An Unsupervised Approach for False Alarm Filtering in Rule-based NIDS

Magdalena Simidžioski

Technische Universiteit Delft

TU Delft

An Unsupervised Approach for False Alarm Filtering in Rule-based NIDS

by

Magdalena Simidžioski

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Specialization: Cyber Security

at the Delft University of Technology,
Faculty of Electrical Engineering, Mathematics and Computer Science.
to be defended publicly on 26 June 2020

Thesis committee:	Dr. Z. Erkin,	TU Delft
	Dr. C. Hernandez Ganan,	TU Delft
	Dr. P. Murukannaiah,	TU Delft

TU Delft Delft University of Technology

Acknowledgements

I wish to thank my supervisor Dr. Carlos Hernandez Ganan for his support and academic guidance during the entire process of writing my thesis. The constructive feedback and various discussions were of tremendous help. I also wish to thank MSc Mathew Vermeer for his input and critical feedback whenever I had various questions and obstacles. I would like to thank Dr. Zekeriya Erkin and Dr. Pradeep Murukannaiah for being part of the thesis committee.
I would like to thank Murat Kurtdaylar for giving me the chance to perform research in a dynamic organisation. I would like to thank Peter van Eijk for giving me the opportunity to collect and analyse the data that was needed for this research. I would also like to thank my colleagues Johan Ruijgt, William Beernink and Reinout Hoornweg for their assistance in both the technical and academic matters of this process. Finally, I would like to thank my family and friends for constantly supporting me during this challenging time.

Magdalena Simidžioski
Delft, June 2020

Abstract

To detect malicious activities in a network, intrusion detection systems are used. Even though these solutions are widely deployed for this purpose they have one serious shortcoming which is the huge amount of false alarms that they are generating. Different measures are taken to tackle this problem such as manually changing the settings of the intrusion detection systems. However, this is an infeasible approach for organisations since a network is changing regularly and specialists that have good knowledge of both the environment and the solution are required.The existing unsupervised approaches cannot be implemented as fully automated solutions because of the need to tune hyper-parameters. Additionally, the implementation of the existing solutions is complicated since often multiple models and the constant updating thereof is required which is a computationally intensive process considering the selected algorithms. In this work, the possibilities to reduce the false alarms in an automated manner are investigated. This is done by applying unsupervised anomaly detection techniques on the resulting alert data to distinguish regular alarms from high priority ones. Real alert data is collected from a network of a large organisation and an additional synthetically generated data set is used to evaluate the proposed approach. Four unsupervised anomaly detection algorithms are chosen to model the regular alerts. These are Local Outlier Factor (LOF), Isolation Forest (IF), Histogram-based Outlier Score (HBOS) and Cluster-based Local Outlier Factor (CBLOF). We show that this approach can greatly reduce the false alarms in real environments. By adding noise to the data we evaluate the performance of the models and propose a method that can be used to determine when the model needs to be retrained. This is done by deriving a metric that is used to trigger the system to automatically retrain on the most recent historic data. This is necessary in order to make the system automated and adaptable to changes in the network.

Contents

1

Introduction

The importance of cyber security is increasing by the day with the major changes that the world has seen in the last decade. It is not a matter of question anymore whether IT systems are at risk of cyber attacks. The problem is so broad that it affects every entity from critical infrastructures and state based actors to small companies providing various services. The need for a fast development of cyber defences is thus increasing in parallel with the continuous search and development of exploits. Over the past decade the number of vulnerabilities has been continuously increasing with the peak being 2019 until now and this number is expected to grow in the next few years [44].

The worldwide spending on cyber security has been estimated to reach approximately $143 billion by 2022 [19]. Exploit frameworks are developed to exploit the most recently discovered vulnerabilities in systems while at the same time vendors are searching for solutions to patch these vulnerabilities. It is unfortunately often the case that systems run outdated software and patches are not installed on time. Public and private organisations face a common problem. They need to defend their networks from all kinds of cyber threats but this has proven to be a very challenging task. Various tools and frameworks have been developed for this purpose that are widely used across the world. However, there is a huge shortage of employees with proper education and understanding in this field and often it is very difficult for organisations to understand their risks and implement an appropriate and effective defence strategy. Additionally, the imbalance between attackers and defenders is significant. For an attacker it takes only a few clicks to use an already available exploit tool and break into a system whereas for a defender it may take days or even months to properly configure and maintain the network and the implemented defences. With all these challenges in mind a simple approach could be of great value for organisations to prevent or at least timely detect threats that are present in their networks and decrease the probability of success for potential attackers. The approach should be cost-effective and simple to implement and maintain in order to be applicable for a

wider range of organisations.

Intrusion Detection Systems have a wide range of uses across this landscape. Their first use was to help administrators monitor logs and audit trails of various systems in order to detect anomalies or other unusual behaviour [6]. However, it was soon found that this type of monitoring was not sufficient to discover all potential threats that a system was facing. As a solution to this problem [15] proposed the first intrusion detection system that could monitor both host and network data, named Intrusion Detection Expert System (IDES). This system incorporates both rule-based and anomaly (statistical) based detection techniques. This system is considered to be the basis for all intrusion detection systems that have been developed ever since [29, 41, 50, 55, 60, 68, 74, 78].

As the role of intrusion detection systems is to detect intrusion in a network, they are designed to capture and analyze huge amounts of network traffic and host data. Because of the volume of the data and the context that is missing it becomes very challenging to select only the real threats that are present in the data. In reality these systems generate lots of false positive alerts, which means that they label observed traffic as malicious when it is completely normal or harmless to the monitored system. There is a large need for improving the existing intrusion detection systems to be fast, more accurate and preventing attacks rather than just detecting them afterwards.

Rule-based Intrusion Detection Systems are currently amongst the most widely used threat intelligence tools by all kinds of organisations. Snort is the most popular open-source IDS with over 5 million downloads which makes it the most widely deployed IDS in the world [42]. Additionally many already known vulnerabilities are still being frequently exploited. The Microsoft Security Intelligence report states that the majority of the customers are victims of vulnerabilities for which patches have been developed several years ago [59, 69]. Surveys show that security patching is difficult and often decide not to apply the necessary patches because of the functionality changes, incompatibilities with their applications and the fear of breaking the system [1, 32] This patching problem is also the cause of approximately 18% of all detected network level vulnerabilities [17]. The Edgescan vulnerabilities statistics report [18] has found that the majority of the components that were vulnerable had working exploits for already known vulnerabilities. Also default configurations and inappropriately configured systems and components were amongst the most common causes of security incidents in 2018. The category that is mostly affected by these attacks and breaches are the small and medium sized organisations, in 2019 43% of the total cyber attack victims were SME's. [77] Another very recent example (2019) are the Microsoft 'Gallium' attacks [26]. These attacks were performed by using cheap hacking tools where the hackers selected vulnerable servers and used already known exploits against these servers to gain access.

Rule-based NIDs have their shortcomings but remain very effective in detecting attacks that are already known in cyberspace. Even though the biggest limitation of these systems is their inability to detect zero days attacks, new rules are added to these systems very rapidly. Zero days attacks are often exploited even after patches have been released and in some cases these vulnerabilities cannot be patched at all. Therefore, rule-based NIDS remain a useful component of the cyber defences. Of course, to increase and have maximum protection, the ideal solution would be to combine both anomaly based and rule based types of NIDS. This research only focuses on rule based NIDS and the improvement thereof, thus no claims are made that this solution is 100% security proof. Rule-based NIDS, if deployed properly can be of great value to any organisation.

1.1. Problem Statement

Many organisations struggle with choosing and implementing the proper security defences for their networks. Network monitoring is a basis for a good defence strategy since it provides visibility in the network and is often the first place where malicious traffic can be observed. Therefore, having solid network visibility is very important. There exist open-source solutions for this purpose which can be deployed by organisations. However, there are several big challenges concerning the effective deployment of a rule-based NIDS. The first and biggest problem is the amount of false alarms. To prevent this several approaches have been adopted. A common approach is tuning the NIDS manually. This action is tightly related to the ever-changing monitored environment. Each alteration of the NIDS configuration and rule set would require a reboot of the system and it is often difficult to decide what to change in the configuration. The second problem that arises is the dilemma about which rules to activate or deactivate since this is very often dependent on the monitored environment thus context remains a very important aspect. Not having relevant information to use for the evaluation of malicious attempts makes it very difficult to distinguish high priority events from irrelevant ones. Additionally, security experts are required to manage this work and it is very time consuming.

Given the challenges at deploying an effective and successful intrusion detection system, this thesis aims at exploring techniques that can be used to develop a system as a module to existing NIDS that is adjustable to the monitored environment to effectively reduce the false alarm rate. The main goal is thus to prevent manual alterations to the NIDS and make the effective deployment of the NIDS independent of human experts.

1.2. Research Question

This research is exploratory and investigates a new approach for reducing the false alarm rate in rule-based NIDS in an unsupervised setting. In order to do this several unsupervised anomaly detection methods are explored and tested on real data, their robustness to changes in the network is evaluated and a threshold for automatic retraining is defined. The main research question is defined as follows:

Main RQ: To what extent can existing rule-based NIDS be effectively deployed by automating the process of false alarm filtering using unsupervised anomaly detection techniques?

To answer this question, multiple sub questions are formulated that will target the different components of this problem:

RQ1: What are the existing techniques for false alarm reduction in rule-based NIDS?

This question is about exploring existing research and techniques related to false alarm reduction in rule-based NIDS. In the scope of this question both knowledge-based and data mining-based techniques will be reviewed and a research gap will be identified.

RQ2: How effective are existing network intrusion detection systems when deployed without any prior alterations to their settings?

This question is about exploring how effective the usage of these intrusion detection systems is and evaluating to what extent a NIDS is able to recognise only relevant events in real world scenarios where often it is the case that there is not enough human expertise or time available to constantly tune their settings. In order to answer this question a NIDS will be set up in a real environment. The NIDS will be used in a default setting and the output of the NIDS will be analysed.

RQ3: Which unsupervised outlier detection techniques can be used for filtering false alarms in rule-based NIDS?

This question concerns the unsupervised approach that is taken to reduce false alarms. This is necessary since the model uses only the output of the NIDS where the alerts are not labelled. To answer this question multiple unsupervised machine learning algorithms will be explored and evaluated that are applied to similar problems.

RQ4: How can the model be retrained in order to remain accurate upon changes in the network ?

The last question concerns the application of the proposed solution in a production environment. To answer this question the model will be tested on data collected on a real network. To evaluate the robustness of the model noise will be added to the data that mimics new processes and attribute values which would make the NIDS generate more false alarms. A metric will be derived to indicate at which point the models need to be retrained on more recent, representative data to remain accurate.

1.3. Research Scope

This research focuses specifically on rule based intrusion detection systems. The alerts that will be included for the implementation of the module are based on rule-based alert output sets. Furthermore, for the exploration of environment information only the CVE vulnerability database is consulted since most events do either include a reference to the CVE database or have no reference at all. Additional information about the network such as open ports, services and operating system information is also considered.
The data that is used represents only the traffic that is detected by the Snort engine as possibly malicious. There are cases where the NIDS misses attacks because of evolved traffic patterns of known attacks. These cases are not considered in this research since the goal is to filter the amount of alarms already detected by the NIDS. For this shortcoming an anomaly detection engine can be used.
The research will be done in collaboration with an organisation which identity will remain anonymous throughout the report. Since the monitored environment is very specific to this organisation, some of the outcomes of this research can be considered specific for this setting. However, an additional synthetic data set is used to test the performance of the proposed method. The methodology of extracting and using the various elements for the reduction of false positives can be applied generally to any data set produced by a rule-based NIDS.

1.4. Contribution

The scientific contribution of this thesis is the proposal of a false alarm filtering technique framed as an outlier detection problem. A novel approach is applied to this problem by using outlier detection techniques to identify true alarms. Also a proposal of a metric to indicate when the model(s) needs to be retrained on more recent data is given. The exploration and evaluation of unsupervised outlier detection techniques can be used to filter false alarms efficiently. This research explores to what extent anomaly detection algorithms can be applied

on rule-based intrusion detection systems and how these models can be made adaptable to change in dynamic environments. Additionally, the effectiveness of existing knowledge-based techniques is evaluated in a real network setting and a comprehensive analysis is provided about the characteristics and main difficulties with the use of rule-based NIDS. Experiments are conducted with these models to evaluate how robust they are on changes and to derive indicators about the thresholds for which these proposed models need to be retrained. The result of this thesis is an automated prototype for false alarm filtering with a selection of anomaly detection techniques and exploration of a a historical data range and recommendations for model retraining. The contributions are listed below:

- Proposal of a novel method to filter false alarms framed as an outlier detection problem independent of expert knowledge.
- An assessment of the extent to which knowledge-based approaches are useful for reducing the false alarm rate in a real network.
- An assessment of the performance of the proposed method with a default set of hyper-parameters in order to prevent manual adjustments
- An assessment of the applicability of this method including an evaluation of the models trained on different ranges of historic data.
- Proposal of a metric to indicate when the model(s) needs to be retrained on more recent data.

1.5. Result Summary

The main finding of this research is that the process of filtering false alarms using anomaly detection systems can be an effective alternative to manual adjustments or knowledge based filtering techniques. The method is tested on multiple data sets and detects all true attacks while having a low false positive rate. Additionally, the proposed model is shown to be robust to changes and an indicator for retraining is defined. The main limitation of this approach is the lack of context. Furthermore, the main cause of these remaining false positives is the similarity in frequency values. The chosen models trained on the selected attributes are not capable of fully distinguishing true from false alarms.

1.6. Report Structure

In this chapter the motivation and problem statement of this research were provided as well as the proposed solutions, the scope and the contribution of this research. The following chapter covers the theory and concepts that should be known to understand the rest of the paper. The third chapter covers a review of

selected works that has been performed on this research topic. In this chapter the most relevant works and their limitations are summarised. Chapter four covers the methodology that is followed during this research. This chapter provides a description of the proposed solution including the feature and algorithm selection and evaluation methods. In Chapter five a description of the data is provided including the main observations. This chapter also covers the different test sets that are used for evaluation. In Chapter six the main results are presented and discussed. In Chapter seven the main limitations are stated and ideas for future research are proposed. In Chapter eight the research questions are answered and the main conclusion of this work is provided.

2

Preliminaries

In this chapter background knowledge is presented that is directly related to the studied problem. The different intrusion detection systems are described including SNORT, the NIDS that is used during this research. The encoding techniques, distance metric and unsupervised algorithms are also described.

2.1. Intrusion Detection Systems

Intrusion detection mechanisms are a broadly studied topic within network and cyber security. Many different techniques are developed for this purpose. The two main categories based on the technique that is used are rule-based and Anomaly-based intrusion detection. Another categorisation of intrusion detection mechanisms can be made based on the place where the system resides. The two main categories are Network-based (NIDS) and Host-based (HIDS) Intrusion Detection Systems. There also exists techniques which deploy both of these and are known as Hybrid or Combined (CIDS) intrusion detection systems.

For the purpose of this research, we will study the alerts generated by a rule-based NIDS which resides on the Network. In this report we will refer to this system as NIDS. The NIDS that will be used is an open source NIDS called Snort[42]. This NIDS was firstly chosen because it is open source and its rule-database gets updated frequently by multiple sources. Furthermore, it is easy to set-up and have it running very quickly.

2.1.1. Anomaly-based IDS

An anomaly based IDS is a system that tries to identify malicious traffic in a continuous stream by learning a model that defines what normal behaviour looks like in the monitored system and alerting on any traffic that deviates from this learned normal behaviour. These systems are capable of detecting zero-day attacks which is one of their strongest points. There exist many different

algorithms that are used for this purpose like neural networks or data mining techniques. Generally, these types of IDS produce a lot of false positives and need continuous adjustments of their thresholds to be effectively deployed.

2.1.2. Rule-based IDS

A rule-based IDS is a system that tries to identify malicious traffic in a continuous stream by matching payloads to predefined rules. The system looks at packet bytes and sequences and matches these to rules of known malicious activity. These systems are thus capable of detecting all known attacks that are contained in their rule set. It is very important to update these rule set very frequently for better detection. These systems are also known to produce many false positives when not configured properly. This means again setting rule thresholds, enabling and disabling rules according to the environment and regularly adjusting the settings for an effective deployment.

2.1.3. Snort

The rule-based Intrusion Detection System that will be studied during this research is Snort [42]. It will be setup to monitor network traffic. This NIDS detects attacks by protocol analysis and content matching. It has been widely employed as a NIDS and has a very rich rule base, which is frequently updated for newly emerging attacks and rules. It has multiple modes and can be used as either an intrusion detection system (IDS) or an intrusion prevention system (IPS). The main difference between these two modes is that IDS only analyses the traffic, generates alerts and logs them, whereas an IPS does all this and additionally it attempts to block the detected malicious packets.

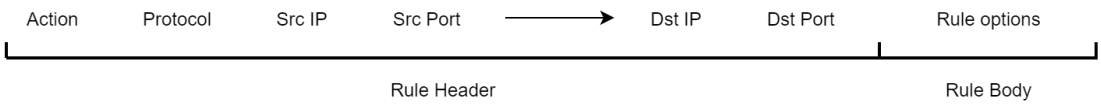


Figure 2.1: Snort Rule

Snort has a separate rule base for each network protocol. The rule-sets in the current snort deployment are classified in 34 different attack classes. Each attack class is supposed to group together events or attacks of a similar type.

Figure 2.2: Snort Architecture

Packet Decoder

Snort relies on an external module used for packet capturing. This module is called libpcap [66]. The packets that are capture are in raw format and need to be

processed. Therefore they are passed on to the packet decoder module. Snort uses raw packet data to also analyze the protocol header information. This module is used to decode specific protocol elements of all network layers and translate these to a custom data structure such that they can be further processed by the preprocessor module.

Preprocessor

The preprocessor module can either be used to preprocess packets before they are passed to the detection engine to be matched with rules or it can be used to detect suspicious activity in the raw headers of the traffic. This is often done by setting thresholds on the values observed in the fields of the header.

Detection Engine

The detection engine takes the preprocessed data and matches each packet against its database of active rules. If a match is found, the IDS acts upon this by, for example, making an alert of the packet and the rule that was a match. Otherwise, no action is taken and the packet is considered benign.

Output Module

The output module is the module that generates alerts or other types of logs about the events detected in the previous module. This module outputs the logs in a readable format which can be parsed and used in various monitoring solutions.

2.1.4. Placement of IDS

The placement of the IDS is typically divided into two categories which are network and host-based placement.

Host Based IDS

A host based IDS (HIDS) is a system that monitors the data generated by internal processes of the host that it resides on. This could be anything from a PC to a Firewall. The IDS can either be anomaly or rule based. Some popular examples of host-based IDS are OSSEC and Wazuh.

Network Based IDS

A network based IDS (NIDS) is a system that monitors the network that it is placed in. It acts as a passive detection engine or an active detection engine with abilities to block traffic when considered malicious. The IDS can either be anomaly or rule based. Popular examples of Network-based IDS are Snort, Suricata and Zeek.

2.2. Data Encoding

The data that is used in this project is an alert set produced by a rule-based IDS. The format of the data consists of a set of only categorical variables. More details about the data set can be found in 5. Many algorithms for data exploration and anomaly detection are directly applicable to numeric data, but this is unfortunately not the case for categorical data. Therefore we consider several encoding methods through which we can get a numeric data set that gives a good representation of the underlying data structure and correlation amongst the different variables.

2.2.1. Label Encoding

Label Encoding is a technique of encoding categorical variables by mapping them to numeric values. Each column consisting of a set of distinct items is mapped to a corresponding number and this operation is performed on all categorical columns in the data set. An example of how this encoding works is the following:

Country	City	Temperature	Precipitation	Wind
Netherlands	Amsterdam	10	20	26
France	Paris	13	60	32
Netherlands	Delft	11	70	23
Germany	Berlin	8	40	30
England	London	10	50	31

Table 2.1: Temperature data example with a mix of categorical and numeric variables.

Country	City	Temperature	Precipitation	Wind
1	1	10	20	26
2	2	13	60	32
1	3	11	70	23
3	4	8	40	30
4	5	10	50	31

Table 2.2: Label Encoding of Temperature data

After using the label encoder the results shown in table 2.2 are obtained. However, based on the numeric values that were assigned to the categories by applying this encoding method it seems as if the categories can be measured. This could lead to a bad performance of machine learning algorithms since a false notion of distance is considered between such categories when in reality these variables have nothing in common.

2.2.2. One-Hot Encoding

An alternative way of encoding categorical variables is by one-hot encoding them. This method creates a vector that has the length of each unique category present in the data. This vector is then filled with ones in the cells if a category appears in the record and zeros in the remaining cells. Using the above mentioned example, the result using the one-hot encoding method is the following:

This type of encoding represents the data better. However, the main issue here is the number of unique values which may be very high resulting in a too large vector to fit into memory and another issue are the new variables that may appear in the data which can not be predicted beforehand.

2.2.3. Frequency Encoding

Frequency encoding is a way of encoding categorical variables such that each category is replaced with a numeric value representing it's relative frequency in the data set. This method applied to the example above results in the following:

Country	City	Temperature	Precipitation	Wind
2/5	1/5	10	20	26
1/5	1/5	13	60	32
2/5	1/5	11	70	23
1/5	1/5	8	40	30
1/5	1/5	10	50	31

Table 2.4: Frequency encoding of Temperature data.

Now each value is encoded with its relative frequency indicating the presence of the specific value in the data. This implicitly captures some behaviour of the sensor, since values with higher frequencies are more likely to be benign.

2.3. Distance Metrics

To be able to measure the difference between the different points in our data set we need to choose an appropriate distance metric. There are several distance metrics used in machine learning which are used to quantify differences between points. Most of these metrics are designed for numeric variables, but there are similarity measures that are also applicable for categorical data and binary vectors. Since the format of the data is only numeric, the euclidean distance is used as a default by the unsupervised methods. Other distance metrics may also be suitable but are not explored because of time limitations.

Euclidean Distance

Euclidean distance is a simple calculation of the distance between two points in a vector space. This distance represents a straight line that connects the two points in space. To calculate this distance the following formula is used:

$$d(\vec{x}, \vec{y}) = ||\boldsymbol{x} - \boldsymbol{y}|| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.1}$$

This formula is used to calculate the distance between to vectors. In this project, we use this formula to calculate the distance of a point to its respective centroid which is part of the clustering phase. The distance is also used to find anomalies, i.e points that lie in low density regions by calculating the distance of each point to its nearest neighbours.

To illustrate how this works in a 2-dimensional space, the distance can be calculated using the Pythagorean Theorem as shown in 2.3. A simple example of calculating euclidean distance in a 3-dimensional space on the vector $\vec{x} = (3, 4, 4)$ and $\vec{y} = (1, -2, 1)$ is the following:

$d(\vec{x}, \vec{y}) = \sqrt{(3-1)^2 + (4-(-2))^2 + (4-1)^2} = \sqrt{2^2 + 6^2 + 3^2} = \sqrt{4 + 36 + 9} = \sqrt{49} = 7$

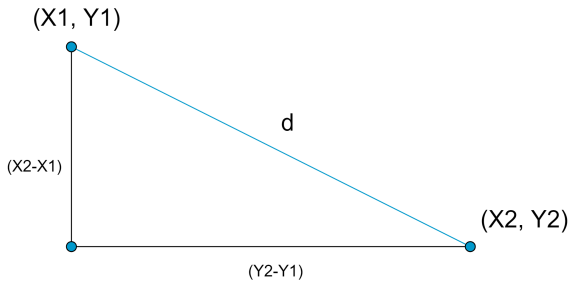


Figure 2.3: Euclidean distance example

2.4. Unsupervised Anomaly Detection Techniques

In many real world scenarios the data that is available to work with is very imbalanced. Meaning that the majority of points in the data are benign. To find the

instances in the data that are anomalous various techniques have been developed. These can generally be split into two categories which are supervised and unsupervised anomaly detection techniques. The supervised techniques are dependent on labels in the data such that these models can learn what is considered anomalous whereas the unsupervised techniques are based on the assumption that the anomalous points will deviate from the majority.

2.4.1. Proximity-based

Density-based techniques calculate the local density of a data point based on the number of neighbouring data points that are located within a specific distance of that point. Based on the values that are obtained for the local densities, an 'outlier' score is assigned [3]. These techniques have an advantage over other anomaly detection techniques because they are able to detect anomalies that are missed by most of the other techniques [81] by considering the local environment of the data point instead of focusing on the global outliers. However, these density-based techniques also have several disadvantages. One of the main disadvantages is that these techniques are very complex and computationally expensive [84]. Another issue is choosing the right value for the measures that are used for these calculations [81].

Local Outlier Factor

The local outlier factor algorithm [12] is based upon the assumption that a normal point lies in a dense region, meaning it is surrounded with many neighbours, whereas an anomalous point lies in a low density region with none, or only a few neighbours.

LOF isolates the points to a certain degree from their neighbour points and focuses more on the local environment of the point instead of the global distribution. This could be very beneficial for finding the local outliers since the data is not anomaly free and if the local environment of the point is not considered, some of the anomalies in the initial data might be counted as inliers instead. LOF is suitable for detecting local outlier in a data set with different density regions since these outliers are considered anomalous only when compared to their direct neighbours. LOF calculates the local density ratio only based on these direct neighbours and does not require any assumption of the global distribution of the data set. The disadvantages of LOF are that it is highly dependent on the predefined outlier measure parameters and that it might become very computationally expensive with a large data set since the distance between each point is calculated. It also is difficult to apply this method to streams since the defined outlier metrics cannot be updated accordingly.
The algorithm computes the k-nearest neighbours for each point in the space and then compares the density of each point with the density of all other points. In

this algorithm the k-distance is used to measure the distance from a point to it's kth neighbour. For example, if k = 5, the k-distance is equal to the distance of a point to it's 5th neighbour. After the k-distance is computed for each point, the algorithm computes the reachability distance. This is calculated with the following formula:

$$reachability(a,b) = max(kdistance(b), distance(a,b)) \tag{2.2}$$

The reachability distance is needed to calculate the local reachability density which is defined as the inverse of the average reachability distances for each point:

$$lrd(a) = \frac{k}{\sum_{n \in points} reachability(a,n)} \tag{2.3}$$

To find anomalies another score is calculated that makes use of the local reachability density score. This is called the local outlier factor and is calculated with the following formula:

$$lof(a) = \frac{\sum_{n} \in points \frac{lrd(n)}{lrd(a)}}{k} \tag{2.4}$$

Defined as the average of the local reachability densities of the points, this score is used to decide whether a point is anomalous or not. Points with a score greater than 1 (LOF > 1) are considered anomalies, whereas all other points whose score is equal or smaller than 1 (LOF <= 1) are considered normal.

$p_1 = (1,2)$, $p_2 = (3,6)$, $p_3 = (3,7)$, $p_4 = (15,10)$

The distance for each pair of points is calculated:

$d(p1,p2) = 4.472$, $d(p1,p3) = 5.385$, $d(p1,p4) = 16.12$, $d(p2,p3) = 1$,
$d(p2,p4) = 12.65$, $d(p3,p4) = 12.37$

In the case where k = 2 the algorithm finds the 2_{nd}-nearest neighbour for each point based on the calculated distance:

$knn(p_1) = p_3$, $knn(p_2) = p_1$, $knn(p_3) = p_1$, $knn(p_4) = p_2$

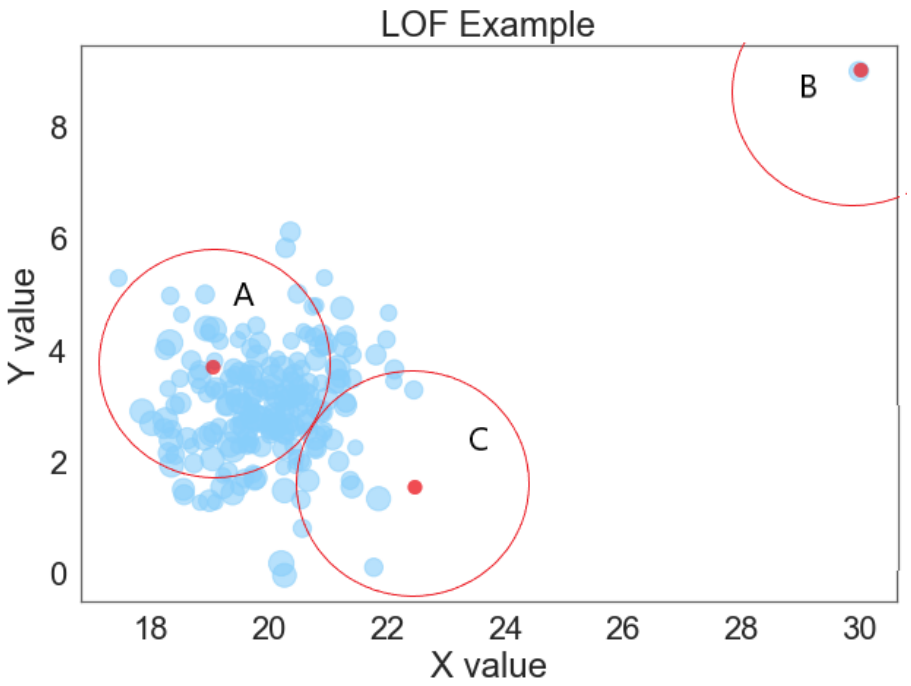


Figure 2.4: Point A represents a normal point surrounded by many neighbours whereas point B represents an anomalous, deviating point located in a low density region. Point B is also called a global outlier. Point C is a local outlier.

The sets of 2-nearest neighbours for each point are the following:

$$neighbours(p1) = p2, p3$$
$$neighbours(p2) = p1, p3$$
$$neighbours(p3) = p1, p2$$
$$neighbours(p4) = p2, p3$$

The reachability distance for each of its two neighbours is then the distance used to calculate the local reachability distance. This is calculated for p1 as follows:

$$reachability - distance(p1, p2) = max(4.472, 5.385) = 5.385$$
$$reachability - distance(p1, p3) = max(5.385, 5.385) = 5.385$$

$$lrd(p1) = \frac{2}{(5.385+5.385)} = \frac{2}{10.77} = 0.18$$

the reachability distances for the other points are the following:

$$lrd(p2) = \frac{2}{5.472} = 0.36 \text{ , } lrd(p3) = \frac{2}{10.77} = 0.18, \; lrd(p4) = \frac{2}{25.3} = 0.07$$

Finally, the local outlier factor for each point is the following:

$$lof(p1) = \frac{0.66+0.66}{2*0.33} = 1.5$$

$$lof(p2) = \frac{0.33+0.66}{2*0.66} = 0.5$$

$$lof(p3) = \frac{0.18+0.36}{2*0.18} = 1.5$$

$$lof(p4) = \frac{0.36+0.18}{2*0.07} = 3.85$$

The points p2 has a LOF < 1 and is considered normal within this set of points. The points p1 and p3 have a slightly higher score than 1 and are considered anomalous. However, point p4 has the highest score, namely 3.85. This point differs a lot from the other three points. In this case, it makes sense to consider only point p4 as a true outlier.

2.4.2. Statistical-based

Statistical-based techniques use the distribution of a model for the detection of anomalies. The statistical-based techniques can be classified into parametric techniques which have a certain assumption about the distribution of the model and define the parameters based on this distribution and non-parametric techniques which do not know the distribution of the model [81]. The parametric techniques use the Gaussian and Regression Model for the detection of anomalies in a data set while the non-parametric techniques mainly use Kernel Density Estimation (KDE) techniques for this purpose. The advantages of using statistical-based techniques is that these techniques are relatively easy to design and they are very fast compared to other anomaly detection techniques such as LOF and clustering. Additionally, the anomaly scores are paired with probability scores which be used as an additional validation metric [84]. Specifically for Histogram-based methods the main disadvantage is that they are unable to capture relationships between the different dimension in the data and rely on the assumption that the data has an underlying distribution [81].

2.5. Histogram-based Outlier Score

The Histogram Based Outlier Score (HBOS) is a statistical based anomaly detection technique [24]. This technique has a linear time complexity $\mathcal{O}(n)$ for a static fixed bin width and a $\mathcal{O}(n * logn)$ complexity for a dynamic bin width. This techniques models the feature densities using histograms with a bin width that can be both static or dynamic dependent on the scenario that it is used for. The created models are subsequently used for the calculation of an anomaly score for each data point. The HBOS algorithm constructs an uni-variate histogram for every feature. This algorithm can work with both categorical and numerical features. If the features are categorical, a frequency is calculated based on the

counts of each feature category and if the features are numerical static or dynamic bin width histograms are used. For the static bin width histogram, k equal bins are used where the density is determined based on the frequency of points that belong to one bin. The dynamic bin widths are determined by sequential values with a fixed amount of $\frac{N}{k}$ that are placed into one bin. The number of bins, k, is often chosen as the square root of the total number of data points N, so $k = \sqrt{N}$. The HBOS for every data point p can be calculated with the following formula:

$$HBOS(p) = \sum_{i=0}^{d} \log(\frac{1}{hist_i(p)}) \tag{2.5}$$

where k represents the number of bins, N represents the number of data points and d represents the dimension. All histograms are normalised in a way so that a maximum value of 1 for the height of each histogram is obtained.

The advantage of HBOS is that it does not require labelling of the data, is effective in detecting global outliers and is very fast and efficient which makes it easier to apply this technique to larger data sets [23, 24, 49, 87]. Also since this technique assigns outlier scores it is easier to estimate how reliable the predictions are. However, this technique does not perform well for the detection of local outliers [24] and it is difficult to get context from the obtained results [87].

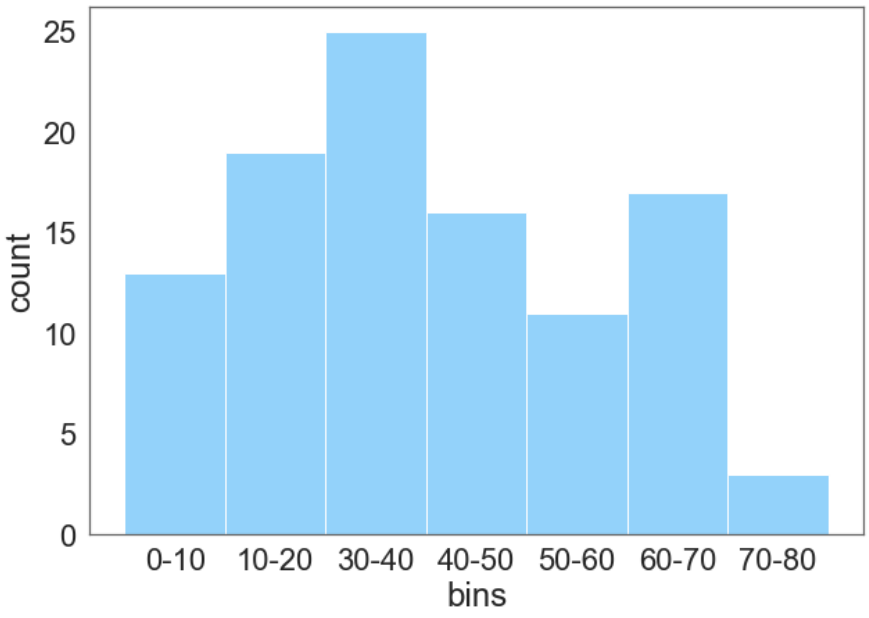


Figure 2.5: Example of Histogram Based Outlier Score

An example of how an anomalous instance would be found by HBOS is the following: The first bin in the range [10, 20] has a height of 13. The score for each point in this bin is thus computed as log(1/13) = -1.1139. The score for the

histogram in range [70-80] would be equal to log(1/2) = -0.30102. This score is closer to 0 and thus these points are considered more anomalous than the points in bin [10-20]. For each attribute included in the training phase a histogram is build and the final score for a point is computed as the sum of each attribute score. Higher scores represent more abnormal points that would be assigned to bins with a low density.

2.5.1. Clustering-based

Clustering-based anomaly detection techniques divide the data points into clusters of a certain size where two data points in the same clusters are more similar to each other while two data points of different clusters have significantly different characteristics [84]. The points that have the largest distance from the clusters or do not belong to any cluster are considered anomalies. Clustering-based techniques can be classified in several categories, namely, Density-based, Partitioning, Hierarchical and Grid-based techniques [81]. The advantages of clustering-based techniques is that they do not require apriori knowledge in order to be applied to detect anomalies in stream data since new points are assigned to the pre-computed clusters. These techniques are also easily scalable and adaptable to different types of data. The disadvantage of these techniques is that the they are sensitive to outliers and that the cluster characteristics such as the number of clusters and distance metric should be pre-defined. These parameters highly dependent on the type of data that is being clustered. Another disadvantage is that these techniques can be very costly and time consuming when applied to large and very complex data sets [81].

Cluster-based Local Outlier Factor

The Cluster Based Local Outlier Factor (CBLOF) [28] is a technique that assigns an outlier factor to each data point which is determined by the distance of the data point to the nearest neighbouring clusters and the size of its own cluster. In order to assign these factors, the data points are first divided into clusters. A similarity function is computed for every tuple and a data point is added to the cluster with the highest similarity value. The complexity of this techniques is $\mathcal{O}(n)$. The cluster-based outlier factor for a point t can be calculated with the following formula:

$$CBLOF(t) = \begin{cases} |C_i^*| \text{ min(distance } (t, C_j)), \text{ where } \text{ t} \in C_i,\ C_i \in \text{SC } \text{ and } C_j \in LC \ for\ j = 1\ to\ b \\ |C_i^*| \text{ min(distance } (t, C_j)), \text{ where } \text{ t} \in C_i\ and\ C_i \in \text{LC} \end{cases} \tag{2.6}$$

where C_i and C_j represent clusters, SC = $C_i, \mid j > b$ represents set with small clusters and LC = $C_i, \mid j \geq b$ represent the set with large clusters and b is the

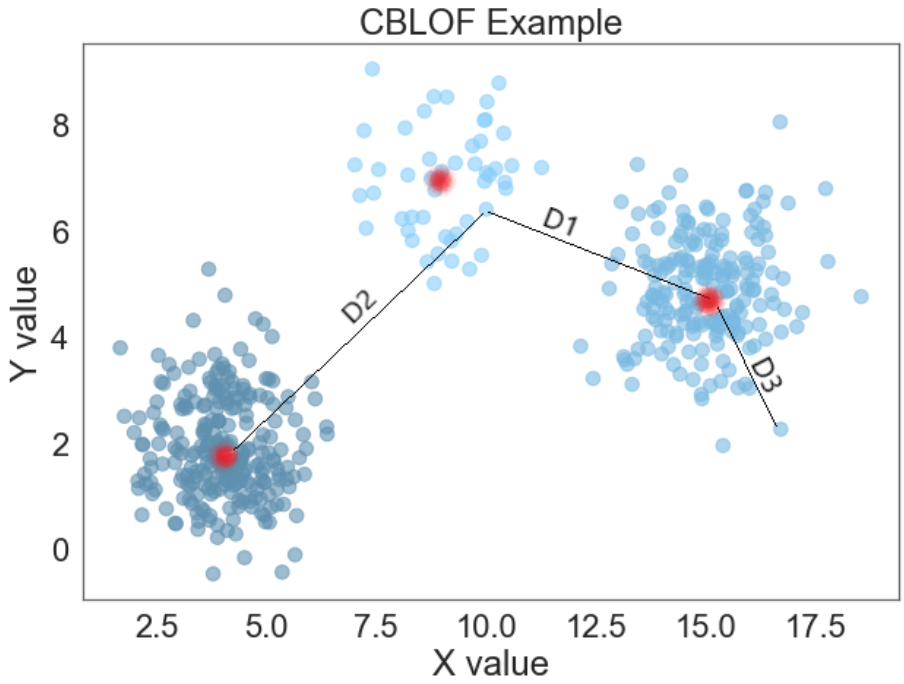


Figure 2.6: Example of CBLOF Score Calculation

boundary of these clusters.
The advantages of using CBLOF for outlier detection are that this technique has a linear complexity which makes it perform fast computations and is easily scalable for large data sets [28].

Because point A is clustered in a small cluster, the score for this point is computed as $score_A = \min(D1, D2)$. Where D1 and D2 are the respective distances from the point to the cluster centres of the two large clusters. The score for point B, which is part of a large cluster, is computed as the score from the point to its own cluster's centre. So the score for point B is $Score_B = D3$.

2.5.2. Ensemble-based

Ensemble-based anomaly detection techniques combine several different algorithms into one and use the advantages of all techniques to increase the accuracy and performance [81]. The ensemble-based techniques can be categorised into sequential ensemble techniques, independent ensemble-techniques, data-centred ensemble techniques and model-centred ensemble techniques [2, 9]. The main advantage of these methods is their robustness to noise in the data which makes them suitable for analysing dynamic and changing data. These models are also suitable for high-dimensional data because of their efficiency in computing outlier scores. The disadvantages of these methods are the difficulty of selecting suitable hyper-parameters and interpreting and evaluating the obtained anomaly scores

is also challenging [71, 81].

Isolation Forest

Isolation Forest [39] is an anomaly detection algorithm designed to detect outliers in high dimensional data. This algorithm does not make assumptions about the distribution of the data and does not require the data to be normally distributed. Additionally, it does not require any labels to detect anomalies and therefore, it is a suitable choice for this purpose.

The Isolation Forest algorithm is very suitable and effective for detecting anomalies in high dimensional data with lots of irrelevant features and noise which has a small number of anomalies or no anomalies at all, since it the majority of these anomalies will be very close to the root of the tree. This algorithm also performs well on normal unbiased data which does not have a lot of noise, is non-parametric which makes it suitable for unsupervised anomaly detection [3] and it is computationally efficient. With using Isolation Forest algorithm even the few true positives that are present will be found. However, this technique is effective in detecting local outliers [13].

The algorithm partitions the domain space. These partitions are created totally at random by selecting a feature and then creating a split value between the minimum and maximum value of the feature The algorithm keeps creating partitions until all data points are isolated. This is done by creating random decision trees. The score for each point in then calculated based on the length of the path needed to isolate that point as shown in 2.7.

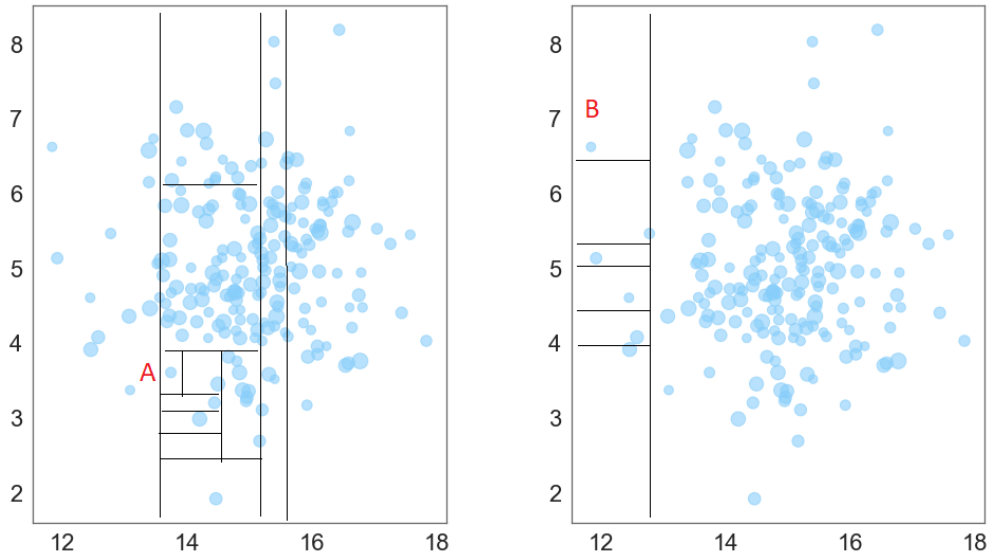


Figure 2.7: Example of Isolating a normal point (A) and isolating an anomalous point (B)

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{2.7}$$

The anomaly score for one point is calculated by taking the average anomaly scores across all trees for that point. A score closer to 1 indicates that this is an anomaly. A score much smaller than 0.5 indicates that this is a normal observation. If all scores are around 0.5 then the data doesn't contain any obvious anomalies. An anomalous point would have a short path to travel to its root.

The decision tree is constructed such that first one feature is chosen and is split based on the min and max value observed. In these splits the next feature is split in the same manner until all features are passed. For anomalous instances, these paths will be close to the minimal length since they would have unique and few splits. For normal instances these splits would be larger than the minimum path length because the splits. Isolation Forest might work better with one-hot-encoding because the uniqueness of observed values within a feature is preserved, whereas with frequency encoding the feature is seen as a continuous variable and the uniqueness of some observed instances is lost.

3

Literature Review

In this section the reader will find an extensive literature review on the problem of false alarm reduction in IDS. There are multiple categories in which all the works can be divided. Additionally, there are many different techniques used by authors including a broad range of machine learning techniques for the purpose of clustering and classification of alerts or to find the root cause, sketch a baseline of normal behaviour and correlate data from multiple sources to verify the nature of the generated alerts. In this report, selected works will be presented that sketch how intrusion detection systems have been developed and improved over time.

The different techniques can be categorised in multiple ways, starting from misuse-based systems and anomaly-based systems. The former detect attacks by means of analysing the captured packets and comparing them to a set of predefined rules. The latter detect attacks by learning a model of normal behaviour for the monitored system, and reporting on any event that fall out this normal baseline behaviour. Unfortunately both types of IDS suffer from high amounts of false alarms due to various reasons such as the dynamically changing environment and zero-day attacks. Due to these problems researchers have explored the improvement of IDS extensively and have developed different techniques that could reduce the high ratio of false alarms and make these systems more accurate and effective.

One general categorisation of the proposed improvements would be techniques based on external knowledge sources and data mining techniques that infer the needed knowledge from the alerts or network traffic without consulting external information sources. This categorisation is however very general, since many works combine techniques from both categories. Another possible categorisation would be based on the place where the IDS is altered. This can be before deployment by updating the rule set or configuration of the IDS or after deployment, filtering the alerts that are already generated by the IDS. Although there are

again hybrid systems, the majority of works perform operations only at one place.

The general state of the research conducted for the problem of false alarms in IDS is that the majority of works focus on the alert log and try to filter out the irrelevant alerts based on correlation and classification techniques. Recently, more advanced and hybrid techniques were studied and developed. There is a smaller subset of works that put focus on solving the problem before the IDS is deployed, to have the IDS only generate the most relevant alerts.

3.1. Background

One of the first studies that explored and introduced intrusion detection systems was [6]. This work is the starting point of active exploration and development of the more advanced intrusion detection systems that we know today. From this work we can already acknowledge the problems that the cyber security community is facing with intrusion detection systems. And one very significant problem is the tremendous false alarm rate. This problem is also addressed in [7]. In this study statistical analysis is performed to the false alarm problem and the phenomenon that is explored is called 'base-rate fallacy'. The authors explain the problem of base-rate fallacy as the probability that even if all true positives are identified correctly, there will be a significant amount of false positives left to investigate. They conclude that the factor limiting the performance of an IDS is to suppress alarms, rather then the ability to truly detect intrusions. This is a very important finding as it sets the base for deployment of intrusion detection systems by laying the focus on the minimisation of false alarm rates in order to have an effective IDS.

[70] evaluate Snort's performance on the DARPA data set and on real network traffic collected from a private network in 40 days which they refer to as their private data set. The alert log is manually analysed with the help of a specialist who labelled the alerts as either true or false positives and the BASE tool. The authors found that the initial scale of false alarms generated by SNORT is huge. They conclude that most of the alerts (96%) generated are either false alarms or irrelevant positives (irrelevant to the network where the traffic is collected and analysed). They selected several rules that generated the most of the false alarms and modified these rules using techniques such as rule modification, event suppression and event threshold. They conclude that these techniques can significantly decrease the false alarm rate, however, there is a chance of missing or ignoring significant alerts. Their conclusion is that this manual process of adjusting the NIDS rules is not efficient and there is a need for an automated alert verification system that is not reliable on human participation. They also state that an intelligent system is needed which finds a relation between the generated alerts and a possible attack scenario.

3.2. Knowledge-based Techniques

In this section several works related to knowledge based techniques will be reviewed. The knowledge based techniques can be divided into manual labelling and labelling based on correlation. For each of these categories relevant works will be explored.

3.2.1. Manual Labelling

A series of relevant works which focus on the reduction of false alarms are [51–53]. The author explores the development of an adaptive learning system for alert classification named ALAC. The author uses a data set of alerts which are all manually labelled by experienced security experts as true or false positives. This information is then provided to a classifier which trains on the historical labelled data and predicts new records as either true or false. The implemented system works in two modes, either as a recommender system which analysts use to manually filter the alerts or an automatic alert filtering system based on the predicted labels. Furthermore, the author extends the ALAC system with an additional module that is an adjustment of the alert clustering system ClaraTy [34].

Another take on the problem was presented by [34] who investigates how to reduce the number of false positives by analysing the root causes of alerts. The author proposes a system that analyses a set of alerts to identify their root cause and concludes that only a few root causes account for over 90% of the total amount of alerts. The author also observes that these root causes are repeating and thus the ultimate solution for reducing the number of alarms is totally removing the root causes in the case these are benign. Therefore, the author introduces a two-step paradigm where the first step consists of identifying the root causes and classifying them as either benign or malicious, and the second step consist of removing the benign root causes.

[63] describe a two-phase alert classification system with the purpose to give insightful information and help analysts to identify possible false positives. The first phase of their method consists of collecting and grouping alerts into ”meta-alerts”. These ”meta-alerts” are then verified with a database holding information about the assets of the monitored environment. Furthermore, the authors also perform alert generalisation through root cause analysis [34] in order to further reduce the generated alerts. The second phase of the system consists of manually labelling the alerts and sending them to a classifier to learn a model. This classifier is then used on newly generated alerts and labels them as either true or false.

[30] highlight that one of the main reason for a high rate of false alarms in IDSs is that most of them are deployed with their default set of rules and configuration. the authors aim to overcome this problem by creating a threat profile of the network and correlating the produced alarms by means of neural networks. An important observation is the fact that there is always only a limited set of vulnerabilities that can be exploited in a network. To tackle this, the authors make an 'enhanced entity relationship' (EER) where the network context and threat profile are presented including the relationship between all the elements. For constructing a threat profile, multiple vulnerability scanners are used and processed. The network context consists of all devices that are present on the network. The final solution takes the information produces by the EER and the set of alarms generated by the IDS and classifies these with neural network to produce a final set of 'effective' alarms.

3.2.2. Correlation-based Labelling

[36, 75] proposed a method for the reduction of false alarms using active alert verification. The authors emphasise the problem that the cyber security community is facing with standardisation in terms of known vulnerabilities and the absence of a common alert format that would be used by different intrusion detection systems. They also emphasise the problem of firewalls and intrusion detection systems being run with in their default setting, without adjusting them to the environment they are placed in. The authors propose an active alert verification method that is integrated as a Snort module and makes use of the Nessus[67] vulnerability scanner to provide the IDS with contextual information about the monitored network. When an alert is generated by the Snort IDS, the Nessus vulnerability scanner is activated and scans the affected host. If the host is found vulnerable the generated alert is kept as a possible true positive, otherwise the alert is labelled as false and not considered in the further analysis. The authors have tested this setup on one Linux and one Windows host and conclude that there is indeed a significant reduction in the number of false positives. The limitation here is that the evaluation is based only upon two hosts with synthetically generated attacks and it remains a question whether this system would work in a real environment. Furthermore, the authors don't describe how the Nessus output is correlated with the Snort alerts.

[10] introduce APHRODITE, an anomaly based architecture for false positive reduction. The authors hypothesise that the main reason for the high amount of false positives is the lack of correlation between input and output traffic. This architecture is meant to work with both misuse-based and anomaly-based intrusion detection systems. The main idea of this system is to apply machine learning on the output traffic of the monitored network to detect anomalies, and correlate these with the alerts generated by the IDS on the input traffic in the network.

The system is evaluated with SNORT[42] (misuse-based) and POSEIDON[11] (anomaly-based) NIDS. They claim to reach a reduction of false positives between 50% and 100%. The main drawback that this technique has is that is needs to be trained on a good data set set full of representative samples such that it classifies the alerts correctly.

[61] perform a sequential analysis on the patterns in order to extract the relevant information from the alert sequences and use a multi-layer decision tree for the identification of the relevant patterns. The alerts are ordered in sequences based on their timestamp. The sequences are first analyses by experts to determine their nature. A window of a predefined size is chosen to divide the ordered alerts in episodes and these episodes are rated. The episodes that have the highest rating are considered as most critical and have to be evaluated.

In [46] the authors propose a two-step model for the effective reduction of false alarms by taking into account the network context. This model consists of firstly verifying the alerts from the IDS with the vulnerability assessment data (EVA) obtained from the network. The second component filters the alerts that are deemed unnecessary i.e. the network is not found vulnerable to those. Again, there is no specification about how the EVA data is obtained and correlated with the generated alerts and how effective this approach is.

[31] develop an algorithm for attack recognition and correlation. They use vulnerability scans, network topology and intrusion detection logs to correlate and cluster the identified alerts into attack scenarios, where each new incoming alert either belongs to some of the identified attack paths (clusters) or is saved as a new cluster. The authors use the CVSS score provided for each CVE_ID. This is a huge limitation since many of the generated alerts and many of the discovered vulnerabilities don't have a CVE_ID and thus this makes it impossible to compute any probability for these events. This paper also focuses on the detection of intentional attacks by correlating multiple steps (alerts).

[62] develop a game-theory based engine that correlates the present vulnerabilities in the network with the IDS alarms that are generated. This system uses firstly, multiple vulnerability scanners to scan the network and create a threat profile consisting of multiple vulnerability sets that model one or more vulnerabilities found. The authors then assign to each vulnerability set a weight that models the severity of the included vulnerabilities. the authors apply a game-theory approach to construct a 'sensible vulnerability set' (SVS) that consists of vulnerability subsets with high weights. Firstly, the IDS alarms are correlated to the set of identified vulnerabilities in order to find and filter only the potential True Positives (TP). After this step, the TPs are correlated with the constructed SVS to further reduce the number of potential TP. The authors evaluate this

system of the DARPA data set and on a private data set.

In [80] The authors propose a system that correlates NIDS and HIDS alerts to generate a more specified and prioritised list of alarms by using machine learning algorithms. The system is automated and reduces the false alarm rate successfully. They identify the main limitations of current IDSs as the high rate of false alarms, separate HIDS and NIDS with no correlation between them, rule-based machine learning techniques, a lack of integrated vulnerability management and the high costs of implementation, especially for small/medium-sized organisations. The authors explain their approach in detail and claim that the false negative rate is successfully reduced, however this approach is only tested on the DARPA 1998 data set. It is not validated and tested in a real network environment.

[40] also propose a novel algorithm for the efficient and effective correlation of alerts. The authors take the alert log as input to the algorithm and cluster the alarms based on similarity. Before this step, the algorithm tries to capture the characteristics that distinguish true alerts and false positives. The last step of the algorithm is linking alerts for a multi-step attack scenario and finding the root cause. Furthermore, the authors propose a method for constructing the attack path from an alert up to its root cause based on an attack association method. Experiments with these algorithms result in a simpler alert set and a reduction of false positives.

[56] find rule-based intrusion detection systems having a high false positive rate. To combat this problem they have developed a machine learning based plug-in for Snort, to analyse the produced alerts. They used an optimised SVM with the firefly algorithm as the machine learning method to classify the results achieving a false alarm rate of 8.6% and a false negative rate of 2.2%.

3.3. Data Mining Techniques

The idea of classifying alerts to learn normal behaviour is explored in [72] where a method for reducing the number of false alarms in IDS is proposed. This is done by training a machine learning algorithm on the alerts generated by the IDS to distinguish between normal behaviour of the sensor and only alert on deviations from this. If the sensor generated alerts for one type of traffic constantly, this will be considered normal behaviour and these alerts will be labelled and filtered as false alarms. The authors conclude that deviations from the 'normal' behaviour of a sensor are very good indicators of actual attacks.

The limitation to this model is that it is solely based on the rule and no other variables are included in determining the normal behaviour. This model is based

on the assumption that the sequences of rules generated by a single sensor are frequent, meaning alerts are generated in the exact same order as learned by the model. The proposed model does not consider new rules or changes in the environment and is not effective in the detection of sudden significant changes or sudden bursts.

[79] use an auto-regressive model to assess regularities between the alters and the normal behaviour of the system. The auto-regressive Kalman filters are used for the re-estimation of the weights while previous observations are used to create a model for the normal behaviour based on the weighted sum of these observations. Anomalies are considered the differences between the observations and the predictions of the model. Because the last n observations are used, the model is somewhat able to adapt to changes in the environment. However, the limitation to this model is again that the prediction is based on data received in the last 20 minutes and this model is also computationally intensive. The model is based upon the frequencies of observed rules meaning that each signal is unique to one rule so multiple models are needed to monitor this behaviour. Additionally, the parameters are selected differently for each signal/model which makes it difficult to apply large it scale in an automated manner.

[54] have presented an approach for classifying alters based on frequent pattern analysis. They use the source and destination hosts as main indicators of regular traffic and construct patterns based on the connectivity between the selected variables. They introduce the concept of stable patterns which represent patters that do not change frequently and update the pattern tree incrementally in a 15-minute interval. The downside of this approach is that it is time sensitive meaning that the frequent pattern updates may cause inconsistency in the model and the time needed to construct a stable pattern tree takes about two weeks if there are no major changes. Additionally, the goal of this work is to reveal behavioural patterns of malicious behaviour within the network thus it is assumed that all alerts that are generated by the NIDS are correct.

In [73] the authors also proposes an alert classifier based on frequent itemset analysis. Frequent episodes are mined from slices of alerts. Once this is done the classifier is used real-time by classifying each alert that is generated as either important alerts or irrelevant, frequently occurring alerts. An alert is modelled as a tuple consisting of the timestamp, source, destination and alert id. The main drawback of this approach is that it only filters alerts that occur very frequently over a longer period of time. Also, in order to avoid the issues of over-generalisation and miss-classifying unusual and intensive short-term malicious network activity, the authors implement additional steps which increase the computational costs.

3.4. Discussion

What we can infer from the literature review is that there are many proposed solutions for tackling the problem of high false alarm rates. The first observation is that it is not specified how these techniques can be actually applied in real networks [38, 80]. Additionally, even when detailed information is provided, this is often complicated and cannot be simply integrated with an exiting NIDS because additional resources are required such as databases to lookup external knowledge [62, 73]. These alarm filters are assumed to work on NIDSs as part of bigger monitoring solutions (SIEM) that collect data from many different sources. Specifically about the inclusion of environmental context in the NIDS, there are several useful techniques developed [25, 36, 47] but not sufficiently described and evaluated in terms of effectiveness in real networks. The majority of techniques is only evaluated on synthetically generated data sets [38, 70, 80] and not in a setting with real network data. The techniques that rely on external knowledge are often limited by too generic rules and variables which makes the correlation of the different environment variables not straightforward [36, 45, 75]. Since this is the basis of many proposed systems where supervised or semi-supervised techniques are used that rely on the quality of the labelled data, the effectiveness of the model is directly affected by these limitations [80]. Of the proposed techniques that do not rely on external knowledge, sequential pattern mining techniques are applied such as Hidden Markov Models. These techniques rely on the correct choice of a window size to capture truly regular and repeating patterns. The biggest limitation with these models is that they are based upon the assumption that certain values will often appear in an initially observed order and small changes would change the regular sequence and decrease the accuracy of these models. Furthermore, in large networks this method can be computationally expensive and large models are more expensive but may be required to better fit the data. Association Rule Learning is also applied to this problem, however, this approach is focused on the most frequently occurring alerts but this is not sufficient to model all false positives since there are multiple groups of 'regular' alerts with different frequencies. Additionally, the computational complexity and memory requirements for this approach can also be considered a limitation, especially if the model needs to be retrained often and enriched with more relations. Often, large databases of rules are constructed which can grow exponentially and are expensive to search through [72, 73]. A common limitation in all the proposed works that is not considered is the frequency with which the gathered information about the environment should be updated and the quality of the data. None of the reviewed studies consider how robust their models are to noise nor the frequency of re-training and updating of the proposed models to correspond to the dynamic changes of the environment and/or how this would affect their applicability in production networks.

3.5. Research Gap

The state-of-the art studies have experimented with and developed several machine learning approaches to reduce the false positive rate. However, these works rely either on labelling based on external knowledge sources or on unsupervised approaches that are computationally expensive and require parameter tuning specific to the environment. Furthermore, the effectiveness of existing solutions is often based on synthetic data and scenarios or on private data sets that are not described in more detail. Most solutions are not evaluated in large dynamic networks in terms of their applicability and integration with existing NIDS deployments. The automation of false alarm minimisation in rule-based NIDS needs to be independent of the quality of external sources and needs to be robust to changes in the environment. To make the false alarm filtering process automated, the need for hyper-parameter tuning needs to be minimised or eliminated. The possibility to filter false alarms in an automated and adaptable way needs to be further explored as to evaluate to what extend it can be made applicable for corporate networks. Since unsupervised outlier detection techniques have been successfully used in various big data analysis problems such as credit card fraud detection or detection of deviating software logs, this technique may also be effective in detecting true alarms in intrusion detection systems as these are only a minority of the total event set. The research gaps are summarised in the list below:

1. The applicability of knowledge-based techniques and the extend to which it is possible to reduce false alarms based on these techniques in real networks is unsufficiently covered.
2. The unsupervised approaches that operate solely on the alert data are still dependent on tweaking parameters, thus still require manual adjustments.
3. There is no analysis or evaluation of the proposed techniques in terms of handling new data, retraining the model on new data and how this will affect the applicability in a production environment.

This thesis addresses the above mentioned gaps. Firstly, novel algorithms are applied to this problem with a lower computational complexity. This is an important factor to take into account in the training and retraining phase. Also for the possibility to constantly update the models in an online setting, even though this is out of scope for this work. Second, the issue of manually tweaking of the parameters is addressed such that the solution can operate in an automated manner. The need of manual adjustments to the parameters is eliminated by choosing to use the default parameters of the algorithms. The only parameter that needs to be chosen is the contamination factor for which recommended ranges are given resulting from the experiments. Lastly the performance of the used

algorithms is evaluated and an indication of when the method should be retrained is given. Since the model retrains itself only when necessary it reduces the overall computational cost and there is no overhead during the real time alert generation. Furthermore, since the chosen models are linearly scalable to the analysed data, these could be also successfully applied in an online manner.

3.6. Conclusion

In this chapter the most important works done on the improvement of NIDS accuracy and effectiveness were reviewed. The techniques that have been researched and developed largely include some sort of machine learning element. Machine learning has been used for pre-processing, post-processing, real-time clustering and correlating of alerts with each other, as well as with other logs and knowledge sources. The conclusion that can be drawn from this review is that there exist techniques that could potentially improve the NIDS effectiveness and accuracy. However, most of the techniques are developed and tested on artificial data sets such as the DARPA data set which is proven not representative enough or on private datasets which are not described, thus the general application of the proposed approaches is questionable. Furthermore, since many models rely on a predefined baseline behaviour or network settings obtained from external sources they are sensitive to changes in the environment. Therefore the adaptability of the models needs to be maintained throughout the whole lifetime in which the NIDS is active. To be able to do this there is a need for an approach that allows for automatic adjustments.

4

Methodology

The goal of this research is to develop a solution for the reduction of false alarms in network intrusion detection systems which works effectively for various networks. To accomplish this a proof of concept solution is designed which learns the behaviour of a sensor and outputs anomalous alerts as true positives. This is divided in three main stages where the first stage is analysing the data and defining the most important features that are descriptive of the sensor behaviour. The second stage is the implementation of the anomaly detection pipeline and finally the last stage is the evaluation of the implemented prototype. In this section the different steps that are taken are described. These include the implementation of the model with all its components, the selected features and selected unsupervised anomaly detection techniques as well as the metrics that are used to evaluate the performance.

4.1. Proposed Solution

To answer whether unsupervised techniques can be effectively used for filtering false alarms from intrusion detection systems a model is designed which is capable of recognizing outliers in the alert set and outputs anomalous alerts as true positives. To develop this model several machine learning techniques are used which filter out all irrelevant or low priority events that are generated by the NIDS resulting in a small set of events for further manual investigation.

The NIDS alerts component represents the alert data obtained directly from the intrusion detection system that is in place. This data is further used by the system to filter out the false alarms. The second component represents the pre-processing part of the model where the alert data is read and parsed. The feature extraction model selects and encodes the features for each alert which are then passed on to the selected algorithms. In the training phase the algorithms are trained on a pre-selected amount of historical data. The trained models are

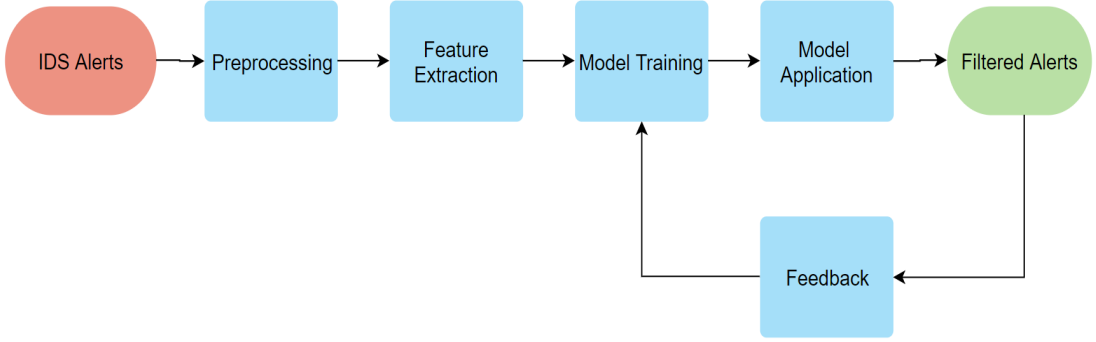


Figure 4.1: Prototype

then used on new alert data to filter false alarms. The prototype has a feedback loop that is used to trigger the model to retrain its algorithms on the most recent data available. The trigger for retraining is based on the ratio between the input and output size of the alert sets. When the output set is larger than a predefined threshold computed as a fraction of the input size of alert data, the model needs to be retrained.

All the components are implemented in Python. For the chosen unsupervised algorithms the implementations from the Sklearn and PYOD libraries are used [14, 85]. The trained models are then saved and used for the classification of new alert data. The new data follows the same pre-processing steps with the difference that the encoding is done based on the pre-calculated frequencies as derived from the training data. For values that are not encountered before, a default value of 0 is assigned which means that these instances get the lowest frequency value.

4.2. Feature Selection

In this section the selection of features is explained. The selection of features to consider for the detection of malicious behaviour can be done by using several methods such as filter based methods which include relevance and redundancy analysis, wrapper based methods that try all possible combinations of features and embedded methods that have already built-in algorithms for selecting the most suitable set of features [65, 83, 86]. Since the attacks in the data are known beforehand and labelled for evaluation purposes we can leverage this information to select a subset of features that will minimise the complexity and computation power of the model while maximising its performance in terms of correct classification. This is done with methods that compute the dependence between each feature and the target variable. The metrics that are used are the Pearson's Chi-Square Test and the Mutual Information Score[82].

4.2.1. Pearson's Chi-Square Test

Pearson's Chi-Square Test is a metric that measures whether the difference between two variables is by chance or not. The degree of freedom used for this test is $df = (r—1) * (c—1)$ where r is the number of rows and c is the number of columns that the data set contains and the p-value indicates the significance level of the obtained results from the test. The Pearson Chi-Square test is calculated with the following formula:

$$X^2 = \frac{(observedfrequency - expectedfrequency)^2}{expectedfrequency} \tag{4.1}$$

4.2.2. Mutual Information Score

Mutual Information measures the amount of information that can be obtained about one variable by observing another variable. It uses the joint probability density function of the two columns and determines how similar the joint distribution is to the marginal distribution of each column respectively. A higher score means a higher degree of relatedness between two columns. The formula is defined as:

$$I(X;Y) = \int_X \int_Y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy \tag{4.2}$$

4.3. Selected Techniques

Anomaly detection is a method of identifying instances in the data that do not conform with the observed general or normal behaviour. There are various techniques developed for this purpose. The majority of them are based on numeric data for which anomalies are found by observing statistics of specific groups. These techniques are categorised in statistical-based techniques, distance-based techniques, density-based techniques, clustering-based techniques, graph-based techniques, tree-based techniques, ensemble-based techniques and learning-based techniques [81]. In the cyber security domain unsupervised machine learning techniques begin to be used more frequently [22]. The main advantage of these techniques is that they do not require large amounts of labelled data as opposed to supervised machine learning techniques. Also, these techniques are able to classify novel instances in the data which have not been observed before. This is important in network intrusion detection since the monitored environments are dynamic.

Multiple different techniques are chosen for filtering false alarms since each of these techniques is designed with different underlying assumptions. This can give more insights into the characteristics of true attacks and the scores of the different techniques can be combined for improved results. Additionally, it is difficult to decide upfront which algorithm would perform better based on the data. The techniques are chosen based on previous research where it has been shown that these techniques have a very good performance for outlier detection problems [3, 16, 33, 37, 48]. The techniques that are selected belong to the density-based, statistical-based, clustering-based and ensemble-based techniques. The density-based techniques are suitable because these techniques do not have to make assumptions about the underlying distribution of the data set and are non-parametric [81, 84]. Specifically from the density-based techniques, the Local Outlier Factor (LOF) [12] is chosen. From the ensemble-based techniques the Isolation Forest (IF) [39] is chosen since it is applied in several cyber security problems with a successful outcome [35, 64]. This algorithm is suitable for analysing large data sets since it is fast and can be implemented in a distributed setting [58]. From the clustering-based techniques Cluster-based Local Outlier Factor (CBLOF) [28] is chosen. This technique is selected because it is know to be very computationally efficient and accurate. Is has also been applied to network security problems [4, 5]. From the statistical-based techniques the Histogram-based Outlier Score (HBOS) [24] is chosen. This technique is shown to be effective in finding global outliers in the data and is additionally very fast and efficiently applicable to large data sets. It has also been applied to cyber security problems [43, 76] with promising results. The selected techniques are also applied to streaming data for the detection of anomalies in the cyber security domain [27].

4.4. Evaluation Criteria

Since the goal of this research is to find a feasible way of reducing false positives in rule-based NIDS without thereby affecting the number of wrongly classified true events, the main evaluation criteria will be the Recall which gives an indication about the total correctly identified true events of the prototype and the False Positive Rate (FPR) which represent the probability of false alarms. Furthermore, we will also take into consideration the Precision, Specificity and False Negative Rate. To calculate these measures we use the values obtained from the confusion matrix.
The selected techniques will also be compared to the other proposed solution in terms of the computational complexity of the applied algorithms. Even though the proposed solution is trained offline and retrained only when necessary as opposed to other solutions, the computational complexity will be included in the evaluation. This is done because a possibility remains of updating the models with new data in an online manner.

Confusion Matrix

A confusion matrix is a representation of the true labels in comparison with the labels assigned as a result of a classification process. In this case a true positive represents an accurately detected attack. A false positive represents a mistake made by the intrusion detection system where an alarm is triggered by a benign event.

Confusion Matrix		**Predicted**	
		Normal	Anomaly
Actual	Normal	TN	FP
	Anomaly	FN	TP

Figure 4.2: Representation of a confusion matrix and the 4 categories that an instance can belong to after classification.

Precision

The precision represents the subset of correctly classified instances in the set of instances classified as anomalies by an algorithm.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.3}$$

Recall

Recall represents the total correctly classified anomalous instances from the complete set of anomalous instances present in the data.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.4}$$

Specificity

The specificity metric is also known as the True Negative Rate (TNR) and represents the proportion of correctly classified negative instances from the total negative instances present in the data.

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{4.5}$$

False Positive Rate

The false positive rate represents the ratio of falsely classified normal instances in comparison with the correctly classified normal instances. This metric is important for the evaluation of the method since the main goal is to automatically reduce false positives.

$$\text{FPR} = \frac{FP}{FP + TN} \tag{4.6}$$

False Negative Rate

The False Negative Rate (FNR) represents the proportion of wrongly classified anomalies out of the total anomalies present in the data.

$$\text{FNR} = \frac{FN}{FN + TP} \tag{4.7}$$

5

Data Exploration

In this chapter the process of data collection is explained as well as the the train and test sets that are used and the methods used to process and analyse the data. Additionally, knowledge-based techniques are evaluated to some extent by collecting information about the environment in order to eliminate false positives. Furthermore the initial findings and characteristics of the data are presented. Two data sets used during this research are collected at an organisation which will remain anonymous throughout this report. The data is stored only for the duration of this project.

5.1. Data Collection

During this thesis we make use of an intrusion detection system to capture network traffic and analyse the content. The experimental setup consists of an NIDS installed on a server running a Linux distribution. The data is collected on a portion of the network. The intercepted traffic is send to the NIDS through a SPAN port which copies all the packets that it receives. The NIDS makes use of a packet sniffing library that captures the incoming packets which are then preprocessed and analyzed. The final output of the NIDS is collected and stored. The network contains many different devices including desktops, laptops, printers and various servers.

One NIDS sensor was used for the monitoring process. The NIDS sensor was placed inside the network, behind the firewall. The reason for this setup is the assumption that the firewall will block most of the unwanted traffic and thus the NIDS will have less false positives and in general less alerts when placed behind the firewall. Another important thing is the strategic placement of the NIDS on the internal network [8]. A good practice is to place the NIDS where more internal subnets come together. In this case the NIDS was placed such to receive traffic from multiple subnets. The data that was collected is exclusively used for

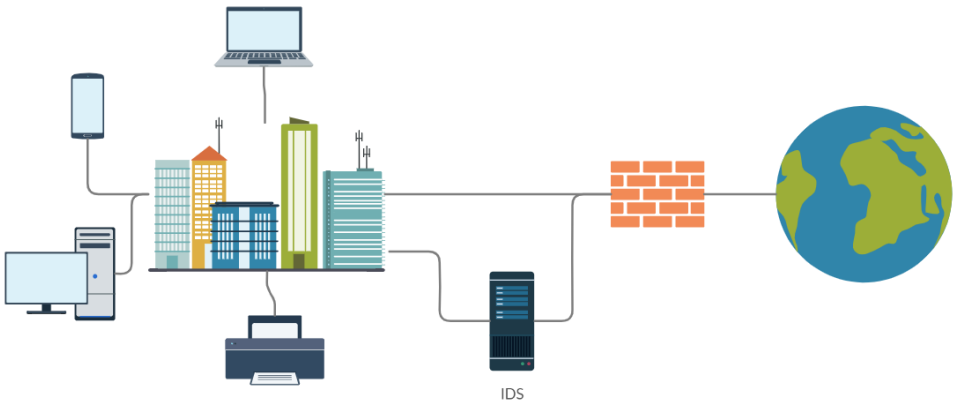


Figure 5.1: Network Topology

this research. A visualisation of the network setup that was used for the data collection can be found in 5.1.

5.1.1. Tools

Multiple tools where used during the data collection and analysis process. These are a network vulnerability scanner, a network intrusion detection system and a visualisation tool that was used to view the alert set and other statistics of the data. There are various tools that can perform these two activities. During this research Nessus, SNORT, the ELK stack and Python were used. Nessus was used for the vulnerability assessment, Snort as the main NIDS, the ELK stack which stands short for Elasticsearch-Logstash-Kibana was used to store and visualise the collected data and Python was used for the implementation and evaluation tasks.

Snort

The chosen IDS for this task was Snort. This IDS is chosen based on positive evaluations and comparisons to the other open source tools which are Suricata and Zeek (Bro). Snort has a simple deployment and rich rule set which gets updated frequently by the community. It is still one of the most used intrusion detection systems for monitoring traffic on network level. During this thesis Snort was deployed in its default mode receiving all the packets that passed the firewall as well as all outbound requests that were either allowed or blocked by the firewall. Table 5.1 shows a sample of the collected data representing the alerts that are triggered by the IDS. The rule sets that were used are the available open-source rule sets, namely, the Community rule set and the Emerging Threat rule set.

These rule sets are used without any alterations as well. They have a subset of rules disabled per default based on feedback and observations that these rules generate a lot of false positives in many environments. These settings are not altered as well.

Class	Prio	Signature_id	Src_ip	Src_port	Dst_ip	Dst_port	Protocol
attempted-recon	2	2101411	src_ip	60826	dst_ip	161	17
policy-violation	1	2009702	src_ip	50852	dst_ip	53	17
trojan-activity	1	2025719	src_ip	1133	dst_ip	445	6

Table 5.1: Example of alerts as output of Snort

Nessus

For creating a network mapping and scanning the network for potential vulnerabilities, Nessus was used [67]. This tool can be characterised as a vulnerability assessment tool which uses various methods to scan the network for existing vulnerabilities, hereby providing information about open ports, running services and OS detection for the hosts in the scanned network. How the scan is being performed can be specified through policies. A set of standard policies is available for internal and external network scans. Nessus has a rich database of plugins which are used by the scanner for discovering vulnerabilities. The program can be used for host and service discovery options as well as port scans, malware scans and web-application scans.

Plugin	CVE	CVSS	Host	Protocol	Port	Name
10223	CVE-1999-0632	None	Host	UDP	111	RPC portmapper Service Detection
78479	CVE-2014-3566	4.3	Host	TCP	3269	SSLv3 Padding Oracle On Downgraded Legacy Encryption Vulnerability (POODLE)
83875	CVE-2015-4000	2.6	Host	TCP	636	SSL/TLS Diffie-Hellman Modulus <= 1024 Bits (Logjam)
97833	CVE-2017-0148	10	HOST	TCP	445	MS17-010: Security Update for Microsoft Windows SMB Server (4013389) (ETERNALBLUE)
10940	None	None	Host	TCP	3389	Windows Terminal Services Enabled
57608	None	5	Host	TCP	445	SMB Signing not required

Table 5.2: Example of a Nessus Scan log file

5.1.2. Pre-processing

Several different data sources were used which have a different structure, therefore separate scripts were developed to parse the data from each source. The scripts

are programmed in Python because it provides a rich set of libraries needed for reading and parsing network related data. Additionally, after the initial analysis, benign exceptional cases in the data were excluded since this behaviour did not represent the regular behaviour observed on the network. See figure 5.2.
From the considered encoding techniques the choice was to use the frequency encoding technique since this is closest to the notion of representing the scale and repetition that some classes and signatures show.

After parsing the data the sources were linked through common identifiers. These identifiers represent the address of the host machines. The address is thus chosen as the main identifier amongst the different data sets since it was the only unique common variable. IP addresses are assigned through the use of NAT and DHCP but the dynamic part is configured to be more static in the sense that each IP address does not get reassigned to another host for at least 3 days of inactivity. This covers the weekend when most employees are inactive.

5.2. Data Description

In this section a description of all data sets will be provided that are used during this research for the analysis and evaluation. This includes information about how the data is divided into a train and test set as well as a description of the labelling process and the attack types that are present in each of the data sets.

5.2.1. Test 1

The training set consists of the first 3/4 and the test set consists of the remaining 1/4 of the total collected data that is analysed in the previous section. The data is split sequentially, meaning, that the timestamp and order of occurrence of events is preserved. The test set is composed of one full week. The attacks that are present in this data set are a Heartbleed vulnerability exploit, alerts related to Trojan infections and alerts related to the Sundown Exploit Kit. The labelling process is done manually by inspecting alerts and verifying their cause for this data set.
Test set 1 is modified for the purpose of evaluating the robustness of the model and defining thresholds for the re-evaluation. Artificial entries are added in the test that represent noise and changes in the network. For example, new IP addresses are introduced which generate new types of traffic that is triggered by the NIDS. These processes are benign in the network but may be seen as anomalies because of the novel, infrequent values. With this type of traffic the goal is to test how robust the models are and what a good indication is to retrain on new, more recent data.

5.2.2. Test Set 2

This data set was collected at a different point in time and on a different network segment to test the validity of this approach. This network segment is from the same network but consists of a different user group. The data was collected over one day. 3 attacks were generated in the afternoon to have ground-truth data to evaluate the prototype. All other observed traffic during this day is considered regular. The data was sequentially split into a train and test set using the first 3/4 of the data for training and last 1/4 for testing. The training set is composed of data observed until the beginning of the afternoon. The training set thus consists of behaviour observed in the morning and partly afternoon. The test set contains alerts generated by the attacks performed in the afternoon. The performed attacks are a port scan, an Internet Explorer 8 exploit and an Eternal Blue exploit on a Windows Host. The labelling process is done manually, the alerts that are labelled as true positives in the evaluation phase are the attacks that were manually generated.

5.2.3. Test Set 3

The third data set was a synthetically generated data set with multiple labelled attacks. This data set is a publicly available data set obtained from The Canadian Institute for Cybersecurity [57]. The alerts are obtained by running through the NIDS while preserving the original timestamps. The data consist of five full days of network traffic generated to mimic a real network with various devices and processes. These days span from Monday to Friday. The data was sequentially split into 3/4 training set and 1/4 test set. The training set consists of the traffic observed on Monday, Tuesday, Wednesday and partly Thursday. The remaining traffic is used as a test set. The attacks that were observed on Tuesday, Wednesday and Thursday are removed from the train set such that the data is representative of regular non malicious traffic. The attacks that remain in the test set on Friday are multiple port scans and OS detection probes, a Brute Force attack, a DOS attack, a Windows exploit with Metasploit, an infiltration attack and a Botnet attack. For this data set the attacks are also known and the alerts are labelled manually based on this information. These labels are not used by the models in the training phase but only for the evaluation of the prototype.

5.3. Data Analysis

In this section the collected data is visually presented. The initial observations and findings are also included. Although several test data sets are used in this research, the data distribution and characteristics of the first test data set are considered to be representative for feature selection and machine learning algorithm selection in real environments. The total amount of alerts collected over a time of approximately 4 weeks was 2,059,759. The alerts were collected in the pe-

riod between July 11th - August 5th. A visual representation can be found in 5.2.

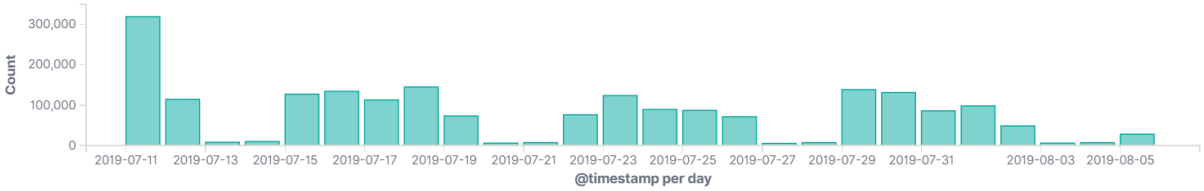


Figure 5.2: The total amount of alerts over time that were generated during the data collection period. The peak at the beginning was a benign, but exceptional process and is not included for further analysis.

From the figure it is clear that during the working week (Monday to Friday) there is a significantly larger amount of alerts generated than during the weekends. This indicates that the alert magnitude is somewhat correlated to the number of people that are using a device on the network during the working week. What this behaviour also indicates is that the NIDS probably labels many benign user actions and automated processes as malicious. Another observation is the repeating distribution of the data. Except for the first day of logging the rest of the days are consistently remaining between a certain interval of alerts per day. Weekends are always below the 1,000 alerts and the rest of the days are always below 150,000 alerts per day.

5.3.1. Rule Correlations

Multiple pairs of rules were observed to be highly correlated. These are shown in Appendix 8. The high correlation of these rule can be used to combine alerts that represent the same event. With this knowledge the number of generated alerts per event can be reduced by merging two alerts is one super-alert that contains both rules if the alert is generated from the same source to the same destination in a predefined time interval. Or this problem could be addressed beforehand by experts that design the rule sets to eliminate duplicate rules or rules that are likely to be generated by the same event.

5.3.2. Data Distribution

In 5.3 the hourly distribution of alerts is presented from days between Monday and Friday over a period of three weeks. The distribution of alerts is very different between 08:00 AM - 15:00 PM and 15:00 PM - 07:00 AM. In the latter case the total alert count per hour doesn't exceed 1,000 whereas in the former case it can get up to 20,000 alerts per hour. From this we can already see a clear base behaviour of the network. This behaviour can even be further dissected when looking at the alerts on hourly level. The working days have a clear distinction

between active (working) and inactive hours which are shown in 5.3. Furthermore, we looked at the distribution of occurrences of rules during a day. This can be seen in 5.6. There is a similar distribution amongst the different days, however sometimes we see slight deviations which are not necessarily indicative of malicious or abnormal behaviour. It is clear that the sensor shows a regular behaviour on the network on an daily basis. This insight can be used to capture the regular behaviour of the network through the sensor.
The second example is from Saturday and Sunday. During these days the majority of employees are free. Therefore only little to no activity is expected on the network during the weekend. The distribution of alerts can be found in 5.4. Furthermore an assumption is made that these rules are triggered by automated processes that are very normal in this environment because of the constant patterns that are observed.

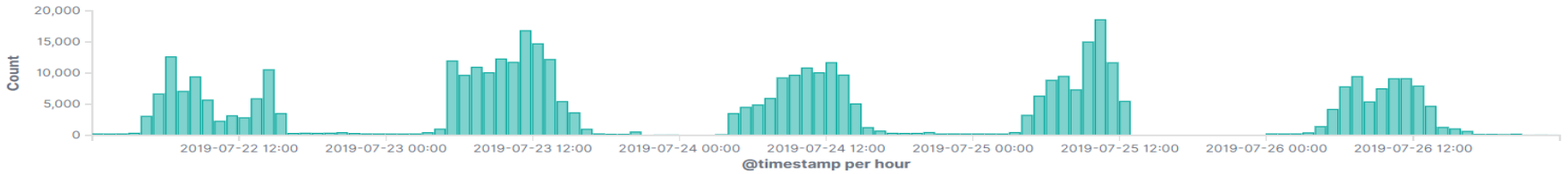


Figure 5.3: This figure represents the alert distribution during the working week, from Monday to Friday. Most of the alerts are generated between 08:00h and 15:00h. Each day is slightly different in terms of frequencies of generated alerts. These differences are due to processes that are not necessarily automated or employees that work on different days.

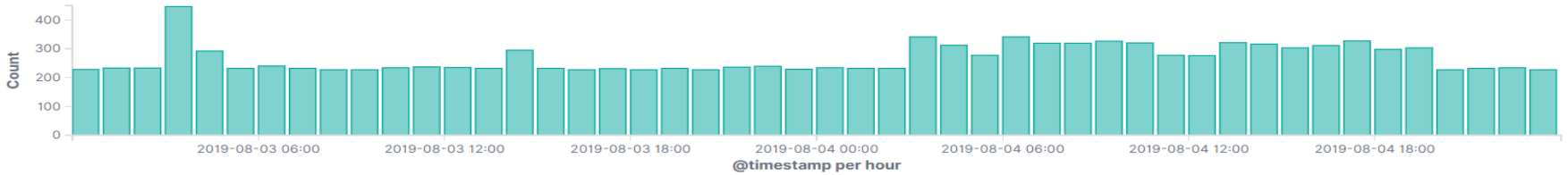


Figure 5.4: This figure represents the alert distribution during the working week, from Monday to Friday. Most of the alerts are generated between 08:00h and 15:00h. Each day is slightly different in terms of frequencies of generated alerts. These differences are due to processes that are not necessarily automated or employees that work on different days.

5.3.3. Vulnerabilities

In 5.5 statistics are shown about the percentage of references present in the alert set and the vulnerability set. Since the vulnerability set is an important information source for obtaining context it is necessary that this source contains sufficiently rich information to label alerts as true or false.

To link an alert to a vulnerability there is a need of a common identifier. In this particular case the main identifier is the destination host. Furthermore, an additional variable is needed that provides information over the actual vulnerability that is found on a host. Such information is also needed from the alert, where it is called a reference to a vulnerability. The most common example of such a reference is a CVE_ID which is a unique enumeration for a discovered vulnerability. The vulnerability scanner also links certain findings to CVE_IDs or other vulnerability databases such as Exploit-DB[21].

The percentage of vulnerabilities found in the network that have a reference to a CVE_ID or other database is only around 5.5%. The vulnerability references obtained from the generated alerts are also very minimal around 7%. What is a more limiting factor is that the references from the vulnerability set and the ones from the alert set have no matching entry. Meaning, there is no case where an alert with CVE_ID is generated against a particular host, and at the same time the scanner actually identified the host to be vulnerable to this particular vulnerability that has been enumerated with the same CVE_ID. This method is proposed and tested by multiple authors as reviewed in Chapter 3.1 but the effectiveness of this method seems not to be evaluated sufficiently.

5.4. Observations

The main observations and characteristics obtained from the data analysis are listed in this section.

5.4.1. Observation 1:

Looking at the network behaviour on a daily level enables us to distinguish between the days which have a lot of user activities and the weekends where these activities are significantly lower. This is important because the traffic observed in these two categories shows significant differences in observed patterns and alert magnitude.

- During the working week days the distribution of alerts is centred mostly between 08:00 and 16:00. The alerts generated during the active hours of the day (08:00 to 16:00) show some peaks and dips but in general have a consistent distribution. The alerts generated during the inactive hours of

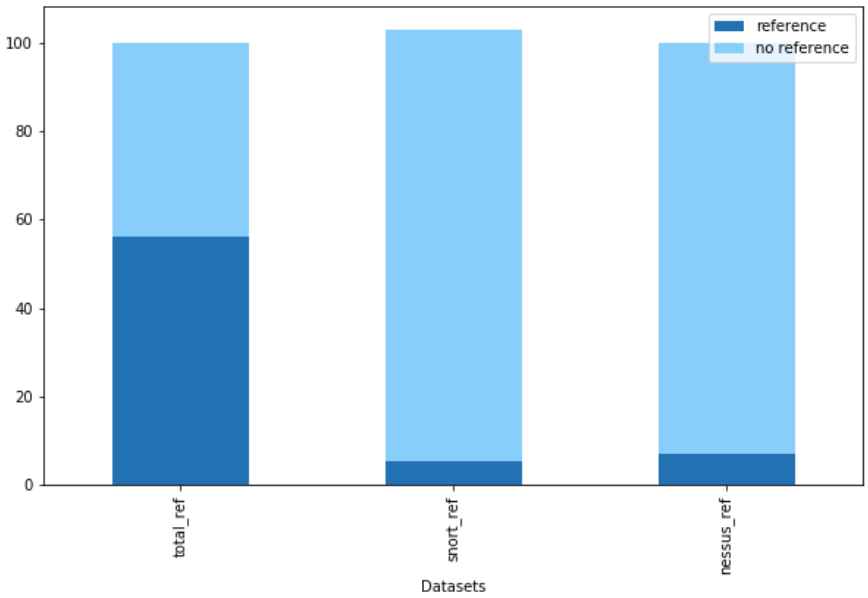


Figure 5.5: Statistics of the vulnerability references included in the data that is used during this research. The first bar represents the total alert set of active rules used during the data collection. The second bar represents all the references that were included in the obtained output of the NIDS (alerts). The third bar represents the references included in the vulnerability scan performed on the monitored network.

the day are almost constant.

- On weekends the distribution of alerts is almost constant throughout each day. On Sundays it is slightly higher than on Saturdays and also more deviating. The cause for these differences can be irregular working times of employees on weekends from different locations. The sensor behaviour during the weekends is clearly distinctive from the rest of the week.

5.4.2. Observation 2:

Several pairs of rules were shown to have highly correlated behaviour. This may be an indication of rules triggered on the same event and causing even more noise in the data. As a prepossessing step these rules could be merged so the volume of alerts will be reduced. This will not be done in this research since these correlation patterns might be too specific for this data set. Additionally, the repetitive behaviour is encoded with the relative frequencies of the alerts and these cases should be filtered by the model.

5.4.3. Observation 3:

Correlating the alerts with context information obtained through network and vulnerability scans is very limited. In specific cases this method might be very

useful, but in general this kind of correlation does not result in a clear indication of whether the alert is true or false.

- The scenarios where these scans are useful are scenarios where the generated alerts and the target host have the same vulnerability reference, or meta data obtained from the alert can be linked with sufficient information obtained about the host to assess whether the attack poses a risk for the target host.
- The scenarios where these scans are not useful are scenarios where the alert doesn't have any specific vulnerability and matches on a very generic content pattern and where the targeted host has also no specific vulnerabilities, or the targeted host is located somewhere on the external network.

5.4.4. Observation 4:

In general, a sub-group of rules is regularly observed throughout the whole period. The repeating rules represent false positives in this case. The regularity with which rules are observed differs. Rules that are observed very regularly will have high relative frequencies, whereas rules that are observed only periodically will have lower relative frequencies.

- True positive alerts are generally absent from a benign set of alerts. Therefore, when represented as frequencies, these instances would get a frequency value of 0. The same holds for other attributes such as IP addresses or ports that have not been intercepted before. Filtering out only the high frequent alerts would still leave a large subset of false positives.
- There exist different groups of regular behaviour within the sensor represented by different frequencies. Clustering the alerts into these groups could reveal malicious behaviour.

5.5. Examples

Two examples are provided to shown in which cases the external knowledge does not help identifying whether an alert is a false positive and in which cases is might still be useful.

5.5.1. Example 1

Here a case is presented which shows how much noise the NIDS actually produces. In 5.6 the total rules generated during that day are shown. In 5.7 a manually filtered set is presented where true positive alerts are visible. In the first figure it is nearly impossible to spot the Trojan alert since it is only a small dot in the data. After filtering the noise we can clearly see more unusual behaviour.

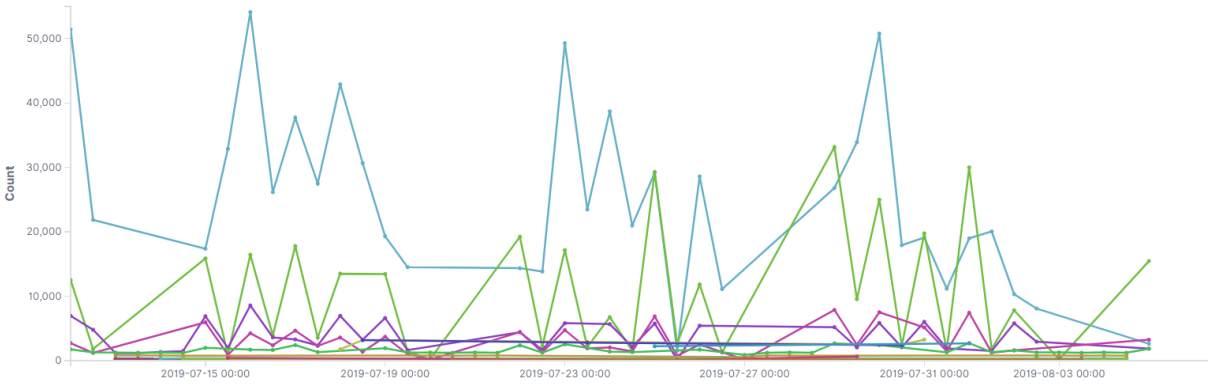


Figure 5.6: Representation of the magnitude if events per unique rule.

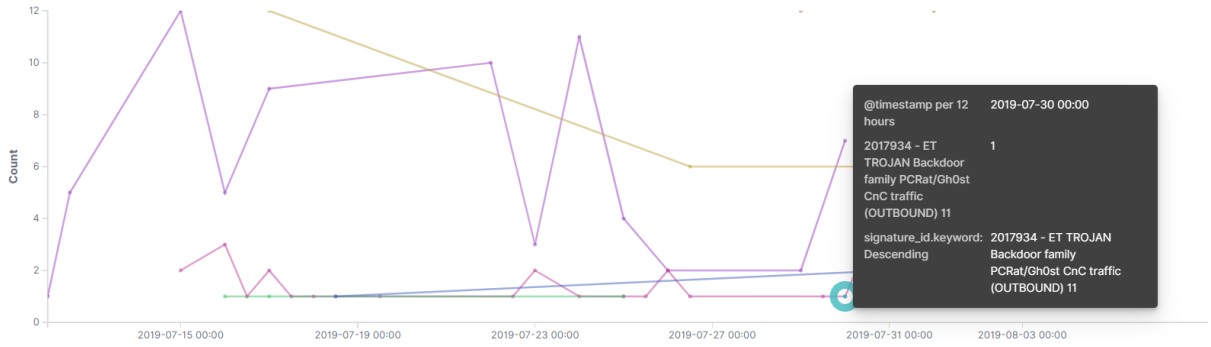


Figure 5.7: Representation of a true positive alert with a low frequency.

5.5.2. Example 2

The list of consecutive alerts shown in table 5.3 is selected and investigated. This behaviour could possibly represents an attack. When examining the network information obtained from the destination host 5.4 we can see there is no specific vulnerability found by the scanner (no CVE reference). The only information that we can extract from the network scanner is that the host is a SQL server. Knowing this information increases the probability that these alerts are indeed true but it is still no clear indication. This is an example of how context information can be useful when available.

Signature	Class	Dest
ET SCAN Suspicious inbound to Oracle SQL port 1521	bad-unknown	H
GPL SQL service_name buffer overflow attempt	attempted-user	H
GPL SQL user name buffer overflow attempt	attempted-user	H

Table 5.3: Alerts indicating a buffer overflow attack.

In both cases the context information is useful, after the event has been iden-

Vulnerability	CVE	CVSS	Protocol	Port	Host
Ping the remote host	None	None	TCP	0	H
Traceroute Info	None	None	UDP	0	H
Oracle Database Service tnslsnr Remote Version Disclosure	None	None	TCP	1521	H
Nessus SYN Scanner	None	None	TCP	1521	H
Oracle Database Detection	None	None	TCP	1521	H
OS Identification	None	None	TCP	0	H

Table 5.4: Vulnerability Scan results associated to the destination host as specified in the buffer overflow alerts.

tified as possibly anomalous. When only relying on the context information these alerts would have been assigned the same probabilities as some repeating rules which clearly are 'regular'. Improving the accuracy of the NIDS based on contextual information about the network is challenging and very limited. It can be better used as an additional check then as the main method for filtering false alarms.

There are mainly two situations which can be distinguished clearly from the initial data analysis. The scenarios where an alert matches a specific vulnerability based on the packet content that is captured is the first scenario and the scenarios in which an alert is too generic to have a vulnerability identifier. In the former case, this information is specifically useful for the elimination of false positives when the target host is (not) vulnerable to that attack because of its configuration. In the latter case, no final decision can be made about the nature of an alert. As most information can be linked only through the name of the alert, we can only conclude whether the alert has a probability of being true when the host has indeed open ports and running services that could potentially be exploited. However, relying solely on this information does not bring us much further than before since the majority of alerts would still need to be analysed manually.

Since we have observed a repetitive behaviour on the sensor, we can leverage this information in combination with the contextual information to improve the NIDS accuracy. By creating a model of regular alerts a lot noise will be filtered out and the remaining alerts can then be additionally cross-checked with the contextual information to be further reduced.

5.6. Features Selection

The data set that is used for this research contains the following basic set of features: signature_id, class, src_ip, dst_ip, src_port, dst_port, ip_protocol,

priority. These features provide information about traffic that is detected in the network. The combination of a selected subset of features may be indicative of anomalous behaviour. Based on the results of the calculated correlation between the features and target variable shown in Figure 5.8 and 5.9 only a subset of features was chosen. The data set was reduced by dropping features that did not provide additional information about the behaviour of the sensor.

In Figure 5.8 the results of the test as calculated on the training data are shown. These scores are calculated for each feature and the target variable which in this case is the label that indicates whether an alert is a true or false positive. A higher score means a higher dependence between the variables.

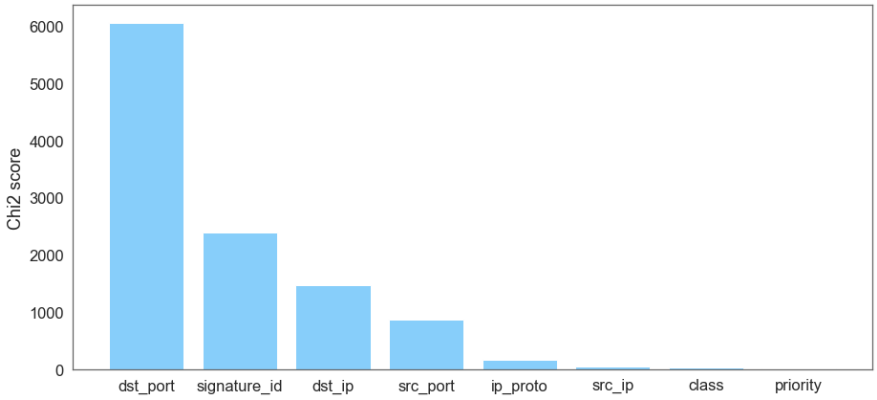


Figure 5.8: Chi Square Test between the features and target variable

In Figure 5.9 the mutual information score for each variable with the target variable is shown. As in the previous case a higher score means a higher dependence between the variables. From the results of both methods that are applied, it can be observed that the destination address, destination port and signature are the features with the highest scores. This indicates that these are the most relevant features.

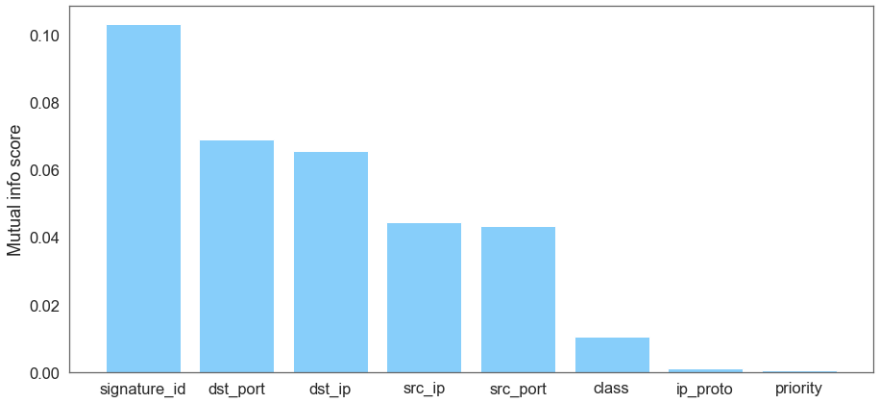


Figure 5.9: Mutual Information Test between the features and target variable

6

Results

In this chapter the results of all applied methods during this research are presented and discussed. The proposed method is tested on two additional data sets. One data set which is collected on the same network and one artificial data set. The anomaly detection methods that are applied are the Local Outlier Factor, Isolation Forest, Cluster-based Local Outlier Factor and Histogram-based Outlier Score.

6.1. Hyper-parameters of Models

Each model is different and requires a different set of hyper-parameters to be defined before usage. The hyper-parameters for each model are different and these are specified in Section 6.2. The main reason to choose default values is the wide range of possibilities that need to be tested and compared. The optimum set of parameters is likely to be different for each setup and this means that manual tweaking of the parameters will be necessary. Since the goal of this research is to develop an automated model and avoid manual adjustments, the hyper-parameters are chosen as the default values with a motivation provided by the authors of the algorithms that in most cases the algorithms will perform well with these settings.

6.2. Threshold Selection

In this section the result obtained for a set of thresholds is presented for each model. The default contamination value for all considered algorithms is 0.1. The threshold is also called the contamination factor and represents the proportion of outliers in the data set. This value is used to define the threshold for the outlier scores computed by the different methods. This threshold can be set as a float value in the range [0, 0.5]. The reason to consider several different thresholds

to evaluate is related to the proportion of true positives within the data which can be lower than the provided contamination factor. This could result in a better accuracy of the model. Furthermore, there can be cases where many alerts are generated as a result of a port scans, for these cases the proportion of true positives in the data would probably be higher than the default factor. In these experiments the thresholds that are used are increasing with 0.05 such that the range of possible thresholds is evaluated sufficiently. Additionally, the confusion matrix of each model with the optimal parameters is shown for each tested data set.

6.2.1. LOF

Local Outlier Factor					
parameters	contamination	n_neighbours	metric	leaf_size	algorithm
values	threshold	10	euclidean	30	auto

Table 6.1: Default set of hyper-parameters for Local Outlier Factor

Of the range of thresholds with which this algorithm was tested, all thresholds perform equally well. This technique did not improve its results with a higher threshold and this pattern is repeated through all the tested data sets. The number of false positives is low. However, in two out of three data sets, this method failed to identify all the important alerts. This indicates that the algorithm fails to correctly distinguish the true positive points based on the calculated scores. In the second data set all true events are correctly identified. This could be due to the data that consists of only one day of traffic and may be fitted better. But even for this case the false positive rate is not improved by setting different thresholds.

	Test Set 1				Test Set 2				Test Set 3			
Threshold	FNR	FPR	Precision	Recall	FNR	FPR	Precision	Recall	FNR	FPR	Precision	Recall
0.05	0.1047	0.0385	0.0551	0.8952	0.0	0.0048	0.1398	1.0	0.0129	0.0878	0.4135	0.9870
0.1	0.1047	0.0385	0.0551	0.8952	0.0	0.0048	0.1398	1.0	0.0129	0.0878	0.4135	0.9870
0.15	0.1047	0.0385	0.0551	0.8952	0.0	0.0048	0.1398	1.0	0.0129	0.0878	0.4135	0.9870
0.2	0.1047	0.0385	0.0551	0.8952	0.0	0.0048	0.1398	1.0	0.0129	0.0878	0.4135	0.9870
0.25	0.1047	0.0385	0.0551	0.8952	0.0	0.0048	0.1398	1.0	0.0129	0.0878	0.4135	0.9870
0.3	0.1047	0.0385	0.0551	0.8952	0.0	0.0048	0.1398	1.0	0.0129	0.0878	0.4135	0.9870
0.35	0.1047	0.0385	0.0551	0.8952	0.0	0.0048	0.1398	1.0	0.0129	0.0878	0.4135	0.9870
0.4	0.1047	0.0385	0.0551	0.8952	0.0	0.0048	0.1398	1.0	0.0129	0.0878	0.4135	0.9870
0.45	0.1047	0.0385	0.0551	0.8952	0.0	0.0048	0.1398	1.0	0.0129	0.0878	0.4135	0.9870
0.5	0.1047	0.0385	0.0551	0.8952	0.0	0.0048	0.1398	1.0	0.0129	0.0878	0.4135	0.9870

Table 6.2: Threshold selection for LOF

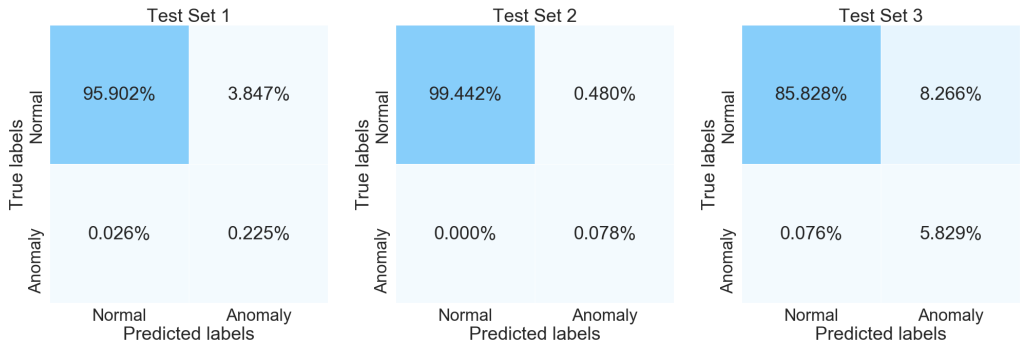


Figure 6.1: Results of the anomaly detection with LOF

6.2.2. IF

Isolation Forest				
parameters	contamination	n_estimators	max_samples	max_features
values	threshold	100	100	1.0

Table 6.3: Default set of parameters for Isolation Forest

The Isolation Forest algorithm performs well in all three data sets. The best threshold for this method slightly varies amongst the different data sets but falls in the range between [0.05, 0.1].

	Test Set 1				Test Set 2				Test Set 3			
Threshold	FNR	FPR	Precision	Recall	FNR	FPR	Precision	Recall	FNR	FPR	Precision	Recall
0.05	0.0	0.0551	0.0436	1.0	0.0	0.2846	0.0027	1.0	0.5549	0.0939	0.2291	0.4450
0.1	0.0	0.0951	0.0257	1.0	0.0	0.2846	0.0027	1.0	0.0	0.1640	0.2766	1.0
0.15	0.0	0.1555	0.0159	1.0	0.0	0.2846	0.0027	1.0	0.0	0.1642	0.2764	1.0
0.2	0.0	0.2012	0.0123	1.0	0.0	0.2846	0.0027	1.0	0.0	0.9468	0.0621	1.0
0.25	0.0	0.2049	0.0121	1.0	0.0	0.1028	0.0075	1.0	0.0	0.9607	0.0613	1.0
0.3	0.0	0.2995	0.0083	1.0	0.0	0.1028	0.0075	1.0	0.0	1.0	0.0590	1.0
0.35	0.0	0.3335	0.0074	1.0	0.0	0.1028	0.0075	1.0	0.0	1.0	0.0590	1.0
0.4	0.0	0.3873	0.0064	1.0	0.0	0.0615	0.0125	1.0	0.0	1.0	0.0590	1.0
0.45	0.0	0.4255	0.0058	1.0	0.0	0.0607	0.0126	1.0	0.0	1.0	0.0590	1.0
0.5	0.0	0.4632	0.0054	1.0	0.0	0.0603	0.0127	1.0	0.0	1.0	0.0590	1.0

Table 6.4: Threshold selection for IF

The Isolation Forest Algorithm tends to compute similar scores for the anomalous and some of the benign instances based on the similar frequency values that these instances have for the same attributes. In this particular case, the destination IP and destination port attributes share similar values and since these are the majority the high frequency value of the signature is suppressed and therefore the score is more anomalous and gets wrongly classified by this algorithm. The weak point here is that even though the signature has been observed before and

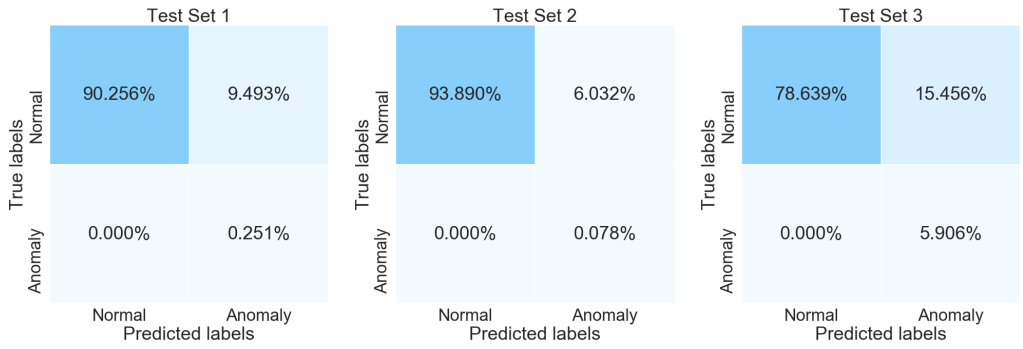


Figure 6.2: Results of the clustering and anomaly detection with IF

represents regular behaviour, because of the novelty of the destination address and destination port the alert gets wrongly classified. There is a subset of points which get a high anomaly score assigned by this algorithm. The common pattern here is that 2 out of the three features are very infrequent (destination port and destination address) but the destination port has a high frequency value. A different selection of features or a more balanced set of attributes might mitigate these false positives.

6.2.3. HBOS

Histogram based Outlier Detection				
parameters	contamination	n_bins	alpha	toll
values	threshold	sqrt(n)	0.1	0.1

Table 6.5: Default set of hyper-parameters for Histogram based Outlier Score

The Histogram-based outlier scores technique also shows satisfying results, all true positives are found for all three data sets. Again the best threshold in each case varies a bit. For the first two data sets which represent alert data collected on a real network, the optimal threshold lies in the range between [0.05, 0.1]. For the synthetically generated data set this threshold is higher, namely, 0.2 of the total size. Therefore, the false positives are also higher for this case. An explanation for this difference could be the underlying distribution of the different events in the data which may be very variable for the last data set.

Since the score for each point is computed as the sum of logs of inverse. The selection of buckets is an important parameter here. If the buckets are not well defined true positive points can end up in buckets of normal points. Therefore this method assigns similar scores to totally different points in the data set.

	Test Set 1				Test Set 2				Test Set 3			
Threshold	FNR	FPR	Precision	Recall	FNR	FPR	Precision	Recall	FNR	FPR	Precision	Recall
0.05	0.0034	0.0528	0.0453	0.9965	0.0	0.2846	0.0027	1.0	0.4450	0.0807	0.3012	0.5549
0.1	0.0	0.0964	0.0254	1.0	0.0	0.2846	0.0027	1.0	0.4450	0.1171	0.2291	0.5549
0.15	0.0	0.1337	0.0184	1.0	0.0	0.2846	0.0027	1.0	0.0258	0.1968	0.2369	0.9741
0.2	0.0	0.1742	0.0142	1.0	0.0	0.2846	0.0027	1.0	0.0	0.2186	0.2230	1.0
0.25	0.0	0.2095	0.0118	1.0	0.0	0.1028	0.0075	1.0	0.0	0.2186	0.2230	1.0
0.3	0.0	0.2323	0.0107	1.0	0.0	0.1028	0.0075	1.0	0.0	1.0	0.0590	1.0
0.35	0.0	0.2712	0.0091	1.0	0.0	0.1028	0.0075	1.0	0.0	1.0	0.0590	1.0
0.4	0.0	0.2712	0.0091	1.0	0.0	0.0615	0.0125	1.0	0.0	1.0	0.0590	1.0
0.45	0.0	0.3605	0.0069	1.0	0.0	0.0607	0.0126	1.0	0.0	1.0	0.0590	1.0
0.5	0.0	0.4359	0.0057	1.0	0.0	0.0576	0.0133	1.0	0.0	1.0	0.0590	1.0

Table 6.6: Threshold selection for HBOS

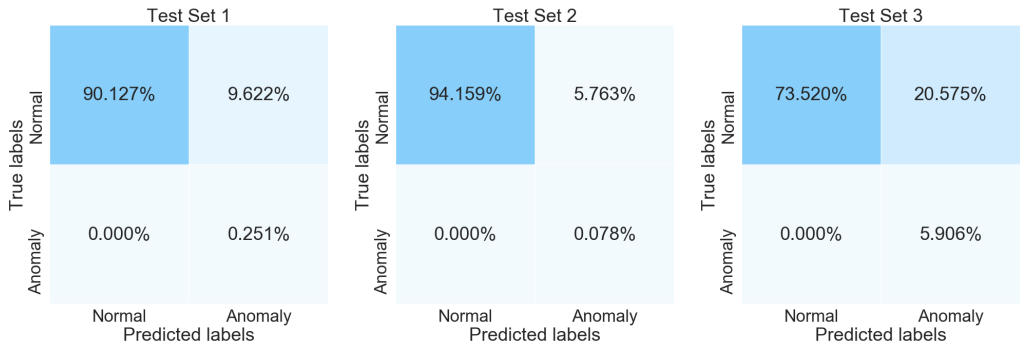


Figure 6.3: Results of the anomaly detection with HBOS

The true anomalies have similar scores with regular patterns where the three attributes have lower frequencies caused by periodical but benign events. Since the algorithm considers the frequency as a continuous variable the frequency 0.00004 and 0.4 may both end up in bins with similar densities while representing two totally different events. Ideally, the true positive points would be all categorised in buckets with low densities and the benign points in bins with high densities.

6.2.4. CBLOF

Cluster Based Local Outlier Factor				
parameters	contamination	n_clusters	alpha	beta
values	threshold	8	0.9	5

Table 6.7: Default set of hyper-parameters for Cluster-based Local Outlier Factor

The Clustering-based Local Outlier Factor technique performs slightly worse than the previous two techniques. All true positive instances are found by this technique but at the cost of more false positives. The optimal threshold, for all three data sets, lies in the range between [0.05, 0.1].

	Test Set 1				Test Set 2				Test Set 3			
Threshold	FNR	FPR	Precision	Recall	FNR	FPR	Precision	Recall	FNR	FPR	Precision	Recall
0.05	0.9965	0.0455	0.0001	0.0034	0.0	0.0497	0.0154	1.0	0.5549	0.0807	0.2569	0.4450
0.1	0.9965	0.1085	0.0008	0.0034	0.0	0.0607	0.0126	1.0	0.0	0.1943	0.2441	1.0
0.15	0.0	0.1731	0.0143	1.0	0.0	0.0615	0.0125	1.0	0.0	0.1943	0.2441	1.0
0.2	0.0	0.2012	0.0123	1.0	0.0	0.1028	0.0075	1.0	0.0	0.1943	0.2441	1.0
0.25	0.0	0.2029	0.0122	1.0	0.0	0.1028	0.0075	1.0	0.0	0.1943	0.2441	1.0
0.3	0.0	0.2110	0.0117	1.0	0.0	0.1028	0.0075	1.0	0.0	0.1943	0.2441	1.0
0.35	0.0	0.3118	0.0080	1.0	0.0	0.1028	0.0075	1.0	0.0	0.1943	0.2441	1.0
0.4	0.0	0.3835	0.0065	1.0	0.0	0.1028	0.0075	1.0	0.0	0.1943	0.2441	1.0
0.45	0.0	0.4091	0.0061	1.0	0.0	0.1028	0.0075	1.0	0.0	0.1943	0.2441	1.0
0.5	0.0	0.4677	0.0053	1.0	0.0	0.1028	0.0075	1.0	0.0	0.1943	0.2441	1.0

Table 6.8: Threshold selection for CBLOF

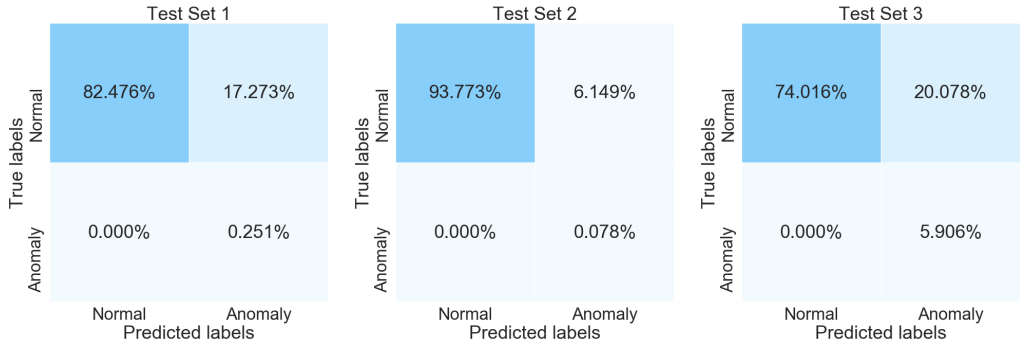


Figure 6.4: Results of the anomaly detection with CBLOF

This algorithm assigns all anomalous points to the same cluster. However, this cluster also contains other more rare events and regular events that happen periodically, but with small frequencies compared to several events that repeatedly trigger the NIDS. The rest of the clusters are composed with at least one of the attributes having a high value. Here the main issue for the false positives is that a subset of normal points with similar attributes is grouped in the smallest cluster. Therefore, the scores for these points are calculated as the distance to the closest large cluster centre. Scores for small clusters are assigned as the minimum distance to one of the large clusters centres. Scores for a large cluster are assigned as the distance of the point to the center of the cluster the point is assigned to. Because the subset of anomalous instances have very infrequent values but are a minority class within the total data points and are therefore assigned to a large cluster that hosts more minority groups with different values but more similar than the other clusters. The algorithm is not capable of building good clusters to distinguish the low frequency events.

6.3. Combining Results

From Figure 6.5 we can see that all 4 methods have an overlap just 1.1% of anomalous labelled instances. This means that the overlapping anomalous in-

stances that are wrongly classified by all four algorithms differ. Since the goal is to reduce the number of false positives it makes sense to combine the methods in order to normalise the anomaly scores of the miss-classified points.

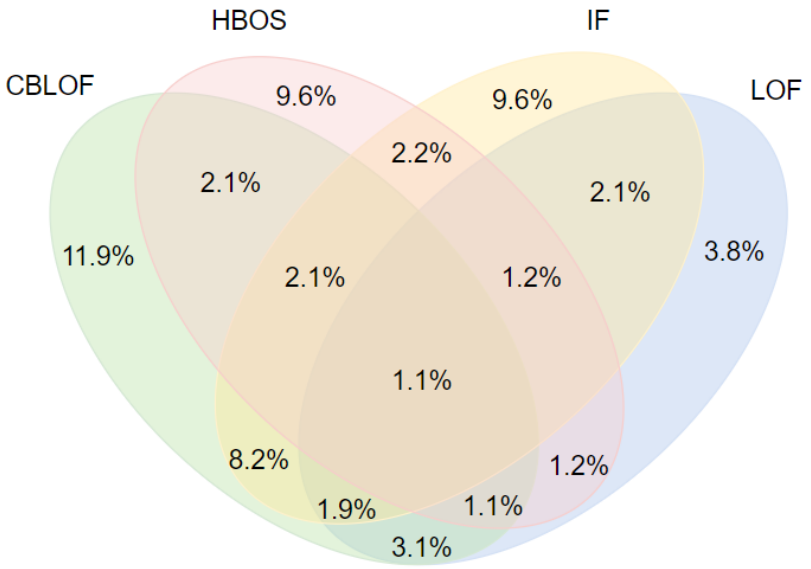


Figure 6.5: The percentage of overlapping instances labelled as anomalous for each tested method against the total number of data points.

It seems that all three methods suffer from the inability to distinguish low frequent alerts from non-observed before alerts. This subset of alerts represent the remaining false positives after the methods are combined. The other false positive points are reduced by combining these methods since the scores for some false positives are normalised. The LOF algorithm has the lowest non-overlapping anomalous instances as shown in Figure 6.5. Additionally, LOF did not find all true anomalies in the data with all tested thresholds. When combining with the other methods it may lead to additional false negatives. Since one of the requirements of the prototype is the detection of all true attacks the LOF method is not further explored.

Several approaches are considered to combine scores of different anomaly detection methods in order to obtain a better performance. These are the mean, median, weighted average score or a majority vote on the classification results. Simple non-parametric measures are shown to be effective in practice [20]. Therefore, the combined score is calculated as the average score of the different methods.

$$Score_{IF-HBOS} = \frac{Score_{IF} + Score_{HBOS}}{2} \tag{6.1}$$

$$Score_{IF-CBLOF} = \frac{Score_{IF} + Score_{CBLOF}}{2} \tag{6.2}$$

$$Score_{HBOS-CBLOF} = \frac{Score_{HBOS} + Score_{CBLOF}}{2} \tag{6.3}$$

$$Score_{IF-HBOS-CBLOF} = \frac{Score_{IF} + Score_{HBOS} + Score_{CBLOF}}{3} \tag{6.4}$$

Since the scores are scaled in the range [0, 1] this range of values will be used to determine whether a point is anomalous or not. In general, for all combinations a score between 0.4 and 0.8 was sufficient to detect all true positives. The results obtained from each combination are shown in the following sections.

IF+HBOS

The combination of an ensemble and statistical based technique does not show significant improvements in comparison to the ensemble based techniques as standalone. According to the results shown in 6.9 the best score for this combination lies in the range [0.5, 0.8].

	Test Set 1				Test Set 2				Test Set 3			
Threshold	FNR	FPR	Precision	Recall	FNR	FPR	Precision	Recall	FNR	FPR	Precision	Recall
1	1.0	0.0	nan	0.0	1.0	0.0	nan	0.0	1.0	0.0	nan	0.0
0.9	1.0	0.0	nan	0.0	0.35	0.0101	0.0476	0.65	1.0	0.0	nan	0.0
0.8	1.0	0.0031	0.0	0.0	0.35	0.0101	0.0476	0.65	1.0	0.0	nan	0.0
0.7	1.0	0.0209	0.1072	1.0	0.0	0.0116	0.0628	1.0	0.3673	0.0618	0.3908	0.6326
0.6	0.9965	0.0270	0.0851	1.0	0.0	0.0355	0.0214	1.0	0.0258	0.1515	0.2875	0.9741
0.5	0.0	0.0893	0.0273	1.0	0.0	0.0602	0.0127	1.0	0.0	0.1619	0.2792	1.0
0.4	0.0	0.2104	0.0118	1.0	0.0	0.0607	0.0126	1.0	0.0	0.1856	0.2526	1.0
0.3	0.0	0.3253	0.0076	1.0	0.0	0.1028	0.0075	1.0	0.0	0.2124	0.2280	1.0
0.2	0.0	0.5535	0.0045	1.0	0.0	0.1028	0.0075	1.0	0.0	0.2186	0.2230	1.0
0.1	0.0	0.6873	0.0036	0.0052	0.0	0.2846	0.0027	1.0	0.0	1.0	0.0590	1.0

Table 6.9: Results of applying the IF-HBOS combination on the test set with a range of different thresholds

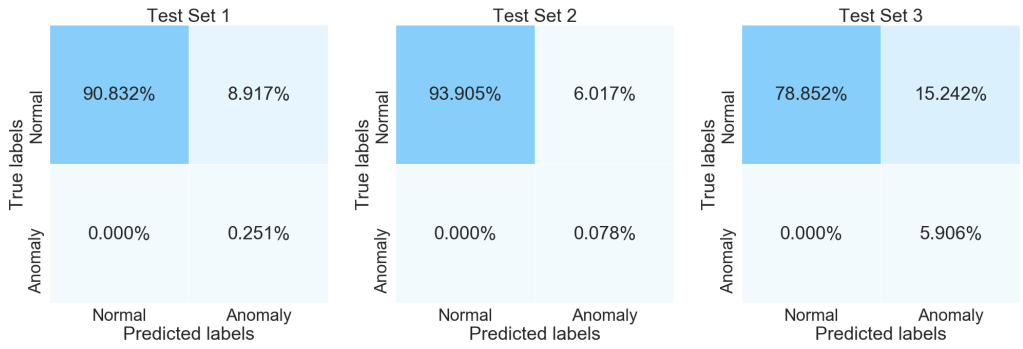


Figure 6.6: IF - HBOS - Confusion Matrices

First, the results obtained from this combination are compared with the results obtained from the HBOS method. For the first data set the improvement is around 1% less false positives. For the second data set the false positives are higher with approximately 1%. For the third data set the results are better and the false positives are reduced with around 5% which is a significant difference. In comparison to the results obtained from the IF method, the first data set has a reduction of around 1%. The results obtained for the second and third data set remain almost unchanged. The HBOS method performs better as standalone on the second data set. When combined with the IF method, the false positives slightly rise. However, in the first and third data sets the results are improved. The conclusion from these results is that the IF method has a positive effect on the scores of HBOS, but the other way around HBOS has no significant effects on the scores of the false positive instances as assigned by the IF method.

IF + CBLOF

The combination of these two techniques also does not show very significant improvements considering the individual performance of the combined methods. The best threshold values fall in the range [0.4, 0.7]. Again, each data set has a different optimal threshold. In the case of the first data set all true positives are found with a maximum threshold of 0.4. Whereas in the second data set this threshold can be set to 0.7. The third data set the threshold is even lower, namely, 0.3. The improvement of this combination is not very significant.

Comparing the results from this combination with the results of both methods individually we see for the first case that the number of false positives increases. This is due to the higher amount of false positives generated by the CBLOF method which has a poor performance. In the second and third data set the results are slightly improved. From these results it seems that these two method do not complement each other well in terms of anomaly scores.

	Test Set 1				Test Set 2				Test Set 3			
Threshold	FNR	FPR	Precision	Recall	FNR	FPR	Precision	Recall	FNR	FPR	Precision	Recall
1	1.0	0.0	nan	0.0	1.0	0.0	nan	0.0	1.0	0.0	nan	0.0
0.9	1.0	0.0052	0.0	0.0	0.35	0.0101	0.0476	0.65	1.0	0.0643	0.0	0.0
0.8	0.9965	0.0088	0.0009	0.0034	0.35	0.0101	0.0476	0.65	0.8033	0.0939	0.1161	0.1966
0.7	0.9965	0.0465	0.0001	0.0034	0.0	0.0353	0.0215	1.0	0.5549	0.0940	0.2290	0.4450
0.6	0.9965	0.0650	0.0001	0.0034	0.0	0.0369	0.0206	1.0	0.5549	0.0940	0.2290	0.4450
0.5	0.9965	0.0984	0.0008	0.0034	0.0	0.0497	0.0154	1.0	0.5549	0.0940	0.2290	0.4450
0.4	0.0	0.1632	0.0151	1.0	0.0	0.0497	0.0154	1.0	0.5549	0.0940	0.2290	0.4450
0.3	0.0	0.1996	0.0124	1.0	0.0	0.0607	0.0126	1.0	0.0	0.1380	0.3125	1.0
0.2	0.0	0.5234	0.0047	1.0	0.0	0.1028	0.0075	1.0	0.0	0.1499	0.2950	1.0
0.1	0.0	0.6245	0.0040	1.0	0.0	0.1028	0.0075	1.0	0.0	0.9598	0.0613	1.0

Table 6.10: Threshold selection for IF-CBLOF

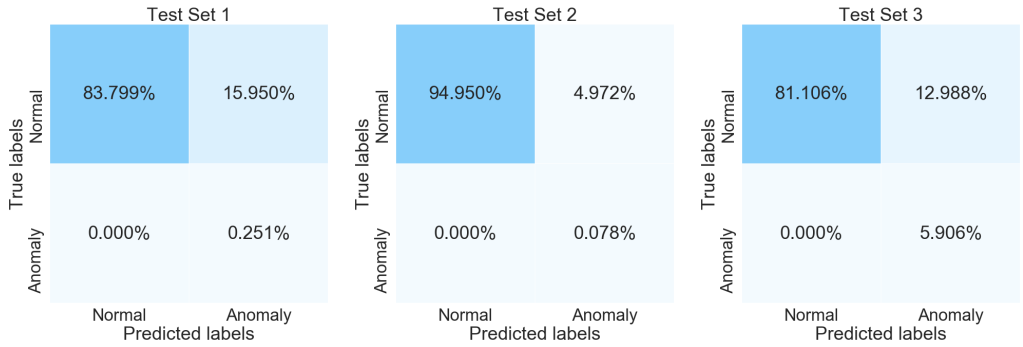


Figure 6.7: IF-CBLOF-Confusion Matrices

HBOS+CBLOF

The results obtained from combining the HBOS and CBLOF methods do reduce the false positives significantly in the first two data sets. The best threshold for this combination can be set to a value in the range [0.4, 0.7]. Even though the optimal threshold for the first data set is 0.4, the false positives are reduced with more than 2% compared to the previous combinations and the HBOS and CBLOF results individually. The same holds for the second data set. In the case of the synthetic data set the results are better than HBOS and CBLOF considered individually. However, these are not better than the results of the IF-CBLOF combination or solely the IF method.

The results of this combination are better than both methods considered separately. The false positives in the first case decrease with almost 10%. For the artificial data set the improvement is not very significant. This may be due to the representation of regular alerts which in this data set is deviating a lot. The events that are flagged as malicious based on their frequency scores may be part of an anomalous cluster according to the CBLOF algorithms and since they get a high anomaly score from that method their overall score is not normalised enough.

	Test Set 1				Test Set 2				Test Set 3			
Threshold	FNR	FPR	Precision	Recall	FNR	FPR	Precision	Recall	FNR	FPR	Precision	Recall
1	1.0	0.0	nan	0.0	1.0	0.0	nan	0.0	1.0	0.0	nan	0.0
0.9	1.0	0.0	nan	0.0	0.35	0.0101	0.0476	0.65	1.0	0.0	nan	0.0
0.8	1.0	0.0	nan	0.0	0.35	0.0101	0.0476	0.65	1.0	0.0	nan	0.0
0.7	1.0	0.0001	0.0	0.0	0.35	0.0101	0.0476	0.65	1.0	0.0427	0.0	0.0
0.6	0.9965	0.0097	0.0008	0.0034	0.0	0.0343	0.0221	1.0	0.5808	0.0940	0.2186	0.4191
0.5	0.9965	0.0434	0.0002	0.0034	0.0	0.0365	0.0209	1.0	0.5549	0.0956	0.2260	0.4450
0.4	0.0	0.0888	0.0275	1.0	0.0	0.0369	0.0206	1.0	0.0	0.1696	0.2699	1.0
0.3	0.0	0.1894	0.0131	1.0	0.0	0.0497	0.0154	1.0	0.0	0.2186	0.2230	1.0
0.2	0.0	0.3902	0.0064	1.0	0.0	0.0497	0.0154	1.0	0.0	0.2186	0.2230	1.0
0.1	0.0	0.5318	0.0047	1.0	0.0	0.0607	0.0126	1.0	0.0	0.2186	0.2230	1.0

Table 6.11: Threshold selection for HBOS-CBLOF

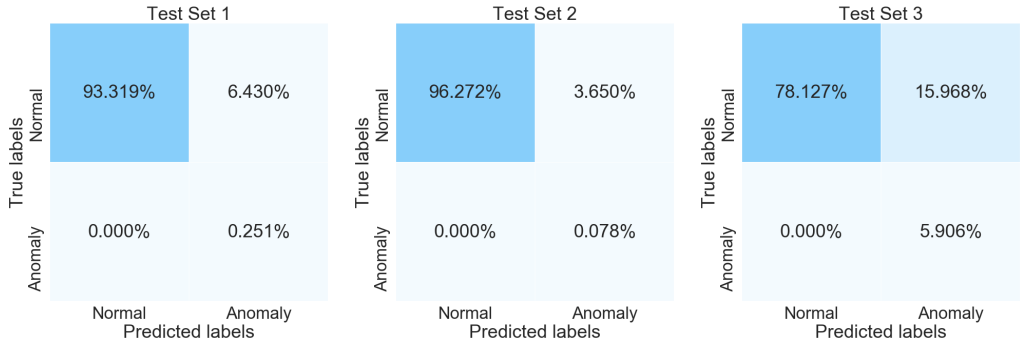


Figure 6.8: HBOS-CBLOF Confustion Matrices

IF+HBOS+CBLOF

The combination of all three selected methods shows the best results in comparison to all other evaluated methods. The optimal threshold for this combination again lies in the range [0.4 - 0.7]. The first data sets has a remaining false positive set of only around 5.5% of the total alerts. For the second data set the remaining false positives are also greatly reduced to a total of 3.7% as well as for the third data set where the remaining false positives are around 5%.

	Test Set 1				Test Set 2				Test Set 3			
Threshold	FNR	FPR	Precision	Recall	FNR	FPR	Precision	Recall	FNR	FPR	Precision	Recall
1	1.0	0.0	0.0	0.0	1.0	0.0	nan	0.0	1.0	0.0	0.0	0.0
0.9	1.0	0.0	0.0	0.0	0.35	0.0101	0.0476	0.65	1.0	0.0	0.0	0.0
0.8	1.0	0.0	0.0	0.0	0.35	0.0101	0.0476	0.65	1.0	0.0036	0.0	0.0
0.7	0.9965	0.0023	0.0036	0.0034	0.0	0.0116	0.0628	1.0	0.5808	0.0939	0.2187	0.4191
0.6	0.9965	0.0172	0.0005	0.0034	0.0	0.0365	0.0209	1.0	0.5549	0.0940	0.2290	0.4450
0.5	0.0	0.0543	0.0442	1.0	0.0	0.0369	0.0206	1.0	0.5019	0.1011	0.2360	0.4980
0.4	0.0	0.0940	0.0260	1.0	0.0	0.0497	0.0154	1.0	0.0	0.1566	0.2860	1.0
0.3	0.0	0.2497	0.0099	1.0	0.0	0.0607	0.0126	1.0	0.0	0.1748	0.2641	1.0
0.2	0.0	0.3349	0.0074	1.0	0.0	0.1028	0.0075	1.0	0.0	0.2186	0.2230	1.0
0.1	0.0	0.5535	0.0045	1.0	0.0	0.1028	0.0075	1.0	0.0	0.2186	0.2230	1.0

Table 6.12: Threshold selection for Combined Method

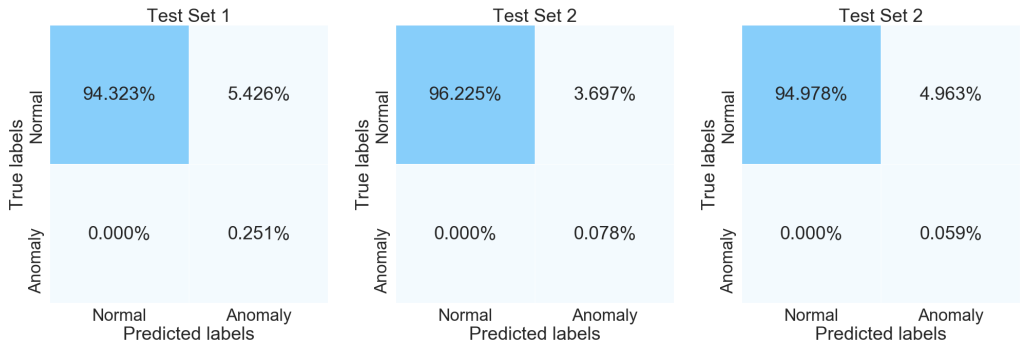


Figure 6.9: Results of the clustering and anomaly detection Combined Scores

The results obtained from this method outperform the rest of the evaluated methods. The conclusion that we can draw from these results is that all three methods are contributing to the normalisation of the scores for the false positives. Each method is complementing the other two and the false positives are normalised significantly by combining all three methods together. The remaining false positives represents events that have very similar values for the chosen attributes as the true positives.

6.4. Evaluation of model

The results of the models presented in the previous section are promising. However, for these models to be applicable in real production environments they need to be robust and adaptable to changes in these environments. Often this is not taken into consideration when evaluating machine learning models and therefore it is unclear to what extent many of the proposed solutions can be used.

Historic data for retraining

An important aspect to take into consideration is the minimal historic data that needs to be stored for training the models. In this evaluation only the first data set will be used since it contains data spanning over approximately 4 weeks of alerts generated from real network traffic. In the previous evaluation of the methods, the training size for this data set was 3/4 of the total size. At this setting the models showed a very good performance. However, environments may change more frequently. In order to determine the optimal amount of historical data needed for training accurate models several experiments are conducted with different time ranges and combinations of days (weekend and weekdays) in both the train and test data sets.

According to the results that are obtained from this experiment as shown in Figure 6.13, the optimal amount of data that needs to be considered for retraining the models is data collected from the 7 previous days. Training on only

one workday and predicting on one workday already shows good results. However, from the traffic patterns that have been observed in the different days during a week it can be seen that these have minor differences between them. Retraining the model on 1 day of historical data might not be representative enough for the following days. These cases are denoted with the letters A and B in the table. Additionally, when the data is trained only on a day from the weekend the performance for filtering false alarms during the working week significantly decreases. In the table these are the cases denoted with K, L, M and N in Table 6.13. An example would be the case when there is a major change in the environment on a Monday morning and the model is set to retrain on the last day of observed traffic. The model would be retrained on traffic from the previous Sunday and this would cause a bad performance triggering the model to retrain again until its performance has sufficiently increased. The optimal choice based on the good performance and the discussed points is therefore to retrain on data of the last 7 days such that traffic observed during all days of the week will be included. In Table 6.13 these are the cases denoted with Q and R.

Table 6.13: The table shows experiments with the prototype on test set 1. The performance of the prototype is tested on different values for the training phase to establish an indication of what the minimum historical data and the type of day should be for an effective model. The percentages represent the remaining amount of data received as input by the prototype. The last column (ALL) represents the combined score of all three techniques (HBOS, CBLOF and IF).

Case	Train		Test		Input Size	Output as % of Input						
	#Days	Type	#Days	Type		HBOS	CBLOF	IF	(IF-HBOS)	(IF-CBLOF)	(HBOS-CBLOF)	ALL
A	1	Workday	1	Workday	47896	23.9%	25.0%	11.3%	8.0%	37.0%	14.9%	9.4%
B	1	Workday	1	Weekend	7008	60.9%	62.1%	1.38%	1.96%	62.6%	60.6&	2.71%
C	2	Workday	1	Workday	47896	20.6%	6.09%	11.2%	5.51%	9.82%	3.40%	3.84%
D	2	Workday	1	Weekend	7008	38.1%	1.38%	2.73%	10.1%	0.64%	19.4%	1.81%
E	3	Workday	1	Workday	47896	20.4%	6.09%	11.8%	4.77%	10.1%	2.23%	1.87%
F	3	Workday	1	Weekend	7008	38.1%	1.38%	2.73%	2.31%	0.64%	6.17%	1.72%
G	4	Workday	1	Workday	47896	20.1%	7.59%	11.7%	5.80%	7.37%	2.65%	3.50%
H	4	Workday	1	Weekend	7008	38.1%	1.38%	2.73%	23.8%	0.64%	9.74%	1.72%
I	5	Workday	1	Workday	47896	19.7%	6.80%	11.3%	7.39%	13.11%	6.66%	4.71%
J	5	Workday	1	Weekend	7008	38.12%	1.55%	1.99%	2.31%	1.55%	3.22%	1.96%
K	1	Weekend	1	Workday	47896	73.6%	87.2%	87.2%	71.3%	75.1%	25.8%	27.82%
L	1	Weekend	1	Weekend	7008	37.9%	24.7%	24.7%	21.0%	21.4%	18.1%	19.5%
M	2	Weekend	1	Workday	47896	73.2%	74.6%	85.7%	69.3%	73.0%	23.5%	32.3%
N	2	Weekend	1	Weekend	7008	29.7%	15.9%	11.4%	9.2%	3.65%	2.63%	3.22%
O	2	Both	1	Workday	47896	23.0%	18.9%	10.2%	9.58%	34.4%	6.81%	7.10%
P	2	Both	1	Weekend	7008	37.3%	4.56%	2.73%	23.8%	6.43%	19.6%	1.99%
Q	7	Both	1	Workday	47896	19.7%	9.82%	10.2%	3.69%	9.70%	3.20%	3.08%
R	7	Both	1	Weekend	7008	38.1%	1.38%	2.73%	6.42%	0.64%	1.79%	1.54%

Model robustness and retraining

For the model to remain effective it needs to be retrained after some time. The frequency for retraining is difficult to set up front since the changes in large networks are often unpredictable. Therefore it is important to define a measure that will trigger the model to retrain itself whenever its effectiveness decreases as a result to new traffic patterns. To do this in a dynamic way the threshold needs to be computed dynamically as well. From the previous experiments it is decided that 7 days of historic data are enough for an accurate model and that the combination of all three models has the best performance overall. Therefore, the following experiments will be conducted with these settings. In these experiments the data is considered on a daily basis.

To evaluate how robust the chosen model is, noise is added to the data sequentially increasing between 1% and 100% of the total input size which represents the total number of alerts produced by the NIDS. Since the model is unsupervised and has no contextual information about the environment this can be done using the input and output size of the set of alerts. When the size of the filtered alert set reaches a predefined threshold compared to the size of the total alerts produced by the NIDS, this might be an indication that the model needs to be retrained. The way that the model knows when to retrain is by computing the threshold with the following formula where A_Output represents the size of the filtered alert set and A_Input represents the size of the original alert set:

$$A_{Output} >= A_{Input} * t \tag{6.5}$$

From the Table 6.14 the optimal threshold should be set at most at 10%. This way the model will still perform reasonably well with a low false positive rate and include novel knowledge only when this threshold is passed. This threshold is chosen by taking into consideration the specificity and recall values. The specificity is an indicator of the false alarms that are correctly filtered by the algorithm which in both cases drops below 0.9 when this threshold is passed. The recall value indicates whether all malicious cases are detected correctly and this remains 1.0 regardless of the threshold.

With this threshold the model should not miss any true positives and still reduce the number of false positives significantly. This threshold range is chosen based on the results of the previous models. The limitation to this approach is the case where the new traffic is malicious traffic but of a large proportion, for example from a DDoS attempt or an aggressive scan. In this case, the possibility exists that the model will be retrained on the malicious examples and wrongly classify them as benign, if these are not filtered by the algorithm (false negatives). Another option could be to consider the size of the alert set observed over a longer time for a more robust range of values.

Table 6.14: In this figure a subset of the data is selected which represents one Workday. Noise is added sequentially in proportion to the size of the data to evaluate how robust the model is. The models are trained on one full week (7 days) including both workdays and weekends. The scores of the combination IF+ HBOS + CBLOF are shown in the table.

Noise %	Input Size	% of Input	FNR	FPR	Precision	Recall	Specificity
None	47896	3%	0.0	0.0268	0.1326	1.0	0.9732
1%	48374	3.7%	0.0	0.0340	0.1068	1.0	0.9660
10%	52685	9.8%	0.0	0.0955	0.0376	1.0	0.9045
20%	57475	15.%	0.0	0.1509	0.0221	1.0	0.9045
30%	62264	18%	0.0	0.1781	0.0174	1.0	0.8219
50%	71844	24%	0.0	0.2475	0.0109	1.0	0.7525
70%	81423	30%	0.0	0.3004	0.0079	1.0	0.6996
100%	95792	6%	0.0	0.3626	0.0056	1.0	0.6374

Table 6.15: In this figure a subset of the data is selected which represents one day of a Weekend. Noise is added sequentially in proportion to the size of the data to evaluate how robust the model is. The models are trained on one full week (7 days) including both workdays and weekends. The precision in this case is 0 because there are no attacks present in this subset of data.

Noise %	Input Size	% of Total	FNR	FPR	Precision	Recall	Specificity
None	7008	1.5%	nan	0.0154	nan	1.0	0.9846
1%	7078	9.5%	nan	0.0190	nan	1.0	0.9810
10%	7708	8.7%	nan	0.0874	nan	1.0	0.9126
20%	8409	14%	nan	0.1493	nan	1.0	0.8507
30%	9110	19%	nan	0.1927	nan	1.0	0.8073
50%	10512	26%	nan	0.2642	nan	1.0	0.7358
70%	11913	33%	nan	0.3316	nan	1.0	0.6684
100%	14016	47%	nan	0.4031	nan	1.0	0.5969

6.5. Discussion

From the results obtained by analysing the causes and possibilities to reduce false alarms a few conclusions can be drawn. Firstly, manually deactivating rules can be very complicated and may still not provide satisfactory results. The activation and deactivation of rules needs to be constantly assessed and regularly adapted to changes that may occur in the environment. Additionally, the remaining set will probably contain a significant amount of remaining false alarms. Since this approach is rather vague and time consuming, it increases the risk of missing important events and taking into consideration that the amount of remaining false alarms is still of significant size, this approach is not considered to be effective.

From the results obtained by applying the unsupervised approach to alarms collected on a real dynamic network several important observations are made. The first observation obtained from the results of the data analysis and feature selection methods is that the majority of false alarms can be represented by a set of attributes. These are the signature, destination port and destination address of the alert. The next important finding resulting from the experiments is that by setting a suitable threshold all true positives are correctly identified. However, not all events are represented sufficiently. As discussed in Section 6.3 this is due to the similar values shared with some of the true alarm's attributes. In some cases, even when the values are not very similar the algorithms compute similar scores for these points for which an explanation could be the rare combination of attribute values.

Furthermore, this approach is tested on multiple data sets. Two collected on a real network and one artificially generated data set. A general observation for all the attempted methods is that these did not perform equally well on the real and artificial data sets. This may be due to the fact that the attributes are chosen based on the real data and because the artificial data set fails to mimic regular network behaviour sufficiently. Even in this case, by setting a sufficiently large threshold, all anomalous events were detected. The LOF algorithm has failed to correctly identify the true alarms. Even with different threshold settings this method computed similar scores for the anomalous points. For both data sets, the combination of all three anomaly detection methods does find all known true positives in the data. The false positives are reduced with at least 90% of the total false alarms generated by the NIDS. A general conclusion that can be drawn from these results is that overall the combination of all 3 algorithms has the best performance. However, the standalone performance of Isolation Forest is also close in performance for all three data sets.
When examining the results obtained from the experiments with different training data settings the Isolation Forest and HBOS methods both have a very good performance in all considered cases. The Recall of these methods is always 100%

while the Specificity varies per case. This is not the case for the CBLOF method. As shown in the Figure 3 this method fails to correctly cluster the anomalous points leading to missing true positives alarms when not trained on sufficiently rich data. The performance for the different combinations varied per case where the performance of the CBLOF method negatively influenced the overall performance of these combinations.

Based on these findings the most optimal amount of training data should contain alert patterns observed during the whole week. In this scenario, when trained on 7 days, the combination of all three methods had the best results with a Recall of 100% and Specificity above 97%. Additionally, the applied algorithms are computationally efficient and in this scenario the calculation of scores is done in seconds. It is therefore a realistic option for this method to be applied in parallel with an intrusion detection system in a stream-like scenario where the model would be updated in real-time with newly observed regular events. This is a point for further research. The remaining alerts that are not filtered out by this method are events with frequencies close to 0. In these cases the infrequent attribute is usually the destination IP. As explained in the previous sections there are multiple reasons that these specific alarms might be depicted wrongly by the chosen attributes such as new destination addresses assigned through DHCP which could potentially be mitigated by a different encoding.

The main difference between the proposed unsupervised methods and the method proposed in this thesis is in the approach that is taken to tackle the problem. The reviewed works try to build profiles of normal sensor behaviour based on time dependent techniques such as Markov Chains and time-series analysis while only considering one variable which is the rule that has triggered the event [72, 79]. In the case of the time-series analysis the solution is framed as uni-variate problems where for each rule a new model needs to be trained and maintained. In the case of the sequential pattern mining technique only a deviating sequence of consecutive signatures is considered as worth investigating. In our case, the model is the same for all events and can handle multivariate data which makes it possible to take into consideration multiple attributes of an alert that could be indicators of true/false alarms such as the source and destination address. The association rule mining technique [73] has the possibility to consider multiple attributes and is therefore more similar to our approach. The main difference here is the complexity of the method and historic data needed to train accurate models. An accurate model needs a minimum of 30 days of alert data and two separate classifiers trained on different time spans in order to capture the frequency patterns of more events. Additionally, the model consists of several different phases where each phase requires a set of parameters to be set such as the minimum support, number of clusters and threshold. Since there is a time constraint for an item to be considered frequent, these steps need to be repeated

very frequently which makes the computational cost high. In our case, all experiments are conducted with a default set of parameters and the model needs only to be retrained when an increase of alerts occurs. This is an indication of a change in the network to which the model needs to be adjusted.

Table 6.16: This table compares the method proposed in this thesis with other similar works. The main characteristics of the works are included in the table for comparison. The accuracy of the various works is not included since this is difficult to compare because of the different data sets and environment settings that are used.

	Finding The Needle: Suppression of False Alarms in Large Intrusion Detection Data Sets [72]	Network IDS Alert Classification with Frequent Itemset Mining and Data Clustering [73]	Processing Intrusion Detection Alert Aggregates with Time Series Modelling [79]	An Unsupervised Approach for False Alarm Filtering in Rule-based NIDS
Technique	Hidden Markov Models	Frequent Itemset Mining	Timeseries Analysis	Outlier Detection (IF/CBLOF/HBOS)
Cost	$\mathcal{O}(k^2n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$, $\mathcal{O}(n)$, $\mathcal{O}(n)$
Dataset	Private	Private	Private	Artificial/Private
Data Description	No	Partially	Yes	Yes
Pre-configured NIDS	Yes	Yes	Not specified	No
Dataset Size	30 Days	8 Weeks	43 Days	4 Weeks, 5 Days, 1 Day
New alarms in testset	No	Not specified	Not specified	Yes
Historical Data	5 Days	2 Weeks & 30 Days	20 Minutes	7 Days
Number of models per sensor	1	2	# Unique rules	1
Dimension	One	Multiple	One	Multiple
Number of parameters to choose	2	8	4	1
Updating	Not specified	Every 24 h	Every 20 min	Upon trigger

7

Limitations and Future Work

In this chapter some notes on the limitations of this approach will be stated as well as ideas for further research on this topic.

7.1. Limitations

The main limitation of this system is that it cannot differentiate between false positives and true attacks during the training phase. In the case that there will be true positives present in the training phase this behaviour will be learned as regular. This is certainly a big limitation in the case where an infiltrator is already present in the network for a longer period of time and periodically generates the same types of traffic. These regular malicious alarms will be considered benign and may be filtered out and remain undetected.

The second limitation of this system is the inability to distinguish successful attacks from unsuccessfully attempts. The unsupervised approach is based on the frequencies of the features that are chosen. Therefore, context is not taken into consideration which in some cases might be important in order to distinguish a true positive from a false alarm. To take context into consideration, an additional filter can be applied on the output of this prototype using a knowledge-based approach. However, the linking of alerts and contextual information is shown to be very challenging. From the attempts made to correlate vulnerabilities with the NIDS output we have seen that the results are unsatisfactory, meaning the false positive rate remains very high and that even some true positives might be labelled with low priority or discarded with this approach. The other issue with this approach is the mismatch between the rules that have references to existing vulnerabilities and the output of a vulnerability scanner, referencing the detected vulnerabilities. In both cases, the majority of data that is produced is not referenced with any type of vulnerability. Considering additional information about the network such open ports, operating systems and running services also did not

provide satisfactory results.

From the results that are obtained using this systems there still remains a small subset of alerts of false alarms. These are alerts that are infrequent and deviating and therefore are considered as anomalies by the chosen algorithms. This is a limitation since events that are benign but rare will also be flagged as anomalous and are required to be further investigated by analysts. This approach is thus good at filtering false positives that are generated as a result of benign and regularly repeating processes of the monitored environment but fails to filter out rare false positive events.

The generalizability of the results is limited by the environment in which the NIDS is placed and the type of data that is being captured by the NIDS. The data that is used during the experiments is not generalizable to all corporate networks. However, it is very likely that the data used here will be similar to data sets from other corporate networks where the majority of traffic by the employees and by automated processes that is captured by the NIDS represents benign events. Since the proposed method is tested on both real network data and synthetically generated network data, where each case depicts a different network environment it can be applicable for similar networks to these. However, the extent to which this method would be generally applicable to any network needs to be further evaluated with different data sets depicting various network environments. The results could be affected by some configuration choices such as the frequency with which the internal addresses get reassigned or the ratio between true and false positives that the NIDS generates.

In this work the data that is used represents the output of a rule-based network intrusion detection system. To reduce the false alarms, only the standard set of attributes of an alert set are used. Therefore, the method that is applied is generalizable to any network with any rule-based intrusion detection systems to perform the filtering operations on the produced alert set.

7.2. Future Work

The system developed during this research shown promising results for the automatic filtering of false alarms. There are however some limitations as discussed in the previous section. Some ideas for improvement or further development of the system are discussed in this section.

Another possibility is to train separate model for a workday and a weekend. This could increase performance and accuracy overall since there are signatures observed during the weekend that are normal, but are not present during the workdays and thus have a lower frequency. If this is considered separately, then the ratio between the different expected signatures on a weekend and on a workday would be represented better. From the experiments conducted on the col-

lected data this did not result in significant changes to the effectiveness of the model, however, in different environment it might make a difference.

In future work, this method may be applied to different networks to evaluate its applicability better. The proposed model can also be tested in a streaming scenario where the algorithm will be updated with new benign events in (near) real-time.

A different set of features might improve the results. New features can be constructed that give more information about the different relationships that certain values have to further reduce the false positives by better representing the rare but benign events. For example the number of unique ports per unique IP address or the relative frequencies of unique connections between source and destination addresses and the relative frequencies of unique signature and destination IP/-Port pairs might better represent these rare events. Also, the IP addresses can be encoded differently to better represent to which (sub)network they belong to.

8

Conclusion

The goal of this research was to explore the possibilities of false alarm filtering in an automated manner. In order to do this real network data was analysed and unsupervised methods were applied. The main intuition behind this approach is that the majority of alerts generated by intrusion detection systems are false positives. In such a set of alerts, since the real attacks only form a small subset of the total data, these can be distinguished from the majority of points by their deviating attribute values. Four different unsupervised anomaly detection algorithms were chosen for this task. The performance of the chosen algorithms is evaluated and these are then combined together to further reduce the number of false positives. Experiments are conducted with different combinations of data for the training phase in order to select an optimal amount of historical data that needs to be stored for retraining the model. Furthermore, the robustness of the proposed approach is tested by adding noise to the data to set a suitable threshold for automatically retraining the system when the number of false alarms increases due to changes in the monitored environment. The main research question and sub-questions as defined in Chapter 1 are answered below.

RQ1: What are the existing techniques for false alarm reduction in rule-based NIDS?

To answer this question a literature review is conducted where the different methods that are proposed to tackle this problem are explored. Different techniques and elements that are used in the proposed solutions are reviewed. The majority of works propose a system that used external knowledge obtained mainly from the monitored environment in order to correlate this to the generated alerts. The goal of these approaches is to deduct which alerts are certain false positives by analysing whether the affected hosts are vulnerable to the attack that triggered the alert. The main elements used in these approaches are the vulnerability data obtained from the network and additional information such as open ports,

services and operating systems. Furthermore, some approaches additionally construct possible attack scenarios from the collected data. The subset of works which use data mining techniques to reduce the false alarms generally use the signature of the alert as main attribute or a combination of the signature, source address, destination address and timestamp of the alert. The main limitation to these works is the challenge to integrate these solutions into a production environment. This is a challenge since these works rely on selecting a suitable set of hyper-parameters that will work in that specific environment and are computationally intensive because of the underlying algorithms that are applied and the frequency with which they are updated.

RQ2: How effective are existing network intrusion detection systems when deployed without any prior alterations to their settings?

To answer this question a NIDS was set up with default settings and public rules were used with their default settings as well. The NIDS monitored traffic from a portion of a network based in a large and dynamic organisation. The data used to answer this question is the resulting alert set. The alerts were analysed and the true positives were labelled for the purpose of evaluating the effectiveness of this setup. The false positives produced by the NIDS in its default setting were approximately 98% of the total alert set as collected over 4 weeks. With this information the answer to this question is that intrusion detection systems are not very effective when not set up properly and continuously tweaked to the monitored environment. For organisations that do not have sufficient time and resources to do this, setting up a custom NIDS solution is a challenging task. Tuning the NIDS configuration requires detailed knowledge of the environment and needs to be constantly adapted to changes that affect the results.

RQ3: Which unsupervised outlier detection techniques can be used for filtering false alarms in rule-based NIDS?

To answer this question several unsupervised anomaly detection algorithms were chosen. Four different techniques were selected for the experiments in order to assess which of these algorithms has the best performance. The algorithms were selected based on recommendations in literature and the application of these algorithms for detecting anomalies in network and other log data. The selected algorithms are Local Outlier Factor, Isolation Forest, Histogram-based Outlier Score and Cluster-based Local Outlier Factor. The algorithms were applied with their default hyper-parameter values in order to assess whether it is possible to avoid manual adjustments to these parameters based on the environment. The only parameter that was evaluated on a range of values was the contamination threshold because of the different portions of true alarms that may be triggered in a network which might vary per environment. The Local Outlier Factor algorithm was the only algorithm that failed to detect all relevant cases. The

Cluster-based Local Outlier Factor detected only a subset of the true alarms in the experiments with different amounts of historic data. This method was more sensitive to the composition of the training data. The Histogram-based Outlier Score and Isolation Forest methods performed well as both standalone methods and in the combined cases. The overall best performing standalone algorithms was Isolation Forest. The remaining false positive alarms remained low through all experiments.

RQ4: How can the model be retrained in order to remain accurate upon changes in the network ?

To answer this question several experiments were conducted on the collected data. The first set of experiments was aimed to define the optimal amount of historical data needed to train accurate models. The second set of experiments was aimed to assess when the model needs to be retrained because of new patterns and derive a threshold which triggers the model to retrain. Firstly, The amount of historical data needed to train accurate model is set to 7 days. From the different scenarios we see that this model performs well even with a smaller training set, but the choice to include all 7 days is made in order to not miss regular traffic patterns that are specific to a single day. Secondly, a metric is derived to act as a trigger for the system to retrain its algorithms based on the most recent available data. The metric is derived by adding noise that mimics new traffic patterns and evaluating its performance in terms of correctly identified true and false alarms. By adding more noise sequentially, it is evident that the amount of false alarms also increases. The daily original alert size is compared to the filtered alert size to measure whether the model filters out the majority of alerts. When a certain threshold is crossed the models knows it needs to retrain on more recent data.

Main RQ: To what extent can existing rule-based NIDS be effectively deployed by automating the process of false alarm filtering using unsupervised anomaly detection techniques?

From the evaluated unsupervised anomaly detection techniques it can be concluded that the number of false alarms can be greatly reduced in a fully automated manner. The proposed methodology has potential to be successfully applied in production environments, hereby not requiring any manual alterations to the NIDS. The selected anomaly detection techniques are at most linear in terms of complexity and when combined reduce the false alarms with more than 90%. The requirement of retraining the models on more recent and representative data can also be effectively automated by measuring the set of filtered alarms as opposed to the total alarms generated by the NIDS. The main shortcoming of this method is the lack of contextual information. This information may be of importance for the analysts to analyse the remaining alerts. Additionally, since the filtering is based on the frequencies of the attributes it may not be straightforward to ex-

plain why something is classified as regular or deviating. There is a remaining subset of false positives which this method is not able to filter out. The reason for this are the similar attribute values shared with the malicious events. These issues may be tackled with a different selection of features or a different encoding.

In conclusion, this approach has a good potential of being used in real production environments by a range of different user groups since it operates in an unsupervised and automated manner. However, the shortcomings need to be taken into consideration and further explored. Additionally, this method needs to be evaluated on more data sets collected in real environments and ideally tested in a production environment where batches of alerts will be filtered in a streaming like scenario.

Appendix A

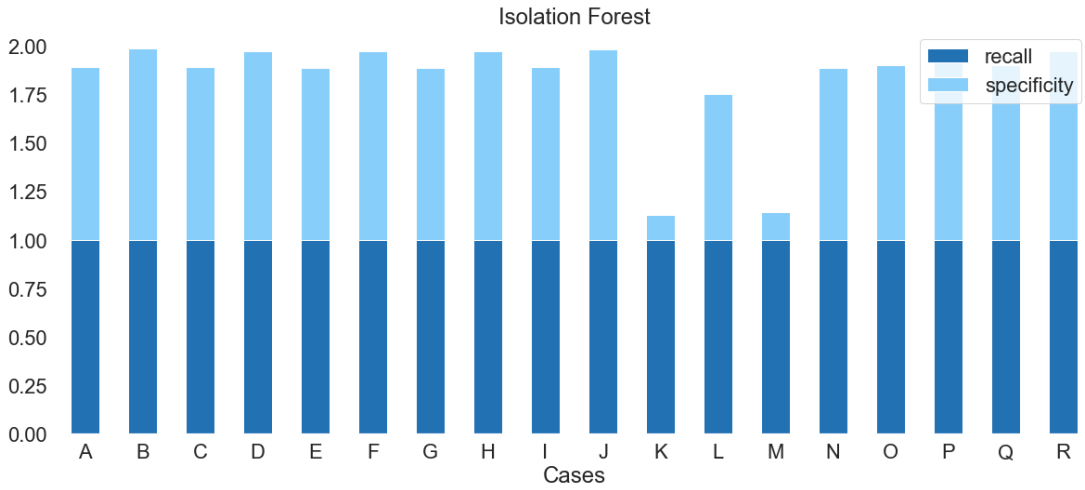


Figure 1: Results with Isolation Forest for cases A to R

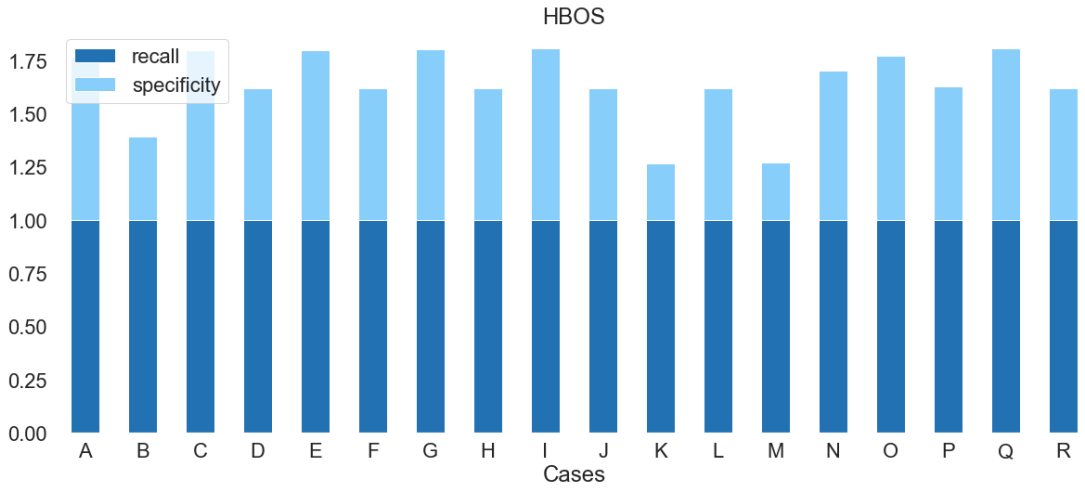


Figure 2: Results with HBOS for cases A to R

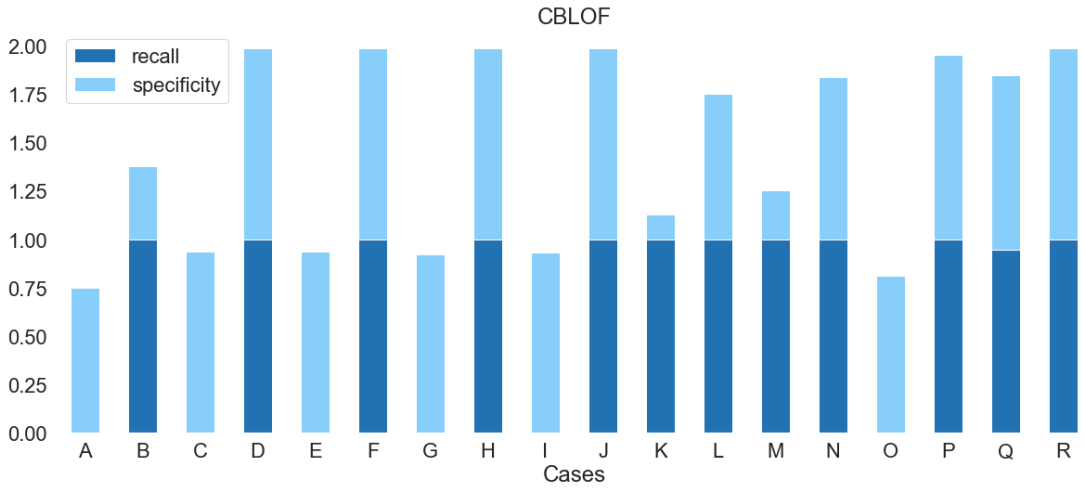


Figure 3: Results with CBLOF for cases A to R

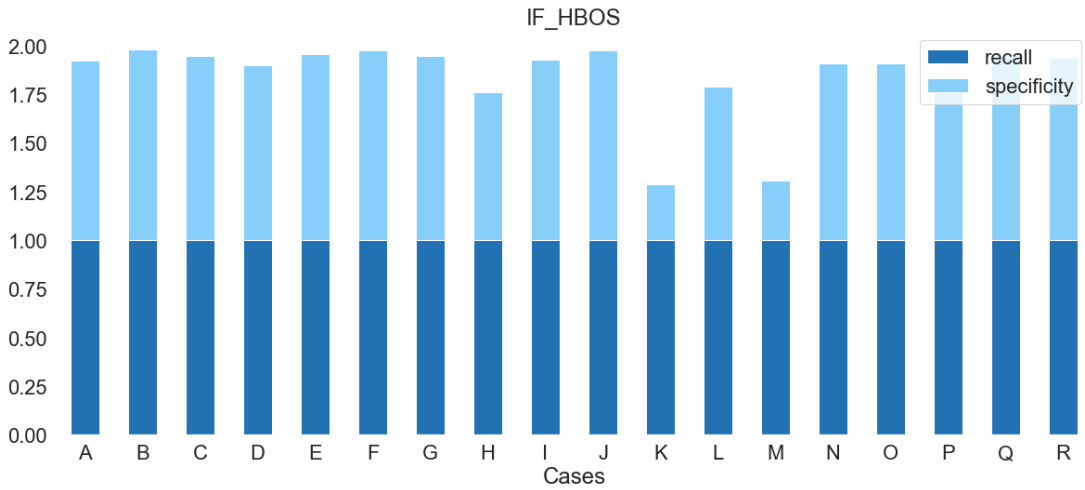


Figure 4: Results with IF-HBOS for cases A to R

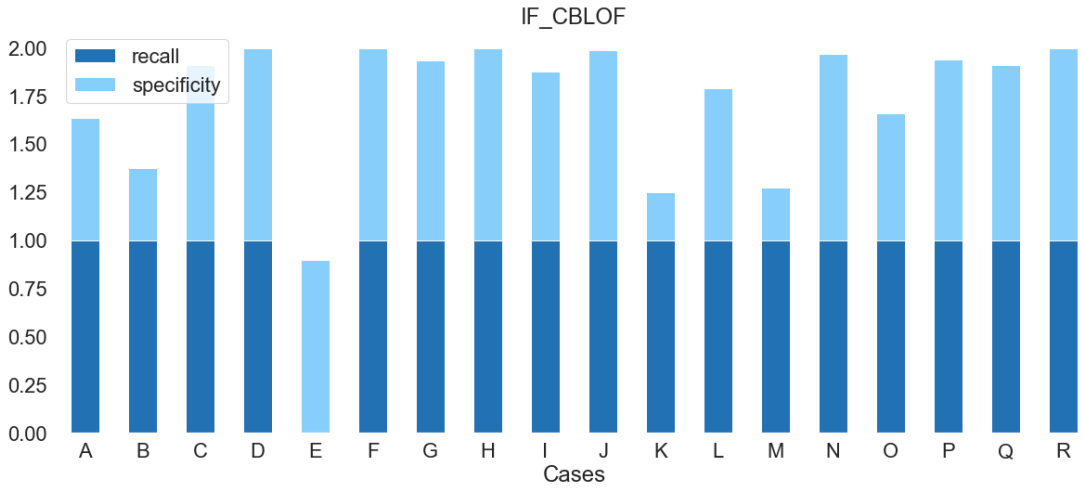


Figure 5: Results with IF-CBLOF for cases A to R

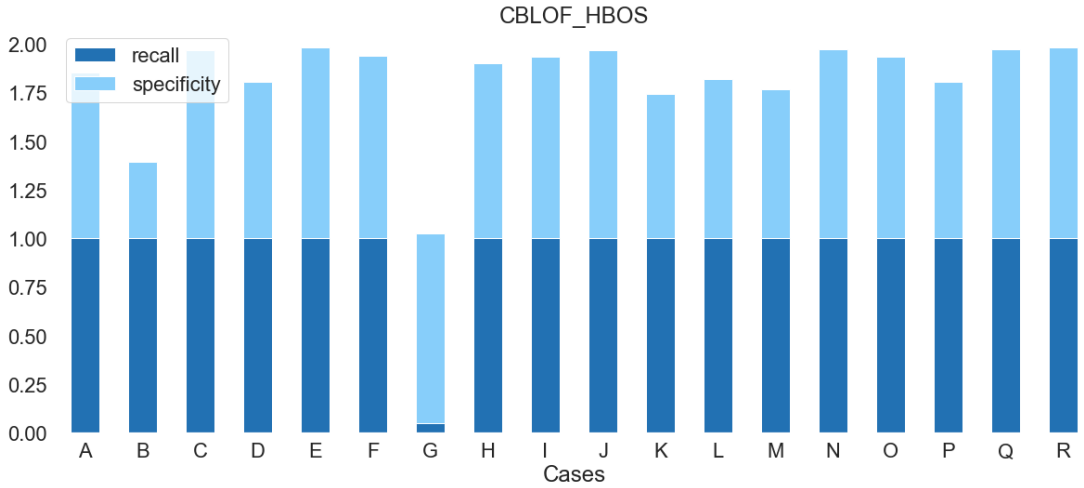


Figure 6: Results with HBOS-CBLOF for cases A to R

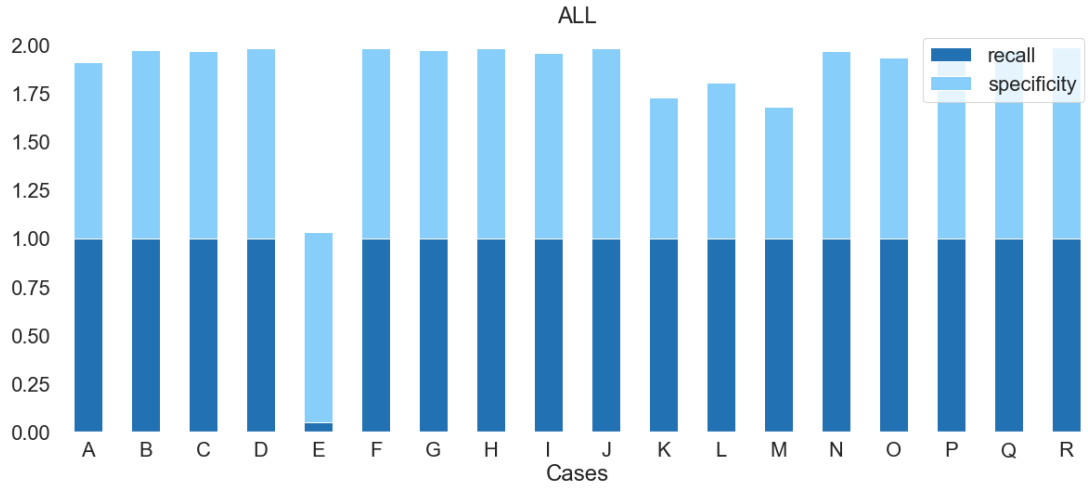


Figure 7: Results with All methods combined for cases A to R

Appendix B

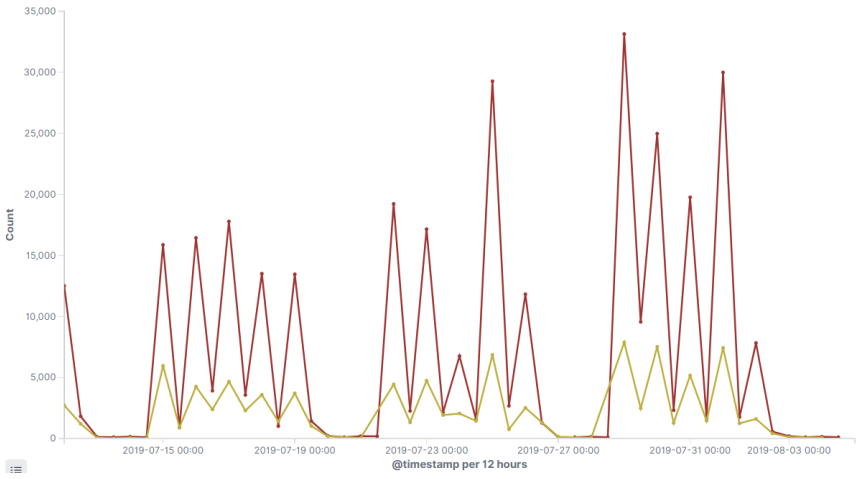


Figure 8: Signature Correlations

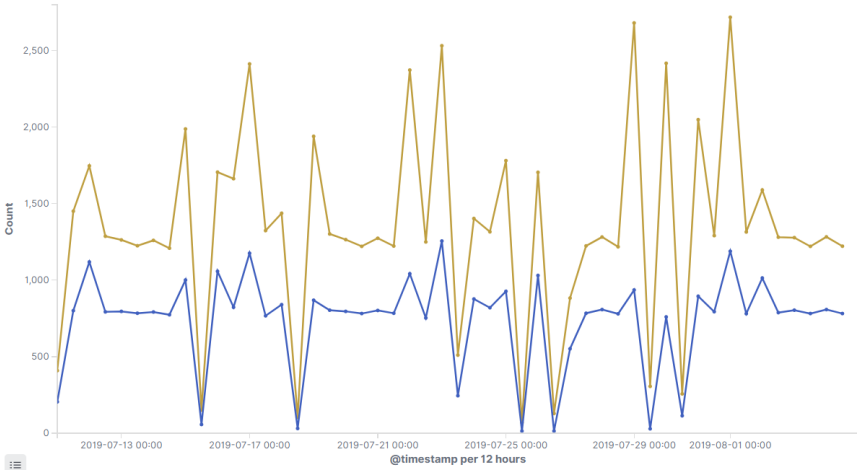
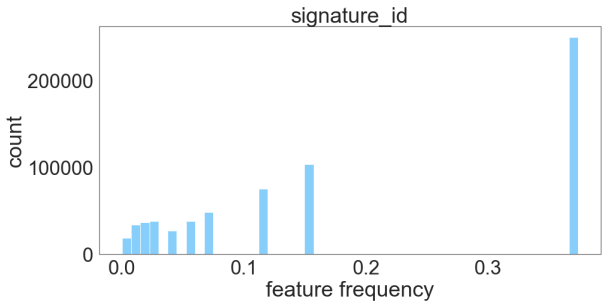
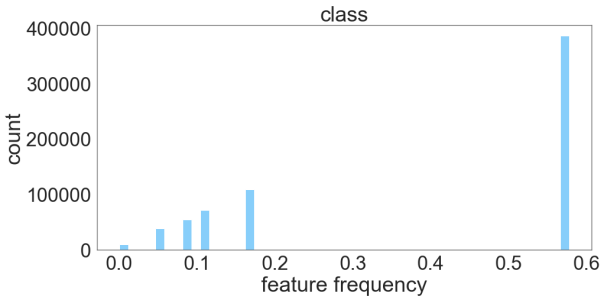
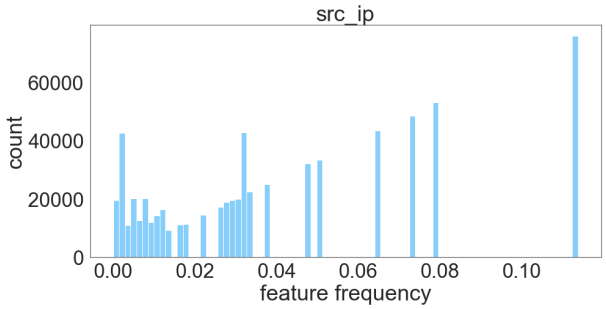


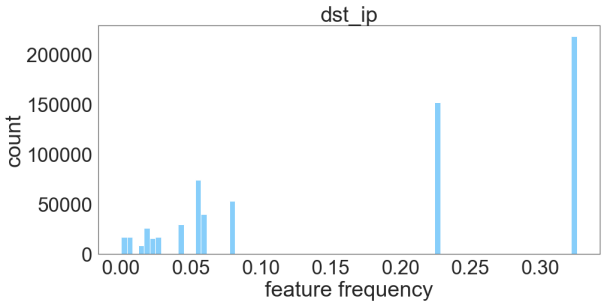
Figure 9: Signature Correlations

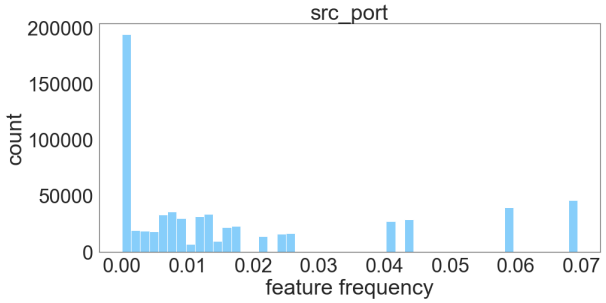
Appendix C

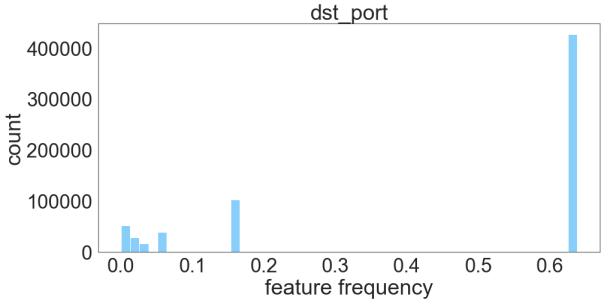

signature_id
count
200000
100000
0
0.0
0.1
0.2
0.3
feature frequency

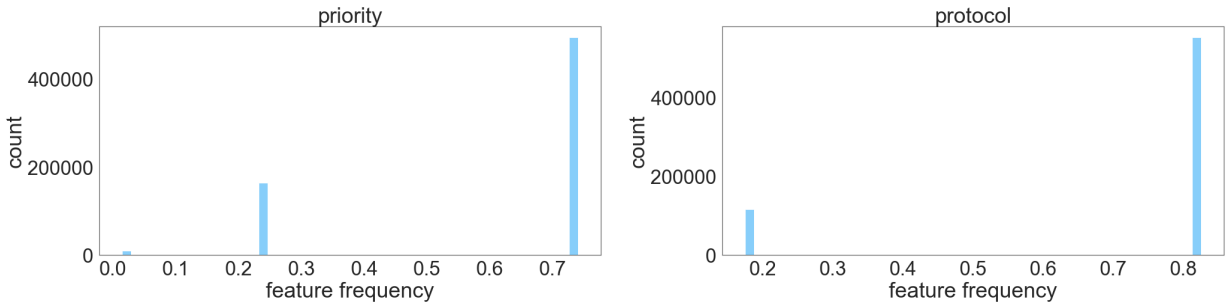

class
count
400000
300000
200000
100000
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
feature frequency


src_ip
count
60000
40000
20000
0
0.00
0.02
0.04
0.06
0.08
0.10
feature frequency


dst_ip
count
200000
150000
100000
50000
0
0.00
0.05
0.10
0.15
0.20
0.25
0.30
feature frequency


src_port
count
200000
150000
100000
50000
0
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
feature frequency


dst_port
count
400000
300000
200000
100000
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
feature frequency



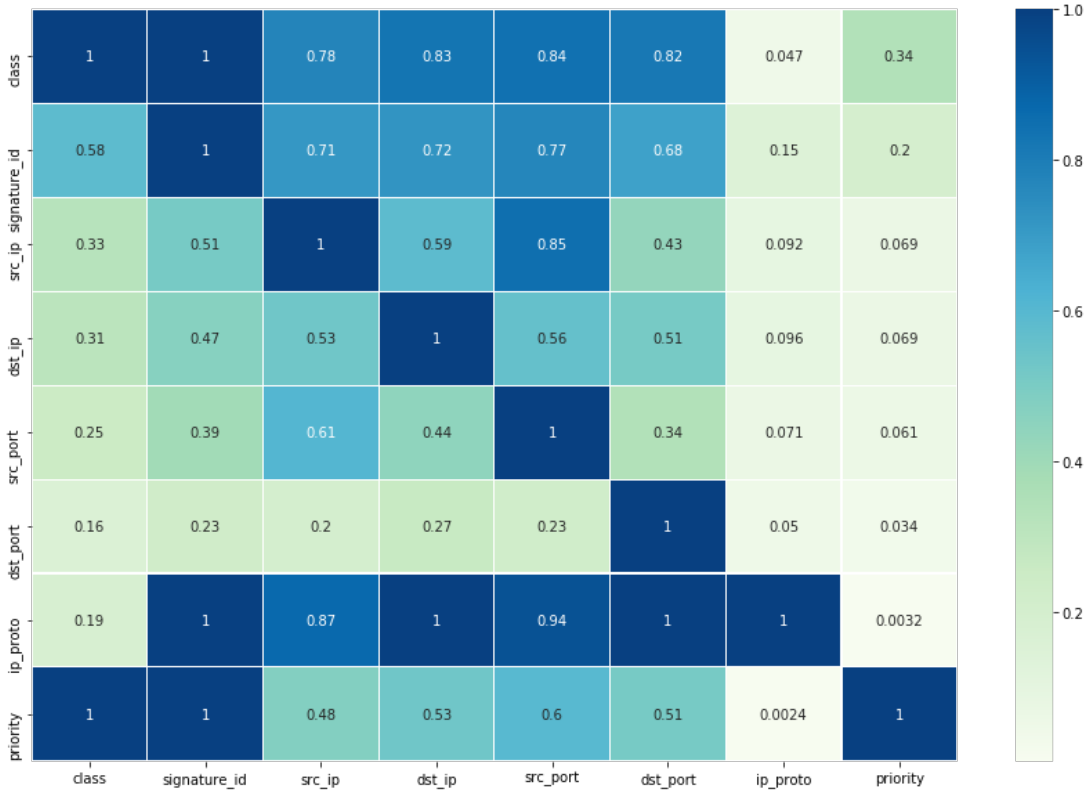


Figure 14: Correlation Matrix between the attributes of the alert data.

Bibliography

[1] 0Patch. Security patching is hard. https://0patch.com/files/SecurityPatchingIsHard_2017.pdf, 2017.

[2] Charu C Aggarwal. Outlier ensembles: position paper. ACM SIGKDD Explorations Newsletter, 14(2):49–58, 2013. doi: 10.1145/2481244.2481252.

[3] Charu C Aggarwal. Outlier analysis. In Data mining, pages 237–263. Springer, 2015. doi: 10.1007/978-3-319-47578-3.

[4] M. Ahmed and A. N. Mahmood. Network traffic analysis based on collective anomaly detection. In 2014 9th IEEE Conference on Industrial Electronics and Applications, pages 1141–1146, June 2014. doi: 10.1109/ICIEA.2014.6931337.

[5] Mohiuddin Ahmed and Abdun Naser Mahmood. Novel approach for network traffic pattern analysis using clustering-based collective anomaly detection. Annals of Data Science, 2(1):111–130, 2015. doi: 10.1007/s40745-015-0035-y.

[6] James P Anderson. Computer security threat monitoring and surveillance. Anderson Co., Fort Washington, PA, 1980.

[7] Stefan Axelsson. The base-rate fallacy and the difficulty of intrusion detection. ACM Transactions on Information and System Security (TISSEC), 3 (3):186–205, 2000.

[8] Longe Olumide Babatope, Lawal Babatunde, and Ibitola Ayobami. Strategic sensor placement for intrusion detection in network-based ids. International Journal of Intelligent Systems and Applications, 6(2):61, 2014.

[9] Luis Basora, Xavier Olive, and Thomas Dubot. Recent advances in anomaly detection methods applied to aviation. Aerospace, 6(11):117, 2019. doi: 10.3390/aerospace6110117.

[10] Damiano Bolzoni and Sandro Etalle. Aphrodite: An anomaly-based architecture for false positive reduction. arXiv preprint cs/0604026, 2006.

[11] Damiano Bolzoni, Sandro Etalle, and Pieter Hartel. Poseidon: a 2-tier anomaly-based network intrusion detection system. In Fourth IEEE International Workshop on Information Assurance (IWIA'06), pages 10–pp. IEEE, 2006.

[12] Markus Breunig, Hans-Peter Kriegel, Raymond Ng, and Joerg Sander. Lof: Identifying density-based local outliers. volume 29, pages 93–104, 06 2000. doi: 10.1145/342009.335388.

[13] Zhangyu Cheng, Chengming Zou, and Jianwei Dong. Outlier detection using isolation forest and local outlier factor. In Proceedings of the Conference on Research in Adaptive and Convergent Systems, pages 161–168, 2019. doi: 10.1145/3338840.3355641.

[14] David Cournapeau. A set of python modules for machine learning and data mining. URL https://pypi.org/project/sklearn/.

[15] Dorothy E Denning. An intrusion-detection model. IEEE Transactions on software engineering, (2):222–232, 1987.

[16] Rémi Domingues, Maurizio Filippone, Pietro Michiardi, and Jihane Zouaoui. A comparative evaluation of outlier detection algorithms: Experiments and analyses. Pattern Recognition, 74:406–421, 2018. doi: 10.1016/j.patcog.2017.09.037.

[17] Edgescan. Vulnerability statistics report. https://www.edgescan.com/wp-content/uploads/2018/05/edgescan-stats-report-2018.pdf, 2018.

[18] Edgescan. Vulnerability statistics report. https://www.edgescan.com/wp-content/uploads/2019/02/edgescan-Vulnerability-Stats-Report-2019.pdf, 2019.

[19] Christian Canales Sid Deshpande Elizabeth Kim, Ruggero Contu. Forecast: Information security, worldwide, 2016-2022. https://www.gartner.com/en/documents/3875867, 2020.

[20] Hans-Peter Kriegel Peer Kröger Erich and Schubert Arthur Zimek. Interpreting and unifying outlier scores. In 11th SIAM International Conference on Data Mining (SDM), Mesa, AZ, 2011. doi: 10.1.1.232.2719.

[21] Exploit Database. Offensive security's exploit database archive. https://www.exploit-db.com/.

[22] Filipe Falcão, Tommaso Zoppi, Caio Barbosa Viera Silva, Anderson Santos, Baldoino Fonseca, Andrea Ceccarelli, and Andrea Bondavalli. Quantitative comparison of unsupervised anomaly detection algorithms for intrusion detection. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, pages 318–327, 2019. doi: 10.1145/3297280.3297314.

[23] Eric Falk, Ramino Camino, Radu State, and Vijay K Gurbani. On non-parametric models for detecting outages in the mobile network. In

2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pages 1139–1142. IEEE, 2017. doi: 10.23919/INM.2017.7987448.

[24] Markus Goldstein and Andreas Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. KI-2012: Poster and Demo Track, pages 59–63, 2012.

[25] Fatma Hachmi, Khadouja Boujenfa, and Mohamed Limam. Enhancing the accuracy of intrusion detection systems by reducing the rates of false positives and false negatives through multi-objective optimization. Journal of Network and Systems Management, 27:93–120, 2018.

[26] Black Hat Ethical Hacking. Microsoft warns this hacking group is targeting vulnerable web servers. https://www.blackhatethicalhacking.com/news/microsoft-warns-this-hacking-group-is-targeting-vulnerable-web-servers/, 2020.

[27] Ayush Hariharan, Ankit Gupta, and Trisha Pal. CAMLPAD: cybersecurity autonomous machine learning platform for anomaly detection. CoRR, abs/1907.10442, 2019.

[28] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. Pattern Recognition Letters, 24(9-10):1641–1650, 2003. doi: 10.1016/S0167-8655(03)00003-5.

[29] L Todd Heberlein, Gihan V Dias, Karl N Levitt, Biswanath Mukherjee, Jeff Wood, and David Wolber. A network security monitor. In Proceedings. 1990 IEEE Computer Society Symposium on Research in Security and Privacy, pages 296–304. IEEE, 1990. doi: 10.1109/RISP.1990.63859.

[30] Neminath Hubballi, Santosh Biswas, and Sukumar Nandi. Network specific false alarm reduction in intrusion detection system. Security and Communication Networks, 4(11):1339–1349, 2011.

[31] Qiu Hui and Wang Kun. Real-time network attack intention recognition algorithm. International Journal of Security and Its Applications, 10(4): 51–62, 2016.

[32] Ponemon Institute. Costs and consequences of gaps in vulnerability response. https://www.servicenow.com/lpayr/ponemon-vulnerability-survey.html, 2018.

[33] Omar Iraqi and Hanan El Bakkali. Application-level unsupervised outlier-based intrusion detection and prevention. Security and Communication Networks, 2019, 2019. doi: 10.1155/2019/8368473.

[34] Klaus Julisch. Using root cause analysis to handle intrusion detection alarms. PhD thesis, Universität Dortmund, 2003.

[35] Dimitar Karev, Christopher McCubbin, and Ruslan Vaulin. Cyber threat hunting through the use of an isolation forest. In Proceedings of the 18th International Conference on Computer Systems and Technologies, CompSysTech'17, page 163–170, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450352345. doi: 10.1145/3134302.3134319. URL https://doi.org/10.1145/3134302.3134319.

[36] Christopher Kruegel, William Robertson, and Giovanni Vigna. Using alert verification to identify successful intrusion attempts. Praxis der Informationsverarbeitung und Kommunikation, 27(4):219–227, 2004.

[37] Aleksandar Lazarevic, Levent Ertoz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In Proceedings of the 2003 SIAM international conference on data mining, pages 25–36. SIAM, 2003. doi: 10.1137/1.9781611972733.3.

[38] Wenke Lee and Salvatore J Stolfo. A framework for constructing features and models for intrusion detection systems. ACM transactions on Information and system security (TiSSEC), 3(4):227–261, 2000.

[39] Fei Tony Liu, Kai Ting, and Zhi-Hua Zhou. Isolation forest. pages 413 – 422, 01 2009. doi: 10.1109/ICDM.2008.17.

[40] Jianyi Liu, Sida Li, and Ru Zhang. Algorithm of reducing the false positives in ids based on correlation analysis. In IOP Conference Series: Materials Science and Engineering, volume 322, page 062016. IOP Publishing, 2018.

[41] Teresa Lunt. Detecting intruders in computer systems. In Proceedings of the 1993 conference on auditing and computer technology, volume 61, 1993.

[42] Martin Roesch. Network intrusion detection prevention system. https://www.snort.org/.

[43] Jaap Mooij et al. A generic approach for detecting security anomalies in isp infrastructures. 2017.

[44] National Institute of Standards and Technology. Cvss severity distribution over time. https://nvd.nist.gov/vuln-metrics/visualizations/cvss-severity-distribution-over-time, 2020.

[45] Subramanian Neelakantan and Shrisha Rao. A threat-aware signature based intrusion-detection approach for obtaining network-specific useful alarms. In 2008 The Third International Conference on Internet Monitoring and Protection, pages 80–85. IEEE, 2008.

[46] Humphrey Waita Njogu, Luo Jiawei, and Jane Nduta Kiere. Network specific vulnerability based alert reduction approach. Security and Communication Networks, 6(1):15–27, 2013.

[47] Humphrey Waita Njogu, Luo Jiawei, Jane Nduta Kiere, and Damien Hanyurwimfura. A comprehensive vulnerability based alert management approach for large networks. Future Generation Computer Systems, 29(1):27–45, 2013.

[48] Tadashi Ogino. Evaluation of machine learning method for intrusion detection system on jubatus. International Journal of Machine Learning and Computing, 5(2):137, 2015. doi: 10.7763/IJMLC.2015.V5.497.

[49] Nerijus Paulauskas and Algirdas Baskys. Application of histogram-based outlier scores to detect computer network anomalies. Electronics, 8(11): 1251, 2019. doi: 10.3390/electronics8111251.

[50] Vern Paxson. Bro: a system for detecting network intruders in real-time. Computer networks, 31(23-24):2435–2463, 1999. doi: 10.1016/S1389-1286(99)00112-7.

[51] Tadeusz Pietraszek. Using adaptive alert classification to reduce false positives in intrusion detection. In International Workshop on Recent Advances in Intrusion Detection, pages 102–124. Springer, 2004.

[52] Tadeusz Pietraszek. Alert classification to reduce false positives in intrusion detection. PhD thesis, Citeseer, 2006.

[53] Tadeusz Pietraszek and Axel Tanner. Data mining and machine learning—towards reducing false positives in intrusion detection. Information security technical report, 10(3):169–183, 2005.

[54] Reza Sadoddin and Ali A Ghorbani. An incremental frequent structure mining framework for real-time alert correlation. computers & security, 28(3-4):153–173, 2009. doi: 10.1016/j.cose.2008.11.010Getrightsandcontent.

[55] Michael M Sebring. Expert systems in intrusion detection: A case study. In Proc. 11th National Computer Security Conference, Baltimore, Maryland, Oct. 1988, pages 74–81, 1988.

[56] Syed Ali Raza Shah and Biju Issac. Performance comparison of intrusion detection systems and application of machine learning to snort system. Future Generation Computer Systems, 80:157–170, 2018.

[57] 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.

[58] Md Amran Siddiqui, Jack W Stokes, Christian Seifert, Evan Argyle, Robert McCann, Joshua Neil, and Justin Carroll. Detecting cyber attacks using anomaly detection with explanations and expert feedback. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2872–2876. IEEE, 2019. doi: 10.1109/ICASSP.2019.8683212.

[59] Skybox Security. Vulnerability and threat trends. https://lp.skyboxsecurity.com/rs/440-MPQ-510/images/Skybox_Report_Vulnerability_and_Threat_Trends_2019.pdf, 2019.

[60] Stephen E Smaha. Haystack: An intrusion detection system. In [Proceedings 1988] Fourth Aerospace Computer Security Applications, pages 37–44. IEEE, 1988. doi: 10.1109/ACSAC.1988.113412.

[61] Mahboobeh Soleimani and Ali A Ghorbani. Critical episode mining in intrusion detection alerts. In 6th Annual Communication Networks and Services Research Conference (cnsr 2008), pages 157–164. IEEE, 2008. doi: 10.1109/CNSR.2008.62.

[62] Basant Subba, Santosh Biswas, and Sushanta Karmakar. False alarm reduction in signature-based ids: game theory approach. Security and Communication Networks, 9(18):4863–4881, 2016.

[63] T Subbulakshmi, George Mathew, and S Mercy Shalinie. Real time classification and clustering of ids alerts using machine learning algorithms. International journal of Artificial & Application, 1(1):20, 2010.

[64] Li Sun, Steven Versteeg, Serdar Boztas, and Asha Rao. Detecting anomalous user behavior using an extended isolation forest algorithm: An enterprise case study, 2016.

[65] N. N. R. R. Suri, M. N. Murty, and G. Athithan. Unsupervised feature selection for outlier detection in categorical data using mutual information. In 2012 12th International Conference on Hybrid Intelligent Systems (HIS), pages 253–258, Dec 2012. doi: 10.1109/HIS.2012.6421343.

[66] Tcpdump Group. Tcpdump/libpcap public repository. https://www.tcpdump.org/.

[67] Tenable®. Nessus. https://www.tenable.com/products/nessus.

[68] Henry S Teng, Kaihu Chen, and Stephen C Lu. Adaptive real-time anomaly detection using inductively generated sequential patterns. In Proceedings. 1990 IEEE Computer Society Symposium on Research in Security and Privacy, pages 278–284. IEEE, 1990. doi: 10.1109/RISP.1990.63857.

[69] Tim Rains. Microsoft security intelligence report volume 20. https://www.microsoft.com/security/blog/2016/05/05/microsoft-security-intelligence-report-volume-20-is-now-available/, 2020.

[70] Gina C Tjhai, Maria Papadaki, SM Furnell, and Nathan L Clarke. Investigating the problem of ids false alarms: An experimental study using snort. In IFIP International Information Security Conference, pages 253–267. Springer, 2008.

[71] An Trung Tran. Network anomaly detection. Future Internet (FI) and Innovative Internet Technologies and Mobile Communication (IITM) Focal Topic: Advanced Persistent Threats, 55, 2017. doi: 10.2313/NET-2017-09-1_08.

[72] James J Treinen and Ramakrishna Thurimella. Finding the needle: Suppression of false alarms in large intrusion detection data sets. In 2009 International Conference on Computational Science and Engineering, volume 2, pages 237–244. IEEE, 2009.

[73] Risto Vaarandi. Real-time classification of ids alerts with data mining techniques. In MILCOM 2009-2009 IEEE Military Communications Conference, pages 1–7. IEEE, 2009.

[74] Hank S Vaccaro and Gunar E Liepins. Detection of anomalous computer session activity. In Proceedings. 1989 IEEE Symposium on Security and Privacy, pages 280–289. IEEE, 1989. doi: 10.1109/SECPRI.1989.36302.

[75] Fredrik Valeur, Giovanni Vigna, Christopher Kruegel, and Richard A Kemmerer. Comprehensive approach to intrusion detection alert correlation. IEEE Transactions on dependable and secure computing, 1(3):146–169, 2004.

[76] Félix Iglesias Vázquez, Robert Annessi, and Tanja Zseby. Analytic study of features for the detection of covert timing channels in networktraffic. Journal of Cyber Security and Mobility, 6(3):225–270, 2017. doi: 10.13052/jcsm2245-1439.632.

[77] Verizon Enterprise. Data breach investigations report. https://enterprise.verizon.com/resources/reports/dbir/2019/summary-of-findings/, 2019. Online, Accessed 30 May 2020.

[78] Eduardo Viegas, Altair O Santin, Andre Franca, Ricardo Jasinski, Volnei A Pedroni, and Luiz S Oliveira. Towards an energy-efficient anomaly-based intrusion detection engine for embedded systems. IEEE Transactions on Computers, 66(1):163–177, 2016. doi: 10.1109/TC.2016.2560839.

[79] Jouni Viinikka, Hervé Debar, Ludovic Mé, Anssi Lehikoinen, and Mika Tarvainen. Processing intrusion detection alert aggregates with time series modeling. Information Fusion, 10(4):312–324, 2009. doi: 10.1016/j.inffus.2009.01.003.

[80] DWYO Waidyarathna, WVAC Nayantha, WMTC Wijesinghe, and Kavinga Yapa Abeywardena. Intrusion detection system with correlation engine and vulnerability assessment. INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, 9(9):365–370, 2018.

[81] Hongzhi Wang, Mohamed Jaward Bah, and Mohamed Hammad. Progress in outlier detection techniques: A survey. IEEE Access, 7:107964–108000, 2019. doi: 10.1109/ACCESS.2019.2932769.

[82] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In Icml, volume 97, page 35, 1997.

[83] Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. Journal of machine learning research, 5(Oct):1205–1224, 2004.

[84] Ji Zhang. Advancements of outlier detection: A survey. ICST Transactions on Scalable Information Systems, 13(1):1–26, 2013. doi: 10.4108/trans.sis.2013.01-03.e2.

[85] Yue Zhao. A python toolbox for scalable outlier detection (anomaly detection). https://pypi.org/project/pyod/.

[86] Z. Zhen, H. Wang, L. Han, and Z. Shi. Categorical document frequency based feature selection for text categorization. In 2011 International Conference of Information Technology, Computer Engineering and Management Sciences, volume 2, pages 65–68, Sep. 2011. doi: 10.1109/ICM.2011.365.

[87] Tommaso Zoppi, Andrea Ceccarelli, and Andrea Bondavalli. On algorithms selection for unsupervised anomaly detection. In 2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC), pages 279–288. IEEE, 2018. doi: 10.1109/PRDC.2018.00050.