation procedure is fourfold. First, the detection performance of the system needs to be evaluated, since, as it was discussed in Chapter 2, a robust and effective detection system should be characterized by high detection rate and a low number of false alarms. Second, it is desired for the proposed system to detect as many types of attacks as possible. To that end, datasets including a diverse set of attacks are selected, so that the evaluation procedure can be as representative as possible of the actual detection performance of the system. Third, the performance of the proposed system is compared to a series of baseline detection methods, to evaluate the impact of its learning module regarding the goal of detection. Finally, it was considered beneficial to additionally compare the detection performance of the proposed methodology to that demonstrated by a state-of-the-art detection technique operating on one of the datasets used in this thesis, so as to further evaluate the detection potential of the proposed system in contrast to one recently developed and published technique. The configuration of the experimental environment and procedure, as well as the obtained results, are discussed in the premises of this chapter.

7.1. Experimental Configuration

The experimental procedure followed in the premises of this thesis resembles the typical evaluation procedure of machine learning tasks, in the sense that each dataset was split into a training set, a validation set, and a test set, with the experimental procedure being divided between the training phase, the tuning/validation phase, and, finally, the testing phase. A descriptive presentation of the formation of these sets is provided in Section 7.2, yet an initial brief illustration is included in this paragraph too. The training set was used solely during the training phase of the system, while the two latter sets were used during the testing phase. In particular, the training set consisted only of benign network traffic, and was utilized to create a "database" of benign behavioral profiles by extracting multivariate FSMs from the NetFlow data, and using this data to fit the detection models on each FSM. As far as the validation phase is concerned, two types of validation sets were used. The first type consisted of benign traffic too, and was provided as a testing input to the designed system, so that the parameters related to the learning process could be further tuned towards a low false alarm rate. More information on this tuning procedure is provided in Section 7.4, thus it will not be further discussed in this section. The second type comprised of mixed traffic and was utilized in the baseline comparison experiments, so that the best performing configuration setting of the proposed system could be selected. Again, more information regarding this process is reported later within this chapter, in Section 7.5 in particular. Finally, the testing set contains both benign and malicious traffic too, and is used to quantify the performance of the detection methodology.

One important aspect of the experiments regards the selection of the level of analysis of the NetFlow data. As it was discussed earlier, the proposed system is able to operate on both host and connection levels of analysis, yet the selection of the preferred level is based on the nature of the input data, and the wanted level of

detection granularity. For example, if the connection level of analysis is selected, a more fine-grained detection procedure can be adopted, since the benign behavioral models derived from the traffic recorded within each benign connection will incorporate much less noise comparing to the models extracted from the communication traffic of each host in the network. On top of that, by operating on connection level, the detection results can be better isolated and interpreted comparing to the results obtained for each host, especially when a host is associated with both benign and malicious behaviors, with the malicious traffic constituting a low minority of its recorded flows. Yet, in order to adopt a connection level detection strategy, a sufficient number of flows shall be available both for the connections used in the training phase of the system, and those used in the testing phase. It can be easily understood that a connection with tens or few hundreds of flows cannot be utilized for the extraction of a robust behavioral profile, while, in case such a connection is encountered in the test set, it is highly doubtful whether the obtained detection results regarding that connection are reliable. In such case, it is preferable to conduct the detection procedure on host level, since the communication traffic associated with each host is definitely of higher (or equal) volume to that associated with the connections of each host. As it was explained in Section 4.2, this is the case for the majority of connections in the datasets used, thus it was decided to evaluate the designed system on host level.

After selecting the level of analysis to be used in the detection procedure, the type of experiments to be included in the evaluation procedure shall be discussed. There are primarily four categories of comparisons performed within the experimental process. Initially, the contribution of the detection algorithms embedded in the FSM structure shall be evaluated. As a result, the performance of the detection system per detection algorithm used for the creation of the benign communication profiles is evaluated. In that way, meaningful conclusions can be drawn regarding the contribution of each detection algorithm to the learning and detection procedures. Secondly, the importance of different NetFlow feature sets regarding the goal of detection is evaluated. As it was mentioned earlier in this thesis, only basic NetFlow features are used in the multivariate learning procedure, in an attempt to render as universal as possible the application of the designed system. Two of the utilized datasets included a wide variety of features derived from NetFlows, yet solely the set of features that can be easily found in most NetFlow datasets is selected. Apart from the universality of the model, this tactic aids in the creation of simple models and increases the interpretability level of the designed system. Bearing this in mind, and taking the feature exploration presented in Section 4.3 into account, the following four distinct feature sets are evaluated:

- **Feature set 1**: protocol, source bytes, destination bytes
- **Feature set 2**: destination port, protocol, source bytes, destination bytes
- **Feature set 3**: source port, destination port, protocol, source bytes, destination bytes
- **Feature set 4**: destination port, protocol, duration, source bytes, destination bytes

The third category of experiments regards the evaluation of the detection impact of the designed system comparing to that of other baseline detection methodologies. This type of experiments aims to evaluate the impact of the key characteristics of the designed system, like the detection techniques incorporated in the FSM structure, as well as the multivariate and sequential nature of the models, towards the goal of detection, by comparing its detection performance with baseline methodologies disregarding some or all of these characteristics. The last category of experiments includes the examination of the detection performance of the proposed methodology in contrast to a state-of-the-art detection technique introduced in [10], and evaluated on the CTU-13 dataset. By comparing the desinged system to such a technique operating on one of the datasets used in this thesis, the extent of the contribution made through this work on the field of anomaly detection in NetFlow traffic can be better determined. More information about the configuration, as well as the results, of these experiments is provided in the sections to follow. Finally, a brief presentation of the tools used for implementing the proposed detection system, as well as the evaluation procedure, shall be provided. Most components of the proposed system, the data exploration and preprocessing procedures, as well as the experiments conducted, were implemented in Python. The only component of the system that did not conform to this pattern regards the multivariate state machine inference process, for which *flexfringe* was used. *Flexfringe* is a tool implemented in C++, thus a Python wrapper was developed to invoke it from a Python script. In addition, the well-known *sklearn* and *scipy* Python libraries were leveraged for the implementation of the detection algorithms incorporated within each multivariate FSM, with *sklearn* being utilized for LOF and Isolation Forest, and *scipy* offering the Gaussian KDE implementation. Finally, it should be mentioned that all experiments were conducted locally on a MacBook Pro with a 16GB memory and a 2.6GHz 6-Core Intel

Core i7 CPU. All the code associated with the development of the proposed system, as well as the experimental procedure followed, can be publicly accessed through `https://github.com/SereV94/MasterThesis`

7.2. Training and Test Sets Selection

A decision that is capable of affecting extensively the performance of the designed detection system, as well as the quality of the experimental evaluation procedure regards the division of the available data into a training and a test set. Similarly to any machine learning task, the training set will be used for both learning the multivariate behavioral profiles from a sample of the available benign traffic, and tuning the learning parameters of the proposed methodology, while the test set will be used to evaluate the detection performance of the followed methodology and hopefully provide answers to the research questions of the current work. The main criteria used for splitting the datasets presented above into these sets are the following: (1) the number of benign flows in each capture, and (2) the timing of each capture. In more detail, given the fact that the proposed detection system is designed to learn behavioral profiles solely from benign network traffic, captures with a high number of benign flows should be utilized as the training set of the system for each dataset. In case that there are many captures with a similar amount of benign traffic, then the timing information incorporated in NetFlows is taken into account by selecting captures in an ascending timely fashion. Following that strategy, the training and test set selection was conducted as it can be seen in Table 7.1.

Dataset	Training set	Test set
CTU-13	Benign flows from Scenario 3	Remaining Scenarios + Malicious flows from Scenario 3
UNSW-NB15	Benign flows from Scenario 1	Remaining Scenarios + Malicious flows from Scenario 1
CICIDS2017	Monday	Rest Days

Table 7.1: Training and Test sets split for each dataset

For the CTU-13 dataset, the 3rd scenario was used for the training phase of this work, since it contains the highest number of benign flows among all scenarios. At this point it should be mentioned that the benign flows of other scenarios could have been selected additionally to extend the training set of the system for this dataset, yet it was considered beneficial to see whether it was possible to acquire high detection performance while avoiding the computational burden that an increase in the size of the training set would entail. For the UNSW-NB15 dataset, the 1st scenario was utilized for training purposes mostly due to the timely precedence of its flows. In this case, the timing information was given priority since most scenarios contained a similar number of benign flows. As far as the CICIDS2017 dataset is concerned, the Monday's capture comprised the training set for this dataset since it consists solely of benign traffic, while constituting the earliest capture of the dataset. Finally, it should be pointed out that portions of the training and test sets presented above were utilized also as validation sets in the experimental procedure followed. As mentioned earlier in this chapter, two types of validation sets were used. The first type was derived by further splitting the training set of each dataset in a 80/20 timely fashion into a pure training set and a validation set meant for parameter tuning. The second type consisted of both benign and malicious flows collected from the test set of each dataset, and was used in the baseline comparison part of the experimental procedure followed within this thesis. More information on both use cases of the validation sets can be found later in this chapter.

7.3. Baseline Methods

As it was mentioned above, in the premises of the third type of experiments conducted, the proposed methodology was compared to some baseline methods in order to evaluate the contribution of its primary characteristics in the detection process. As a result, the best performing behavioral model identified through the previous two experimentation categories is compared to three baseline methods with different characteristics. First, the impact from the incorporation of the three selected detection algorithms (LOF, Isolation Forest, Gaussian KDE) in the FSM structure is evaluated. To do so, the same methodology as the one followed in the proposed system is used to create multivariate behavioral profiles from benign data, with the only difference being the fact that the aforementioned sophisticated detection algorithms, fitted in each state of the models, are replaced by a much simpler detection technique. This technique is based on the calculation of the distance between a given testing data point and the mean of the training data points residing in each

state after the replaying procedure is completed. In more detail, during the training phase of the system the mean and the standard deviation of each feature of the flows composing the state-local training sets of each multivariate behavioral model are calculated. Subsequently, the absolute distance between the mean of each feature and the corresponding feature value of the state-local testing flows is compared to the recorded standard deviation multiplied by a threshold value. If the product of this comparison is positive in at least one of the examined features, then the given testing flow is considered as anomalous by the state in which the detection process is undertaken. This decision is made under the assumption that any positive results would suggest that the given testing flow lies "far" from the mean of benign flows captured by the current state within the context of the features producing these positive comparison results. As mentioned earlier, this method adopts the exact same pipeline as the original system, apart from the detection technique used in each state, thus it is referred as the baseline behavioral method for the rest of this thesis.

The second baseline method used in this type of experiments aims to evaluate the adoption of a multivariate approach in the detection methodology employed within the designed system. Thus, a similar approach to the one followed is adopted with the difference being the creation of behavioral profiles through the extraction of simple symbolic state machines, instead of multivariate ones. To do so, a symbolic encoding should be derived from the features associated with each event in the set of traces extracted from the NetFlow data. The strategy employed for the creation of this encoding is the same as that described in [68]. In particular, the categorical features, like the communication protocol, are simply assigned a unique numerical value, as it was explained in Section 4.2, while percentile clustering is used for the numerical features included in the detection process, like the bytes transmitted, and the port numbers. The ELBOW method [30] is utilized to select the optimal number of percentiles for each numerical feature according to the identification of a "break point" in the Within-Cluster Sum of Squares (WCSS) metric calculated for a selected range of bins (1 to 10 in this case). After this optimal number of percentiles is identified for every numerical feature, each feature value is assigned the index of the percentile in which it resides. For example, if for a given feature v the optimal number of percentiles is 4, then the feature values residing before the 25th percentile will be assigned with 0, those residing between the 25th and the 50th percentile will be assigned with 1, etc. This process is called discretization and is used to provide a symbolic representation to the recorded feature values. After all features are discretized, the encoding presented in [68] is used to derive a symbolic representation from the combination of these discretized feature values. At this point, it should be mentioned that the discretization bins are acquired only from the benign training data, so that the integrity of the learning process is maintained. Subsequently, *flexfringe* is utilized to learn state machines from the symbolic traces derived from the benign training flows using the Alergia heuristic [80]. As in the case of the proposed multivariate methodology, a state machine is learnt from each benign host in the training set, with the traces of the hosts in the test set replayed to all these state machines and the "at least one" rule of detection, presented in Section 6.2.4, employed to identify benign and anomalous hosts. In this context, the match between a state machine representing some benign behavior and a given set of traces belonging to a host is expressed in a way similar to the error-based symptom detection strategy presented in [68], which is based on the difference between the transition distributions produced by the traces of the reference state machine and those of the host under examination.

Finally, LOF-based clustering on aggregated views of the provided NetFlows is employed as the third and final baseline method used in the premises of this thesis. Through the incorporation of this method in the evaluation procedure, the contribution of the sequential nature of the proposed system towards detection is evaluated. This method is quite naive, since it groups the provided NetFlows in host level, and applies aggregation functions on the NetFlow features used for detection to create a basic profile of each host. For numerical features the mean of the grouped values is used, while for categorical attributes the mode is utilized as the aggregation function. Of course, this aggregation entails significant loss regarding the information associated with each host, yet if the whole set of flows associated with each host were used, the application of this clustering technique would become computationally infeasible due to the nearest neighbour nature of LOF. Nevertheless, it was considered beneficial to also evaluate a clustering technique, which, surprisingly, produced excellent results for one of the datasets used, as it will be seen in Section 7.5.

7.4. Parameter Tuning

Both the designed detection methodology, and the baseline methods included in the evaluation process, base their operation on some tunable parameters, that affect their detection potential. As a result, it is considered beneficial to adjust these parameters towards the achievement of a high detection performance, in terms of

both high detection accuracy, and a low false alarm rate. As in most ML tasks, this tuning procedure is conducted on a validation set, which, as mentioned in Section 7.2, was retrieved through a 80/20 timely split of the training set of each dataset used in the experimental procedure, and belongs in the first category of validation sets used for the needs of this work. A tuning procedure was adopted mostly for the sequential methods used in the premises of this thesis, since they include more hyperparameters, thus this section will focus in the presentation of the tuning strategy followed on the designed system, as well as the baseline multivariate and symbolic detection methods. Another interesting point that shall be discussed before presenting the tuning procedure regards the type of parameters needed to be tuned in the aforementioned methodologies. In this thesis, two types of parameters were taken into account in the tuning process. The first type refers to the parameters associated with the detection algorithm fitted in each state of the state machines, while the second type regards the detection threshold used to match a learnt benign model to the set of traces corresponding to a given host from the test set. Finally, it should be mentioned that both types of parameters were tuned solely on benign data, since one of the initial goals when designing the proposed detection system regarded its ability to operate on an unsupervised level.

As it is mentioned above, the first type of hyperparameters regards the detection technique used in each state of the sequential models extracted for each of the compared methods. In particular, since the proposed system uses three different methods, namely LOF, Isolation Forest, and Gaussian KDE, the state-dependent hyperparameters of each method included the number of neighbours used, the number of samples used for fitting the base estimators, and the threshold to which the multivariate Gaussian pdf of each test point is compared, respectively. In the case of the baseline multivariate model, the value with which the standard deviation of the features of the training points is multiplied constitutes the state-dependent hyperparameter, while in the case of the symbolic sequential model there is no such hyperparameter, since solely the existence of a symbol is evaluated in each state. As far as the second type of parameters is concerned, all these sequential methodologies base their final detection decision regarding a given host on the match between the benign profiles maintained by the system and the traces of the examined host. This match is expressed through the comparison of a threshold value to the ratio of flows predicted as benign to the total number of flows examined for the host in the case of the two multivariate approaches, and the difference between the reference transition distribution and the one produced by the host in the case of the symbolic approach. It can be easily seen that these values need to be tuned wisely. Given the fact that the "at least one" rule of detection is used, meaning that the identification of one profile matching the set of testing traces of a host is enough to consider the host as benign, the extent to which the second type of hyperparameters can affect the detection performance can be easily perceived. In the case of the two multivariate approaches, if the detection threshold is set too low, then it is highly probable that most hosts will be considered as benign, since the low threshold would entail the need of high confidence when identifying a host as malicious. On the opposite case, if this threshold is set too high, then most hosts would be considered as malicious, resulting to a system with a high detection rate, but in parallel a high false alarm rate. The exact opposite hold true in the case of the symbolic approach, since a low threshold would entail the need of small difference between the compared distributions for a host to be considered benign, leading to more alarms being raised. On the other hand, a high difference threshold would lead to the inclusion of much less confidence when classifying a host as benign, which as a matter of fact could lead to low detection rate. Thus, it is highly important to tune all these parameters with caution, while bearing in mind that the detection method should avoid overfitting on the training and validation data.

Taking all these things into consideration, a grid search was conducted on all the aforementioned parameters and the produced models were evaluated on the validation set derived from the timely 80/20 split of the training set. Ideally, the values should be set in such a way that all the hosts in the validation set would be identified as benign, yet a combination of values performing well for the benign hosts associated with the majority of the communication traffic could also be accepted by the process. In fact, selecting a combination of values that produces good results for the majority, rather than the entirety of hosts, would potentially lead to models producing better detection performance in the test set, since the risk of overfitting can be better mitigated. Apart from the grid search strategy, the tuning of the detection thresholds of each model can be conducted in a manual way too. Since the value of the decision thresholds affects the number of alarms raised in the testing phase, one can set these values according to the number of alarms a network administrator can inspect within the time granularity in which the system operates. For example, if the aim is to create a system with few false alarms then the detection threshold can be set quite low, so that only hosts for which the system is highly confident about classifying them as malicious would actually raise an alarm. In such a case though, the designed system should be associated with a high detection rate too, so that most of these few alarms

raised represent actual malicious behavior too. In the premises of this thesis, the automated grid search approach was adopted, avoiding though cases where the threshold would be set in extreme values since they were considered prone to overfitting.

7.5. Results

The detection performance of the proposed system was evaluated through a series of experiments on different feature sets and datasets, with multiple baseline methods being employed to evaluate different characteristics of the system, and a state-of-the-art detection technique included in the experimental procedure so as to examine the effectiveness of the designed system against a recently published approach. The results of these experiments were evaluated using the evaluation metrics presented in Chapter 2, meaning the number of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions, as well as the accuracy, precision, and recall of the predictions. As in most anomaly detection tasks, the malicious class was considered as the positive one. In addition, it should be pointed out that for some parts of the experiments two versions of results are presented. The first version includes the results obtained when only hosts with a significant number of flows recorded in each dataset (major hosts of the dataset) were included in the experimental procedure, while the second version includes the results obtained when all hosts are considered. The decision of separately presenting the evaluation results regarding the major hosts of each dataset might seem controversial, since it entails the exclusion of various hosts from the evaluation procedure, yet it is justified by the goal of this thesis and the nature of the designed system.

As it was explained in Chapter 1, the goal of this thesis is to evaluate the feasibility of creating and designing a detection system based on mutlivariate sequential models operating on basic NetFlow features, rather than optimizing the performance of such a system. On top of that, the number of flows available for each aggregation entity (host in this case) affects both the training and the testing phase of the proposed system. In particular, in the training phase it can be easily understood that if there is not a sufficient number of flows representing the communication behavior of a host, the derived sequential profile will most probably be of poor modelling quality. Yet, even if all the benign profiles are extracted from hosts with a sufficient amount of benign traffic, making predictions on hosts with few flows (tens or few hundreds) would lead to highly unreliable and possibly random results. Taking all these things into consideration, it was decided to apply the proposed detection methodology solely on the major hosts available in each dataset in an attempt to secure the robustness and integrity of the obtained results and conclusions on the designed system. As far as the remaining non-major hosts are concerned, it was decided to be treated as benign, since the designed system could not provide a robust prediction regarding the nature of their behaviour. Of course, such a strategy could entail a significant increase in the number of unidentified malicious activities, in case these activities are expressed through a low number of flows. Nevertheless, for the sake of the completeness of the conducted experiments, this strategy was followed in order for the evaluation results obtained from all hosts to be included in this work.

7.5.1. Results on CTU-13 Dataset

The initial experiments conducted on the CTU-13 dataset regard the evaluation of the detection performance of the designed system using different detection algorithms and feature sets. The obtained results including traffic solely from major hosts on each scenario of this dataset can be seen in Table 7.2, where the number of TP, TN, FP, and FN predictions for each of the four feature sets and each of the three detection techniques considered are reported. It is worth noting that the flows associated with the major hosts of the training set of the CTU-13 dataset accounted for 99.9% of the total number of flows in the training set, which is true also in the case of the flows in the test set of this dataset. Furthermore, it should be highlighted that the benign flows of scenario 3 have been used in the training phase of the evaluation process, yet given the fact that this scenario includes also one malicious host it was considered beneficial to include its detection results into this table.

There are various interesting points that can be derived from the results presented in Table 7.2. Before delving deeper into these points, it should be noted that the designed system should ideally have a high number of TPs and TNs (high accuracy), and in parallel a low (or zero) number of FPs (high precision) and FNs (high recall). Bearing this in mind, it can be easily seen that the configurations of the system associated with the third feature set (source port, destination port, protocol, source bytes, destination bytes) seem to fit the aforementioned conditions. The results obtained from these three configurations are relatively similar, with the LOF based approach slightly outperforming the rest two settings. In particular, the systems associ-

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	1	3	2	0	0	5	0	1	0	5	0	1
2	0	1	1	1	1	2	0	0	1	2	0	0
3	1	4	1	0	1	4	1	0	1	4	1	0
4	0	3	1	1	1	4	0	0	1	4	0	0
5	1	3	0	0	1	3	0	0	1	3	0	0
6	0	2	1	1	1	2	1	0	0	2	1	1
7	1	3	0	0	1	3	0	0	1	3	0	0
8	0	3	1	1	1	4	0	0	0	4	0	1
9	1	3	2	9	1	5	0	9	3	5	0	7
10	10	3	2	0	10	5	0	0	10	3	2	0
11	2	1	1	0	2	1	1	0	2	1	1	0
12	3	3	0	0	3	3	0	0	0	2	1	3
13	1	3	1	0	1	4	0	0	1	4	0	0
Total	21	35	13	13	24	45	3	10	21	42	6	13

Results for feature set 1

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	1	3	2	0	0	4	1	1	0	4	1	1
2	1	1	1	0	1	2	0	0	0	2	0	1
3	1	3	2	0	1	4	1	0	1	4	1	0
4	1	4	0	0	1	4	0	0	0	4	0	1
5	1	3	0	0	1	3	0	0	1	3	0	0
6	1	2	1	0	1	2	1	0	0	3	0	1
7	1	3	0	0	1	3	0	0	0	3	0	1
8	1	4	0	0	1	4	0	0	0	4	0	1
9	1	4	1	9	0	4	1	10	0	4	1	10
10	10	4	1	0	10	4	1	0	10	4	1	0
11	2	1	1	0	2	1	1	0	2	1	1	0
12	3	3	0	0	3	3	0	0	3	3	0	0
13	1	4	0	0	1	4	0	0	1	4	0	1
Total	25	39	9	9	23	42	6	11	17	43	5	17

Results for feature set 2

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	1	4	1	0	1	4	1	0	1	4	1	0
2	1	2	0	0	1	2	0	0	1	2	0	0
3	1	5	0	0	1	4	1	0	1	4	1	0
4	1	4	0	0	1	4	0	0	1	4	0	0
5	1	3	0	0	1	3	0	0	1	3	0	0
6	1	3	0	0	1	2	1	0	1	3	0	0
7	1	3	0	0	1	3	0	0	1	3	0	0
8	1	4	0	0	1	4	0	0	1	4	0	0
9	10	4	1	0	10	4	1	0	10	4	1	0
10	10	4	1	0	10	4	1	0	10	4	1	0
11	2	1	1	0	2	1	1	0	2	1	1	0
12	3	3	0	0	3	3	0	0	3	3	0	0
13	1	4	0	0	1	4	0	0	1	4	0	0
Total	34	44	4	0	34	42	6	0	34	43	5	0

Results for feature set 3

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	1	3	2	0	0	4	1	1	0	4	1	1
2	1	1	1	0	1	2	0	0	1	2	0	0
3	1	3	2	0	1	4	1	0	1	4	1	0
4	1	3	1	0	0	4	0	1	1	4	0	0
5	1	3	0	0	1	3	0	0	1	3	0	0
6	1	2	1	0	1	2	1	0	1	2	1	0
7	1	3	0	0	1	3	0	0	1	3	0	0
8	1	3	1	0	1	4	0	0	1	4	0	0
9	1	3	2	9	0	4	1	10	1	5	0	9
10	10	3	2	0	10	4	1	0	10	4	1	0
11	2	1	1	0	0	1	1	2	2	1	1	0
12	3	3	0	0	3	3	0	0	3	3	0	0
13	1	3	1	0	1	4	0	0	1	4	0	0
Total	25	34	14	9	20	42	6	14	24	43	5	10

Results for feature set 4

Table 7.2: Detection results per feature set and detection algorithm used on the CTU-13 dataset with only the major hosts included

ated with all three configurations manage to identify the entirety of the malicious hosts, yet the LOF based approach demonstrates the fewest number of false alarms. It should be pointed out that there are also other settings providing low false alarm rates, like the Isolation Forest based approach operating on the first feature set (protocol, source bytes, destination bytes), and the Gaussian KDE based approach operating on the fourth feature set (destination port, protocol, duration, source bytes, destination bytes), yet these settings are connected to a relatively lower detection rate, since they misclassify approximately 30% of the malicious hosts.

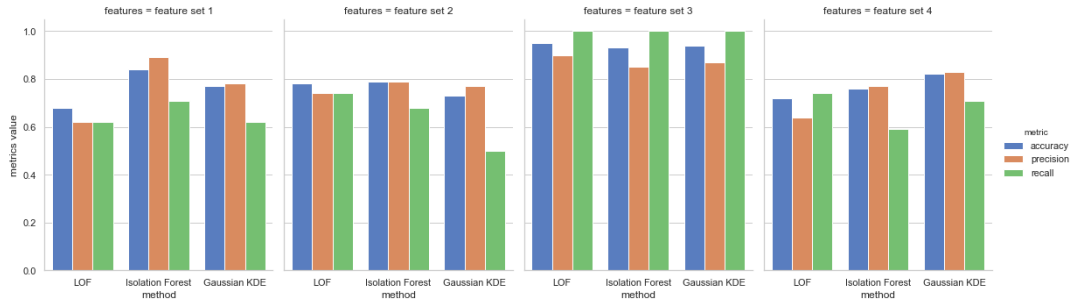


Figure 7.1: Aggregated evaluation metrics per detection method and feature set for the CTU-13 dataset with only the major hosts included

Moving on from the best performing settings, it shall be pointed out that the best performing detection algorithm varies among the different feature sets used for evaluation. When the first feature set is used, the Isolation Forest models clearly outperform the rest models in every metric presented in Table 7.2. For the second feature set (destination port, protocol, source bytes, destination bytes) both the LOF and the Isolation Forest algorithms demonstrate comparable results, with the first identifying slightly more malicious hosts, and the second associated with slightly less false alarms. As it was mentioned above, in the case of the third feature set the LOF algorithm seems to provide the best performance in all the considered metrics, while for the last feature set the Gaussian KDE based models produce the most promising results. Finally, it is worth noting that the designed system is able to detect all the malicious hosts only when the third feature set is taken into account. In the majority of the best performing settings of the rest configurations, the designed system

fails primarily to identify the malicious hosts of scenarios 1 and 9, resulting to the number of FNs being close to 10 in these cases. Delving deeper into the cause of this behaviour, and revisiting Table 4.1 presented in Chapter 4, it can be seen that both the aforementioned scenarios are associated with the NERIS bot, meaning that the anomalous communication traffic of this bot unfortunately cannot be detected when the source port is not included in the feature set on which the system operates.

In Figure 7.1 the achieved accuracy, precision, and recall metrics for each detection algorithm and feature set used across the entire dataset are presented. It can be easily seen that the LOF based models operating on the third feature set attain the highest accuracy (95%) and precision (90%) among all configurations, with the rest two model categories operating on the same feature set achieving a similar recall value (100%), but slightly lower accuracy and precision values. This observation is justified by the fact that the LOF based models produce a lower number of false alarms comparing to the rest two model categories operating on the third feature set (in this case two less than the Isolation Forest based models, and one less than the Gaussian KDE based ones). In addition, it is worth noting that the models operating on the third feature set seem to dominate all the settings configured on the rest feature sets, with only the Isolation Forest based models operating on the first feature set demonstrating relatively similar precision, accompanied though by significantly lower accuracy and recall values. This observation indicates the impact of the feature selection procedure on the detection potential of the designed system, since as it was mentioned above the inclusion of the source port in the third feature is highly beneficial towards the detection of the malicious behavior produced by the NERIS bot.

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	1	4	2	0	0	6	0	1	0	6	0	1
2	0	4	1	1	1	5	0	0	1	5	0	0
3	1	5	1	0	1	5	1	0	1	5	1	0
4	0	5	1	1	1	6	0	0	1	6	0	0
5	1	6	0	0	1	6	0	0	1	6	0	0
6	0	5	1	1	1	5	1	0	0	5	1	1
7	1	5	0	0	1	5	0	0	1	5	0	0
8	0	6	1	1	1	7	0	0	0	7	0	1
9	1	5	2	9	1	7	0	9	3	7	0	7
10	10	5	2	0	10	7	0	0	10	5	2	0
11	2	5	1	1	2	5	1	1	2	5	1	1
12	3	6	0	0	3	6	0	0	0	5	1	3
13	1	5	1	0	1	6	0	0	1	6	0	0
Total	21	66	13	14	24	76	3	11	21	73	6	14

Results for feature set 1

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	1	4	2	0	0	5	1	1	0	5	1	1
2	1	4	1	0	1	5	0	0	0	5	0	1
3	1	4	2	0	1	5	1	0	1	5	1	0
4	1	6	0	0	1	6	0	0	0	6	0	1
5	1	6	0	0	1	6	0	0	1	6	0	0
6	1	5	1	0	1	5	1	0	0	6	0	1
7	1	5	0	0	1	5	0	0	0	5	0	1
8	1	7	0	0	1	7	0	0	0	7	0	1
9	1	6	1	9	0	6	1	10	0	6	1	10
10	10	6	1	0	10	6	1	0	10	6	1	0
11	2	5	1	1	2	5	1	1	2	5	1	1
12	3	6	0	0	3	6	0	0	3	6	0	0
13	1	6	0	0	1	6	0	0	0	6	0	1
Total	25	70	9	10	23	73	6	12	17	74	5	18

Results for feature set 2

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	1	5	1	0	1	5	1	0	1	5	1	0
2	1	5	0	0	1	5	0	0	1	5	0	0
3	1	6	0	0	1	5	1	0	1	5	1	0
4	1	6	0	0	1	6	0	0	1	6	0	0
5	1	6	0	0	1	6	0	0	1	6	0	0
6	1	6	0	0	1	5	1	0	1	6	0	0
7	1	5	0	0	1	5	0	0	1	5	0	0
8	1	7	0	0	1	7	0	0	1	7	0	0
9	10	6	1	0	10	6	1	0	10	6	1	0
10	10	6	1	0	10	6	1	0	10	6	1	0
11	2	5	1	1	2	5	1	1	2	5	1	1
12	3	6	0	0	3	6	0	0	3	6	0	0
13	1	6	0	0	1	6	0	0	1	6	0	0
Total	34	75	4	1	34	73	6	1	34	74	5	1

Results for feature set 3

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	1	4	2	0	0	5	1	1	0	5	1	1
2	1	4	1	0	1	5	0	0	1	5	0	0
3	1	4	2	0	1	5	1	0	1	5	1	0
4	1	5	1	0	0	6	0	1	1	6	0	0
5	1	6	0	0	1	6	0	0	1	6	0	0
6	1	5	1	0	1	5	1	0	1	5	1	0
7	1	5	0	0	1	5	0	0	1	5	0	0
8	1	6	1	0	1	7	0	0	1	7	0	0
9	1	5	2	9	0	6	1	10	1	7	0	9
10	10	5	2	0	10	6	1	0	10	6	1	0
11	2	5	1	1	0	5	1	3	2	5	1	1
12	3	6	0	0	3	6	0	0	3	6	0	0
13	1	5	1	0	1	6	0	0	1	6	0	0
Total	25	65	14	10	20	73	6	15	24	74	5	11

Results for feature set 4

Table 7.3: Detection results per feature set and detection algorithm used on the CTU-13 dataset with all hosts included

After analysing the results obtained when only major hosts were considered, the total host-related traffic was evaluated too, with the attained results presented in Table 7.3. As it can be easily seen, if the major hosts' results presented in Table 7.2 are compared to the total results presented in Table 7.3, the sole differences recorded regard the number of TN and FN predictions. In particular, the number of TN has notably increased in all settings, while the number of FN has been incremented by one in all settings. This phenomenon can be easily explained if the tactic followed for the predictions made on minor hosts is taken into consideration. As it was explained in the introductory paragraphs of this section, all minor hosts, for the malice of which the proposed system is not able to provide robust predictions, were regarded as benign. If this strategy is further inspected, it can be easily understood that the only metrics that can be affected, comparing to the ones recorded for the major hosts, are the number of correctly identified benign hosts (TN), as well as the number of unidentified malicious hosts (FN). The first metric is affected by the portion of minor hosts that are actually benign, while the second is regulated by the portion of minor hosts that demonstrate malicious

activity. As a result, it is apparent that, in the case of the CTU-13 dataset, the majority of the minor hosts are indeed benign, while there is only one malicious minor host encountered in scenario 11.

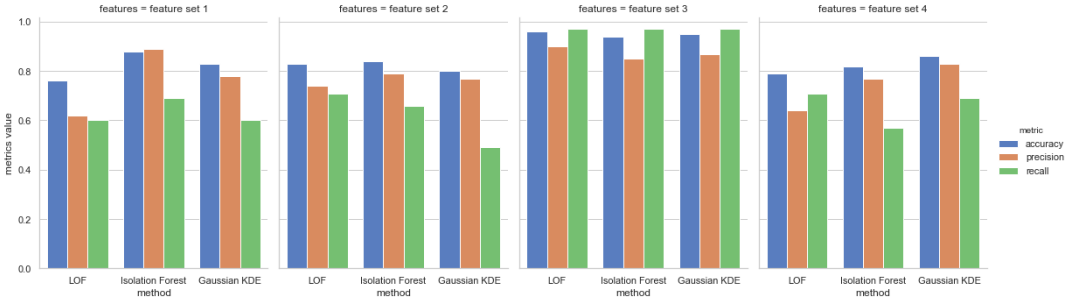


Figure 7.2: Aggregated evaluation metrics per detection method and feature set for the CTU-13 dataset with all hosts included

As in the case of the major hosts' results, the achieved accuracy, precision, and recall metrics for each detection algorithm and feature set used across the entire dataset are presented in Figure 7.2. As expected, the primary differences between the metrics presented in this figure and those presented in Figure 7.1 regard mostly the accuracy, and in a much smaller extent the recall. In particular, the accuracy achieved in all settings is prominently increased due to the proportional increase of the number of TN predicted by the models, while the attained recall is slightly decreased due to the misclassification of one more malicious host when the total network traffic is considered. Of course, the precision values remain the same across all settings, since the metrics contributing to its calculation (TP, FP) are not affected by the prediction strategy followed for minor hosts. Taking these things into consideration, it can be understood that the conclusions to be drawn from the evaluation of the different settings of the proposed system are not affected from the incorporation of the minor hosts' results into the initial result set.

Scenario	Best Multivariate				Baseline Multivariate				Symbolic				LOF			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	1	5	1	0	1	5	1	0	1	5	1	0	1	4	2	0
2	1	5	0	0	1	5	0	0	1	4	1	0	1	5	0	0
3	1	6	0	0	1	5	1	0	1	5	1	0	1	5	1	0
4	1	6	0	0	1	6	0	0	1	6	0	0	1	4	2	0
5	1	6	0	0	1	6	0	0	1	6	0	0	1	5	1	0
6	1	6	0	0	1	6	0	0	1	6	0	0	1	4	2	0
7	1	5	0	0	1	5	0	0	1	2	3	0	1	4	1	0
8	1	7	0	0	1	7	0	0	1	7	0	0	1	6	1	0
9	10	6	1	0	10	6	1	0	9	7	0	1	10	3	4	0
10	10	6	1	0	10	6	1	0	10	6	1	0	10	3	4	0
11	2	5	1	1	2	5	1	1	1	4	2	2	2	4	2	1
12	3	6	0	0	3	6	0	0	3	6	0	0	3	6	0	0
13	1	6	0	0	1	6	0	0	0	5	1	1	1	6	0	0
Total	34	75	4	1	34	74	5	1	31	69	10	4	34	59	20	1

Table 7.4: Comparative results on the CTU dataset between the proposed system and the baselines

Before discussing about the baseline comparison results, it should be pointed out that the proposed system seems to provide a solid detection performance on the CTU-13 dataset, since at its best setting it attains an accuracy of 96% (95% on major hosts), a precision of 90%, and a recall of 97% (100% on major hosts). Subsequently, in order to proceed to the comparison of the designed detection system to the baseline methods presented in Section 7.3, the best performing configuration of the proposed system shall be selected. For this purpose, as mentioned in Chapter 4, a validation set comprising of the mixed traffic captured in scenarios 3, 4, 5, 7, 10, 11, 12, and 13 was utilized. The selection of these scenarios was based on the training/cross-validation setting suggested by the creators of the CTU-13 dataset in [28]. At this point, it should be clarified that the utilization of a validation set including mixed traffic in this part of the experiments does not compromise the unsupervised nature of the system. This decision is made only so that the best performing methodology can be identified and tested against the implemented baseline methods. Through this process,

there were multiple configurations producing closely comparable results, yet the Isolation Forest based models operating on the first feature set, as well as the LOF based ones operating on the third feature set seem to slightly dominate the rest configurations, with both of these approaches achieving the same exact metrics on the validation set. In order to decide on which of these two settings should be promoted as the best performing one in the premises of the baseline experiments, the confidence of the predictions made was taken into account. In particular, since the desinged system bases the final classification decision on the ratio of flows predicted as benign to the total number of flows associated with a host, the setting that produced the higher such ratio for benign hosts and the lower ratio for malicious hosts was chosen. Through this process, the LOF based models were eventually selected and considered in the baseline experiments.

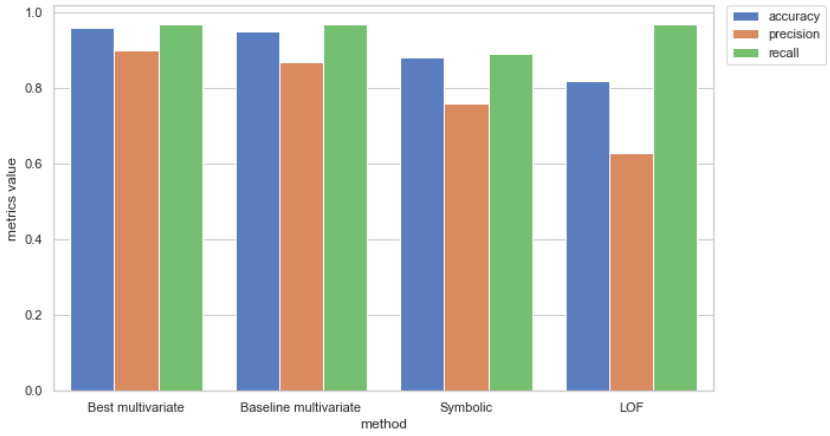


Figure 7.3: Comparison between the aggregated evaluation metrics of the proposed system and the baseline methods for the CTU-13 dataset

After identifying the best performing configuration of the designed detection system, a comparison between its performance and that attained by the three baseline methods presented in Section 7.3 is conducted. The results achieved for each method on each scenario of the CTU-13 dataset (both those comprising the validation set and those comprising the test set) are presented in Table 7.4, while Figure 7.3 summarizes the accuracy, precision, and recall metrics achieved by each method across the entire dataset. It should be pointed out that the baseline methods were evaluated on the same feature set as that used by the best configuration of the proposed multivariate approach. On top of that, the minor hosts were treated, for all baseline methods, in the same way as that followed within the experiments conducted on the designed methodology, meaning that all minor hosts were by default predicted as benign. This decision was made to maintain the integrity of the evaluation procedure, as well as to apply a sense of fairness among the experiments conducted for each baseline method. Taken into account the results presented both in Table 7.4 and in Figure 7.3, it can be easily seen that the proposed multivariate approach produces similar results to the ones achieved by the baseline multivariate method. In particular, both methods identify the majority of the malicious hosts, with only one malicious host being undetected, yet the proposed multivariate approach raises one less false alarm comparing to the baseline multivariate approach, which as a matter of fact explains the slightly higher accuracy and precision values attained by the first method. As far as the rest two baseline methods are concerned, it can be clearly seen that the LOF clustering technique produces the worst results, which as a matter of fact was expected due to the simplistic nature of this method, resulting to a system producing a high number of false alarms, while the symbolic sequential approach produces obviously better results than the LOF clustering technique, yet not comparable to the ones attained by the multivariate methods.

7.5.2. Results on UNSW-15 Dataset

As in the case of the CTU-13 dataset, the initial experiments conducted on the UNSW-NB15 regarded the evaluation of the detection performance of the proposed methodology on different configurations of the feature set and the detection algorithms fitted in the states of the multivariate models. The obtained results including traffic only from major hosts for each of the four partitions of the dataset can be found in Table 7.5. Again only the major hosts of the dataset were initially taken into account, so that the detection performance of the proposed methodology could be better evaluated. As in the case of the CTU-13 dataset, the flows linked with the major hosts represent the high majority of the total flows captured in the dataset, with the

flows associated with the major hosts in the training set accounting for 97.1% of the total flows of that set, and the flows associated with the major hosts in the test set accounting for 95.9% of the total flows included in the test set. Table 7.5 incorporates also the results obtained in the first partition of the dataset, despite the fact that the benign flows of that partition composed the training set of this dataset. The reason why these results are reported too stems from the existence of malicious hosts among the hosts of this partition.

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	1	10	0	3	1	10	0	3	1	10	0	3
2	1	10	0	3	2	10	0	2	2	10	0	2
3	1	10	0	3	1	10	0	3	1	10	0	3
4	1	10	0	3	2	10	0	2	2	10	0	2
Total	4	40	0	12	6	40	0	10	6	40	0	10

Results for feature set 1

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	1	10	0	3	0	10	0	4	0	10	0	4
2	2	10	0	2	0	10	0	4	2	10	0	2
3	1	10	0	3	0	10	0	4	1	10	0	3
4	2	10	0	2	1	10	0	3	2	10	0	2
Total	6	40	0	10	1	40	0	15	5	40	0	11

Results for feature set 2

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	3	10	0	1	1	10	0	3	1	10	0	3
2	3	10	0	1	2	10	0	2	2	10	0	2
3	4	10	0	0	1	10	0	3	1	10	0	3
4	3	10	0	1	0	10	0	4	2	10	0	2
Total	13	40	0	3	4	40	0	12	6	40	0	10

Results for feature set 3

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	1	10	0	3	0	10	0	4	0	10	0	4
2	1	10	0	3	0	10	0	4	4	10	0	0
3	1	10	0	3	0	10	0	4	1	10	0	3
4	0	10	0	4	1	10	0	3	2	10	0	2
Total	3	40	0	13	1	40	0	15	7	40	0	9

Results for feature set 4

Table 7.5: Detection results per feature set and detection algorithm used on the UNSW-NB15 dataset with only the major hosts included

As far as the best performing configuration is concerned, it can be easily seen that the LOF based approach operating on the third feature set provides by far the best metrics comparing to the rest configurations. Yet, when examining the results obtained for each feature set, different detection algorithms tend to be the best performing ones. In fact, it can be seen that the LOF-based approach dominates when the second and third feature sets are used, with the Gaussian KDE based model performing the best on the fourth feature set. As far as the first feature set is concerned, both the Isolation Forest and the Gaussian KDE based approaches seem to be suitable. Finally, it is worth mentioning that all configurations manage to identify all the benign hosts correctly, with the best performing configuration managing to identify approximately 80% of the malicious hosts. These results are better visualised in Figure 7.4, in which the aggregated results over the entire dataset are presented for each configuration.

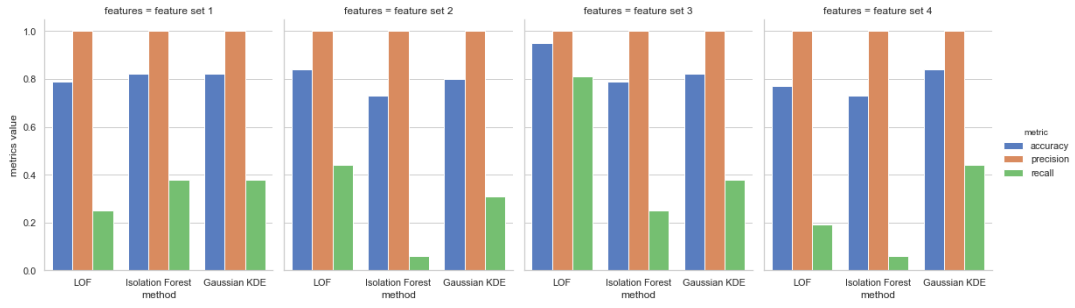


Figure 7.4: Aggregated evaluation metrics per detection method and feature set for the UNSW-NB15 dataset

As it can be seen in Figure 7.4, the LOF-based approach operating on the third feature set provides the best aggregated results among all the evaluation metrics. Of course, as it was mentioned above, the precision value of all configurations is equal to 100%, since all the benign hosts are identified successfully, yet the recall values recorded are significantly low for most of the tested settings. In fact, solely the best performing configuration achieves a relatively high recall value (81%), while the rest configurations manage to detect less than half of the malicious hosts. After better inspecting the dataset to identify the reason of that result, it was discovered that the hosts, that were not identified correctly in each of the partitions, were also associated with a significant number of benign flows, which as a matter of fact worsens the detection procedure of the proposed methodology. Yet, the combination of a LOF based approach with the third feature set produces a system capable of partially mitigating the aforementioned phenomenon. To further deal with such an issue, a more fine-grained detection approach could be employed, meaning that it might be beneficial to conduct a connection level analysis in this dataset. This possibility was also discussed in Chapter 4, where the dataset

was presented, yet it was not explored since a host level analysis was better suited for the rest two datasets. Nevertheless, such a possibility could be considered highly interested as future work on this specific dataset.

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	1	36	0	3	1	36	0	3	1	36	0	3
2	1	36	0	3	2	36	0	2	2	36	0	2
3	1	36	0	3	1	36	0	3	1	36	0	3
4	1	34	0	3	2	34	0	2	2	34	0	2
Total	4	142	0	12	6	142	0	10	6	142	0	10

Results for feature set 1

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	1	36	0	3	0	36	0	4	0	36	0	4
2	2	36	0	2	0	36	0	4	2	36	0	2
3	1	36	0	3	0	36	0	4	1	36	0	3
4	2	34	0	2	1	34	0	3	2	34	0	2
Total	6	142	0	10	1	142	0	15	5	142	0	11

Results for feature set 2

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	3	36	0	1	1	36	0	3	1	36	0	3
2	3	36	0	1	2	36	0	2	2	36	0	2
3	4	36	0	0	1	36	0	3	1	36	0	3
4	3	34	0	1	0	34	0	4	2	34	0	2
Total	13	142	0	3	4	142	0	12	6	142	0	10

Results for feature set 3

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	1	36	0	3	0	36	0	4	0	36	0	4
2	1	36	0	3	0	36	0	4	4	36	0	0
3	1	36	0	3	0	36	0	4	1	36	0	3
4	0	34	0	4	1	34	0	3	2	34	0	2
Total	3	142	0	13	1	142	0	15	7	142	0	9

Results for feature set 4

Table 7.6: Detection results per feature set and detection algorithm used on the UNSW-NB15 dataset with all hosts included

As in the case of the CTU-13 dataset, after presenting and discussing the results obtained when only major hosts are considered, the total host-related traffic is evaluated too, with the achieved results presented in Table 7.6. If the results presented in this table are compared with those presented in Table 7.5, which regarded solely the major hosts, it can be easily seen that the only difference between these tables concern the increased number of TN predictions. The explanation behind this observation is identical to the one provided in the case of the CTU-13 dataset, with the difference being that all minor hosts in the UNSW-NB15 dataset are purely benign. As a result, only the number of correctly identified benign hosts (TN) is affected by the incorporation of the entire recorded traffic in the illustrated results. Of course, such an observation suggests that the main conclusions drawn from the major hosts' results are still valid when the entirety of the recorded traffic is taken into account.

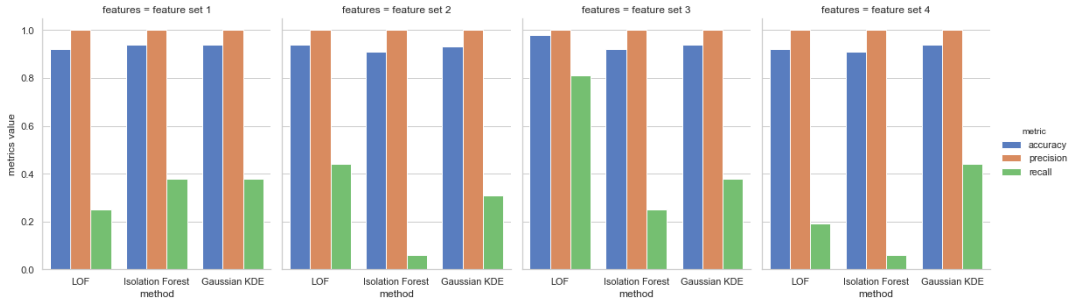


Figure 7.5: Aggregated evaluation metrics per detection method and feature set for the UNSW-NB15 dataset with all hosts included

In Figure 7.5, the achieved accuracy, precision, and recall metrics for each detection algorithm and feature set used across the entire dataset are recorded and presented. As expected, the sole difference between the metrics presented in this figure and those presented in Figure 7.4 regards the accuracy achieved. In more detail, the accuracy attained in all of the examined configurations is visibly increased, which as a matter of fact is explained by the proportional increase in the number of TN predicted by the models. Of course, the values of the precision and recall achieved by each configuration remain the same across all settings, since the metrics contributing to their calculation (TP, FP, FN) are not affected by the prediction strategy followed for minor hosts. Accordingly, in order to compare the designed system with the implemented baseline approaches, its best performing configuration shall be identified. As in the case of the CTU-13 dataset, a validation set, including mixed traffic from the first two chunks of the dataset, is used towards this purpose. The decision to include these two chunks of the dataset was based mostly on the time precedence of their flows, since all four chunks include the same kinds of attacks. This process pointed the LOF based models operating on the third feature set out as the best performing configuration, since the system learnt on this setting managed to detect a highest number of malicious hosts on the validation set.

After identifying the best performing configuration of the proposed multivariate approach, the baseline comparison experiments are conducted. The results obtained from these experiments are presented in Table

Scenario	Best Multivariate				Baseline Multivariate				Symbolic				LOF			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	3	36	0	1	1	36	0	3	4	36	0	0	4	36	0	0
2	3	36	0	1	2	36	0	2	4	36	0	0	4	36	0	0
3	4	36	0	0	2	36	0	2	4	36	0	0	4	36	0	0
4	3	34	0	1	3	34	0	1	4	34	0	0	4	34	0	0
Total	13	142	0	3	8	142	0	8	16	142	0	0	16	142	0	0

Table 7.7: Comparative results on the UNSW dataset between the proposed system and the baselines

7.7 and visualised in Figure 7.6. It can be easily seen that the obtained results show a significantly different pattern comparing to that observed in the CTU-13 dataset. In that dataset the proposed multivariate detection methodology significantly overperformed the last two baseline methods, and demonstrated equally high results with the baseline multivariate method. In this case, a completely opposite effect is recorded. In particular, the baseline multivariate method produces degraded results, since it manages to correctly detect only half of the malicious hosts existing in the dataset, while both the symbolic sequential method and the LOF clustering method identified all the malicious hosts. This last observation is quite surprising, especially when the simplistic nature of the LOF clustering approach is taken into account.

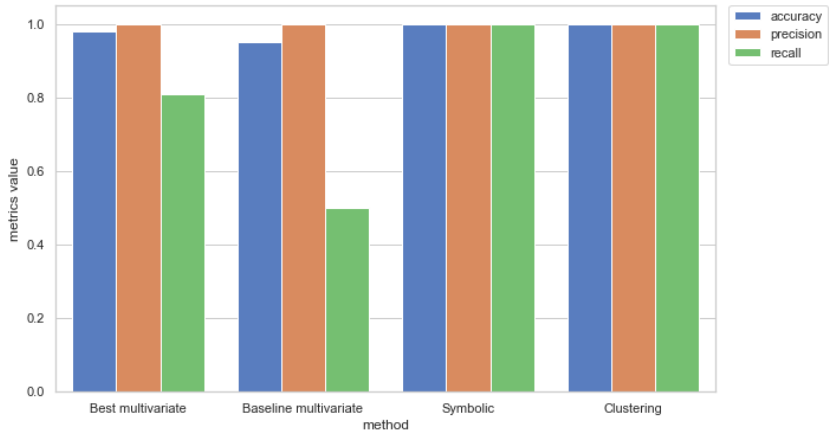
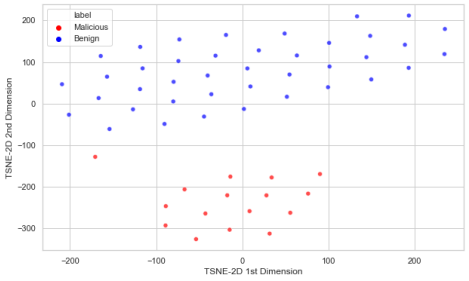


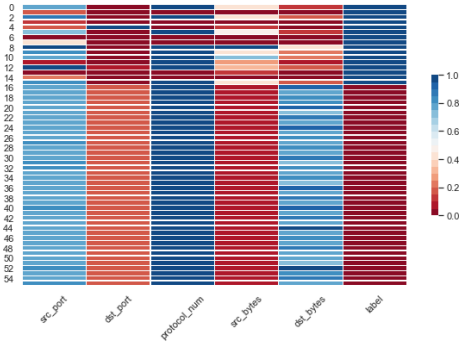
Figure 7.6: Comparison between the aggregated evaluation metrics of the proposed system and the baseline methods for the UNSW-NB15 dataset

To further investigate the origin of this observation, the aggregated profiles of the hosts used within the LOF clustering approach were visualised and the correlation between the utilized features and the benignity of a host was examined, as it can be seen in Figure 7.7. In more detail, Figure 7.7a illustrates the τ-SNE [57] projection of the aggregated profiles onto a 2D space, while Figure 7.7b depicts the heatmap of the relation between the features of each aggregated host's profile. As it can be easily seen from Figure 7.7a, the aggregated representations of the hosts can be clearly separated according to the benignity of their nature. In particular, the high majority of the malicious hosts form a relatively tight cluster depicted in the bottom part of Figure 7.7a, that lies relatively far from the cluster of benign hosts visualised on the top part of this figure. Of course this 2D projection is not a totally accurate visualization of the relative placement of the aggregated profiles of each host, yet it provides an insight on the reason behind the excellent results achieved by the LOF clustering baseline approach. In order to better understand which features of these aggregated views contribute mostly on the clear separability between the aggregated profiles of the benign and malicious hosts, the heatmap presented in Figure 7.7b is leveraged. This heatmap depicts the relations between the features used and the label of all 56 major hosts included in the dataset, with the 16 malicious ones visualised on the top part of the heatmap, and the 40 benign ones following in the bottom of the heatmap. At this point, it should be mentioned that all features have been scaled in the range $[0,1]$ for visualization purposes. Having said that, it can be easily seen that the destination bytes seem to be the feature that contributes mostly in separating the aggregated profiles according to their label of benignity, since there is an evident contrast between the normalized values associated with each set of hosts. All things considered, it can be concluded that detection techniques of simple nature are sufficient for detecting malicious behavior in this dataset, yet performing

the detection process through the proposed system in a connection level could be an interesting direction of future work on this dataset.



(a) TNSE projection on 2 dimensions of the aggregated view of the data from the UNSW-NB15 dataset



(b) Heatmap between the aggregated data points and the considered features on the UNSW-NB15 dataset

Figure 7.7: Visualization of the aggregated points in the LOF clustering context for the the UNSW-NB15 dataset

7.5.3. Results on CICIDS2017 Dataset

The CICIDS2017 dataset constitutes the last dataset used in the evaluation process of the proposed detection methodology, with the experiments conducted on this dataset following the same pattern as the one presented for the rest two datasets. Initially, only the traffic linked to the major hosts of the dataset was included in the experiments, with the flows used in the training phase accounting for 80% of the total flows included in the training set, and the flows used in the testing phase representing 91.6% of the total testing flows. These major hosts' results obtained for each day of this dataset on the different evaluations examined can be seen in Table 7.8. In the contrary to the previously presented datasets, the training set in this case (Monday flows) did not include any malicious hosts. Yet, the results recorded on that day are incorporated in this table for reasons of completeness.

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
Monday	0	13	0	0	0	13	0	0	0	13	0	0
Tuesday	1	12	1	0	1	12	1	0	1	12	1	0
Wednesday	1	13	0	0	1	13	0	0	1	13	0	0
Thursday (morn.)	1	11	2	0	1	11	2	0	1	13	0	0
Thursday (aft.)	1	11	2	0	1	11	2	0	0	13	0	1
Friday (morn.)	1	10	1	2	1	10	1	2	1	11	0	2
Friday (aft. - DDoS)	1	11	2	0	1	12	1	0	1	13	0	0
Friday (aft. - port scan)	1	12	1	0	1	11	2	0	1	13	0	0
Total	7	93	9	2	7	93	9	2	6	101	1	3

Results for feature set 1

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
Monday	0	13	0	0	0	13	0	0	0	13	0	0
Tuesday	1	12	1	0	1	12	1	0	1	12	1	0
Wednesday	1	13	0	0	1	13	0	0	1	13	0	0
Thursday (morn.)	1	11	2	0	1	11	2	0	1	13	0	0
Thursday (aft.)	1	11	2	0	1	11	2	0	1	11	2	0
Friday (morn.)	1	9	2	2	1	10	1	2	1	11	0	2
Friday (aft. - DDoS)	1	12	1	0	1	11	2	0	1	11	2	0
Friday (aft. - port scan)	1	12	1	0	1	12	1	0	1	12	1	0
Total	7	93	9	2	7	93	9	2	7	96	6	2

Results for feature set 2

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
Monday	0	13	0	0	0	13	0	0	0	13	0	0
Tuesday	0	12	1	1	1	11	2	0	1	12	1	0
Wednesday	0	12	1	1	1	12	1	0	1	13	0	0
Thursday (morn.)	0	13	0	1	1	12	1	0	1	12	1	0
Thursday (aft.)	0	12	1	1	0	9	4	1	0	10	3	1
Friday (morn.)	1	11	0	2	1	10	1	2	1	11	0	2
Friday (aft. - DDoS)	1	13	0	0	1	12	1	0	1	12	1	0
Friday (aft. - port scan)	1	12	1	0	1	12	1	0	1	12	1	0
Total	3	98	4	6	6	91	11	3	6	95	7	3

Results for feature set 3

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
Monday	0	13	0	0	0	13	0	0	0	13	0	0
Tuesday	0	12	1	1	0	12	1	1	1	12	1	0
Wednesday	0	13	0	1	1	12	1	0	1	11	2	0
Thursday (morn.)	0	12	1	1	1	11	2	0	1	11	2	0
Thursday (aft.)	1	12	1	0	1	11	2	0	1	11	2	0
Friday (morn.)	1	10	1	2	1	9	2	2	1	9	2	2
Friday (aft. - DDoS)	1	12	1	0	1	11	2	0	1	11	2	0
Friday (aft. - port scan)	0	13	0	1	1	11	2	0	1	11	2	0
Total	3	97	5	6	6	90	12	3	7	89	13	2

Results for feature set 4

Table 7.8: Detection results per feature set and detection algorithm used on the CICIDS2017 dataset with only the major hosts included

It can be easily seen that the best results among all metrics are provided by the Gaussian KDE based approach operating on the first feature set, recording only one false alarm among the predictions made. At this point, it should be highlighted that there are multiple other settings that demonstrate a slightly higher number of detected hosts (one more host in particular), yet the Gaussian KDE based approach operating on the first feature set is considered as the best performing one, since it significantly outperforms all of these configurations in terms of the number of false alarms raised. Another interesting observation regards the fact that there is not such a great variation on the best performing detection algorithms among different feature sets, as that observed in the other two datasets. In particular, the Gaussian KDE based models tend to

outperform the other two kinds of models in all four feature sets. At this point it should be mentioned that the malicious hosts in this dataset are associated with both malicious and benign flows, with the malicious flows being the high majority in most occasions. This is not the case only for three hosts, the sole malicious host on Thursday afternoon, and two of the malicious hosts on Friday morning. This observation explains the three FNs recorded on these exact hosts by the best performing configuration. On top of that, this fact suggests that the only attacks that are not captured by the proposed system include a minor infiltration, consisting of 36 flows, and two short bot communications, consisting of 2 flows each.

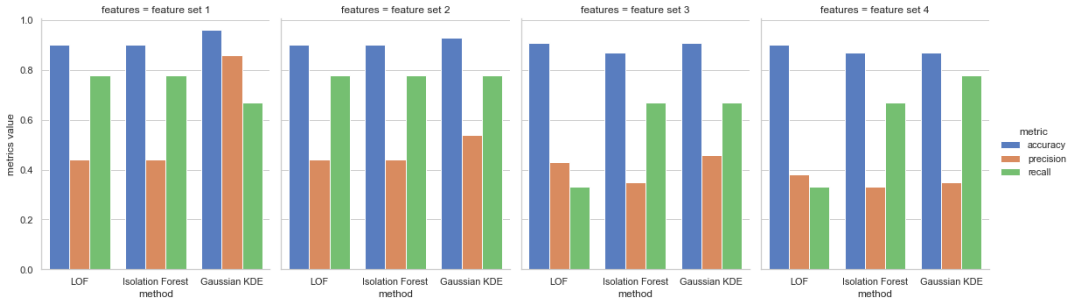


Figure 7.8: Aggregated evaluation metrics per detection method and feature set for the CICIDS2017 dataset with only the major hosts included

Figure 7.8 presents the accuracy, precision, and recall values achieved in all of the evaluated configurations across the entire dataset. It can be easily seen that the Gaussian KDE based approach operating on the first feature set outperforms all other configurations in two of the three considered metrics, since it achieves an accuracy of 96% and a precision of 86%. Yet, there are multiple configurations demonstrating better recall with similar accuracy values. As it was explained above, the main reason why one of these methods is not regarded as the best performing one instead regards their low precision values. The Gaussian KDE approach connected with the first feature set records a precision that is almost double than any other value observed, while maintaining a relatively high recall, even if it is not the highest one. Of course, as in the cases of the other two datasets considered in this work, the total host-related traffic is evaluated too, with the associated results presented in Table 7.9.

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
Monday	0	8239	0	0	0	8239	0	0	0	8239	0	0
Tuesday	1	7190	1	0	1	7190	1	0	1	7190	1	0
Wednesday	1	7688	0	0	1	7688	0	0	1	7688	0	0
Thursday (morn.)	1	4200	2	0	1	4200	2	0	1	4202	0	0
Thursday (aft.)	1	5098	2	0	1	5098	2	0	0	5100	0	1
Friday (morn.)	1	4460	1	2	1	4460	1	2	1	4461	0	2
Friday (aft. - DDoS)	1	2064	2	0	1	2065	1	0	1	2066	0	0
Friday (aft. - port scan)	1	3665	1	0	1	3664	2	0	1	3666	0	0
Total	7	42604	9	2	7	42604	9	2	6	42612	1	3

Results for feature set 1

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
Monday	0	8239	0	0	0	8239	0	0	0	8239	0	0
Tuesday	1	7190	1	0	1	7190	1	0	1	7190	1	0
Wednesday	1	7688	0	0	1	7688	0	0	1	7688	0	0
Thursday (morn.)	1	4200	2	0	1	4200	2	0	1	4202	0	0
Thursday (aft.)	1	5098	2	0	1	5098	2	0	1	5098	2	0
Friday (morn.)	1	4459	2	2	1	4460	1	2	1	4461	0	2
Friday (aft. - DDoS)	1	2065	1	0	1	2064	2	0	1	2064	2	0
Friday (aft. - port scan)	1	3665	1	0	1	3665	1	0	1	3665	1	0
Total	7	42604	9	2	7	42604	9	2	7	42607	6	2

Results for feature set 2

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
Monday	0	8239	0	0	0	8239	0	0	0	8239	0	0
Tuesday	0	7190	1	1	1	7189	2	0	1	7190	1	0
Wednesday	0	7687	1	1	1	7687	1	0	1	7688	0	0
Thursday (morn.)	0	4202	0	1	1	4201	1	0	1	4201	1	0
Thursday (aft.)	0	5099	1	1	0	5096	4	1	0	5097	3	1
Friday (morn.)	1	4461	0	2	1	4460	1	2	1	4461	0	2
Friday (aft. - DDoS)	1	2066	0	0	1	2065	1	0	1	2065	1	0
Friday (aft. - port scan)	1	3665	1	0	1	3665	1	0	1	3665	1	0
Total	3	42609	4	6	6	42602	11	3	6	42606	7	3

Results for feature set 3

Scenario	LOF				Isolation Forest				Gaussian KDE			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
Monday	0	8239	0	0	0	8239	0	0	0	8239	0	0
Tuesday	0	7190	1	1	0	7190	1	1	0	7190	1	0
Wednesday	0	7688	0	1	1	7687	1	0	1	7686	2	0
Thursday (morn.)	0	4201	1	1	1	4200	2	0	1	4200	2	0
Thursday (aft.)	1	5099	1	0	1	5098	2	0	1	5098	2	0
Friday (morn.)	1	4460	1	2	1	4459	2	2	1	4459	2	2
Friday (aft. - DDoS)	1	2065	1	0	1	2064	2	0	1	2064	2	0
Friday (aft. - port scan)	0	3666	0	1	1	3664	2	0	1	3664	2	0
Total	3	42608	5	6	6	42601	12	3	7	42600	13	2

Results for feature set 4

Table 7.9: Detection results per feature set and detection algorithm used on the CICIDS2017 dataset with all hosts included

If the results presented in this table are compared with those presented in Table 7.8, which regard solely the major hosts, it can be easily observed that the number of TN predictions is significantly increased. This phenomenon can be explained by the "physiology" of this dataset. In more detail, the CICIDS2017 consists of a considerably large set of benign hosts involved in short-lived network communication, along with a handful of major hosts associated with the high majority of the recorded traffic. Thus, the incorporation of the aforementioned large set of minor benign hosts in the evaluation procedure results to the significant increase of the number of correctly identified benign hosts (TN) recorded. Of course, given the fact that all the added

minor hosts are benign, the primary conclusions drawn from the major hosts' results remain valid also when all the captured flows are taken into account.

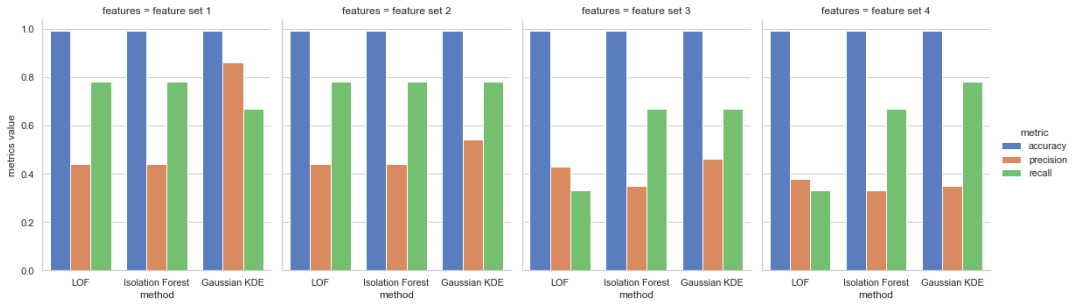


Figure 7.9: Aggregated evaluation metrics per detection method and feature set for the CICIDS2017 dataset with all hosts included

In Figure 7.9 the achieved accuracy, precision, and recall metrics for each detection algorithm and feature set used across the entire dataset are visualised. As expected, accuracy constitutes the only metric the value of which has changed comparing to that presented in Figure 7.8. In particular, the accuracy attained in all of the examined configurations is close to 100%, which as a matter of fact is explained by the significantly larger number of TN predictions included in the result set. Of course, as in the case of the UNSW-NB15 dataset, the same precision and recall values are recorded for each configuration across all settings, since the metrics affecting their calculation (TP, FP, FN) are not altered by the prediction strategy followed for minor hosts. Subsequently, as in the rest datasets, the designed detection system is compared to a series of baseline methods. To do so, the configuration performing the best on a validation set should be identified. The validation set used in the premises of this dataset consists of three captures with mixed network traffic, namely the Tuesday, Wednesday, and Thursday morning captures. These captures were selected since they included a considerable amount of malicious flows, with the time order of the flows among all captures taken into account again in this validation process. The Gaussian KDE based models operating on the first feature set were promoted through this process as the best performing configuration of the designed system, since they produced the lowest number of false alarms, while identifying the same number of malicious hosts as the other configurations did.

Scenario	Best Multivariate				Baseline Multivariate				Symbolic				LOF			
	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
Monday	0	13	0	0	0	13	0	0	0	13	0	0	0	13	0	0
Tuesday	1	12	1	0	1	13	0	0	1	11	2	0	0	13	0	1
Wednesday	1	13	0	0	1	13	0	0	1	13	0	0	0	13	0	1
Thursday (morn.)	1	13	0	0	1	13	0	0	1	13	0	0	0	13	0	1
Thursday (aft.)	0	13	0	1	0	12	1	1	0	11	2	1	0	13	0	1
Friday (morn.)	1	11	0	2	1	11	0	2	1	11	0	2	1	11	0	2
Friday (aft. - DDoS)	1	13	0	0	1	13	0	0	0	12	1	1	0	13	0	1
Friday (aft. - port scan)	1	13	0	0	1	13	0	0	0	13	0	1	0	13	0	1
Total	6	101	1	3	6	101	1	3	4	97	5	5	1	102	0	8

Table 7.10: Comparative results on the CICIDS dataset between the proposed system and the baselines

Finally, the baseline comparison results on each day of the dataset are presented in Table 7.10, and visualised in Figure 7.10. It should be noted that, in contrary to the previous two datasets, the comparison results include only the major hosts of the dataset. This decision was made in order to achieve better interpretability of the results, since the excessively large amount of minor benign hosts of this dataset would significantly bias the accuracy values (all of them would be set close to 100%). It can be easily seen that the only baseline method providing comparable results to those achieved by the best performing configuration of the proposed methodology is the multivariate one, demonstrating actually the same results as those of the proposed system. As far as the rest two baseline methodologies are concerned, the symbolic sequential approach demonstrates equally moderate precision and recall value, originating from the same number of misclassified benign and malicious hosts, while the LOF based clustering approach, despite attaining an excellent precision due to the lack of false alarms, fails to detect more than one malicious host, resulting to a remarkably low recall value. As a result, it can be concluded that in the premises of this dataset, the multivariate sequential nature of the designed system is quite impactful towards the goal of detection, which as a matter of fact is not true for the sophisticated outlier detection algorithms introduced in the structure of the

sequential models.

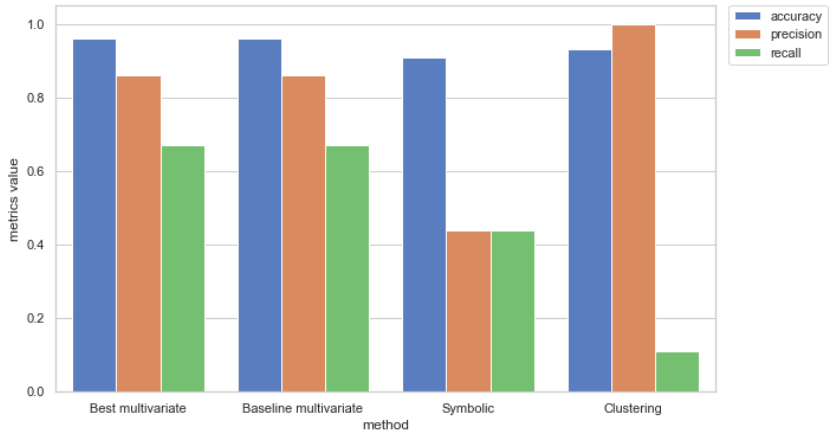


Figure 7.10: Comparison between the aggregated evaluation metrics of the proposed system and the baseline methods for the CICIDS2017 dataset

7.5.4. Comparison to State-of-the-art

The final part of the experimental procedure includes the comparison of the detection performance of the proposed system with that of a state-of-the-art detection technique, so as to examine the potential of the proposed methodology against a recently developed and published approach. To do so, the detection technique presented in [10] was selected, since the authors of this work evaluate their method, named *BotFP*, on one of the datasets used in this thesis, the CTU-13 dataset in particular. The functionality of this technique was touched upon in Chapter 3, yet a brief overview will be provided in this section too. In particular, as in the premises of this work, the creators of BotFP perform detection on host level by grouping the available flow records according to their source IP address. The feature set on which the detection procedure is based includes basic NetFlow attributes, like the source and destination port numbers and the destination IP address, and a frequency distribution signature is extracted for each host on each feature with the flows being separated by the communication protocol used. Since in the case of the CTU-13 dataset only three types of protocols are encountered (TCP, UDP, ICMP), the aforementioned procedure results to a feature set of $9xb$ features, where b denotes the number of bins used to extract the frequency distributions of each feature on each protocol category. After extracting the signature of each host in the training set, the DBSCAN clustering algorithm is utilized to cluster the hosts according to their signatures, and a label of benignity is assigned to each cluster based on the existence of at least one malicious host in its premises. In the testing phase each host is classified according to the label of the closest signature-based cluster derived during training. Finally, it should be pointed out that, as in [28], where the CTU-13 dataset was introduced, the creators of BotFP decide to use scenarios 3, 4, 5, 7, 10, 11, 12, and 13 for training purposes, utilizing in that way scenarios 1, 2, 6, 8, and 9 during testing. The comparative results attained by the designed system and BotFP on those scenarios are presented in Table 7.11 in a way similar to the one used in [10].

Scenario	Best Multivariate				BotFP			
	TP	TN	FP	FN	TP	TN	FP	FN
1	1	166	0	0	1	163	3	0
2	1	131	0	0	1	131	0	0
6	1	111	0	0	1	111	0	0
8	1	167	3	0	1	165	5	0
9	10	133	1	0	10	133	1	0

Confusion results

Method	1	2	6	8	9
Best Multivariate	1	1	1	0.98	0.99
BotFP	0.98	1	1	0.97	0.99

Accuracy results

Table 7.11: Comparative results between the designed system and BotFP on the CTU-13 dataset

After an initial inspection of the results presented in Table 7.11, it can be easily seen that there is a significant increase to the number of hosts included in these experiments comparing to those considered in the CTU-13 related experiments conducted so far in the premises of this thesis. This increased number of hosts

results from the attempt made to directly compare the detection performance of the designed methodology to that of BotFP, since its creators included a proportionally high number of hosts in their experiments. As a result, instead of considering as benign solely the hosts from which benign flows are originated, all hosts connected with benign as well as background flows were treated as benign. Subsequently, as in the work presented in [10], only the hosts, the source IP address of which is in the host network, while linked with at least 150 packets, were eventually utilised in the experimental procedure. Finally, an important difference between the designed system and BotFP regarding the training set used in the evaluation process of each technique should be pointed out. In the premises of this work, solely the benign hosts of scenario 3 are utilized to train the designed system, while this is not true in the case of BotFP, where both the benign and the malicious hosts of scenarios 3, 4, 5, 7, 10, 11, 12, and 13 are used for training purposes, meaning that BotFP is of a supervised nature. Bearing all these things in mind, it can be seen that both systems manage to identify all malicious in the test set, with the system designed in this work raising a lower number of false alarms comparing to BotFP. In particular, the proposed methodology produces the same results with BotFP in scenarios 2, 6, and 9, while raising less false alarms in scenarios 1 and 8 (three less and two less respectively). Given the aforementioned observations, it can be concluded that the proposed detection methodology manages to provide comparable detection performance to that of a state-of-the-art technique.

7.6. Discussion

Taking into consideration all the results presented so far, multiple conclusions can be drawn regarding the detection performance of the proposed methodology. The most important of these conclusions are briefly discussed in this section, yet a more comprehensive presentation of the main conclusions regarding the entirety of the current work can be found in Chapter 8. First, the fact that the best performing version of the proposed detection pipeline for each dataset varies in terms of the detection algorithm and the feature set used suggests that the nature of the dataset highly affects the appropriateness of the detection technique. Yet, the fact that the proposed methodology achieves quite promising results in most cases implies that an unsupervised multivariate sequential approach operating on basic NetFlow features can indeed be used in the premises of a network-oriented anomaly detection task. In fact, the undetected malicious hosts, in most cases, were associated with either mixed flows (both benign and malicious), with the benign flows representing the majority of the host-specific traffic, or with a significantly low traffic volume (tens of flows). In such cases it would be potentially beneficial to use a more fine-grained detection approach by modelling the communication traffic of benign connections instead of hosts. In addition, it is worth noting that the selection of the NetFlow features to be included in the learning process is of high importance for the detection performance of the designed system. In fact, in the case of the CTU-13 dataset the identification of the malicious behaviour of a specific bot (Neris) was achieved only after the source port numbers associated with each flow were incorporated in the learning process. As far as the baseline comparison results are concerned, it is important to note that in two out of the three datasets used in the experimental procedure the baseline multivariate approach demonstrated similar results to the ones attained by the best performing configuration of the proposed system. This observation suggests that simpler detection techniques could be also used in each state, which as a matter of fact would reduce significantly the detection complexity and render the proposed system more efficient, especially in cases that the computationally demanding LOF based approach is selected. Furthermore, the multivariate sequential approach followed in the proposed methodology seems to outperform the symbolic and clustering baselines, with the surprisingly good results of these two methods in the UNSW-NB15 dataset being attributed to the direct correlation of the benignity of hosts to certain NetFlow features. Finally, the competitive detection results that the designed system produced, when examined in comparison to a state-of-the-art detection technique on the CTU-13 dataset, indicate its high detection potential.

8

Conclusion and Final Remarks

The primary goal of this work was the development of an effective anomaly detection system able to operate in an unsupervised manner on aggregated privacy preserving representations of network traffic. Towards that end, the designed system was based on the assumption that, if robust models reflecting the various benign behavioral patterns in the recorded communication traffic of a given network are extracted, behaviors not conforming to the derived benign models can be labelled as malicious. As a result, a multivariate state machine learning approach boosted by multiple well-known detection techniques was utilized to model the benign communication traffic of hosts in multiple datasets. The designed system was evaluated both on different configurations of its own components, and in comparison to some baselines and a state-of-the-art methodology, so that meaningful conclusions regarding its nature and its detection capabilities can be drawn. This chapter aims to summarize these conclusions and answer the main questions set in the beginning of this thesis, while presenting the primary strong points and limitations of the designed system. Finally, some directions, towards which the work conducted in this thesis can be extended in the future, are briefly discussed.

8.1. Main Conclusions

An initial discussion on the conclusions drawn from the evaluation procedure followed in this thesis was conducted at the end of Chapter 7. In this section, these conclusions will be reflected on the primary research questions set in the beginning of this work in the following question-to-answer format:

1. **How effective can multivariate behavioral communication profiles extracted solely from basic NetFlow features of benign data be proved for the purpose of malware detection in network traffic?**

   As it can be seen from the results obtained on all the datasets used in the evaluation procedure, the best performing configurations of the proposed system manage to attain relatively high detection performance, along with a low number of false alarms. Apart from few cases, which will be addressed further in Section 8.3 of this chapter, the high detection performance achieved on different datasets incorporating a great variety of malicious network behavior indicates the capabilities of the proposed methodology. The high detection prospect of the designed system is further supported by the fact that it provides competitive results when examined in comparison to a state-of-the-art detection technique on the CTU-13 dataset. Of course, there are certain shortcomings associated with the designed pipeline, yet the attained results seem promising regarding its detection potential.

2. **What type of malicious behaviour can be successfully identified from the proposed detection system?**

   The datasets used for evaluating the designed detection system contain different types of attacks, ranging from bots, DoS attacks, and port scans, to infiltration, web attacks, and sql injection attacks. The proposed methodology manages to identify the majority of these malicious behaviors, which as a matter of fact strengthens the opinion that the utilization of benign traffic in the training phase of the proposed pipeline provides great detection flexibility to the designed system, allowing it to identify different types of malicious behaviors not conforming to the normal communication patterns captured during the learning process.

3. **What is the impact from the incorporation of outlier detection models in the structure of multivariate state machines towards the goal of detection?**

   As it was discussed in Chapter 7, the multivariate baseline method, which utilized the same structure as the designed system, with the main difference being the incorporation of a simple model in each state of the derived state machines, tended to produce comparable results to those achieved when more complex outlier detection algorithms were fitted in each state. Given the significant boost in the computational performance of the system, when a simple detection technique is fitted in each state, it is considered beneficial to further examine the possibility of using much simpler detection algorithms fitted in each state of the multivariate sequential models used for profiling the benign behavior recorded in the network under examination.

8.2. Strong Points

The strongest point of the work presented in the premises of this thesis can be summarised by the fact that the proposed pipeline provides a solid overall detection performance in all the datasets taken into account in this thesis, which as a matter of fact was the primary goal of this work. This result is of high importance given the characteristics of the designed pipeline. First, the designed system operates in an unsupervised manner, since it only needs benign communication traffic in its training phase, the acquisition of which is a much easier task comparing to the case of malicious traffic. In addition, the behavioral models used within the system are learnt from basic features of NetFlow data, avoiding the use of highly sophisticated attributes that might not be provided by some network monitoring tools. Furthermore, it utilizes a multivariate sequential model in order to learn benign communication profiles from certain network-related aggregation entities, like hosts, and connections, enabling in that way the incorporation of underlying temporal patterns in the learning and detection processes, while providing a more fine-grained analysis on the network data comparing to the typical ML dataset-like representation. To the best of our knowledge, this is the first work that incorporates all the aforementioned characteristics in one system. On top of that, the relatively high detection performance achieved in datasets containing different type of cyber attacks and malicious network behaviours indicates that the designed system is able to detect a wide variety of malicious activity, while the competitive detection results attained, when the proposed system is examined in comparison to a state-of-the-art technique, highlight further its high detection potential. Finally, the adoption of a multivariate sequential approach frees the detection process from the information loss associated with the needed encoding procedure in most sequential methodologies.

8.3. Limitations

There are three main limitations of the work conducted in this thesis. The first is reflected by the malicious hosts that were not identified in the evaluation process. Most of these hosts were associated either with a significant number of benign flows along with their malicious traffic or with a short-lived malicious behavior comprising of tens of flows. The first observation suggests that the designed system is not able to identify hosts of mixed nature, when the malicious flows associated with them constitute the minority of their entire traffic. The second observation is linked to the fact that the designed system is able to provide robust prediction only when a sufficient number of flows is available for each network entity, which as a matter of fact suggests that when the malicious traffic is expressed through short-lived connections the proposed methodology would most probably fail to provide high detection performance. Finally, the detection phase of the proposed system involves the replay of the set of traces of the examined network entity on all the benign behavioral models extracted during the training phase. Such a strategy can work when there is a relatively low number of benign profiles, yet, if the number of benign profiles is increased up to a significant extent, the detection process could become computationally infeasible especially if computationally intensive detection algorithms, like LOF, are fitted in each state of the benign behavioral models.

8.4. Future Work

As with every newly developed method, there is significant room for improvement, while there are several promising research paths that have not been explored or thoroughly covered in the premises of this work. This section provides a brief presentation of the main future directions that could be followed in continuance of the work presented in this thesis. These directions can be seen as follows:

- As it was mentioned in the discussion regarding the experimental results presented at the end of Chap-

ter 7, it would be interesting to evaluate the proposed system on connection level, since such an analysis could offer more fine-grained detection results. Especially, in cases where the number of flows per connection is large enough, as it is in the case of the UNSW-NB15 dataset, such an approach could potentially lead to better results than the host level based approach, since each modelling entity would be associated with less noise.

- In the premises of this thesis, the inference algorithm implemented in *flexfringe* was utilized without any task-specific adjustments made. Such adjustments could lead to the extraction of models that could capture even better the communication patterns in the premises of a network.
- The multivariate FSMs used as the behavioral profiles of certain network entities are capable of capturing any sequential temporal patterns present in the recorded traffic, yet, if an anomaly is of a collective nature, like the excessive increase in the number of attempted connections to a host during a DoS attack, it is possible that such models will be unable to detect it. As a result, it could be beneficial to examine the incorporation of aggregated features in the modelling process, so that such cases can be effectively treated.
- The baseline multivariate method seems to produce comparable detection performance to that achieved when more complex and computationally intensive detection algorithms are fitted in each state of the derived profiles. As a result, it would be of great interest to fit a detection technique that would combine both effectiveness and simplicity in the structure of the multivariate FSMs.
- Finally, in order to deal with the latency introduced in the detection process by the exhaustive search for a match between the set of testing traces examined and all the benign models used by the system, a hashing scheme could be utilized to reduce the number of models that need to be evaluated for a potential match.

Bibliography

[1] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[2] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[3] Mennatallah Amer, Markus Goldstein, and Slim Abdennadher. Enhancing one-class support vector machines for unsupervised anomaly detection. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, pages 8–15, 2013.

[4] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and computation*, 75(2):87–106, 1987.

[5] Ia M Barzdin. *Finite automata-behavior and synthesis*. North-Holland Publishing Company, 1973.

[6] Ulrich Bayer, Paolo Milani Comparetti, Clemens Hlauschek, Christopher Kruegel, and Engin Kirda. Scalable, behavior-based malware clustering. In *NDSS*, volume 9, pages 8–11. Citeseer, 2009.

[7] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[8] Alan W Biermann and Jerome A Feldman. On the synthesis of finite-state machines from samples of their behavior. *IEEE transactions on Computers*, 100(6):592–597, 1972.

[9] Leyla Bilge, Davide Balzarotti, William Robertson, Engin Kirda, and Christopher Kruegel. Disclosure: detecting botnet command and control servers through large-scale netflow analysis. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 129–138, 2012.

[10] Agathe Blaise, Mathieu Bouet, Vania Conan, and Stefano Secci. Botfp: Fingerprints clustering for bot detection. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–7. IEEE, 2020.

[11] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[12] Daniela Brauckhoff, Xenofontas Dimitropoulos, Arno Wagner, and Kavè Salamatian. Anomaly extraction in backbone networks using association rules. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pages 28–34, 2009.

[13] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[14] Matthias Büchler, Karim Hossen, Petru Florin Mihancea, Marius Minea, Roland Groz, and Catherine Oriat. Model inference and security testing in the spacios project. In *2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, pages 411–414. IEEE, 2014.

[15] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[16] Tsun S. Chow. Testing software design modeled by finite-state machines. *IEEE transactions on software engineering*, (3):178–187, 1978.

[17] Benoit Claise, Brian Trammell, and Paul Aitken. Specification of the ip flow information export (ipfix) protocol for the exchange of flow information. *RFC 7011 (Internet Standard), Internet Engineering Task Force*, pages 2070–1721, 2013.

[18] William W Cohen. Fast effective rule induction. In *Machine learning proceedings 1995*, pages 115–123. Elsevier, 1995.

[19] Paolo Milani Comparetti, Gilbert Wondracek, Christopher Kruegel, and Engin Kirda. Prospex: Protocol specification extraction. In *2009 30th IEEE Symposium on Security and Privacy*, pages 110–125. IEEE, 2009.

[20] Guillaume Dewaele, Kensuke Fukuda, Pierre Borgnat, Patrice Abry, and Kenjiro Cho. Extracting hidden anomalies using sketch and non gaussian multiresolution statistical detection procedures. In *Proceedings of the 2007 workshop on Large scale attack defense*, pages 145–152, 2007.

[21] Catalin Dima. Real-time automata. *Journal of Automata, Languages and Combinatorics*, 6(1):3–24, 2001.

[22] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[23] Ricardo Dunia and S Joe Qin. Subspace approach to multidimensional fault identification and reconstruction. *AICHE journal*, 44(8):1813–1831, 1998.

[24] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[25] Marina Evangelou and Niall M Adams. Predictability of netflow data. In *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, pages 67–72. IEEE, 2016.

[26] Yudan Fan, Yuna Zhu, and Lin Yuan. Automatic reverse engineering of unknown security protocols from network traces. In *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, pages 1139–1148. IEEE, 2018.

[27] Gilberto Fernandes Jr, Luiz F Carvalho, Joel JPC Rodrigues, and Mario Lemes Proença Jr. Network anomaly detection using ip flows with principal component analysis and ant colony optimization. *Journal of Network and Computer Applications*, 64:1–11, 2016.

[28] Sebastian Garcia, Martin Grill, Jan Stiborek, and Alejandro Zunino. An empirical comparison of botnet detection methods. *computers & security*, 45:100–123, 2014.

[29] E Mark Gold. Complexity of automaton identification from given data. *Information and control*, 37(3): 302–320, 1978.

[30] Cyril Goutte, Peter Toft, Egill Rostrup, Finn A Nielsen, and Lars Kai Hansen. On clustering fmri time series. *NeuroImage*, 9(3):298–310, 1999.

[31] Margaret Gratian, Darshan Bhansali, Michel Cukier, and Josiah Dykstra. Identifying infected users via network traffic. *Computers & Security*, 80:306–316, 2019.

[32] Roland Groz, Nicolas Bremond, Adenilso Simao, and Catherine Oriat. hw-inference: A heuristic approach to retrieve models through black box testing. *Journal of Systems and Software*, 159:110426, 2020.

[33] Christian Hammerschmidt, Samuel Marchal, Radu State, Gaetano Pellegrino, and Sicco Verwer. Efficient learning of communication profiles from ip flow records. In *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, pages 559–562. IEEE, 2016.

[34] Christian Hammerschmidt, Samuel Marchal, Radu State, and Sicco Verwer. Behavioral clustering of non-stationary ip flow record data. In *2016 12th International Conference on Network and Service Management (CNSM)*, pages 297–301. IEEE, 2016.

[35] Edward James Hannan. *Multiple time series*, volume 38. John Wiley & Sons, 2009.

[36] Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

[37] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003.

[38] Marijn JH Heule and Sicco Verwer. Software model synthesis using satisfiability solvers. *Empirical Software Engineering*, 18(4):825–856, 2013.

[39] Rick Hofstede, Pavel Čeleda, Brian Trammell, Idilio Drago, Ramin Sadre, Anna Sperotto, and Aiko Pras. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *IEEE Communications Surveys & Tutorials*, 16(4):2037–2064, 2014.

[40] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. Introduction to automata theory, languages, and computation. *Acm Sigact News*, 32(1):60–65, 2001.

[41] Wenjie Hu, Yihua Liao, and V Rao Vemuri. Robust anomaly detection using support vector machines. In *Proceedings of the international conference on machine learning*, pages 282–289. Citeseer, 2003.

[42] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

[43] Malte Isberner, Falk Howar, and Bernhard Steffen. The ttt algorithm: a redundancy-free approach to active automata learning. In *International Conference on Runtime Verification*, pages 307–322. Springer, 2014.

[44] Teuvo Kohonen, MR Schroeder, TS Huang, and Self-Organizing Maps. Springer-verlag new york. *Inc., Secaucus, NJ*, 43(2), 2001.

[45] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[46] Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffic feature distributions. *ACM SIGCOMM computer communication review*, 35(4):217–228, 2005.

[47] Kevin J Lang. Random dfa's can be approximately learned from sparse uniform examples. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 45–52, 1992.

[48] Kevin J Lang, Barak A Pearlmutter, and Rodney A Price. Results of the abbadingo one dfa learning competition and a new evidence-driven state merging algorithm. In *International Colloquium on Grammatical Inference*, pages 1–12. Springer, 1998.

[49] Xin Li, Fang Bian, Mark Crovella, Christophe Diot, Ramesh Govindan, Gianluca Iannaccone, and Anukool Lakhina. Detection and identification of network anomalies using sketch subspaces. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 147–152, 2006.

[50] Yihua Liao and V Rao Vemuri. Use of k-nearest neighbor classifier for intrusion detection. *Computers & security*, 21(5):439–448, 2002.

[51] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144, 2007.

[52] Qin Lin, Sridha Adepu, Sicco Verwer, and Aditya Mathur. Tabor: A graphical model-based approach for anomaly detection in industrial control systems. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 525–536, 2018.

[53] Qin Lin, Yihuan Zhang, Sicco Verwer, and Jun Wang. Moha: a multi-mode hybrid automaton model for learning car-following behaviors. *IEEE Transactions on Intelligent Transportation Systems*, 20(2):790–796, 2018.

[54] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[55] David Lo, Leonardo Mariani, and Mauro Santoro. Learning extended fsa from software: An empirical assessment. *Journal of Systems and Software*, 85(9):2063–2076, 2012.

[56] Davide Lorenzoli, Leonardo Mariani, and Mauro Pezzè. Automatic generation of software behavioral models. In *Proceedings of the 30th international conference on Software engineering*, pages 501–510, 2008.

[57] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[58] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *Proceedings*, volume 89. Presses universitaires de Louvain, 2015.

[59] Leonardo Mariani and Fabrizio Pastore. Automated identification of failure causes in system logs. In *2008 19th International Symposium on Software Reliability Engineering (ISSRE)*, pages 117–126. IEEE, 2008.

[60] Leonardo Mariani, Fabrizio Pastore, and Mauro Pezze. Dynamic analysis for diagnosing integration faults. *IEEE Transactions on Software Engineering*, 37(4):486–508, 2010.

[61] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205, 2017.

[62] Aziz Mohaisen, Omar Alrawi, and Manar Mohaisen. Amal: High-fidelity, behavior-based automated malware analysis and classification. *computers & security*, 52:251–266, 2015.

[63] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[64] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE, 2015.

[65] Tony Ohmann, Michael Herzberg, Sebastian Fiss, Armand Halbert, Marc Palyart, Ivan Beschastnikh, and Yuriy Brun. Behavioral resource-aware model inference. In *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*, pages 19–30, 2014.

[66] José Oncina and Pedro Garcia. Inferring regular languages in polynomial updated time. In *Pattern recognition and image analysis: selected papers from the IVth Spanish Symposium*, pages 49–61. World Scientific, 1992.

[67] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[68] Gaetano Pellegrino, Qin Lin, Christian Hammerschmidt, and Sicco Verwer. Learning behavioral fingerprints from netflows using timed automata. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 308–316. IEEE, 2017.

[69] Roberto Perdisci, Wenke Lee, and Nick Feamster. Behavioral clustering of http-based malware and signature generation using malicious network traces. In *NSDI*, volume 10, page 14, 2010.

[70] Michael O Rabin. Probabilistic automata. *Information and control*, 6(3):230–245, 1963.

[71] Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *Ann. Math. Statist.*, 27(3):832–837, 09 1956. doi: 10.1214/aoms/1177728190. URL `https://doi.org/10.1214/aoms/1177728190`.

[72] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[73] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.

[74] David W Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[75] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP*, pages 108–116, 2018.

[76] Michael Sipser. Introduction to the theory of computation. *ACM Sigact News*, 27(1), 1996.

[77] Qing Song, Wenjie Hu, and Wenfang Xie. Robust support vector machine with bullet hole image classification. *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, 32(4): 440–448, 2002.

[78] Thomas A Sudkamp and Alan Cotterman. *Languages and machines: an introduction to the theory of computer science*, volume 2. Addison-Wesley Reading, Mass., 1988.

[79] Florian Tegeler, Xiaoming Fu, Giovanni Vigna, and Christopher Kruegel. Botfinder: Finding bots in network traffic without deep packet inspection. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 349–360, 2012.

[80] Franck Thollard, Pierre Dupont, Colin De La Higuera, et al. Probabilistic dfa inference using kullback-leibler divergence and minimality. In *ICML*, pages 975–982, 2000.

[81] Wesley van der Lee and Sicco Verwer. Vulnerability detection on mobile applications using state machine inference. In *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 1–10. IEEE, 2018.

[82] SE Verwer, MM De Weerdt, and Cees Witteveen. An algorithm for learning real-time automata. In *Benelearn 2007: Proceedings of the Annual Machine Learning Conference of Belgium and the Netherlands, Amsterdam, The Netherlands, 14-15 May 2007*, 2007.

[83] Sicco Verwer and Christian A Hammerschmidt. flexfringe: a passive automaton learning package. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 638–642. IEEE, 2017.

[84] Sicco Verwer, Mathijs de Weerdt, and Cees Witteveen. One-clock deterministic timed automata are efficiently identifiable in the limit. In *International Conference on Language and Automata Theory and Applications*, pages 740–751. Springer, 2009.

[85] Sicco Verwer, Mathijs de Weerdt, and Cees Witteveen. A likelihood-ratio test for identifying probabilistic deterministic real-time automata from positive data. In *International Colloquium on Grammatical Inference*, pages 203–216. Springer, 2010.

[86] Sicco Verwer, Mathijs de Weerdt, and Cees Witteveen. The efficiency of identifying timed automata and the power of clocks. *Information and Computation*, 209(3):606–625, 2011.

[87] Sicco Verwer, Mathijs de Weerdt, and Cees Witteveen. Efficiently identifying deterministic real-time automata from labeled data. *Machine learning*, 86(3):295–333, 2012.

[88] Jiacun Wang. *Formal Methods in Computer Science*, pages 34–36. CRC Press, 2019.

[89] Xiaogang Wang, Weiliang Qiu, and Ruben H Zamar. Clues: A non-parametric clustering method based on local shrinking. *Computational Statistics & Data Analysis*, 52(1):286–298, 2007.

[90] Nong Ye and Qiang Chen. An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. *Quality and Reliability Engineering International*, 17(2):105–112, 2001.