Universiteit Leiden
Opleiding Informatica

Adaptive Cyber Security

for a Thesis

Name:	Lennart van Efferen
Studentnr:	s1398296
Date:	06/02/2017
1st supervisor:	Dr. Amr Ali-Eldin
2nd supervisor:	Dr. W.J. Kowalczyk

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Adaptive Cyber Security: Machine Learning Techniques for a Flow–Based Anomaly Intrusion Detection System

Lennart van Efferen

1 Abstract

The increase in successful cyber attacks on systems with firewalls and encryption techniques has lead to the creation of Intrusion Detection Systems. Machine learning techniques are often used for these systems to predict malicious behavior in the vague and unbalanced data. In this research, an Intrusion Detection System is developed that can only access the packet headers of network traffic and not the attached data, to keep up with the growing bandwidth of networks and maintain the privacy of the users. Both a Multi–Layer Perceptron and a J48 Decision Tree are used to compare this system with one that can also access the payload–data of network packets. Secondly, the two different detection methods of IDS are compared to gain insight in the likely direction of future Adaptive Cyber Security systems. Finally, the two machine learning techniques are juxtaposed in order to analyse their performance as network traffic classification algorithms.

Contents

List of Figures

List of Tables

2 Introduction

With so much information online nowadays, cyber attacks represent a massive threat to the safety and privacy of society. The American computer security software company McAfee for example reported, in the Center for Strategic and International Studies of June 2014, that the estimated annual cost of the global economy from cybercrime is more than 400 billion dollars. Traditional techniques such as encryption or firewalls fail frequently to protect systems from penetration attacks or malware. However, it is possible to develop a system that can detect and deny intrusions at an early stage, so that the damage can be minimised. A system that identifies malicious activites in network resources is called an Intrusion Detection System (IDS). As the amount of usage on networks grows every year, one of the problems IDS systems thus face is the size of input data. Big companies, universities or government instances, for example, already exceed network traffic usages over hundreds of Mbps. Another problem is the increase in malicious behaviour in general, mostly because of the increase of economical value of the attack.

A possible solution for this network monitoring problem is to investigate the flows in the network data instead of the packet payload. The flow in network traffic describes the packets from a source to a destination in a certain time interval, or all the packets with common properties. By looking at flows instead of the payloads, the data size for investigation drops drastically. Some attacks however are less likely to be detected since they can be transported by the payload. It is therefore of great importance that the flow–based system is created in a good way.

Machine learning techniques are often used in IDS to try to predict this malicious behaviour to reduce the damage that can be caused by it. The reason for this common practice is due to the ability of these techniques to handle uncertain data. A neural network, for example, transforms inputs to a meaningful output with respect to the characteristics of the input nodes and the weights given to the inner connection nodes. For the achievement of adaptive cyber security in the near future, some techniques seem promising. These machine learning techniques function as a classification algorithm and can work in two different ways. The first way is called a misuse detection system which tries to classify the attacks by learning the characteristics of these penetration attempts. The second way, called a anomaly detection system, tries to identify only abnormal data from normal data, instead of classifiying each attack. In this research a flow–based anomaly detection system is proposed.

The aim of this research is to develop a flow–based anomaly detection system. This flow–system will then be compared with a system that can access the payload of network flows as well. The same dataset will be used for the comparison of the systems and the true positive and false positive rates are used as the performance evaluation. The goal is a high true positive rate, as this indicates the detection rate of the classes, and a low false positive rate, since this indicates the attacks that went through the system. Additionaly, the flow–based anomaly system will be compared with a flow–based misuse system. There should be a difference in the classification results of the system with just two classes (benign and malicious behaviour) and the system with more classes (benign behaviour and different attack types). As a final experiment, the two machine learning techniques used in this research are compared for the classification of network traffic.

In the following chapter, IDS and the problems they face are elaborated. Then a model for a multi–layer perceptron and a J48 tree are given and as to why these machine learning techniques are used. In chapter three classification techniques of network traffic will be explained, with flow–based as the main subject. Also, the problems flow–based systems face are explained. Then existing datasets are discussed and an argument is formulated as to why the UNSW–NB15 and Winters dataset are used in this paper. Feature selection of the datasets is then elaborated. Then the results of the experiments are showed and finally the paper concludes by indicating the next area of research to be conducted based on the results of the experiments in this paper and results of other research.

3 Research background

In the last years a lot of research has been conducted in the field of cyber security, with special interest regarding computational intelligence. Algorithms have been used for both anomaly detection and misuse detection. The inputs vary between command sequences from user input, network flows or system information. The biggest problem however is a representative dataset, since most available sets do not represent the wide variety of instances that can arise in the outside world. The DARPA and KDD dataset are still the most publicly available sets, even after the criticism by Mc.Hugh[22] and Malhony and Chan[3] on the DARPA set and the criticism by Sabnani et al[4] on the KD99 set. Nowadays a lot of researchers generate their own dataset to avoid these unfavorable sets or to overcome their limitations (of incomplete training sets for example). A honeypot is a way to generate a dataset, but it does not guarantee that the set of normal instances is free of malicious behaviour. A. Sperotto et al[21] created the first flow labelled dataset, generated from a honeypot. Winter [26] extracted his own dataset from these flows for the use of Support Vector Machines as analysis.

Artificial Neural Networks are one of the most succesfully employed data processing units since they have the ability to generalise from incomplete and noisy data. In this research network data is used as the input for the detection of anomaly's, with the research conducted by James Cannady as one of the main examples. In the article "Artificial Neural Networks for Misuse Detection"[27] he got promising results with the implementation of a multi–layer perceptron as a misuse system, eventhough the time to build the model was relatively high. Other researchers used a neural network for malicious detection in different input data, for example K.Tan [28] in user behaviour data or Gosh[29] in sequences of system calls.

For the classification of network traffic as input for the machine learning algorithms, specially payload and flow–based systems have been investigated thoroughly. P. Gogoi et al.[30] and H. Alaidaros et al[31] give a great overview on how the performance and accuracy are compared and where the advantages and disadvantages of these systems lie. In the research "Anomalous Payload-based Network Intrusion Detection" by K.Wang et al [11] a payload–based anomaly detector called PAYL is presented with almost 100 percent accuracy on traffic for port 80.

M. Soysal et al.[47] already conducted research for machine learning algorithms for flow–based IDS. Three different techniques are compared and evaluated: Bayesian networks, decision trees and multi–layer perceptrons. The decision trees performed superior compared to the Bayesian networks in classification accuracy, but required a longer training time. The multi–layer perceptron showed poor accuracy values compared to the decision tree and had a longer training time than the Bayesian networks. Z. Jadidi et al.[50] used a neural network optimized with a Gravitational Search Algorithm (GSA) on Winters dataset for a flow–based system. The system resulted in a 99.43 percent accuracy on classifying benign and malicious flows. Y. Abuadlla et al.[52] used three different training algorithms for a two stage neural network as a flow–based system. Resilient Backpropagation, Radial Basis Function net and Levenberg–Marquardt were used for training the neural network on a NetFLow dataset. The first stage was the anomaly detection stage and the second stage the detection and classification stage. The analysis firstly showed an improvement of prediction accuracy in the second stage compared to the first (anomaly detection) stage. Secondly, a multi–layer perceptron with Levenberg–Marquardt has low memory consumption and a low false alarm rate compared to the Radial Basis Function and was faster compared to Backpropagation.

4 Intrusion Detection System

The components of an IDS are data collection, data pre–processing, intrusion recognition, reporting and response. If a system is reactive, it is called an Intrusion Prevention System (IPS) or Intrusion Detection and Prevention System (IDPS), and responds by blocking the connection or reprogramming the firewall. A system that merely detects attackers, is called an Intrusion Detection System (IDS) and is a good way to begin implementing adaptive behaviour against cyber attacks, since intrusion recognition is the most vital part. The problems an intrusion detection system faces however are not easy to overcome. The data they retrieve as input must be relevant for the detection of cyber attacks, therefore it could be network traffic (data packets), command sequences from user inputs, or low–level system information (e.g log files or CPU usage) of the system or network. However, the volumes for this sort of data can be huge, especially when the inspected network is of some size. Furthermore, the data distribution is highly imbalanced and there is not a realizable boundary between normal and abnormal behaviour. To make it even more difficult, people come up with new penetration techniques every day so there is a need for continuous adaption for this changing environment. There are two different detection methods IDS systems use[1]:

- *Misuse Detection System (MDS)*: a system that identifies intrusions by matching observed data with pre-defined data. The system requires descriptions of data regarding intrusive behaviour in order to properly detect intrusions.

- *Anomaly Detection System (ADS)*: this system builds a model for 'normal' data and uses the model to detect anomalies/outliers in the observed data by detecting deviations from the model.

Most systems use MDS where the system compares the data in the network with a database of known attack patterns, because it is effective when the intrusion signature is created in a good way. Moreover, since network traffic tends to be vague and rarely is 'normal', ADS that attempt to build a model for this 'normal behaviour' frequently fail [2]. The false alarm rate is therefore high compared to MDS. However, MDS only detect attacks when prior knowledge of the attack is available. For responsive behaviour in a detection system without prior knowledge, an ADS is preferable as MDS lack of self-teach abilities and therefore fail to defend against new attack types if the signatures are not updated[5].

Machine learning can be approached via supervised or unsupervised learning. Unsupervised learning creates a model without the need of a labelled dataset by modelling the universal properties of the data. Supervised learning trains a classifier on a labelled dataset in order to determine to which class instances befit. The problem with supervised learning in IDS is the lack of available labelled datasets. Two commonly used data sets consisting of network traffic and audit logs are available online and are known as DARPA and KD99. These sets have been used extensively by researchers to learn and develop IDS. Yet, as numerous research showed, these data sets have problems that need to be overcome in order to create a more useful detection system [3, 4, 22].
Most of the IDS systems use machine learning techniques to distinguish normal data from abnormal data in the large amounts of data that they retrieve as input. There are different machine learning techniques for these cybersecurity systems, such as artificial neural networks, evolutionary computation, artificial immune systems, fuzzy systems, swarm intelligence or soft computing. These techniques are used because they can adapt to a changing environment, they are resilient against noisy information and they can exhibit fault tolerance. In this paper, an artificial neural network is proposed for the classification of network traffic. Artificial neural networks (ANN) seem promising for the classification of network traffic [27]. They can generalize from limited noisy and incomplete data, which are often characteristics of network data. Furthermore, if the network structure is well developed, high computational speed can be achieved and more attacks can be prevented [7, 8]. Normal multi–layer feed forward (MLFF) neural networks require long training time, so a lot of times Radial Basis Function (RBF) neural networks are used instead [9]. Another way to improve the detection rate is good feature selection, which will be done in for this research.
Additionaly a decision tree is proposed since this technique seems promising for the classification problem of network traffic [33]. The first place at the KD99 detection competition was with the use of a decision tree and the second place a decision forest [34]. With the use of more machine learning techniques, the comparisons

between the systems are more valuable. Moreover, the time to generate a model from the input data is smaller compared to the multi–layer perceptron and the resulting tree is easy to understand for the users.

5 Machine Learning Techniques

5.1 Multi–Layer Perceptron

Engineers and computer scientists often look at nature for intelligent systems to forge it in artificial ways. Two good examples are swarm intelligence and neural networks. Swarm intelligence looks at natural complex systems in (e.g ant colonies or bird flocking), which consist of non–complex agents, that can perform intelligent tasks jointly, impracticable to the individual agent. The engineer attempts to build such a collective self organized system consisting of a population of simple agents. Artificial neural networks emerged from the way our biological neurons in the brain work. An biological neuron will fire an action potential if the cumulative input of the signals arriving exceeds a certain threshhold, represented as θ. This threshhold however varies around an average value and is not the same for every neuron so that it is uncertain if a neuron is doing what is expected given a potential. The firing threshholds are being updated continuously, which is the key factor for the adaptive learning abilities of the network. The same is true for a single layer artificial perceptron which can only solve linear problems:

$$y = \sum_{i=1}^{n} W_i X_i \tag{1}$$

Where n represent the number of inputs, corresponding tot the features. The weights per input are denoted to W_n and X_i is the input data. The perceptron then translates the inputs to an output signal, with respect to the threshhold, using a transfer function. For example the output will be 1 (firing state) when $\sum_{i=1}^{n} W_i X_i > \theta$. Commonly used examples of these functions are Unit Step, Sigmoid or Gaussian. The weights determine the slope of the transfer function and the Bias allows to shift the transfer function horizontally along the axis while leaving the curvature unaltered. Like the biological neurons, learning arises in updating the weights and Bias in order to reduce the error rate. The perceptron weight adjustment is denoted by:

$$\Delta W = \eta \times \delta \times X \tag{2}$$

With η as the learning rate (usually < 1) and δ as the predicted output $-$ desired output.

When a multilayer perceptron (MLP) is used non linearly separable data can be distinguished as well. A MLP has the same structure of a single layer with the addition of one or more hidden layers with all the nodes connecting each other between layers. The network trains itself with an algorithm called backpropagation. This supervised learning algorithm first computes outputs using a sigmoid function and then propagates the errors backwards. Each unit receives the amount of error it generated this way and the weights are adjusted. In short, backpropagation uses the output error to adjust the weights of inputs at the output error and then continues this adjustment for the previous layers. The error in one of the output nodes is then denoted as:

$$\delta_o = output \times (1 - output) \times (expected - output) \tag{3}$$

And the error rate for a node h in the hidden layer can be calculated as:

$$\delta_h = output_h \times (1 - output_h) \times (W_h - \delta_o) \tag{4}$$

For this research a MLP is used since the problem is not linear solvable. The system has to predict if an attack is happening from all the inputs it receives. The problem can therefore be expressed as in (1). If the inputs exceed a certain threshold, then an incoming attack is likely. The weight can be adjusted by the backpropagation algorithm. For the activation function in the nodes, the sigmoid function will be implemented:

$$\sigma(t) = \tfrac{1}{1+e^{-x}} \tag{5}$$

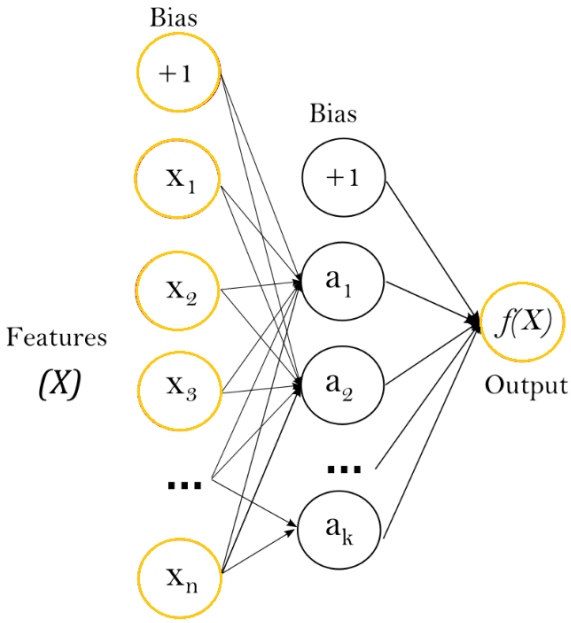


Figure 1: Multilayer perceptron with one hidden layer

The neural network in this research aims at both a binairy classification (anomaly) and a multi class problem (misuse), where instead of just predicting an attack, also the type of attack is predicted. Similair work has been conducted with high results [19, 20].

5.2 J48 Decision Tree

The J48 tree is the open source implementation of the C4.5 decision tree algorithm, which itself is an extension of the earlier ID3 algorithm. The algorithm utilises a tree–like structure with a root, nodes and leaves for the decisions in the classfication problems. Like the MLP, it is a supervised classification algorithm, since it needs all the outcomes of possible attributes to build itself. The outcome as a tree structure is easy to understand for the end users, unlike the blackbox problem with the MLP[27, 35] whereas the analysis of the data responses occurring in the network doesnt give any insight of the structure of the function being approximated. Additionally, the decision tree can handle various inputs or missing data[36] as well and is therefore useful in a wide range of applications.
The C4.5 algorithm uses the concept of information entropy of the attributes to split the instances. Information gain increases with the average purity of the created subsets. The formula of the entropy is denoted as:

$$H(p) = -p\log(p) - (1-p)log(1-p) \tag{1}$$

A node with $H(0) = 0$ is then pure node (non equal distribution) and with $H(0.5) = 1$ a mixed node (equal distribution). Sometimes instances with the same attributes result in the same class and further splitting is impossible. These identical examples then result in a leaf still mixed and not pure, which is unfavorable. A tree with only pure nodes on the training set however could be overfitting on the data, which does not guarantee the same results for unseen data. To generalise superior and overcome this overfitting problem, pruning a decision tree is often necessary. There are two ways to prune a decision tree:

- *Pre–pruning*: This technique is performed during the construction of the tree. It is therefore faster compared to post–pruning. When information from an attribute at a particular node is not statistically significant, this branch is not further developed.

- *Post–pruning*: Post–pruning is executed after the tree is finished building. With either subtree replacement or subtree raising some branches are pruned to reduce the estimated error.

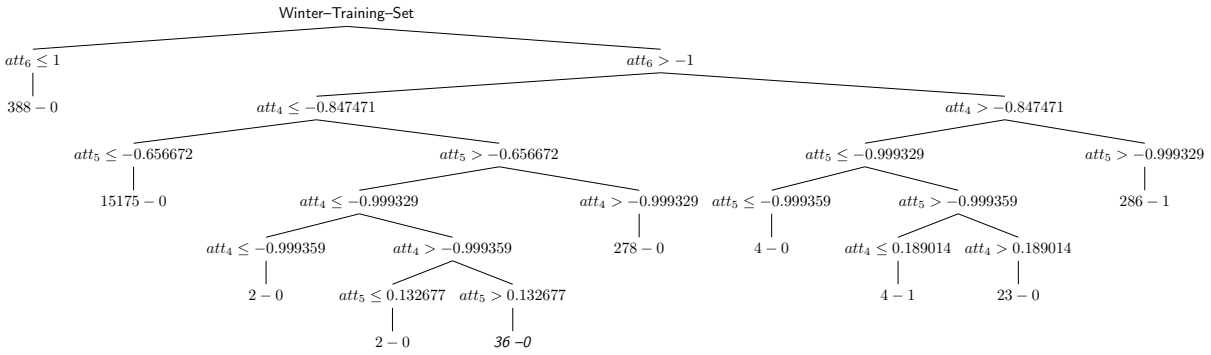
The C4.5 algorithm pruning is based on statistical confidence estimates. With N instances in the training set, an observed error rate f is measured. The true error rate p, which is the estimated error over all the available data, can then be calculated with a confidence interval by the following formula:

$$p = f \pm z \times \sqrt{\frac{f \times (1-f)}{N}} \tag{2}$$

With the value of z depending on the desired level of confidence. For the transformation of a branch and its leaves to a single leaf node the algorithm compares the (upper) error confidence intervals of the unpruned and the pruned tree. Since the pruned tree results in a leaf, the error estimation can be calculated by the formula seen at (2). For the unpruned tree, the weighted error estimates of all the leaves sum up to the upper error estimate of this subtree or can be calculated with the following formula:

$$e = \frac{f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}} \tag{3}$$

Note: The following decision tree is the actual constructed tree on the training set of one the experiments in this research. The training set consisted of 16200 instances, all the results are elaborated under the leaves. The size of the tree is 19 and the number of leaves is 10. This J48 tree correctly classified 99.98 percent of the 8630 instances in the test set



6 Flow–Based Intrusion Detection

There are three different ways a network based intrusion detection system can classify network traffic data. The first method is port-based, where the classification is based on the 16–bit transport layer port numbers used by servers for traffic flow. It unwraps the predefined layers of the packet and inspect them with a protocol analysis method. Anything that is deviating from the standard use of the protocol is likely malicious. There are some problems with this technique, such as the fact that the FTP protocol can assign ports dynamically according to the traffic load. Additionally, many protocols used by applications today are not registered to IANA (Internet Assigned Numbers Authority) or they use a varying set of ports in order to remain anonymous (like P2P applications). Due to these issues, the results from this port-based technique suffer from a low degree of accuracy [10]. The second technique for network traffic classification is the inspection of packet payload data [11]. This technique yields satisfactory accuracy but it demands a thorough investigation of the data. Privacy cannot be maintained unless the data is encrypted, which renders inspection of the packet payload data ineffective. The third technique is called flow–based classification [16], where the important features of network flows are collected from the packet headers (the transport ports can be features as well). Each traffic flow is characterised by a set of determined features with values that depend on the network class. Other features can then be created with these sequences of packets, like the connection duration or the amount of packets retrieved from one source. Since this data is gathered from the packet headers, privacy is maintained, data encryption is possible and less data is needed to be examined in comparison with the packet payload data technique or the protocol technique [17] (since flow–based only investigates an aggregated set of packets instead of every packet). It however got some drawbacks regarding certain attack types. Some malware, for example, worms or virusus, deliver their malicious code in the payload data of the packets and will therefore be hard to detect by just investigating the flow between source and destination.

For this paper the flow–based and the payload–based will be used for the neural network to teach itself. For time efficiency only the relevant features will be identified. This will be done with a literature study.
Flow data normally derives from modules placed in network routers. The packets have at least one layer 3 IP header, which normally consists of 20 bytes of data and thirteen items, some examples are:

- *IP Header Length* (number of 32 -bit words forming the header)
- *Size of Datagram* (The combined length of the header and the data in bytes)
- *Identification* (uniquely identifies this packet together with the source addresss)
- *Time To Live (TTL)* (Number of hops which the packet may be routed over)
- *Protocol* (type of transport packet being carried (e.g. 1 = ICMP; 6 = TCP; 17= UDP)
- *Source Address* (the IP address of the source)
- *Destination Address* (the IP address of the destination)

<table>
<tr><th>4</th><th>8</th><th>16</th><th colspan="2">32 bits</th></tr>
<tr><td>Ver.</td><td>IHL</td><td>Type of service</td><td colspan="2">Total length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td>Fragment offset</td></tr>
<tr><td colspan="2">Time to live</td><td>Protocol</td><td colspan="2">Header checksum</td></tr>
<tr><td colspan="5">Source address</td></tr>
<tr><td colspan="5">Destination address</td></tr>
<tr><td colspan="5">Option + Padding</td></tr>
<tr><td colspan="5">Data</td></tr>
</table>

Figure 2: Packet Header information

The last field of the packet is the data that is sent from the host to the source. With all the TCP or UDP headers, this data field starts with a layer 4 TCP header. This header gives additional information to the application that will recieve the incoming data. Some examples of fields in the layer 4 TCP header are:

- *Source port* (The port number of the application that sends the data)
- *Destination port* (The port number of the application that receives the data)
- *Control flags* (Bit string that indicates which of the six control flags are on and off)
- *Checksum* (The checksum of the remaining data that needs to be received)
- *Data* (The actual data that is being sent from the host to the received)

<table>
<tr><td colspan="8">16</td><td>32 bits</td></tr>
<tr><td colspan="8">Source port</td><td>Destination port</td></tr>
<tr><td colspan="9">Sequence number</td></tr>
<tr><td colspan="9">Acknowledgement number</td></tr>
<tr><td>Offset</td><td colspan="2">Reserved</td><td>U</td><td>A</td><td>P</td><td>R</td><td>S</td><td>F</td><td>Window</td></tr>
<tr><td colspan="8">Checksum</td><td>Urgent pointer</td></tr>
<tr><td colspan="9">Option + Padding</td></tr>
<tr><td colspan="9">Data</td></tr>
</table>

Figure 3: TCP Header information

7 The Datasets

Finding the right dataset is one of the most difficult tasks for IDS. Most of the datasets available are outdated and contain old attacks. Moreover, almost all available sets contain unrealistic data. For example, the DARPA 98 and 99 are simulated in a military network environment and are some of the most commonly used datasets of network traffic; however they have three major issues. Firstly, the sets do not contain modern attack types. Secondly, Mahoney and Chan [3] discovered that all of the packages in the 99 set, that contain a time to live (TTL) of 126 or 253 are malicious data packages. This is undesirable for data mining algorithms as a predictor, since they would teach themselves in a incorrect way. Thirdly, the TTL values are artificially high in comparison to real traffic data. For these reasons, the data is not representative and the DARPA 98 and 99 are no longer used for research. Another popular dataset is the KDD CUP 1999 set. However, it is essentially an extended of the DARPA 98 version (adding more features) and thus suffers from the same constraining characteristics. Moreover the set is missing records and the training set contains a volume of redundant data. This is the reason why the NSLKDD dataset, an improvement of the KD99 set, was issued. However NLSKDD is still not a good representation of an real life attack environment [22]. This is the reason the UNSW–NB15 data set[52] is used in this research as the labelled dataset for the misuse system. For a flow–based dataset an improved version of the honeypot data, gathered by Sperotto[21], is used.

7.1 Winter

The first publicly available flow–based dataset was captured by monitoring a honeypot, an environment that attracts attackers and analyses the network data. This was done by Sperotto, Sadre and Pras from the University of Twente [21]. The data contains ten features that provide flow information of the traffic collected. The features are eloborated in the following table:

Table 1: Features of the flow–based labeled dataset

Name.	**Description**
id	Unique Id of the flow
src_ip	The (anonymized) ip address of the source
dest_ip	The (anonymized) ip address of the destination
packtets	Amount of packets in the flow
octets	Amount of bytes in the flow
start_time	Start time of the connection in number of seconds
start_msec	Start time of the connection in miliseconds
end_time	End time of the connection in number of seconds
end_msec	Start time of the connection in miliseconds
src_port	Source port number
dst_port	Destination port number
tcp_flags	All the obtained TCP flags
prot	Protocol number

The dataset consists apart from malicious traffic of side effect traffic (not malicious) and unknown traffic (traffic that could not be classified). Winter [26] modified this dataset in some ways that could be benificial for training algorithms. The dataset is reduced in size and features are dropped (IP adresses since they have been anonymized) or combined for the training time. The old dataset was time consuming since it consists of 14.2 million flows (more then 98 percent has been labeled). By selecting only the relevant flow attributes, deleting the unlabeled flows and all flows belonging to other protocols then SSH and HTTP and reducing the size, this set is easy to use and not time consuming. The new set consists of 22942 flows, gathered through a random sampling process with a probability chance of 1/600. The features are given in the following table. All values of the features are set in the range between 0 and 1 for the One–Class Support Vector Machines used in the paper.

Table 2: Features of Winter's data set

Name.	Description
IP protocol	IP protocol number of the flow
Source port	Source port of the flow.
Destination port	Destination port of the flow.
Duration	Duration of the flow in seconds.
Octets per flow	Amount of transferred bytes in the flow.
TCP flags	TCP flags in the flow.
Packets per flow	Amount of transferred ackets in the flow.

7.2 UNSW–NB15

The UNSW–NB15 dataset was created in the Australian Centre for Cyber Security to simulate real modern attack activities. The UNSW–NB15 contains nine types of attacks and 49 generated features. Both anomaly detection and attack classification are conducted with the neural network on this dataset. The features and their descriptions are provided in the appendix and can be divided into several different categories (see table 31): basic features, flow features, content features, time features and additional generated features. The flow features are the results, saved as Packet capture (PCAP) files, from the TCP dump and are the contents of the network packets. They consist of the normal features that are necessary for a connection, such as IP-addresses, the length of the packet and the protocol used. All of the other features are generated by the tools BRO-IDS and Argus. The basic features can be derived from the packet headers without inspecting the payload of the packets (see table 32). The content features are of flow statistics, information from the TCP/IP headers or payload information. Time features describe the data that supply a specific time or need to mature over a temporal window. Some examples of such features are the records start time, the recorded last time or the time between the SYN and the SYN_ACK packets of the TCP (see table 35). The remaining features are depicted in the dataset. The labelled features 48 and 49: attack category and label, are the target features for data mining algorithms. The former refers to the name of each attack category among the nine categories: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. The latter feature specifies whether there is an attack or not, thus taking the binary form. Out of this dataset, most of the features can be gathered from the network flow or generated additionaly from these features.
The UNSW dataset contains several attack types:

- *DoS* (An common attack attempt to exhaust the destinations resources so that normal traffic becomes denied)
- *Fuzzers* (A big random data feed to discover a flaw in the system and make it crash)
- *Backdoors* (A remote attack to gain unauthorized access to a system)
- *Exploits* (An instruction data feed that uses a flaw in a system)
- *Analysis* (A port based intrusion attack against web applications)
- *Generic* (A hash function against a block–cipher to cause a collision)
- *Reconnaissance*, (Now known as Differentiated Services Code Point (DSCP) (usually set to 0), but may indicate particular Quality of Service needs from the network, the DSCP defines the way routers should queue packets while they are waiting to be forwarded)
- *Shellcode* (Small program with instructions for the victim's computer)
- *Worms* (Self multiplying malicious code that, instead of virussus, doesnt need a program to attach itself to)

7.2.1 UNSW–NB15 Feature Selection

A lot of work is done on the right feature selection of network traffic data, for example the work of H. Gunes Kayacik et al. [13]. The study utilises the KD99 dataset (which contains 41 different features, some more important than others and some not important at all). The paper identifies four type of attacks: Denial of Service (DOS), Probes, Remote to Local (R2L) and User to Root (U2R). Correct feature selection is crucial to decreasing the time–and data complexity. The feature for example that gives the number of outbound commands in an FTP session (feature 20 in the KD99 CUP and feature 40 in the UNSW dataset) evidently contributes very little to the classication of an attack, and can be dropped.
Studies by Nour Moustafa and Jill Slay [15] compare the features of the KD99 and the UNSW dataset for their relevance in classification and also delivered great insight for this research. The result are depicted in table 3. Other work shows that attacks are patently dependent on a combination of features [12]. Unfortunatly not all the features can be used for a flow–based ADS. For this research all the unnecessary features, according to the work of Nour Mastafa and Jill Slay, have been dropped to reduce the time complexity.

Table 3: UNSW–NB15 feature relevance

Normal	11, 34, 19, 20, 21, 37, 6, 10, 11, 36, 47
DoS	6, 11, 15, 16, 36, 37, 39, 40, 42, 44, 45
Fuzzers	6, 11, 14, 15, 16, 36, 37, 39, 40, 41, 42
Backdoor	6, 10, 11, 14, 15, 16, 37, 41, 42, 44, 45
Exploits	10, 41, 42, 6, 37, 46, 11, 19, 36, 5, 45
Analysis	6, 10, 11, 12, 13, 14, 15, 16, 34, 35, 37
Generic	6, 9, 10, 11, 12, 13, 15, 16, 17, 18, 20
Reconnaissance	10, 14, 37, 41, 42, 43, 44, 9, 16, 17, 28
Shellcode	6, 9, 10, 12, 13, 14, 15, 16, 17, 18, 23
Worms	41, 37, 9, 11, 10, 46, 23, 17, 14, 5, 13

The features from this dataset chosen for this research regarding the flow–based system are given in the following table. These features can be accessed through the layers 3 and 4 of the TCP/IP connection and dont need access to the payload of the packet. For the payload based system, all relevant features will be used.

Table 4: UNSW–NB15 relevant flow features

1	proto	nominal	Transaction protocol
2	state	nominal	Indicates to the state and its dependent protocol
3	dbytes	Integer	Destination to source transaction bytes
4	sttl	Integer	Source to destination time to live value
5	dttl	Integer	Destination to source time to live value
6	sloss	Integer	Source packets retransmitted or dropped
7	dloss	Integer	Destination packets retransmitted or dropped
8	service	nominal	http/ftp/smtp/ssh/dns/ftp-data/irc and (–) if not much used service
9	Sload	Float	Source bits per second
10	Dload	Float	Destination bits per second
11	Spkts	integer	Source to destination packet count
12	Dpkts	integer	Destination to source packet count
13	swin	integer	Source TCP window advertisement value
14	dwin	integer	Destination TCP window advertisement value
15	stcpb	integer	Source TCP base sequence number
16	smeansz	integer	Mean of the flow packet size transmitted by the src
17	Djit	Float	Destination jitter (mSec)
18	synack	Float	TCP connection setup time. The time between the SYN and the SYN_ACK packets.
19	ackdat	Float	TCP connection setup time. The time between the SYN_ACK and the ACK packets.
20	is_sm_ips_ports	Binary	If source (1) equals to destination (3) IP addresses and port numbers (2)(4) are equal, this variable takes value 1 else 0
21	ct_state_ttl	Integer	No. for each state (6) according to specific range of values for source/destination time to live (10) (11).
22	ct_srv_src	Integer	No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26).
23	ct_srv_dstt	Integer	No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26).
24	ct_dst_ltm	Integer	No. of connections of the same destination address (3) in 100 connections according to the last time (26).
25	ct_src_ ltm	Integer	No. of connections of the same source address (1) in 100 connections according to the last time (26).
26	ct_src_dport_ltm	Integer	No of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26).
27	ct_dst_sport_ltm	Integer	No of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26).
28	ct_dst_src_ltm	Integer	No of connections of the same source (1) and the destination (3) address in in 100 connections according to the last time (26).

8 Experimental Work

In this section, all the results of the conducted experiments are elaborated. The tables show all the relevant and interesting outcomes for comparison between the proposed system and possible alternatives. This work is done in the tool Weka, with the standard built in MLP and J48 tree.

8.1 Flow–Based Anomaly Detection versus Payload–Based Anomaly Detection

The main experiment in this research was the comparison between a flow–based anomaly IDS and a payload–based anomaly IDS. Winters dataset is used to investigate the possibiliy of such a flow–system. The UNSW–NB15 is used for the comparison between a ADS and a MDS. The following tables show the results of the classifications with the true positive rates (TPR) and false positive rates (FPR) as an indication of the performance of the systems[41]. All these systems in this experiment have been trained with the provided training sets.

8.1.1 Results of the Flow–Based Systems

Table 5: The MLP flow–based anomaly detection system on different datasets

Data set	**Total instances**	**Normal instances**	**Attacks**	**Detection rate**	**Avg. TPR**	**Avg. FPR**
Winter	8630	942	7688	99.59%	99.6 %	1.8%
UNSW–NB15	175341	56000	119341	93.29%	93.3%	14.3%

Table 6: The J48 tree flow–based anomaly detection system on different datasets

Data set	**Total instances**	**Normal instances**	**Attacks**	**Detection rate**	**Avg. TPR**	**Avg. FPR**
Winter	8630	942	7688	99.98%	100 %	0.2%
UNSW–NB15	175341	56000	119341	88.53%	88.5%	6.8%

Table 7: Confusion matrices MLP

Table 8: Winter

Classified as normal	**Classified as attack**	
923	19	**Normal**
16	7672	**Attack**

Table 9: UNSW–NB15

Classified as normal	**Classified as attack**	
44265	11735	**Normal**
26	119315	**Attack**

Table 10: Confusion matrices J48 tree

Table 11: Winter

Classified as normal	**Classified as attack**	
940	2	**Normal**
0	7688	**Attack**

Table 12: UNSW–NB15

Classified as normal	**Classified as attack**	
54475	1525	**Normal**
18582	100759	**Attack**

The results of these flow–based systems are promising since the detection rate is high and the false alarm rate low. Especially the results derived from Winters dataset with almost 100 percent detection rates and low false alarm rates. Moreover, with only 7 features to predict the attacks, the time complexity is very low.
The MLP performed inferior on the UNSWB–NB15 dataset with a 93.3 percent of the instances correctly classified and a 20.96 percent false alarm rate. Even with the 28 features presented the MLP could not perform better in separating the malicious behaviour from the benign. The contribution of the second dataset shows the importance of right feature selection and how results can differ given the input.
The J48 tree performed almost perfectly on Winters dataset with 0 succesfull attacks and only 2 false alarms. The results generated from the UNSW–NB15 dataset differ from the MLP in the severity of the system. The MLP only let 26 attacks through but showed a 20.96 false alarm rate, which renders the system as severe. The decision tree did not detect 18582 attacks but only has a 2.72 percent false alarm rate.

8.1.2 Results of the Payload–Based Systems

Table 13: The MLP payload-based anomaly detection system on the UNSWB–NB15 dataset

Data set	**Total instances**	**Normal instances**	**Attacks**	**Detection rate**	**Avg. TPR**	**Avg. FPR**
UNSW–NB15	175341	56000	119341	93.33%	93.3%	14.2%

Table 14: The J48 tree payload-based anomaly detection system on the UNSWB–NB15 dataset

Data set	**Total instances**	**Normal instances**	**Attacks**	**Detection rate**	**Avg. TPR**	**Avg. FPR**
UNSW–NB15	175341	56000	119341	88.47%	88.5%	6.9%

Table 15: Confusion matrices payload–based systems

Table 16: MLP

Classified as normal	**Classified as attack**	
44337	11663	**Normal**
27	119314	**Attack**

Table 17: J48 tree

Classified as normal	**Classified as attack**	
54478	1522	**Normal**
18699	100642	**Attack**

The MLP achieved a high detection rate and let few attacks pass, only 27 out of the 119341. The amount of false alarms is high with 11663 out of the 56000 classified unsuccessfully. The decision tree showed a lower detection rate and lower false alarm rate. However, a IDS with 15.67 percent of the attacks penetrating the system, resulting in 18699 attacks, could be concluded as a impractical system.

8.1.3 Comparison of the Systems

The data of the payload–based systems show similair results to the flow–based systems. Most of the outcomes show little difference. Because the time complexity is higher with a payload–based system (since more features are used), the argument could be made that the difference should be significantly higher for the practical use of such a system. Especially in an environment with high network usage, the flow–based system seems more logical. With even better feature selection and/or combination (see the results in table 7), a better and faster IDS could be created without the information gathered from the inspection of the packet payload, since these features do not make a significant difference.

Table 18: MLP

Flow–based	**Payload–based**	
93.29%	93.33%	**Detection Rate**
79.0%	79.2%	**True Positive Rate Benign Instances**
100%	100%	**True Positive Rate Malicious Instances**
0.0%	0.0%	**False Positive Rate Benign Instances**
21.0%	20.8%	**False Positive Rate Malicious Instances**
11735	11663	**No. False Alarms**
26	27	**No. Undetected Attacks**

Table 19: J48 tree

Flow–based	**Payload–based**	
88.53%	88.47%	**Detection Rate**
97.3%	97.3%	**True Positive Rate Benign Instances**
88.5%	84.3%	**True Positive Rate Malicious Instances**
15.6%	15.7%	**False Positive Rate Benign Instances**
2.7%	2.7%	**False Positive Rate Malicious**
1525	1522	**No. False Alarms**
18582	18699	**No. Undetected Attacks**

8.2 Anomaly Detection versus Misuse Detection

The following tables show the results of the MLP and the J48 decision tree as a misuse system. All the attack catagories could be an outcome instead of the binairy outcome form, seen previously at the ADS.

8.2.1 Results of the Misuse Systems

Table 20: Results of both misuse systems

Algorithm	Overall detection rate	Avg. TPR	Avg. FPR	Benign TPR	Benign FPR
MLP	70.16%	70.2%	11.1%	97.7%	25.3%
J48	75.20%	75.2%	6.7%	97.3%	15.4%

Table 21: Multilayer perceptron flow–based misuse system

Normal	Reconnaissance	Backdoor	DoS	Exploits	Analysis	Fuzzers	Worms	Shellcode	Generic	← *Classified as*
54723	1	0	3	1273	0	0	0	0	0	**Normal**
7106	0	0	0	3365	0	0	0	0	20	**Reconnaissance**
259	0	0	0	1487	0	0	0	0	0	**Backdoor**
1217	0	0	0	11007	0	0	0	0	40	**DoS**
4211	0	0	0	29169	0	0	0	0	13	**Exploits**
25	0	0	0	1975	0	0	0	0	0	**Analysis**
15947	0	0	0	1941	0	11	0	0	285	**Fuzzers**
20	0	0	0	110	0	0	0	0	0	**Worms**
1133	0	0	0	0	0	0	0	0	0	**Shellcode**
236	0	0	0	649	0	0	0	0	39115	**Generic**

The results of this MLP flow–based misuse system are interesting for multiple reasons. The TPR of the normal behaviour is a 97.72 percent, and therefore 2.28 percent of the normal instances resulted in a false alarm. However, 25.27 percent of the malicious behaviour is classified as benign, which makes the system undeployable for a real world network. It failed to detect any form of reconnaissance, backdoor, DoS, analysis, worms or shellcode attacks. The MLP classified most instances as normal behaviour, exploits attacks or generic attacks. The weights of the nodes in this neural network seem unbalanced and therefore the outcome unfavorable.

Table 22: J48 flow–based misuse system

Normal	Reconnaissance	Backdoor	DoS	Exploits	Analysis	Fuzzers	Worms	Shellcode	Generic	← *Classified as*
54469	52	6	40	488	0	918	3	14	10	**Normal**
366	7593	2	1356	1048	0	82	3	4	37	**Reconnaissance**
114	25	105	1127	291	1	37	2	9	35	**Backdoor**
1045	106	17	8065	2411	1	248	4	51	316	**DoS**
2305	559	19	10121	19079	2	506	49	88	665	**Exploits**
634	6	0	1139	162	0	20	0	0	39	**Analysis**
13547	38	4	1177	542	0	2450	0	69	357	**Fuzzers**
7	0	0	5	31	0	4	74	1	8	**Worms**
257	52	0	59	123	0	31	0	587	24	**Shellcode**
70	5	2	272	185	0	25	1	6	39434	**Generic**

The J48 tree also got a high detection rate of normal instances, which resulted in a merely 2.73 percent false alarm ratio. It failed completely to detect any analysis attack, and classifed most instances as exploits, DoS or generic attacks. With 15.4 percent of attacks penetrating the system, the IDS seems unfavorable.

8.2.2 Comparison of the systems

Considering the MLP as a flow–based ADS previous with a 20.96 percent false alarm rate, this improvement of the MLP as a MDS is noticeable. However, the system also let 30154 attacks pass, a 25.27 percent of the total attacks. In this area of research, the number of successful attacks on a system weigh more heavily on the performance than the amount of false alarms. The ADS system was too severe resulting in high amounts of false alarms and the MDS too mild, resulting in a system failing to classify malicious behaviour.
The decision trees scored similar results with little difference in the TPR and FPR on the benign behaviour. Both the ADS and the MDS failed to recognize the difference between the malicious and benign behaviour however and respectively let 15.6 and 15.4 percent of the attacks through by classifiying those instances as normal.

Table 23: MLP

ADS	**MDS**	
79.0%	97.7%	**True Positive Rate Benign Instances**
0.0%	25.3%	**False Positive Rate Benign Instances**
11735	1277	**No. False Alarms**
26	30154	**No. Undetected Attacks**

Table 24: J48 tree

ADS	**MDS**	
97.3%	97.3%	**True Positive Rate Benign Instances**
15.6%	15.4%	**False Positive Rate Benign Instances**
1525	1531	**No. False Alarms**
18582	18345	**No. Undetected Attacks**

8.3 Multilayer Perceptron versus J48 Tree

The following results show the difference between a MLP and a J48 decision tree for the classification of attacks. Both have been trained with all the relevant features and with 10 fold cross validation on the test set. This way both the machine learning techniques have the same information and are trained in the same way. Also, the results are given as a misuse system so that the difference between each attack catagory can been seen and more information of the classification is visible and the TPR and FPR of every attack catagory can be compared.

8.3.1 Results of the Multi–Layer Perceptron

Table 25: Results of the MLP payload–based misuse system

Algorithm	Overall detection rate	Avg. TPR	Avg. FPR	Benign TPR	Benign FPR
MLP	78.29%	78.3%	4.5%	88.4%	3.7%

Table 26: Multilayer perceptron (10 fold cross validation) payload–based misuse system

Normal	Backdoor	Analysis	Fuzzers	Shellcode	Reconnaissance	Exploits	DoS	Worms	Generic	← *Classified as*
49484	1	68	5200	16	342	858	14	0	17	**Normal**
12	3	0	41	4	178	1264	244	0	0	**Backdoor**
238	0	165	8	2	13	1302	267	0	5	**Analysis**
3331	0	9	11223	65	729	2473	272	0	82	**Fuzzers**
33	0	1	206	208	449	226	10	0	0	**Shellcode**
291	0	1	284	2	7185	2376	346	0	6	**Reconnaissance**
425	0	19	1239	77	1053	28394	2157	0	29	**Exploits**
72	2	3	416	50	365	9692	1646	0	18	**DoS**
0	0	0	18	1	20	89	2	0	0	**Worms**
41	0	0	249	6	53	642	48	0	38961	**Generic**

8.3.2 Results of the J48 Decision Tree

Table 27: Results of both misuse systems

Algorithm	Overall detection rate	Avg. TPR	Avg. FPR	Benign TPR	Benign FPR
J48	88.64%	86.6%	2.4%	98.9%	0.6%

Table 28: J48 Tree (10 fold cross validation) payload–based misuse system

Normal	Backdoor	Analysis	Fuzzers	Shellcode	Reconnaissance	Exploits	DoS	Worms	Generic	← *Classified as*
55391	0	11	465	13	15	91	13	1	0	**Normal**
1	411	20	22	16	33	864	374	2	3	**Backdoor**
25	107	430	15	0	4	1029	387	0	3	**Analysis**
522	28	33	15621	121	53	1347	444	0	15	**Fuzzers**
21	13	0	130	781	14	125	47	0	2	**Shellcode**
9	61	10	63	2	7906	1905	531	1	3	**Reconnaissance**
73	86	114	461	151	664	28592	3113	41	98	**Exploits**
22	68	68	149	82	118	8421	3283	2	51	**DoS**
0	0	0	6	0	2	40	2	78	2	**Worms**
1	4	3	31	12	5	263	254	2	39425	**Generic**

8.3.3 Comparison of the systems

The J48 decision tree exhibited superior performance in classification over the MLP with respectively a 86.64 percent and a 78.29 percent correctly classified instance rate. However, this result includes the wrong classifications of all types of instances. The most important are the true negatives and the false positives in indications of normal behaviour. The dataset consists of 56000 normal behaviour and 119341 malicious behaviour instances. The J48 tree predicted 55391 times normal instances correctly whereas the MLP only did 49484 times, or a 98.9 percent true positive versus a 88.4 percent. This means that the rate in which the systems classified normal behaviour as attacks, the amount of false alarms, is 1.09 percent from the J48 tree versus a 11.64 percent of the MLP. Moreover, on every attack category, the decision tree scored better with a higher TPR. The decision tree let 674 attacks through, or a 0.56 percent of the attacks and the MLP let 4443 attacks pass, a 3.72 percent. Another interesting thing that can be seen in table 26 is that the MLP completely failed to recognize worm attacks in any way and did not classify any instance as this type. The following table shows the classification rates on every attack type of both the flow–based and the payload–based systems.

Note: The flow–based system and the payload–based systems proposed here should not be compared for the reason that the algorithms in table 29 are trained with the provided training set and the algorithms in table 30 with 10 fold cross validation on the test set. The only comparison is between the two different machine learning techniques.

Table 29: Flow–based misuse systems

MLP	**J48**	
97.3%	97.3%	**Normal**
0.0%	72.4%	**Reconnaissance**
0.0%	6.0%	**Backdoor**
0.0%	65.8 %	**DoS**
87.4%	57.1%	**Exploits**
0.0%	0.0%	**Analysis**
0.1%	13.5%	**Fuzzers**
0.0%	56.9%	**Worms**
0.0%	51.8%	**Shellcode**
97.8%	98.6%	**Generic**
70.2%	75.2%	**Weighted Average**

Table 30: Payload–based misuse systems

MLP	**J48**	
88.4%	98.9%	**Normal**
0.2%	23.5%	**Reconnaissance**
8.3%	21.5%	**Backdoor**
61.7%	85.9 %	**DoS**
18.4%	68.9%	**Exploits**
68.5%	75.4%	**Analysis**
85.0%	85.6%	**Fuzzers**
13.4%	26.8%	**Worms**
0.0%	60.0%	**Shellcode**
97.4%	98.6%	**Generic**
78.3%	86.6%	**Weighted Average**

9 Discussion

The following paragraphs will elaborate on the problems regarding the conducted experiments in this research. These are known problems in this field of research and seem to be hard to overcome. Afterwards, some ideas for future work, that arose during this research, are shared to contribute to the achievement of adaptive cyber security.

9.1 Flow–Based Anomaly Detection versus Payload–Based Anomaly Detection

The results derived from the UNSW–NB15 dataset for the first experiment are in some ways promising. The MLP only let 26 attacks through as the payload based let 27 attacks pass. This almost 100 percent true positive rate on the classification of attacks seems to be a good IDS. However, with such a high false alarm rate the implementation of such a system is rather impractical. With 11735 false alarms, a 21 percent false positive rate, the system is too severe for real world deployement since it would also generate too much false alarms. However, it does conclude too little difference in performance between the network classification techniques. Other research also prefers the flow–based systems since they can handle the growth in internet bandwidth and still can achieve high detection rates [18, 30, 31]. A problem regarding this conclusion however is the feature selection. Since almost all of the relevant features shown in table 3 are flow features, the payload–based system has almost no extra information for this classification problem. The only two additional features are *is_ftp_login* (describes if the ftp session is accessed by user and password) and *ct_ftp_cmd* (desribes the number of flows that has a command in ftp). With so little addition, the systems are too similair for a good comparison between a flow–based and payload–based system. For this reason, Winters dataset was used to check a possible flow–based system with less features. The results from Winters data are promising since the detection rates are high and the false alarm rates low. There are however still problems and it does not guarantee the same results in a real world deployment. One of the reasons for this is that the set is still not representative as a real world environment. The set consists of a small number of network flows and therefore can't be compared to the diversity of data that can arise out in the real world. The system might have a high false positve rate if deployed and would therefore generate too much false alarms for the network manager to handle.

9.2 Anomaly Detection versus Misuse Detection

The results in table 21 show a unbalanced neural network that classified all instances as either normal behaviour, exploits or generic attacks. Neural networks however seem to be better in classifiying instances with more classes as possible outcome. Anomaly detection often shows high false alarm rates[2], as can be seen in the results as well. The problems in the comparison between a anomaly detection and misuse detection system are big however and hard to overcome. The main problem is the lack of publicy available representative datasets. For a more valuable conclusion, more datasets should be used to compare both systems. Another way to improve the results is to combine the testing and training set and train the system with n–fold–cross validation instead of only leaning the algorithms on the training set.

9.3 Multilayer Perceptron versus J48 Tree

This experiment was done with all the features and the same training technique for the right comparison. The payload–based J48 tree showed good results on both the TPR and FPR on the benign behaviour. This system had a merely 1.1 percent false alarm rate and only failed to detect 0.6 percent of the attacks. The decision tree classified every attack type with higher percentages. For a better conclusion however, more datasets should be used to conclude that a decision tree is superior in classifying network data. With only one dataset for the comparison of two machine learning techniques, this conclusion could be misguided and incorrect.

10 Conclusion

In this paper, anomaly detection has been chosen over misuse detection, due to the failure of misuse detection to detect new attacks. A multi–layer perceptron was selected for the ability to adapt to a changing environment, resilience to noisy information, and fault tolerance. A J48 decision tree was chosen for the possible insight in data processing, resilience to uncertain or missing data and a proven high detection accurracy. A flow–based, rather than payload–based system is proposed, due to the ability of the former to keep up with growing network traffic, and protect privacy. The results of the first experiment show that a flow–based system does not perform inferior compared to a payload–based system. Also, seen from the results derrived from Winters dataset, with right feature selection a high detection rate and a low false alarm rate are achievable. Flow–based systems seem to be, apart from inevitable, the right way for IDS in the future.

The comparison between the J48 MDS and the J48 ADS showed little difference in the classification results. The MDS with the use of a MLP scored better than the anomaly detection regarding the false alarm rate but decreased evidently on the permeability. Instead of 26 successfully penetrating attacks (0.02 percent), a rough 30154 attacks (25.26 percent) went through undetected. As mentioned earlier, succesful penetrations can inflict enormous damage to systems and could result in huge economic losses, invasion of privacy or data loss. For these reasons, undetected attacks are the most important indicator of the performance of IDS. With a high false alarm rate of 20.96 percent on the flow–based ADS and a 25.26 percent of undetected attacks on the MDS, both systems seem unfit for deployment in a real world environment.

The J48 decision tree performed superior as a classification technique for network traffic than the MLP. The detection rate of 98.9 percent is promising whereas a 88.4 percent detection rate is far away from realization in the outside world. Furthermore, the 0.56 versus a 3.72 percent on the undetected attacks and a 1.09 versus a 11.64 as false alarm rate shows the difference in the overall performance of these systems. The research of Y. Bouzida [33] showed the superior performance of decision trees over a MLP in detecting new attacks, which could support the conclusion that this machine learning technique is indeed the right direction for adaptive cyber security. Also the work of M. Soysal[47] achieved better detection results and a shorter training time with a decision tree than a MLP.

11 Future Work

This research still encourages the creation of a new dataset for the development of IDS, since this is one of the major problems regarding learning algorithms for adaptive cyber security. The research conducted by Robin Summer and Vern Paxson called "Outside the Closed World: On Using Machine Learning For Network Intrusion Detection"[31], indicates the problems with machine learning for the development of IDS very clear.

As for future research, a Radial Basis Function neural network could be compared with the neural network with back propagation. Also fuzzy techniques could be combined with the neural network to see if this improves the results and reduces the training time. Since numeric attributes can be converted to linguistic values it is expected to drop in training time. Also as research show, fuzzy rules can be very usefull for the classification of the vague network data, since the values dont have clear boundaries. For an adaptive algorithm as a cyber security system, anomaly detection is the right way to achieve Adaptive Cyber Security in the future, since misuse detection fails to adapt to this fast changing environment. Also flow–based is necessary to keep up with the growing data in this environment, even if payload–based achieves superior results. Right feature selection or combination could result in a flow–based anomaly IDS with high detection rates and a low false alarm rate. Another possibility is the combination of the techniques whereas the payload–based inspects random packets in the flows afterwards, so that certain attacks like worms could be detected more often.

In the last years special interest arised for the combination of certain computational techniques like neural networks and fuzzy logic[40, 49] or neural networks and an artificial immune algorithm[47]. Nowadays statistics are the most commonly used technique for IDS [2]. The disadvantage of most data mining techniques, like neural networks or the association rule induction[14], is the time to learn the model with big data. There is some work done to improve the learning time, such as quantitative associations, in which, e.g integers and numerical attributes, get a domain to reduce the number of inputs. Another way to improve learning time is with the combination of a fuzzy approach [23],[14], so that crisp values turn into fuzzy linguistic values. This seems to be the right way for the classification of network data since the problem involves many numeric attributes, which results in high detection errors (true false or negative true). And secondly, the boundaries between the values are not well defined, which makes it hard to build a model for normal data. A neuro–fuzzy system can generate the fuzzy rules to classify the vague network data in the right way and build a model for normal behaviour, for example with a Fuzzy C-means algorithm[24]. Most fuzzy systems however, still make use of human knowledge to create the fuzzy rule base, but lack in self-learning. A neuro–fuzzy system seems to be a successful solution to overcome this problem of need for human intervention and should therefore be a next area of research [25, 16].

References

[1] Dayuan CAO, *Intrusion Detection Technology*, R Beijing: People Post Press, (2007).

[2] E. Biermann, E.Cloete , L.M.Venter *A comparison of Intrusion Detection systems*, Computers & Security, 20 , (2001), pp. 676–683.

[3] Matthew V. Mahoney and Philip K. Chan, *An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection*, Recent Advances in Intrusion Detection Vol. 2820, (2003), pp. 220–237.

[4] M. Sabhnani and G Serpen, *Why machine learning algorithms fail in misuse detection on KDD intrusion detection data set*, Intelligent Data Analysis 8, (2004), pp. 403–415.

[5] V O Ferreira, V V Galhardi, L B L Gonalves, R C Silva and A M Cansian, *A model for anomaly classification in intrusion detection systems*, Journal of Physics: Conference Series, Volume 633, (2015)

[6] J. Cannady, *Applying CMAC–based on–line learning to intrusion detection*, Proceedings of the IEEE–INNS–ENNS International Joint Conference on Neural Networks (IJCNN 00) vol 5, (2000), pp. 405–410.

[7] Xiaotao Wei, Houkuan Huang, and Shengfeng Tian, *A Modied RBF Neural Network for Network Anomaly Detection*, ISNN 2006, LNCS 3973, (2006), pp. 261–266.

[8] J. Jiang, C. Zhang, M. Kane *RBF–based real–time hierarchical intrusion detection systems*, Proceedings of the international Joint Conference on Neural Networks (IJCNN 03) vol 2, (2003), pp. 1512–1516.

[9] A.P.F. Chan, W.W.Y. Ng, D.S. Yeung, E.C.C. Tsang *Comparison of different fusion approaches for network intrusion detection using ensemble of RBFNN*, Proceedings of 2005 International Conference on Machine Learning and Cyernetics vol 6, (2005), pp. 3846–3851.

[10] Andrew W. Moore and Konstantina Papagiannaki *Toward the Accurate Identification of Network Applications*, Passive and Active Network Measurement: 6th International Workshop, PAM 2005, Boston, MA, USA, March 31 –April 1, 2005. Proceedings, (2005) pp. 41–54.

[11] Ke Wang, Salvatore J. Stolfo, *Anomalous Payload-Based Network Intrusion Detection*, proceedings, Lecture Notes in Computer Science, vol. 3224, (2004), pp. 203–222.

[12] Suseela T. Sarasamma, Qiuming A. Zhu, and Julie Huff, *Hierarchical Kohonenen Net for Anomaly Detection in Network Security* , Computer Science Faculty Publications Paper 29, (2005)

[13] H. Günes Kayacik, A. Nur Zincir–Heywood, Malcolm I. Heywood, *Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets*, Third Annual Conference on Privacy, Security and Trust, (October 2005)

[14] Arman Tajbakhsh, Mohammad Rahmati, Abdolreza Mirzaei, *Intrusion detection using fuzzy association rules*, Applied Soft Computing 9, (2009), 462–469

[15] Nour Moustafa, Jill Slay *The significant features of the UNSW-NB15 and the KDD99 Data sets for Network Intrusion Detection Systems*, Proceedings of the 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security , (November 2015)

[16] Shelly Xiaonan Wu, Wolfgang Banzhaf, *The use of computational intelligence in intrusion detection systems: A review*, Applied Soft Computing 10, (2010), pp. 1–35

[17] Mario Golling, Robert Koch, Rick Hofstede, *Towards Multi–layered Intrusion Detection in High–Speed Networks*, 6th International Conference on Cyber Conflict, (2014),

[18] Anna Sperotto, Aiko Pras, *Flow–Based Intrusion Detection*, Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management, (May 2011), pp. 23–27

[19] M. Moradi, M. Zulkernine, *A Neural network based system for intrusion detection and classification of attacks*, Proceeding of the 2004 IEEE International Conference on Advances in Intelligent Systems–Theory and Applications, (November 2004), p. 15–18

[20] Mohammed Reza Norouzian, Sobhan Merati, *Classifying Attacks in a Network Intrusion Detection System Based on Artificial Neural Networks*, International Journal of Scientific and Engineering Research Volume 4, (Februari 2013)

[21] Anna Sperotto, Sadre, R., van, F. V., and Pras, A., *A Labeled Data Set For Flow-based Intrusion Detection* IP Operations and Management, Lecture Notes in Computer Science 5843. Berlin (October 2009), pp. 39–50

[22] J. McHugh, *Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory* ACM Transactions on Information and System Security, vol. 3, no. 4, (2000) pp. 262–294

[23] C. Kuok, A. Fu, M. Wong, *Mining fuzzy association rules in databases*, SIGMOD Record 27 (1), (1998), pp. 41–46

[24] James C. Bezdek, Robert Ehrlich, William Full, *FCM: The Fuzzy C-means clustering algorithm*, Computers and Geosciences Vol. 10, No. 2-3, (1984), pp. 191–203

[25] J.S.R. Jang, *ANFIS: adaptive-network-based fuzzy inference systems*, IEEE Transactions on Systems, Man, and Cybernetics 23 (3), (1993), pp. 665–685

[26] Philipp Winter, Eckehard Hermann, Markus Zeilinger , *Inductive Intrusion Detection in Flow-Based Network Data using One-Class Support Vector Machines*, 4th IFIP International Conference on New Technologies, Mobility and Security, (2011)

[27] J.Cannady , *Artificial Neural Networks for Misuse Detection*, National Information Systems Security Conference, (1998), pp. 443–456

[28] K.Tan, *The application of neural networks to unix computer security*, Proceedings of IEEE International Conference on Neural Networks vol 1, (1995), pp. 476–481

[29] A.K.Gosh, A.Schwarzbard, *A study in using neural networks for anomaly and misuse detection*, Proceedings of the 8th USENIX Security Symposium vol 8, (1999), pp. 141–152

[30] Prasanta Gogoi, Monowar H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, *Packet and Flow Based Network Intrusion Dataset*, Contemporary Computing: 5th International Conference, (2012), pp. 322–334

[31] Hashem Alaidaros, M.Mahmuddin, A.Al Mazari , *An overview of flow–based and packet–based intrusion detection performance in high speed networks*

[32] Robin Sommer, Vern Paxson *Outside the Closed World: On Using Machine Learning For Network Intrusion Detection*, In Proc. of IEEE Symposium on Security and Privacy, (2010), pp. 305–316

[33] Yacine Bouzida, F. Cuppens *Neural networks vs. decision trees for intrusion detection*, IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM). Vol. 28, (2006)

[34] C. Elkan *Results of the KDD 99 classifier learning*, ACM SIGKDD Explorations Newsletter 1, (2000), pp. 63–64

[35] Fu, L. *A Neural Network Model for Learning Rule-Based Systems*, Proceedings of the International Joint Conference on Neural Networks, (1992), pp. 343–348

[36] N. Bhargava, G.Sharma, R.Bhargava, M.Mathuria *Decision Tree Analysis on J48 Algorithm for Data Mining*, International Journal of Advanced Research in Computer Science and Software Engineering, (2013)

[37] Amr Ali-Eldin, J. van den Berg, H. A. Ali *A risk evaluation approach for authorization decisions in social pervasive applications*, Computers and Electrical Engineering, (2016) pp. 1–15

[38] P.Carcia–Teodoro, J. Diaz–Verdejo, G. Macia–Fernandez, E. Vasquez *Anomaly–based network intrusion detection: Techniques, systems and challenges*, Computer and Security 28, (2009) pp. 18–28

[39] Tom Auld, Andrew W. Moore, Member, IEEE, and Stephen F. Gull *Bayesian Neural Networks for Internet Trafc Classication*, IEEE Transactions on Neural Networks Vol.18 No.1 (2007)

[40] Muhammad Fermi Pasha, Rahmat Budiarto, Mohammad Syukur and Masashi Yamada, *EFIS: Evolvable-Neural-Based Fuzzy Inference System and Its Application for Adaptive Network Anomaly Detection*, Advances in Machine Learning and Cybernetics, (2005) pp. 662–671

[41] Gulshan Kumar *Evaluation Metrics for Intrusion Detection Systems - A Study* , International Journal of Computer Science and Mobile Applications, (2014) pp. 11–17

[42] Alan Piszcz, Nicholas Orlans, Zachary Eyler–Walker and David Moore, *Engineering Issues for an Adaptive Defense Network*, Mitre Technical Report, (2001)

[43] S. Revathi, Dr. A. Malathi, *A Detailed Analysis on NSL–KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection*, International Journal of Engineering Research and Technology (IJERT), (2013)

[44] He Liang , *An Improved Intrusion Detection based on Neural Network and Fuzzy Algorithm*, Journal of Networks Vol. 9 No.5, (2014)

[45] Adel Nadjaran Toosi, Mohsen Kahani *A new approach to intrusion detection based on an evolutionary soft computing model using neuro–fuzzy classifiers*, Computer Communications 30, (2007), pp. 2201–2212

[46] Wu J, Peng D, Li Z,Zhao L,Ling H *Network Intrusion Detection Based on a General Regression Neural Network Optimized by an Improved Artificial Immune Algorithm*, PLoS ONE 10, (2015)

[47] Murat Soysal, Ece Guran Schmidt *Machine learning algorithms for accurate flow–based network traffic classification: Evaluation and comparison*, Performance Evaluation 67, (2010), pp. 451–467

[48] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani *A Detailed Analysis of the KDD CUP 99 Data Set*, Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications, (2009)

[49] Dahlia Asyiqin Ahmad Zainaddin and Zurina Mohd Hanapi *Hybrid of Fuzzy Clustering Neural Network over NSL Dataset for Intrusion Detection System*, Journal of Computer Science 9 (3) (2013) pp. 391–403

[50] Zahra Jadidi , Vallipuram Muthukkumarasamy, Elankayer Sithirasenan and Mansour Sheikhan *Flow–Based Anomaly Detection Using Neural Network Optimized with GSA Algorithm*, IEEE 33rd International Conference on Distributed Computing Systems Workshops (2013)

[51] Yousef Abuadlla, Goran Kvascev, Slavko Gajin, and Zoran Jovanovi *Flow-Based Anomaly Intrusion Detection System Using Two Neural Network Stages*, Computer Science and Information Systems 11(2) pp. 601–622

[52] Nour Moustafa, Jill Slay *UNSW–NB 15: a comprehensive data set for network intrusion detection systems (UNSW–NB15 network data set*, Military Communications and Information Systems Conference (MilCIS), (November 2015)

12 Appendix

Table 31: Flow Features

No.	Name	Type	Description
1	srcip	nominal	Source IP address
2	sport	integer	Source port number
3	dstip	nominal	Destination IP address
4	dsport	integer	Destination port number
5	proto	nominal	Transaction protocol

Table 32: Basic Features

6	state	nominal	Indicates to the state and its dependent protocol
7	dur	Float	Record total duration
8	sbytes	Integer	Source to destination transaction bytes
9	dbytes	Integer	Destination to source transaction bytes
10	sttl	Integer	Source to destination time to live value
11	dttl	Integer	Destination to source time to live value
12	sloss	Integer	Source packets retransmitted or dropped
13	dloss	Integer	Destination packets retransmitted or dropped
14	service	nominal	http/ftp/smtp/ssh/dns/ftp-data/irc and (–) if not much used service
15	Sload	Float	Source bits per second
16	Dload	Float	Destination bits per second
17	Spkts	integer	Source to destination packet count
18	Dpkts	integer	Destination to source packet count

Table 33: Content Features

19	swin	integer	Source TCP window advertisement value
20	dwin	integer	Destination TCP window advertisement value
21	stcpb	integer	Source TCP base sequence number
22	dtcpb	integer	Destination TCP base sequence number
23	smeansz	integer	Mean of the flow packet size transmitted by the src
24	dmeansz	integer	Mean of the flow packet size transmitted by the dst
25	trans_depth	integer	Represents the pipelined depth into the connection of http request/response transaction
26	res_bdy_len	integer	Actual uncompressed content size of the data transferred from the servers http service

Table 34: Time Features

27	Sjit	Float	Source jitter (mSec)
28	Djit	Float	Destination jitter (mSec)
29	Stime	Timestamp	record start time
30	Ltime	Timestamp	record last time
31	Sintpkt	Float	Source interpacket arrival time (mSec)
32	Dintpkt	Float	Destination interpacket arrival time (mSec)
33	tcprtt	Float	TCP connection setup round–tip time. The sun of the synack and ackdat.
34	synack	Float	TCP connection setup time. The time between the SYN and the SYN_ACK packets.
35	ackdat	Float	TCP connection setup time. The time between the SYN_ACK and the ACK packets.

Table 35: Additional Generated Features

36	is_sm_ips_ports	Binary	If source (1) equals to destination (3) IP addresses and port numbers (2)(4) are equal, this variable takes value 1 else 0
37	ct_state_ttl	Integer	No. for each state (6) according to specific range of values for source/destination time to live (10) (11).
38	ct_flw_http_mthd	Integer	No. of flows that has methods such as Get and Post in http service.
39	is_ftp_login	Binary	If the ftp session is accessed by user and password then 1 else 0.
40	ct_ftp_cmd	Integer	No of flows that has a command in ftp session.
41	ct_srv_src	Integer	No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26).
42	ct_srv_dstt	Integer	No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26).
43	ct_dst_ltm	Integer	No. of connections of the same destination address (3) in 100 connections according to the last time (26).
44	ct_src_ ltm	Integer	No. of connections of the same source address (1) in 100 connections according to the last time (26).
45	ct_src_dport_ltm	Integer	No of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26).
46	ct_dst_sport_ltm	Integer	No of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26).
47	ct_dst_src_ltm	Integer	No of connections of the same source (1) and the destination (3) address in in 100 connections according to the last time (26).