NETWORK INTRUSION DETECTION USING MACHINE LEARNING

by

Seyed Pedrum Jalali Mosallam

Master of Science in Structural Engineering from Sharif University of Technology, 2014

Bachelor of Science in Civil Engineering from Sharif University of Technology, 2011

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the program of

Computer Science

Toronto, Ontario, Canada 2018

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

Abstract

Network Intrusion Detection Using Machine Learning

Seyed Pedrum Jalali Mosallam, Master of Science in Computer Science

Ryerson University, 2018

In this research we have studied the use of machine learning techniques in detecting network intrusions. Most research in the field has used the very outdated dataset (KDDCup99) which consists of a set handcrafted features. In our research we present models that work well on both the older dataset and on newer datasets such as ISCX2014 and ISCX2012. We also present methods for extracting features from these datasets. Another issue we found with most research in this field is that they do not study the effect of surges in regular network traffic and how that might affect the model. We put our model to test in 10x traffic and show its effectiveness under these conditions. We also study how semi-supervised models can be used in training NIDS models without directly showing them labeled data.

Acknowledgement

We thank NSERC and J-SAS Inc. for the funding of this research. We also thank the Canadian Institute of Cybersecurity with the University of New Brunswick for providing us with the datasets to train and test our models.

Contents

Abstract iii

List of Tables viii

List of Figures ix

List of Equations xi

1 Introduction 1

1.1 Motivation ... 3

1.2 Research Statement ... 3

1.3 Novelty and Contribution ... 4

1.4 Organization ... 5

2 Attack Taxonomy 7

2.1.1 TCP/IP Hijacking ... 7

2.1.2 Denial of Service Attacks [4] ... 7

2.1.2.1 TCP SYN Flood Attack [5] ... 7

2.1.2.2 TCP Reset Attack [6] ... 8

2.1.2.3 UDP Flood Attack ... 9

2.1.2.4 ICMP Attack ... 9

2.1.2.5 The ping of Death ... 10

2.1.2.6 Teardrop ... 10

2.1.2.7 CGI Attack ... 10

2.1.2.8 Mail Bomb Attack ... 10

2.1.3 Amplification Attacks ... 11

2.1.4 Distributed Dos Flooding ... 11

2.1.5 Port Scans [3] ... 12

2.1.5.1 Stealth Syn Scan ... 12

2.1.5.2 FIN, X-Mas and Null Scans ... 12

2.1.5.3 Spoofing Decoys ... 12

2.1.5.4 Idle Scanning ... 13

3 Machine Learning Review [7] 14

3.1 Logistic Regression ... 14

3.1.1 Threshold ... 14

3.1.2 Non-Numeric Predictors 16
3.1.3 Error Calculation 17
3.1.4 Model Validation 19
3.1.5 Feature Selection 22
3.2 Linear Discriminant Analysis (LDA) 25
3.3 Quadratic Discriminant Analysis (QDA) 26
3.4 K-Mean Clustering Algorithm 26
4 Literature Review 28
4.1 Software Exploitation 29
4.2 Sampled Packets 30
4.3 Web Related Classification Methods 32
4.4 NIDS Using Machine Learning 33
4.5 Dataset Preparation 35
4.6 Network Attack Through Software Exploitation 36
4.7 Anomaly Detection 36
5 Statistical Model of Network Under Attack (Part 1) 39
5.1 Dataset 39
5.2 Theory 39
5.2.1 Stage 1, Generating Clean Traffic Model 40
5.2.2 Stage 2, Generating The 20 Minute Probability Distribution 41
5.2.3 Stage 3, Unfiltered Traffic Distribution 43
5.3 Attack Detection 45
5.4 Results 45
5.5 Issues 46
6 Supervised Model of Network Under Attack (Part 2) 48
6.1 Dataset 48
6.1.1 ISCX2012 Dataset 49
6.1.2 Darpa 1998 Dataset 50
6.2 Model 52
6.2.1 Predictors 52
6.2.1.1 Application Type 53

6.2.1.2 Protocols 55
6.2.1.3 Unique Local and Remote IPs 56
6.2.1.4 Direction 57
6.2.1.5 Packet Count 57
6.2.1.6 Bytes Transferred 57
6.2.2 Machine Learning Models 57
6.2.3 Intervals 57
6.2.3.1 Feature Selection 58
6.2.3.2 Low Count of Attack Records 58
6.2.3.3 Sensitivity and Specificity 59
6.2.3.4 Binary or Multiple Classes 59
6.2.3.5 Naming convention 60
6.2.3.6 Training, Validation and Test Set 67
6.3 Surge Test 68
6.4 Implementation 69
6.4.1 ISCX2012 Dataset 70
6.4.1.1 Phase 1, Obtaining the Dataset 70
6.4.1.2 Phase 2, Setting Up the Database 71
6.4.1.3 Phase 3, Data Generation 72
6.4.1.4 Phase 4, Running the Machine Learning Models 73
6.4.1.5 Phase 5, Unseen Traffic and Attack Evaluation 74
6.4.1.6 Phase 6, Surge Test 75
6.4.2 Darpa Dataset 76
6.4.2.1 Phase 1, Obtaining the Dataset 76
6.4.2.2 Phase 2, Setting Up Database 77
6.4.2.3 Phase 3 Flow Generation: 78
6.4.2.4 Phase 4, Matching Labels 80
6.4.2.5 Phase 4, Post Process 81
6.4.2.6 Step 5, Applying Surge, Generating Features and and Running Model 83
6.5 Deployment and Architecture 83

6.6 Results 84
6.6.1 ISCX 2012 Dataset 84
6.6.1.1 Model Evaluation 89
6.6.1.2 Interpretation of the Results 91
6.6.2 Surge Test 98
6.6.3 Comparison with Other Papers 101
7 Semi-Supervised Model of Network Under Attack (Part 3) 103
7.1 Dataset 103
7.1.1 ISCX2012 Dataset 103
7.1.2 ISCX2014 Dataset 103
7.2 Model 104
7.3 Training Phase 105
7.3.1 Step 1, Obtaining Data Points 105
7.3.2 Step 2, Clustering 106
7.4 Detection Phase 107
7.4.1 Step 1, Obtaining Data Points 107
7.4.2 Step 2, Cluster Assignment 107
7.4.3 Step 3, Attack Detection 107
7.4.3.1 Test 1 108
7.4.3.2 Test 2 108
7.5 Implementation 108
7.6 Results 109
8 Conclusion and Future Work 112
Appendix A, Interpreting the Results of the Semi-Supervised Model 114
A.1 Applications 114
A.2 Protocols 117
A.3 Local Vs Remote Nodes 118
A.4 Direction 119
A.5 Other 121
References 123

List of Tables

Table 1 The different models considered in the research 35

Table 2 Results 46

Table 3 Results 46

Table 4 Different applications used in the ISCX2012 dataset 54

Table 5 Different protocols used in the ISCX2012 dataset 55

Table 6 Sample flows 56

Table 7 Naming convention 60

Table 8 The different logistic models tested 61

Table 9 The different linear discriminant and quadratic discriminant analysis models tested 63

Table 10 Results 85

Table 11 Top 20 90

Table 12 Day 7 Results 91

Table 13 Comparison with other work 102

Table 14 ISCX2014 top models based on threshold 111

Table 15 ISCX2012 top models based on threshold 111

List of Figures

Figure 1, Data point generated from linear model with noise 19

Figure 2 Linear regression 20

Figure 3 Higher order regression 20

Figure 4: Markov chain model used in analysis. Some actions have been omitted for clarity..... 40

Figure 5 Clean traffic log probability distribution 43

Figure 6 cumulative clean traffic log probability 43

Figure 7 Unfiltered traffic log probability distribution 44

Figure 8 difference in probability buckets between clean and unfiltered traffic 45

Figure 9 Obtaining the dataset 70

Figure 10 Setting up the database 71

Figure 11 Data generation 72

Figure 12 Running the machine learning models 73

Figure 13 Test set 74

Figure 14 Surge test 75

Figure 15 Obtaining the dataset 76

Figure 16 Setting up the database 77

Figure 17 Flow generation 79

Figure 18 Maching labels 80

Figure 19 Post Process 81

Figure 20 Deployment Architecture 83

Figure 21 Effects of different interval sizes 91

Figure 22 Ratio of interval size selected by the top 20 models 92

Figure 23 Ratio of top 20 models that considered copying the attack records 93
Figure 24 Ratio of the top 20 models that the feature evaluation was based on the attack records 94
Figure 25 Ratio of the top 20 models that the feature evaluation was based on the validation set 95
Figure 26 Predictors selected in the top 20 models 96
Figure 27 Performance deterioration with increase in normal traffic using the ratio with the prior all method..... 98
Figure 28 Performance deterioration with increase in normal traffic using the ratio method and the F1 score 99
Figure 29 Performance deterioration with increase in normal traffic using full flow counts and the prior all methods 100
Figure 30 Performance deterioration with increase in normal traffic using the full flow counts and the f1 score 100
Figure 31 Obtaining the data points..... 105
Figure 32 Data points..... 106
Figure 33 Results 110

List of Equations

Equation 1 14

Equation 2 17

Equation 3 18

Equation 4 18

Equation 5 19

Equation 6 22

Equation 7 22

Equation 8 23

Equation 9 23

Equation 10 23

Equation 11 24

Equation 12 24

Equation 13 25

Equation 14 27

Equation 15 27

Equation 16 30

Equation 17 42

Equation 18 101

Equation 19 101

Equation 20 101

Equation 21 101

Equation 22 101

Equation 23 101

Equation 24 102

Acronyms

NIDS: Network Intrusion Detection System

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

IPV4: Internet Protocol Version 4

MAC: Media Access Control Address

CSV: Comma Separated File

DDOS: Distribute Denial of Service

DOS: Denial of Service

1 Introduction

Today we have become more dependent on our computers and the internet than ever before. Hospitals, banks, businesses are all connected to the internet. While this connectedness has facilitated a lot of our daily tasks it has also opened up the doors to security issues. As attacks become more and more sophisticated it become clear that we need better methods for tackling these attacks. In order to protect us against network attacks a set of tools have been made. They generally fall into two categories:

- Intrusion Detection Systems
- Intrusion Prevention Systems

Intrusion detection systems are used to detect an attack. They do not take any action against the attack though. They could be used to notify system admins or to trigger another software to stop the attack. Intrusion prevention systems on the other are used in stopping the attack once it has been detected. These two systems are normally used together.

The topic of this research is mainly focused on intrusion detection. Generally, intrusion detection systems are classified into 3 different categories:

- Host Intrusion Detection Systems
- Signature Based
- Anomaly Based

Host intrusion detection systems are generally installed on hosts in the network. Information on the host is used in detecting attacks. The information could include cpu utilization, memory usage, files accessed, network connections, ... If changes are noticed from regular usage an attack would

be detected. One issue with these types of systems is that the intrusion detection system needs to be installed on all the hosts on the system, this may not be possible on all networks.

Signature intrusion detection systems are based on known attacks. In these types of systems, the network pattern is compared with a set of well known attacks. If a match is detected, then it is assumed that there is an attack. The problem with these types of systems is that there needs to be an exact match for the system to detect the attack.

In anomaly detection systems, different techniques are used to detect unseen attacks. They are generally divided into three categories:

- State based
- Supervised
- Unsupervised

In state, based methods a group of features regarding the network are considered. The probability of transitioning from one of these states to another for regular network traffic is considered. It is then compared to the observed probabilities of the actual network traffic. In Supervised models, regular and anomalous traffic is shown to the model. The model learns to differentiate between the two. When actual network traffic appears, depending on which pattern it is more similar to it will either be classified as attack or normal traffic. In unsupervised models only, normal traffic is shown to the model. If the actual traffic deviates more than a certain amount form the normal traffic, then it is considered an attack.

Three datasets were used in this research. The first being the Darpa intrusion detection data sets [1]. The second and third datasets were the 2012 and 2014 New Brunswick ISCX datasets [2]

1.1 Motivation

This research was part of an industrial project funded jointly by NSERC and J-SAS Inc. The object of this project was to find an AI model that would be able to detect attacks in the network environment. The model must be able to detect both local and external anomalous behaviour.

There are some issues with the current research in this field. To point out to a few:

- Most of the research in this field use a very outdated dataset (KDDCup99) which was generated for a competition in the year 1999. As network patterns have greatly changed over the years we believe that a practical AI model must be tested on newer datasets to prove its efficiency.
- The KDDCupp99 dataset consists of a set of 42 hand crafted features. However, there is much more information in network traffic that can be used towards building AI models. As most papers are using the KDDCup99 dataset they fail to utilize the vast amount of data available in network traffic. We demonstrate an architecture that can be used for extracting data from the incoming traffic and using them in our AI models.
- Another issue we find with research in this field is they do not take variations of network traffic into account. In other words, they don't take into account how well the model will perform if there are sudden increases in network traffic due to unpredictable events. As an example, consider a university campus network where all grades are released on the same day. On that day there will probably be a surge in network traffic. The method we proposed worked well both under regular traffic and increased traffic.

1.2 Research Statement

The research is divided into 3 different parts:

Part 1: Statistical model using the markov chain

Part 2: Supervised model

Part 3 Semi-Supervised

In order to find an effective machine learning model at detecting network attacks we first start off by testing a simple statistical method in detecting network attacks. In this part of the project we test a statistical based method using the markov chain that monitors the incoming and outgoing connections to a particular node in order to detect anomalies.

After running this model, we point out to some of the difficulties of using such models in this research and continue our research in a supervised machine learning direction. In the second part of the research we test different supervised learning methods in detecting attacks on two different datasets. We also compare the results with previous research and perform surge tests to see the model's effectiveness when there is an increase in regular traffic.

A perfect model would be a fully unsupervised model that only requires the regular network traffic to train on and would be able to detect attacks without being trained on them. Although we don't look into fully unsupervised models in this research, however in the last part of the research we test semi supervised models on two different datasets.

1.3 Novelty and Contribution

- Most of the research in this field use a very outdated dataset (KDDCup99) which was generated for a competition in the year 1999. For the supervised model we use both the KDDCupp99 dataset and the ISCX2012 dataset. We also compare our results with previous research and show that our model can work well on both the newer datasets and the older.

- For the semi-supervised model, we use the ISCX2012 and ISCX2014 datasets showing that our model can perform well on new datasets unlike previous research which have used an outdated dataset.
- For the supervised model, we apply surges to the traffic and show our model works well even under increased traffic, something that other research does not point out to.
- As most research in the field limit themselves to the KDDCupp99 dataset they fail to use the vast amount of information available in the network traffic. We demonstrate methods for extracting additional features from the network traffic.

1.4 Organization

The chapters have been organized as follows:

- In chapter 2 we explain the different types of threats and attacks that can be used by hacker to perform malicious activity.
- In chapter 3 we provide a brief review of some of the machine learning techniques that have been used in this research.
- In chapter 4 we provide a literature review of the different research that has been performed in this fields
- In chapter 5 we discuss the initial statistical model that was used in detecting network attacks and point out to some its drawbacks in the end
- In chapter 6 we discuss our supervised machine learning model, compare it with previous work and apply surges to test its effectiveness on increased traffic.
- In chapter 7 we test out a semi supervised method in detecting attacks.
- In chapter 8 we discuss the results and provide directions for future work

- In appendix A we have plotted the feature selection plots for the supervised learning models
- In appendix B we try to interpret the meaning of the clusters of the semi-supervised model of the research.
- In appendix C we plot the results of the different models tested in the semi-supervised model.

2 Attack Taxonomy

In this section we explain some of the more common attacks. The attack we aim at detecting during this research are mainly network attacks. Network Attacks [3]

2.1.1 TCP/IP Hijacking

This form of attack only works when the attacker can sniff the packets sent between the two hosts. This is usually the case when the attacker is on the same network as the victim. The way it works is the attacker constantly monitors the packets being sent between the two hosts. During this process the attacker keeps track of the acknowledgement and sequence numbers. It then sends a spoofed packet to the target host spoofed with the victims IP address. Since the sequence and acknowledgement numbers work out the target host will respond to the sent packet. Using this technique, the attacker can gain important information from the host.

2.1.2 Denial of Service Attacks [4]

In these type of attacks, the attacker does not gain any form of additional privileges or access to any form of restricted information. The attacker

2.1.2.1 TCP SYN Flood Attack [5]

In a typical TCP connection, a SYN packet is sent to the server. The server stores information about the node that has been trying to connect. It then responds with SYN ACK packet. The server then waits for the host to respond with an ACK packet, after which regular communication will start.

In a SYN attack the attackers send a large number of SYN packets to the server using a spoofed IP address. For each SYN packet sent the server adds a record in a table of hosts that it is waiting

on for an ACK. If the number of such packets increases beyond a certain amount then the table will be filled up and there will be no more room for legitimate requests.

Some of the methods currently employed to reduce the impact of such attacks includes:

- Filtering certain IPs: If the range of IPs that can legitimately connect to the server is limited, one option would be to use a firewall and only allow traffic through those IPs pass through.
- Increase Capacity: As memory becomes cheaper it become less of an issue to increase the size of the backlog table that stores the open connections.
- Reducing the timeout: Normally when a SYN-ACK packet is sent the server waits a certain amount of time before closing the connection. One option would be to reduce the timeout period to reduce the effectiveness of SYN attack.

2.1.2.2 TCP Reset Attack [6]

A TCP connection has a set of flags. One of such flags is the reset flag. The reset flag indicates that the host wishes that the other host close the connection. One scenario where this might occur is if the for any reason one of the hosts loses information about the connection. For example, the host crashes or is restarted. The other side of the TCP may not know that this has happened and would continue sending packets. In such a scenario, the host that has lost the relevant information about the connection would send a reset packet back to the other host indicating that it no longer has information about the connection and the connection should be closed.

An attacker could however exploit this mechanism to disrupt legitimate TCP traffic. This can be achieved by sending reset packets to either side of the connection and spoofing all other information. When the host receives the packet it will not be able to differentiate between the attacker and the actual host due to the fact that all other fields appear valid. It will then terminate

the connection.

One of the ways such attacks are countermeasure is by using IPSec layer encryption. Doing so would prevent the attacker form actually reading the TCP headers and being able to spoof the conversation.

2.1.2.3 UDP Flood Attack

When a server receives a UDP packet it first determines the port number of the packet. Once the port number has been figured out it then searches for the process that is listening on the port. If no such process is found, it then sends an ICMP packet back to the host that sent the UDP packet to notify it that the target port is closed.

An attacker could exploit this vulnerability by sending a large number of packets to the victim with random port numbers. The return address is normally spoofed so the attacker could remain anonymous. For each packet the host receives it needs to look up the port number, check the open sockets and fin a matching process and respond with an ICMP message. This could overload the target machine to a point that it may not be able to respond to legitimate users.

In order to reduce the impact of such attacks it is often recommended to disable the ICMP response mechanism and to only keep ports that are absolutely essential open.

2.1.2.4 ICMP Attack

This form of attack occurs when the attacker sends a large number of ICMP requests to the host. If the number of such requests exceeds a certain amount the hosts resources will be overloaded an therefore the host will be unable to respond to legitimate users.

2.1.2.5 The ping of Death

In an ICMP message it is assumed the size of the message is at most 65,536 bytes. There was a time where if you sent an ICMP message that was larger than the maximum specified size it would cause the system to crash. This type of attack shows the importance of not making any form of assumptions about the type of input that can be received from users or the outside world

2.1.2.6 Teardrop

When the size of the transmitting message is long it is fragmented into multiple packets. The packets are then reassembled at the destination. In the packet header, there is a field that specifies the offset with which the packets must be reassembled. In some of the older systems when these offsets did not align it would cause the system to crash.

2.1.2.7 CGI Attack

In this attack, the attacker first needs to find a cgi script located on the server. The attacker would then constantly invoke the cgi script using spoofed source address. This would cause system resources to be consumed to a point that the server becomes irresponsive. Cgi scripts can often be found in web applications with backend capabilities.

2.1.2.8 Mail Bomb Attack

Mail bomb attacks are generally performed against email servers. In these type of attacks, the attacker attempts to overload the email server with email messages to a point where it either become unresponsive because of the volume of incoming messages to process or it runs out space to store the emails. There are different variations of this type of attack. In the most simple type the attacker constructs a large number of emails and floods the emails server with the emails. This

attack is normally performed as distribute denial of service attack. A simple DOS attack could easily be detected due to the fact that the source address cannot be spoofed in this type of attack.

When a user wants to send an email, the email client looks up the ip address associated with the email. It the initiates the appropriate protocol with the target email server in order to send the email. The email server will be contacted regardless of whether the email exists or not. This will end up consuming the server's resources. One form of attack would be to register a large number of random email addresses from the email server's domain in a large list of subscription based mailing lists. The mailing lists would constantly send emails to the email server thus using up all of its resources.

2.1.3 Amplification Attacks

Some networks allow communication to the broad cast address. What happens in this case is the attacker will send a ICMP request to the broadcast address of this network. By doing this all the hosts in the network will receive the request. There might be hundreds of hosts on this network. Then source address will be set as the victims address. This will cause all the hosts on the broadcast network to send an ICMP request to the victim. By doing this the attacker will be amplifying his attack without using too much bandwidth of his own.

2.1.4 Distributed Dos Flooding

In this attack, the attacker first gains access to a set of hosts. Then using those hosts the attackers performs attacks towards the target machine.

2.1.5 Port Scans [3]

Before being able to perform any form of software exploits on the target host we need to figure out what applications are active on the target machine. This is usually done through port scans. Every application that requires network access normally operates on a set of port numbers. By figuring out what ports are open and listening we can gain an understanding as to what applications are active on the target host.

2.1.5.1 Stealth Syn Scan

In this type of port scan a SYN packet is sent over a range of ports to a node. If an application is listening on that port using the tcp protocol, it will respond with a SYN/ACK packet. If no SYN/ACK packet is received then it can be implied that the port is not open. Also a RST packet could be sent if a SYN/ACK packet is received in order to prevent a DOS attack.

2.1.5.2 FIN, X-Mas and Null Scans

In the FIN port scan a FIN packet is sent to a range of ports of a node. X-Mas sends a packet with FIN, URG and PUSH set and NULL sends a packet with no flags set. If the target port is closed the node will respond with an RST packet. If nothing is received it can be implied that the port is open.

2.1.5.3 Spoofing Decoys

One issue with port scans is that they are easily detectable. When packets are sent back to back from a single node to another node over a range of ports this can easily be detected. One technique that is used to make detection more difficult is interleaving some tcp packets with spoofed addresses in between the port scans.

2.1.5.4 Idle Scanning

Another way to avoid being detected is using idle scanning. In idle scanning, initially a target that is idle needs to be found. An idle target would be a node that is not sending or receiving too many packets. The attacker can then send a SYN packet to the victims spoofed with the address of the idle node. If the port is open the victim will respond with a SYN/ACK packet to the idle host. The attacker can then send a SYN/ACK packet to the idle node. If the identification number has increased that would indicate that the port was open, else the port was closed.

3 Machine Learning Review [7]

In this section we provide a review of some the machine learning techniques and theories used in this research

3.1 Logistic Regression

The logistic regression classifier is a machine learning technique used for classification. The classifier returns a value between zero and one, indicating the likelihood that the input predictors belongs to a particular class.

The general form of the classifier is shown below:

Equation 1

$$p = (Y = category\ 1, X, \beta)$$

In the equation above we have:

- p: The probability that X belongs to category 1
- X: The input predictors
- β: The coefficients of the model to be determined by training

In the binary case only one model is trained. However, in the case of multiple categories, a separate model needs to be fit for every category.

3.1.1 Threshold

As mentioned in the previous section the logistic regression classifier returns the probability that a data point belongs to a particular category. One way of classifying data points is to assign them to the category with the highest probability. For example, in the binary case we would assign the data point to the category where:

$if\ P(X\ belongs\ to\ category\ 1) > 0.5$

$X\ is\ a\ category\ 1\ point$

$else$

$X\ is\ a\ category\ 2\ point$

In a non-binary case the data point would be assigned to the category with the highest probability:

$maxP$

$= \max\{P(X\ belongs\ to\ category\ 1), P(X\ belongs\ to\ category\ 2), \dots, P(X\ belongs\ to\ category\ M)\}$

$if\ P(X\ belongs\ to\ category\ 1) = maxP$

$X\ is\ a\ category\ 1\ point$

$else\ if\ P(X\ belongs\ to\ category\ 2) = maxP$

$X\ is\ a\ category\ 2\ point$

…

However, in some cases we may deliberately change this threshold. For example, consider Category 1 the data points where a network attack is not happening and all other categories network attacks. Misclassifying Category 1 as an attack wouldn't be a big deal, it will only result in a false alarm. However, misclassifying an attack as category 1 could be devastating. Therefore, we can change the classification as follows:

$if\ P(X\ belongs\ to\ category\ 1) > 0.9$

$$X\ is\ a\ category\ 1\ point$$

$else$

$$X\ is\ a\ category\ 2\ point$$

As seen above only when we are more than 90% certain do we assign a data point to the non-attack category.

3.1.2 Non-Numeric Predictors

In the equations above it is assumed that the predictors are numeric. In other words, they are values such as:

- Packets count
- Packet size
- Connection Duration

However, there may be cases where the predictors are not numeric such as:

- IP Protocol
- Application type
- …

In such cases for each possible value we create a new predictor. Each predictor will be able to take on a value of one or zero. For example, consider the case for application type, let's assume the following applications are possible:

- Ping
- SSH

- MySQL
- HTTP

Therefor 4 predictors will be created one for each application:

$$P_{Ping} = (1, 0, 0, 0)$$

$$P_{SSH} = (0, 1, 0, 0)$$

$$P_{Mysql} = (0, 0, 1, 0)$$

$$P_{HTTP} = (0, 0, 0\ 1)$$

Where P_{Ping} refers to a data point, where the application that was used was Ping. It can be seen that for this application the first predictor is set to 1 and the other 3 to zero.

3.1.3 Error Calculation

Assume that we have N predictors and M categories. Assume that we have gathered K labeled data points:

Equation 2

$$P_i = (Y_{1i}, Y_{2i}, \dots Y_{Mi} | X_{1i}, X_{2i}, \dots, X_{Ni})$$

Where:

- P_i: labeled point i
- Y_{ji}: A binary value one or zero.
  - $Y_{ji} = 1 \Rightarrow P_i$ belongs to category j.
  - $Y_{ji} = 0 \Rightarrow P_i$ does not belong to category j.

- $\sum_j Y_{ji} = 1$ in other words, each data point can belong to at most one category

- X_{ji}: The numerical value of the predictor

For each data point we refer to the category it belongs to by Y_i. Where Y_i is a number between 1 and M referring to the category the data point belongs to. It is clear that:

$$Y_{Y_i i} = 1$$

And all other $Y_{ji} = 0$.

For each data point P_i we evaluate the following:

Equation 3

$$Y_j(P_i) = \frac{e^{\beta_{0j}+X_{1i}\beta_{1j}+X_{2i}\beta_{2j}+\cdots+X_{Ni}\beta_{Nj}}}{1 + e^{\beta_{0j}+X_{1i}\beta_{1j}+X_{2i}\beta_{2j}+\cdots+X_{Ni}\beta_{Nj}}}$$

Where:

- $Y_j(P_i)$: The probability the data point P_i belongs to category j.
- β_{1j}, β_{2j}, ...: Coefficient for the different predictors for category j.
- X_{1i}, X_{2i}, ...: Predictors for the different data points.

For each data point the following values will be calculated:

$$Y_1(P_i), Y_2(P_i), \ldots Y_{M-1}(P_i)$$

And

Equation 4

$$Y_M = 1 - \sum_j Y_j(P_i)$$

Using the discussion in the threshold section one of the categories will be assigned to each data point. We will refer to the category assigned to each data point Y_i'. Y_i' will be a number between 1 and M. The measure of fit quality is calculated as follows:

Equation 5

$$MSE = \frac{\sum_{i=1}^{K} Y_i' \neq Y_i}{K}$$

3.1.4 Model Validation

One of the issues that might arise in a machine learning model is overfitting. For example, consider the data points in Figure 1:

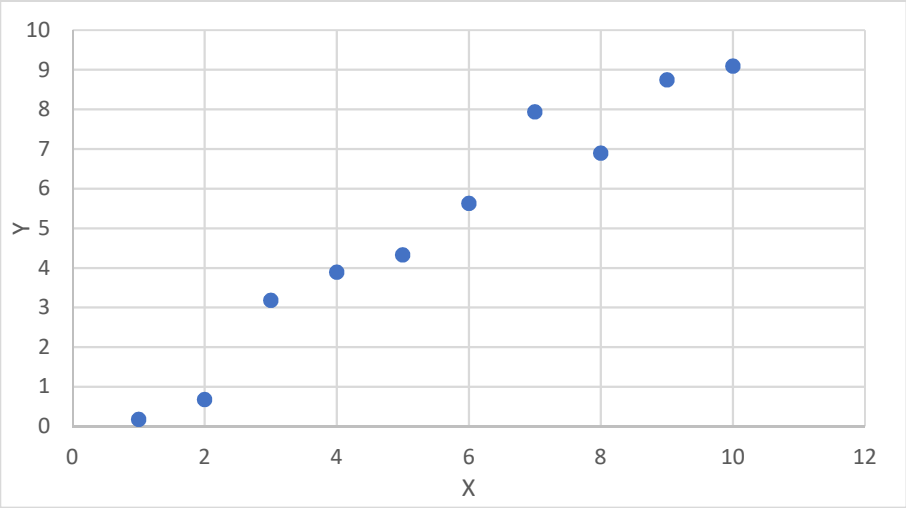

Figure 1, Data point generated from linear model with noise

Fitting a line through the model above we get Figure 2:

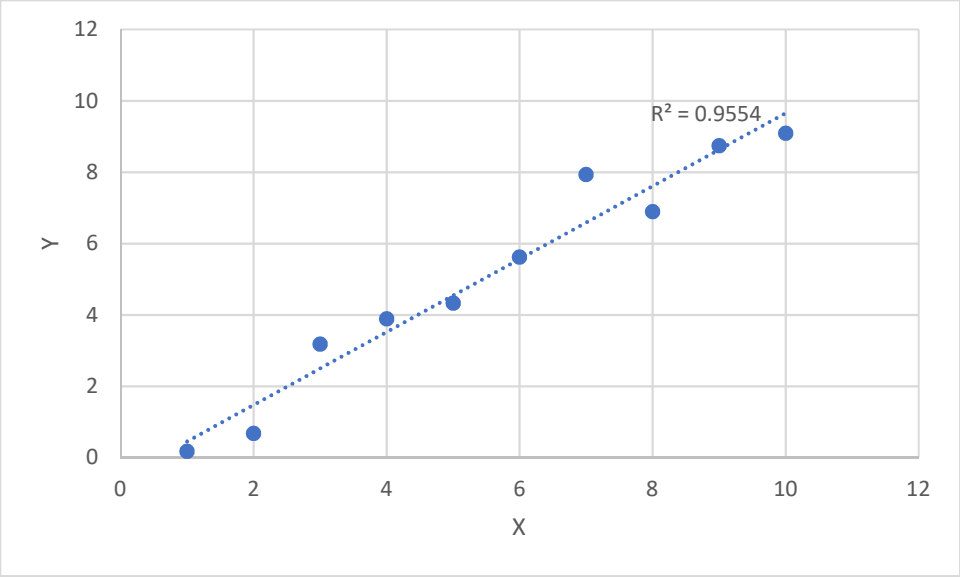

Figure 2 Linear regression

However, we could also fit a higher order polynomial and get a smaller error (Figure 3):

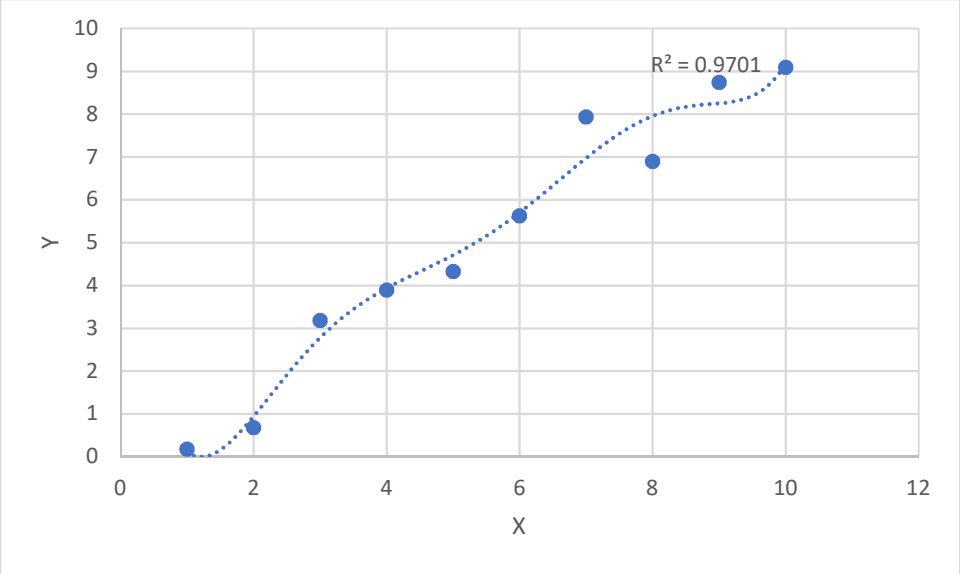

Figure 3 Higher order regression

The consequence of this overfitting is that once another dataset comes along, the performance of the higher order regression will be far worse than the linear model. Therefore, in order to avoid

the overfitting, issue we need to a method for evaluating our model. There are different methods for performing this evaluation.

The Validation Set Approach: In this method the initial data is split in 2 equal sets. The first set is used to train the model. The second set is used for validation. Models are compared based on their performance on the validation set.

Leave One Out Cross Validation: The problem with the validation set approach is that we are putting half our data points aside. With statistical methods, the more data points we have the better our model will be. In cases where we have very limited data points this could greatly reduce the accuracy of our model. In the leave one out cross validation method, one of the data points is put aside and the model is trained using the rest of data. The error of the model is calculated using the single point. This process is repeated for every point. The errors obtained are averaged and the model performance is evaluated based on the average error obtained.

K-Fold Cross Validation: This method is similar to the leave one out cross validation, however instead of keeping one data point out, we put 1/k of the data aside. Normally K is chosen to be a number around 10. There for 10% of the data is put aside. The model is trained using the rest of the 90% data. The error is calculated using the 10% put aside. This is repeated for different 10% subsets and the average error is calculated. The benefits of this method in comparison with the previous method is that only K models need to be trained, while the previous method required on model to be generated for each data point.

3.1.5 Feature Selection

One of the purposes of running machine learning models is to find the relevant predictors. Considering all the available predictors will also result in overfitting. There are different methods for performing feature selection

Forward Selection: In forward selection we start with a single predictor model. We select one of the predictors and fit the data points to the model below:

Equation 6

$$Y_1 = \frac{e^{\beta_0 + X_1\beta_1}}{1 + e^{\beta_0 + X_1\beta_1}}$$

This is repeated for all other predictors:

Equation 7

$$Y_i = \frac{e^{\beta_{0i} + X_i\beta_{1i}}}{1 + e^{\beta_{0i} + X_i\beta_{1i}}}$$

Where

- Y_i: Model fit using predictor i
- β_{1i}: Coefficient for the predictor in model i
- β_{0i}: Constant coefficient for model i.

The error for each model is calculated and the minimum error is found:

Equation 8

$$MSE_{min} = MIN\{MSE_1, MSE_2, \dots, MSE_N\}$$

Where:

- MSE_{min}: The lowest error obtained among the different models
- MSE_i: The error obtained considering predictor i.

The predictor associated with the smallest error is chosen. We will call this $X_{min,1}$.The same process is repeated:

Equation 9

$$Y_i = \frac{e^{\beta_{0i} + X_{min,1}\beta_{1i} + X_i\beta_{2i}}}{1 + e^{\beta_{0i} + X_{min,1}\beta_{1i} + X_i\beta_{2i}}}$$

However, this time we will consider models with 2 predictors. The first predictor is the best predictor chosen in the previous step. The second predictor is chosen from the remaining predictors. And again, the best predictor is chosen by finding the minimum error obtained:

Equation 10

$$MSE_{min} = MIN\{MSE_1, MSE_2, \dots, MSE_{N-1}\}$$

Note that this time there will only be N-1 error values.

After each new predictor is added, the model will generally be evaluated with a validation set. If the error is decreasing the process continues and a new predictor is chosen. If the error has increased, then we are probably overfitting and the process is stopped.

Backward Selection: In this method initially all the predictors are chosen, and a model is fit. The error of this model is calculated:

Equation 11

$$Y_N = \frac{e^{\beta_{01}+X_1\beta_{11}+X_2\beta_{21}+\cdots+X_N\beta_{N1}}}{1+e^{\beta_{01}+X_1\beta_{11}+X_2\beta_{21}+\cdots+X_N\beta_{N1}}}$$

$$MSE_N = error\ of\ model\ using\ validation\ set$$

One at a time each of the predictors are removed and a model is fit with the remaining data points. Similar to forward selection the error of each model is evaluated.

Equation 12

$$Y_{N-1,i} = \frac{e^{\beta_{0i}+X_1\beta_{1i}+X_2\beta_{2i}+\cdots+X_{i-1}\beta_{i-1i}+X_{i+1}\beta_{i+1i}+\cdots+X_{iN}\beta_{Ni}}}{1+e^{\beta_{0i}+X_1\beta_{1i}+X_2\beta_{2i}+\cdots+X_{i-1}\beta_{i-1i}+X_{i+1}\beta_{i+1i}+\cdots+X_{iN}\beta_{Ni}}}$$

Where:

- N: The number of predictors

- $X_1, X_2, \ldots$ The model predictors
- $Y_{N-1,i}$: The model the where the I'th predictor has been removed
- β_{1i}, β_{2i}: The coefficient for predictors in the I'th model.

The error values are calculated:

Equation 13

$$MSE_{min} = MIN\{MSE_1, MSE_2, \ldots, MSE_{i-1}, MSE_{i+1}, \ldots, MSE_N\}$$

The predictor that its removal that results in the minimum error is remove from the model. Similar to forward selection this process is continued. In the next step the predictor selected in the original state along with another predictor will be remove from the model. This process continues until our error increases in the validation set.

Mixed Selection: In this method a combination of forward and backward selection is performed. Initially forward selection is performed. After each new predictor is added, an iteration is preformed over all currently selected predictors to see if removing the predictor will result in improved accuracy.

3.2 Linear Discriminant Analysis (LDA)

The linear discriminant analysis is based off the bays theorem. Basically, it attempts to provide us with an answer of the following question:

$$Assuming\ we\ see\ predictor\ X\ what\ is\ the\ probability\ the\ Category\ is\ Y_i$$

It assumes that the data points have a gaussian distribution and attempts to find the best model that fits the data. The parameters that need to be determined are the covariance matrix and the mean of

the data points. In the multiple output case it assumes that the different classes share the same covariance matrix.

3.3 Quadratic Discriminant Analysis (QDA)

In quadratic discriminant analysis the covariance matrix is not considered to be the same for different classes. Each class is assumed to have its own covariance matrix. Therefore, there will be more predictors to evaluate and therefore makes the model more flexible

3.4 K-Mean Clustering Algorithm

The k-mean clustering algorithm attempts to cluster a set of data points in a manner that results in the least amount of inter-cluster variance. In other words, each data point is assigned to the cluster where the Euclidean distance between the data point and the cluster mean is smallest. The procedure is iterative. For each cluster, the mean is calculated. For each data point the distance to all the cluster centroids is calculated. If the distance between a point to the centroid of another cluster is less that the distance to the current cluster the data point is assigned to the other cluster. After this reassignment, the cluster centroid for both the initial and the second cluster is recalculated, and the process is repeated until convergence is obtained.

The algorithm has an initialization phase. In this phase, a set of initial clusters are chosen. The initial set of clusters are normally chosen entirely at random. However, [8] proposes a much more efficient method of making this selection. The improved algorithm is called the k-means++. The algorithm proceeds as follows:

1. With equal probability, a data point is chosen from among the provided data points. This will be considered the centroid for the first cluster.

2. A second point is chosen among the remaining points. The probability of each point being chosen as the second points is as bellow. The second points will be considered the centroid for the second cluster:

Equation 14

$$P(X_i) = \frac{d^2(X_i, X_1)}{\sum_j d^2(X_j, X_1)}$$

Where $d^2(X_i, X_1)$ is the Euclidean distance between point X_1 and X_i. It can be seen that points that are farther away from the original point have a higher chance of being selected

3. The rest of the centroids are chosen similar to step 2 however the probability of a point being selected as a centroid is as follows:

Equation 15

$$P(X_i) = \frac{d^2(X_i, C_m)}{\sum_j d^2(X_j, C_m)}$$

Where C_m is the cluster closest to X_i.

The k-means clustering algorithm converges to a local minimum. Therefore, the final clusters depend on the initially chosen clusters. Therefore, the model needs to be run several times with different initial clusters in order to obtain better results.

4 Literature Review

Shiravi et al. explain how the ISCX datasets were generated. The authors present a way to generate synthetic labeled data sets. In preparing the data an actual physical lab was prepared. Two different network traffics were considered, an α set and a β set. The α set are network attacks. They are manually applied to and from the nodes in the lab. The β set are profiles built from regular users. Based on these profiles the computers are setup to mimic real user behavior with some additional randomness. With this setup an entire network is simulated with intervened attacks.

Other researchers have also used the ISCX datasets in their research. Zhao et al. use a decision tree model to classify attack from normal attack. In their model they consider a time interval T. During that time interval they calculate certain features such as average packet payload, length of connection interval, or the average time between the packets in the time interval. They also consider some features not related to the time interval such ip and port numbers. Their model resulted in fairly accurate results.

Yassin et al. perform a combination of k-means clustering and the naive Bayes classifier. The k-means clustering is initially performed to reduce the number of data points. After that the NBC is performed to correctly classify the data points as attack or normal traffic. The results showed an accuracy of over 98% with false alarms around 2%.

D Lin et al. explain how PCAPLib can be used to capture packets in real time and classify and anonymize the data for future research. Although the paper does not go into the details of how the software performs the anomaly detection phase, but it does provide test results with the ISCX dataset showing over 96% accuracy.

4.1 Software Exploitation

In security analysis, we are often encountered with binary data in which we are required to determine which file type this data belongs to. While in a typical scenario we could open the file with a hex editor and read the header to determine the file type, this is not always the case. The binary data may be sampled data from a network packet therefore showing data from the middle of the file. Conti et al propose statistical mapping techniques for classifying binary data to file types.

They considered 14 commonly known files types and found 1000 fragments of each of these file types. A fragment was considered 1024 bytes. On these 1024 bytes they obtained different statistical features such as:

- Shannon Entropy
- Arithmetic mean
- Chi Square
- Hamming Weight

They applied these statistical models to the individual bytes in the fragment. Applying these statistical models for each file type they obtained a mean and variance for each statistical model. By applying the same statistical models to random dataset of binary files they were able to determine how likely it is that the file belongs to the specified category.

Cho et al suggest using a Markovian model for detecting intrusions through software exploits. The Markov chain has 2 states, privileges and unprivileged. The transitions in the model were the system calls. For each user and application, a typical usage model is built. Should the users or applications behavior deviate from the normal model it will be assumed that an attack is happening.

A similar approach has also been proposed by N Ye.. the authors suggest a Markov chain model for intrusion detection. In this model the states are the system calls. The author considered 284 states for the different system calls. In a typical use case it is assumed that system calls for an application follow a certain pattern. If system call A is made then normally either B, C or D is called next. Should system call E be made after A then there is probably an anomaly. The general assumption in the model was that system calls follow specific patterns. In order to catch these effects, the authors suggest considering window sizes of 100 system calls. Having built a model from previously seen data, we know the probability of transition from state A to the next state. Therefore, the probability of making the 100 transitions are known:

Equation 16

$$P = P_1 P_2 \dots P_{100}$$

Such probabilities are calculated for the normal use case. A distribution will be obtained. If an attack happens and the P value obtained is not in the normal range obtained in the previously obtained range, then an attack is detected.

Ourston et al use a similar approach using the Markov mode. However, this time they have a training set for attack data. They train the model using the attack data. When applying the model if the probability follows closely to the trained model then there is probably an attack happening.

4.2 Sampled Packets

One of the goals in NIDS is to detect attacks as they are occurring. In research we are usually working with previously logged data and our only goal is to accurately determine if an attack has happened in the logged data. In practice however only detecting whether an attack has happened or not is not enough and it is required to detect the attack in a timely manner. If the amount of

traffic in our network is large, monitoring every single packet may be infeasible. Mai et al. have studied the effect of sampling on network statistics required for detecting anomalies. The research was mainly focused on finding the effects of sampling on two types of anomalies, volume anomalies and port scans.

Volume anomalies are the type of anomalies that cause a significant change in the volume of network traffic such as DOS attacks. For volume anomalies the statistic that they considered was the rate of flow arrival. In both cases it was shown that flow sampling greatly reduces the accuracy of detection and increases the false alarms.

Duffield et al also explain the impact of sampling on flow statistics. In this paper statistics refers to the mean and variance. The properties they consider are the flow counts per interval time, byte, packet count and flow duration. They measure how sampling effects the statistics of these features.

Brauckhoff et al study the impact of packet sampling on the detection of the blaster worm. They show that although packet sampling greatly effects flow counts, however it does not affect volume metrics such packet and byte count very much. They suggest that while some metrics change a lot by sampling however using entropy methods the blaster worm can still be detected with reasonable accuracy.

Most machine learning research performed on network classification has been done on the whole dataset, while the actual application of the model has been done on sampled data. Nguyen et al suggest performing the machine learning models on sampled datasets. In this research the authors aimed at classifying the type of application generating the flow though machine learning models. The models they chose were the Naïve Bayes and the C4.5 Decision tree model. In their models

they only consider the latest N packets from a flow. The results show an increased performance when the training is performed on sub flows rather than full flows.

4.3 Web Related Classification Methods

While in the field of NIDS we are aiming at classifying network traffic, researchers have also been working on classifying web related patterns. While these research attempts are not related to security, however their ideas may extended to the detection of network intrusions.

One of the areas of research is finding certain users on social media that are trend makers and also spotting users that are good at finding these trend makers. The benefits of being able to detect these users is twofold:

1. It enables us to better detect trends on the internet
2. It improves the recommender system on such systems

Sha et al suggest using support vector machines in classifying users as trend makers or trend spotters. Their results show reasonable accuracy.

Another field of research that could potentially be related to NIDS is predicting the number of users during different hours and days of the week. Having such models could potentially be used to detect anomalies. Amico et al study just that. In their research they consider 3 different datasets. IM, GW and KAD. IM is a dataset extracted from an instant messaging server in Italy. GW is from a dataset extracted from an ISP In France. KAD is a dataset extracted mainly from users of the eDonkey2000 client. Their model uses a combination of the logistic regression, Bayesian inference

and the LA approximation to implement their model. Using this model obtain relatively accurate results.

Hsieh et al provide hidden Markov solution to classify applications through their network traffic. In their model they consider the handshake phase of the applications. They suggest that since different applications have a different initial handshake the first few packets sent between client and server can be used to detect the type of application. In their research they generate a hidden Markov model for each application. When a new connection is established the connection, pattern is compared with the different Markov models. The connection is then associated with the application that results in the highest probability. Knowing the application could greatly help in NIDS research as different application may pose different threats. There is also room for future research to investigate the potential of finding different handshakes for attacks compared to regular traffic.

4.4 NIDS Using Machine Learning

Other researchers have previous worked on the field of NIDS using machine learning. [9] discusses some of the recent researches performed on the field. The papers also lists some of latest NIDS application implemented in the industry along with the companies that have implemented them. Tsai et al review some of the recent papers published in the field on NIDS using machine learning. They also show the trend as to which machine learning models recent research is going towards. Liao et al provide a very comprehensive review of different intrusion detection techniques. It breaks down the models into different categories:

- Statistical, Pattern Based, Rule Based, State Based, Heuristic
- Anomaly-based, signature based, stateful protocol analysis

And also by how well the methods perform. Other papers which have reviewed NIDS systems are [10]

One of the issues of implementing machine learning techniques in network intrusion detection is the computational cost of the models. Models such as KVM have proven to be very effective, however applying such models to very large datasets may not be computationally feasible. Horng et al suggest a combination of a supervised and unsupervised method for detecting network intrusions. The authors suggest initially applying a clustering algorithm to the dataset. The clustering algorithm will greatly reduce the number of data points required to consider. After the clustering phase an SVM [11] is applied to the clusters.

The datasets used in this research was the KDD Cup (1999). The dataset consisted of 41 different network features. A backward feature selection algorithm was performed on the features to select only the relevant features. The clustering algorithm performed was the Birch Hierarchal Clustering Algorithm [12]. The algorithm provides a method of clustering that does not required all the data to be present at once. The clusters are built incrementally as more data is presented. This is very important when working with very large datasets. As new data points are introduced they are replaced by clusters. If the variance in a cluster gets to big the algorithm splits the cluster in two. For each cluster 3 parameters are stored. Using those three parameters the mean of the cluster can be obtained.

In [13] two different machine learning techniques are compared. The Cascading K-means Clustering and C4.5 Decision Tree Algorithm [14]. The authors perform the machine learning models on the KDD 99 Cup dataset. The research showed generally good results.

Bouzdia et al propose a machine learning model on the KDD 99 CUP. In the research they use one unsupervised learning method and two supervised learning methods. The unsupervised method was the principal component method. The supervised methods were the nearest neighbor and the decision tree models. They consider 4 different models (Table 1):

Table 1 The different models considered in the research

Model	Apply unsupervised method	Supervised method
1	Yes	PCA
2	Yes	Decision Tree
3	No	PCA
4	No	Decision Tree

In 2 of the models they initially apply the unsupervised learning method. In the other 2 they went directly to the supervised methods. The results show that the results were pretty close for the cases where the unsupervised learning method was applied with the cases where the unsupervised learning methods were not applied. Applying the PCA method can greatly reduce the number of predictors, hence making computations much more efficient.

4.5 Dataset Preparation

One of the challenges in NIDS research is preparing the required datasets. Most of the labeled datasets publicly available are out of date. Sangster suggest using war games to prepare the required datasets. Such competitions generally require two teams. Each team will have a set of computers they will need to protect. These computers must be able to provide a minimum required service level at all times. While protecting their computers they will also be required to perform

attacks on their opponent's computers. Other versions of these warfare games include cases where there is a third team that performs attacks, and the other teams are only required to protect their computers.

4.6 Network Attack Through Software Exploitation

While monitoring network traffic is good way to detect network anomalies however it is not enough to detect all attacks. Some attack might have a perfectly normal traffic pattern, but may be exploiting vulnerabilities in software. It is evident that monitoring network traffic alone will not be enough. Chen et al propose applying machine learning techniques to the process system calls pattern in order to detect normal usage from attacks. The dataset they used in this research was from the DARPA 1998 dataset. They considered the frequency of the different system calls made by each process. Therefore, for each process they obtained a vector. Each vector was labeled as attack or normal. The machine learning models they considered were the support vector machines and the artificial neural networks methods. The results showed that the SVM methods performed much better than the ANN.

4.7 Anomaly Detection

Generally applying machine learning techniques to labeled datasets is good for building models to detect known or previously seen attacks. However as new attacks are discovered every day it is also necessary to come up with a model that can detect attacks that have not yet been seen. Such models are generally referred to as anomaly detection models. In [15] three different methods have been proposed for detecting anomalies. The first method is a cluster based method. In this method each feature is placed in an element of a vector in N dimensional space. The number of neighbors surrounding each point in a radius of R are considered. The points that have few neighbors are

considered anomalies. In the second method the sum of the distance to its closest k neighbors is calculated. If this number is large then the point is an anomaly. The third method is an SVM method. In this model it is assumed that the normal points are close to the origin and the anomalies are far away from the origin. A hyper plane if found that will best separate the points in the origin from the points far out. The best hyper plane is the one that creates the largest margin.

Leung et al propose using a clustering algorithm for detecting network intrusions. They use the MAFIA clustering method [16]. In this clustering method the space is split into cells, initially of equal size. Each data point is assigned to once cell. Cells are considered to be adjacent if they have at least one common side. Clusters are groups of adjacent cells. Not all the cells in the cluster need to be adjacent but there must be some path of adjacent cells between any two cells in the cluster. In regions where there are more data points the cell size is adaptively reduced. Leung et al use the 1999 KDD Cup Data set. They used the training set for training the model and the test set for validation. Data points are considered anomalies if they are not part of the clusters obtained in the training set.

While good accuracy was obtained however the model should have been tested on a second dataset to confirm its effectiveness. Also, the 1999 KDD Data set is a very old and outdated dataset. A lot of the different network usage seen today (such as video streaming, …) were not in use at that time.

Pransta et al suggests using a clustering algorithm for anomaly detection. In this paper the author assumes that each predictor can only take on a finite set of values. Data points are added to clusters based on the number of similar predictors. The more predictors that share the same value the similar the data points are. If a matching cluster is not found a new cluster is only created if the data point has a certain amount of similarity with the rest of the data points. While the approach

was able to obtain high accuracy, however the same issues as the previous paper exist. While the authors claim this is unsupervised it is really a supervised algorithm as tuning parameters need to be set. Also in the case that the predictors are continuous parameters, the continuous parameters would need to be converted to discrete parameters. This granularity for which this is done will have an impact on the clusters, therefore this is also another tuning parameter that needs to be set.

Monowar et al use a tree method in order to perform clustering. Similar data points are clustered together by assigning them the same parent. There are two tuning parameters for building the tree, α and ϵ. Where α determines if nodes are similar enough to fall in the same cluster and ϵ controls how the height of the tree is increased.

Casas combine two different methods for detecting anomalies. In the first step they generate flows based off the arriving packets. They then split the flows into time intervals. They consider different properties of the flows in the time interval, such as bytes transferred, packets transferred and other similar properties. If a change is noticed during a certain time slot then that time slot is taken to the next step. In this step a clustering algorithm is performed on the flows in that time slot. Outliers will be considered as anomalies.

Portnoy et al perform a clustering algorithm on the KDD 1999 dataset. They initially normalize the data to standard gaussian distribution. They then apply a clustering algorithm to the data. They take the clusters with the most amount of data points as clean traffic and the rest as anomalies. The issue with taking such an approach is that you are assuming that only certain points of space contain anomalies. This contradicts the idea of assuming all unknowns are anomalies.

5 Statistical Model of Network Under Attack (Part 1)

Today whenever a complicated problem involving a large amount of data is encountered the first thing that comes to mind is to use machine learning. However, we decided to start out with a much simpler approach and see the draw backs before attempting a more complicated machine learning approach. The purpose of this chapter is to do just that.

The statistical model used in this chapter is the markov chain. In this chapter we demonstrate a simple model in detecting network attacks by looking only at the flow-in and flow-out patterns. The dataset used was the Darpa intrusion detection dataset of the year 1998.

5.1 Dataset

The dataset used in this research was the Darpa Intrusion detection dataset of the year 1998. The dataset consists of 7 weeks of captured network traffic. During the 7-week period multiple controlled attacks were performed on the network. The network attacks that were present in the dataset were DOS Attacks (Back, land, Neptune, pod, smurf, teardrop, syslog), Dictionary attacks, FTP Attacks, port sweep (isweep, portsweep, spy) and warez (warez, warezclient, warezmaster)

5.2 Theory

In this model we attempt at predicting network attacks by building a markov chain model based on the connectivity to a particular node. The model detects when an attack is happening targeted at a particular node.

By obtaining the probability of transition from one state to another we can find abnormal behavior and flag them as attacks. States in the model are defined based on two parameters.

Connection Count: The number of connections that were initiated with the server during that one-minute period.

Repeat: The number of previous intervals that the connection count property remained the same.

The model is built in three stages

5.2.1 Stage 1, Generating Clean Traffic Model

In the first stage the clean traffic behavior was modeled using the markov chain model. The traffic was split into one-minute periods.

Each state has a series of transitions. The transitions show the probability that the state went from its current state to the next state. Therefore, the markov chain model would look something like Figure 4. To avoid clutter in the figure below only the transitions propagating from the states in the middle row have been drawn:

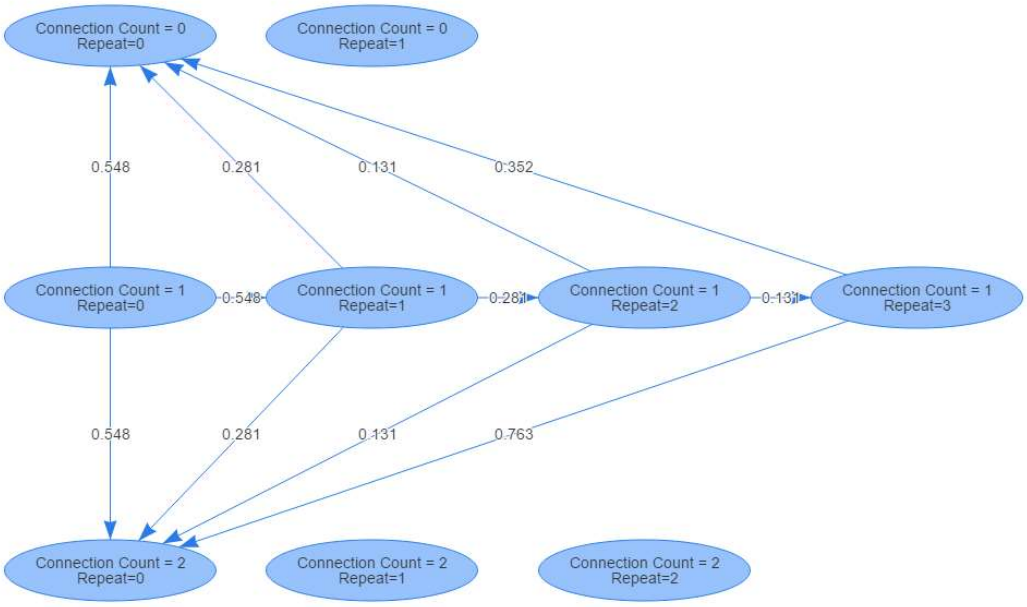

Figure 4: Markov chain model used in analysis. Some actions have been omitted for clarity

The algorithm for determining the next state is as follows:

Step 1: S = S(0, 0) and $T_i = T_0$

Step 2: $T_i = T_{i+1}$

Step 3: Find the number of connections initiated during T_i.

If $C_i = C_{i-1}$ go to step 4.

If $C_i \neq C_{i-1}$ go to step 5.

Step 4: $S_i = S(C_i, 0)$. Go to step 2.

Step 5: $R_i = R_{i-1} + 1. Go\ to\ step\ 6.$

Step 6: $S_i = S(C_i, R_i). Go\ to\ step\ 2$

Where

$\boldsymbol{S(C, R)}$: The state with connection count C and repeat R

$\boldsymbol{T_i}$: The I'th one-minute time interval.

$\boldsymbol{C_i}$: The number of connections initiated with server during time interval i.

$\boldsymbol{R_i}$: The number of consecutive times this connection count has been repeated up to interval i.

5.2.2 Stage 2, Generating The 20 Minute Probability Distribution

Once the markov chain model for the clean traffic has been generated, the following distribution probability is calculated:

Equation 17

$$P_i = \prod_{j=1}^{20} A(S_j, S_{j+1})$$

Where:

P_i: $Probability\ obtained\ by\ passing\ clean\ traffic\ through\ the\ markov\ model\ during\ a\ 20\ minute\ period\ starting\ at\ minute\ i$

S_j: $The\ server\ state\ during\ time\ interval\ (j+i)$

S_{j+1}: $The\ server\ state\ during\ time\ interval\ (j+1+i)$

$A(S_j, S_{j+1})$: $The\ transition\ probablity\ for\ clean\ traffic\ to\ traverse\ from\ state\ j\ to\ j+1$

The result of generating the probability distribution for the clean traffic are shown in Figure 5 and Figure 6. The 95% cut-off point can be obtained as follows:

$$Prob(Log(P_i) > -9.2) = 0.95 \Rightarrow Log(P_i) > -9.2 \Rightarrow P_i > 0.000101$$

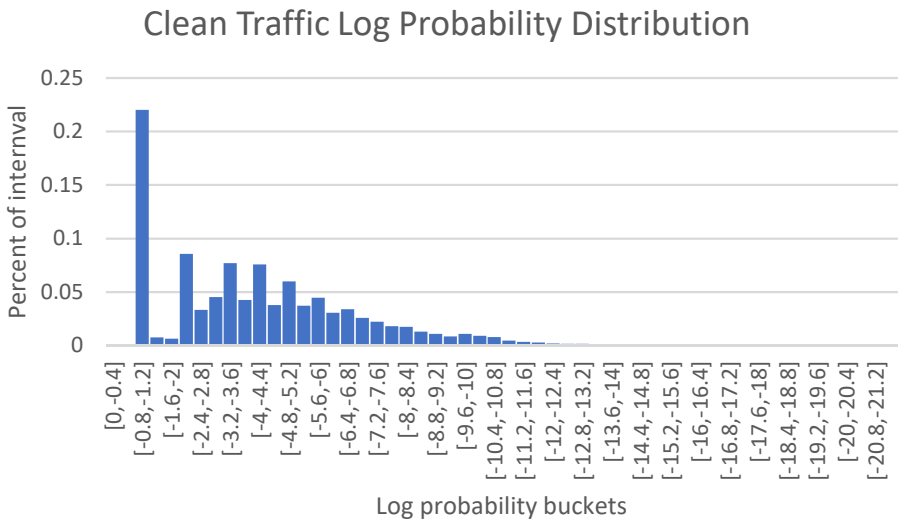

Figure 5 Clean traffic log probability distribution

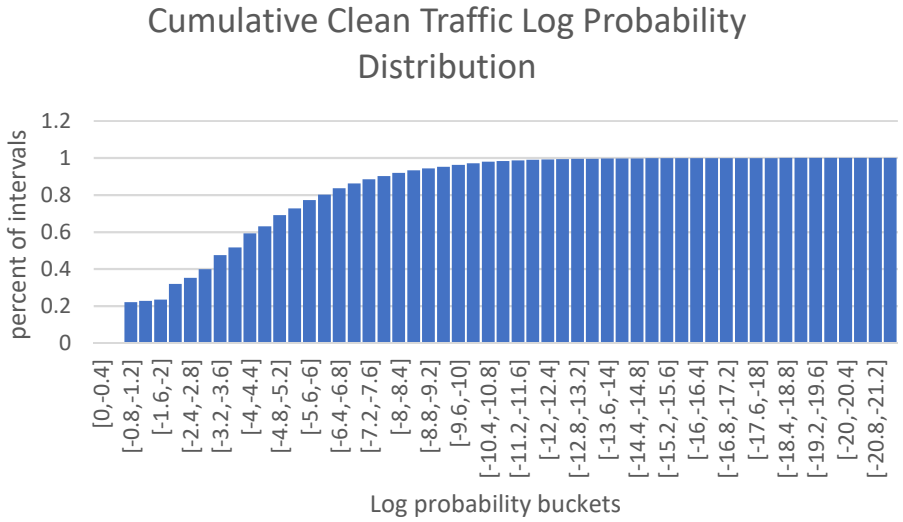

Figure 6 cumulative clean traffic log probability

5.2.3 Stage 3, Unfiltered Traffic Distribution

Using the markov model for clean traffic, the unfiltered traffic was passed through the model and the 20-min probability distribution was calculated. The results have been plotted in Figure 7. The

large number of observations in the right most bucket was due to observations with zero probability.

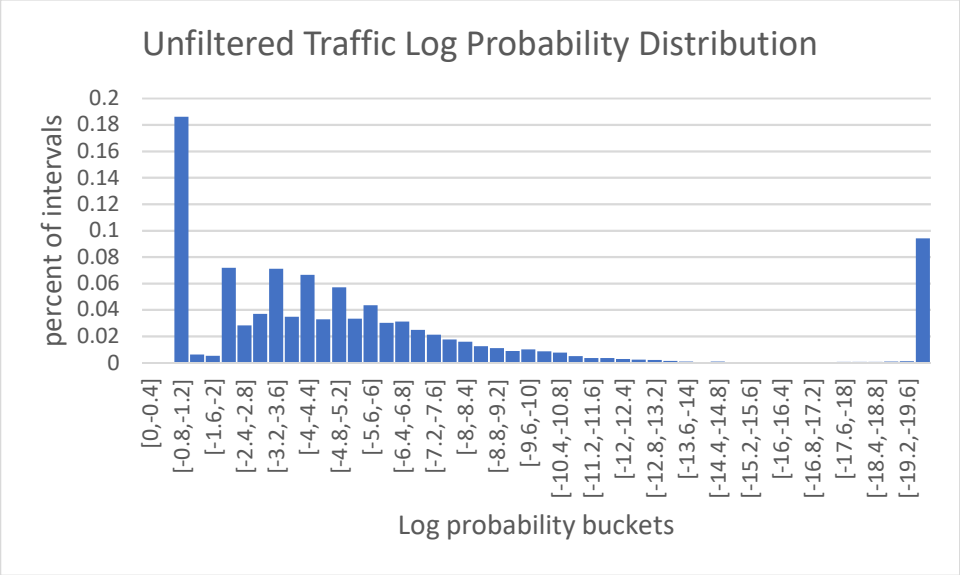

Figure 7 Unfiltered traffic log probability distribution

In the Figure 8 the difference in log probability distribution between the two traffics is plotted. The really low and high-end buckets have been removed for better visualization. It can be seen that the left-hand buckets are positive which shows that clean traffic has a better correlation with the markov model. As we move to the right the values become negative indicating that the uncorrelation is more in the unfiltered traffic.

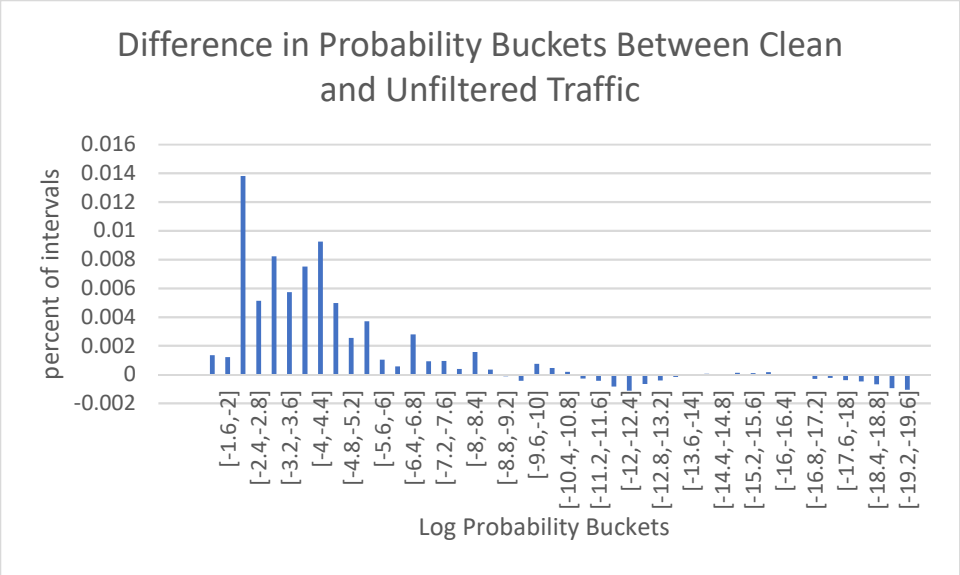

Figure 8 difference in probability buckets between clean and unfiltered traffic

5.3 Attack Detection

In this paper our main goal was to find attacks where there is a significant change in incoming connections. Therefore, our main targets where the following type of attacks:

- Network mapping
- Illegal upload of copyright content using Warez
- Illegal download of copyright content using Warez
- Syn flood denial of service
- Port sweep
- Network probing tools
- DOS attack using misfragmented UDP packets.
- DOS using ping of death

We considered 24 different attacks from the dataset.

5.4 Results

We considered Log(P)<-18 as our cut-off point for detecting attacks. The results are based on a 7-week period of monitoring (Table 2 and Table 3):

Table 2 Results

	True Detections	Missed Attacks
Count	20	4
Accuracy	83%	16%

Table 3 Results

	False Alarms	Total Connections
Count	7	25181
Accuracy	0.027%	

5.5 Issues

There were a number of issues with this model that motivated the research towards machine learning models.

- In the model we are specifically looking at connection counts. The issue with such an approach is that we are not taking any of the other predictors into account. There might be other features that could also help in detecting attacks such as the number of bytes or packets being transferred or the type of application responsible for generating the flow.
- The model was evaluated on a relatively old dataset. The network traffic was much less complex than today's traffic. Due to the simplicity in the network traffic there were very few states which made the model work well. Had we used a dataset with more recent network traffic there would have been a much larger variety in states. Such a variety in

states would results in a large number of transitions not being observed in the initial training phase and hence would results in false alarms.

- The model only takes into account a single node. Some attacks are not visible by just observing a single attack and would require looking at the entire network.

6 Supervised Model of Network Under Attack (Part 2)

One of the issues with most of the research done in the field of network intrusion detection using machine learning techniques is that they all use a very old and outdated dataset (KDDCup99). In this section we propose a supervised machine learning technique that performs well both on the old dataset and on a newer dataset (ISCX2012). The ISCX dataset was provided by the New Brunswick Institute of Security [17]. In [18] the authors explain how the ISCX datasets were generated

Also, one of the issues with most research in this field is that they limit themselves to the limited handcrafted features provided by KDDCup99 dataset. We demonstrate a method for extracting features from network logs. Using this method researchers will no longer be limited to the features provided to them.

Another issue we find with research in this field is they do not take variations of network traffic into account. In other words, they don't take into account how well the model will perform if there are sudden increases in network traffic due to unpredictable events. As an example, consider a university campus network where all grades are released on the same day. On that day there will probably be a surge in network traffic. The method we proposed worked well both under regular traffic an increased traffic.

6.1 Dataset

Two different datasets were used in this part of the research:

- The ISCX2012 dataset
- The Darpa 1998 Dataset

6.1.1 ISCX2012 Dataset

This dataset was prepared by the University of New Brunswick. The dataset consists of 7 days of network traffic with controlled attacks.

Day 1: Normal Activity

Day 2: Normal Activity + small amount of brute force attacks

Day 3: Normal Activity + Infiltrating the network from inside

Day 4: normal activity + HTTPS denial of service

Day 5: normal Activity + DDOS using IRC Botnets

Day 6: normal Activity + small amount of brute force attacks

Day 7: Normal Activity + Brute force ssh

The dataset consists of labeled flows. The flows consist of the following data:

- **AppName**: The name of the application used in the connection. I am assuming this is based on port number. Not all flows have AppName. For unknown applications this field is set to “Unknown TCP” or “Unknown UDP”
- **Total Source Bytes:** The total number of bytes transferred from the source node to the destination node during the duration of the flow.
- **Total Dest Bytes:** The total number of bytes transferred from the destination node to the source node during the duration of the flow.
- **Total Source Packets:** The total number of packets transferred from the source node to the destination node during the duration of the flow.

- **Total Destination packets:** The total number of packets transferred from the destination node to the source node during the duration of the flow.
- **sourcePayloadAsBase64:** The source payload in base 64 format.
- **sourcePayloadAsUTF:** The source payload in UTF format
- **destinationPayloadAsBase64:** The destination payload in base 64 format.
- **destinationPayloadAsUTF:** The destination payload in UTF format.
- **direction:** This value is set to one of the following {L2L, R2L, L2R, R2R} which determines the direction of the flow.
- **sourceTCPFlagsDescription:** TCP flags that were set by the source node.
- **destinationTCPFlagsDescription:** TCP flags that were set by the destination node.
- **source IP:** IP address of the source node
- **destination IP:** IP address of the destination node.
- **protocolName:** The name of the protocol used in the flow, tcp, udp, icmp, ..
- **source Port:** The port number used by the source node.
- **destination Port:** The port number used by the destination node.
- **startDateTime:** The start time of the flow.
- **stopDateTime:** The end time of the flow
- **Tag:** Normal or Attack

6.1.2 Darpa 1998 Dataset

The second dataset that was used in validating the model was the Darpa 1998 dataset. This dataset was prepared by the MIT Lincoln laboratory. Unlike the ISCX2012 dataset where all the required features were extracted, the Darpa dataset was missing some of the required features. Therefore, we had to extract all the features ourselves.

The dataset consists of 7 weeks of data. Each week consisted of 7 days of data. For each data two files were provided:

- A libpcap file
- A csv file with labels

The libpcap file is a binary file with network data. The csv consisted of flow data with labels. Due to the fact that csv file did not contain all the features required by our model, the required features had to be extracted from the libpcap file and matched with the labels in the csv file.

The csv file consisted of the following data:

- **Date:** The date when the flow was captured
- **Start Time:** The time when the flow started
- **End Time:** The time when the flow ended
- **Application:** The application that generated the flow. This was most probably guessed based off the port number as only the well-known ports had this field set. For the not well-known ports a port number was used in this column
- **Source Port:** The port number the source node used for communication (if applicable).
- **Dest Port:** The port number used by the destination node for communication (if applicable)
- **Source Node:** The source node in the flow
- **Destination Node:** The destination node in the flow
- **Tag:** 1 determines an attack and 0 determines a normal flow
- **Attack Type:** The name of the attack if any

6.2 Model

Our goal is to come up with a model that satisfies the following:

1- Is generalized: It can be applied to multiple datasets and still provide satisfactory results

2- Perform well under surge conditions

In coming up with such a model a lot of questions arise:

- What type of machine learning model do we choose? Linear Regression? Linear Discriminant Analysis? Support Vector Machines?
- How do we take temporal data into account?
- What predictors do we consider in our model and how do we choose the relevant predictors?
- How do prevent the model fitting to closely to normal data points since there may be many more normal data points then attack points.

Rather than taking a guess at the questions above, we performed 120 different machine learning models with varying assumptions to find the models that perform best on the datasets. In the sections below, we describe in further detail how the models where generated and how they were varied.

6.2.1 Predictors

While flow data provides useful information about a single flow however it lacks the required information to capture temporal data. In order to better capture temporal data, we generate a new set of predictors from the flow information.

In order to generate these new predictors, we consider intervals of T seconds. For each interval of time we end up with one vector of predictors. The intervals are overlapping considering a granularity of 1 second. For example, considering 10 second intervals, the first vector of predictors will be for the time interval 0~10, the second vector of predictors will be for the time interval 1~11 and so on.

In our model we consider three different intervals:

- 1 second intervals
- 5 second intervals
- 10 second intervals

6.2.1.1 Application Type

Each flow is generated by a certain application. As an example, the application could be an http or ssh server. The application could also be a network layer protocol, for example ICMP. For the ISCX 2012 dataset the environment was controlled and the applications for most the flows where known. Table 4 displays the applications used in the dataset:

Table 4 Different applications used in the ISCX2012 dataset

1	NA	18	Gnutella	35	IRC
2	Anet	19	Google	36	Kazaa
3	AOL-ICQ	20	Groove	37	LDAP
4	Authentication	21	GuptaSQLBase	38	ManagementServices
5	BGP	22	H.323	39	MDQS
6	BitTorrent	23	Hosts2-Ns	40	MGCP
7	Blubster	24	Hotline	41	MicrosoftMediaServer
8	Citrix	25	HTTPImageTransfer	42	Misc-DB
9	Common-P2P-Port	26	HTTPWeb	43	Misc-Mail-Port
10	Common-Ports	27	iChat	44	Misc-Ports
11	DNS	28	ICMP	45	MiscApp
12	DNS-Port	29	IGMP	46	MiscApplication
13	dsp3270	30	IMAP	47	MS-SQL
14	Filenet	31	Ingres	48	MSMQ
15	Flowgen	32	Intellex	49	MSN
16	FTP	33	IPSec	50	MSN-Zone
17	giop-ssl	34	IPX	51	MSTerminalServices
52	Nessus	71	Real	90	Tacacs
53	NETBEUI	72	rexec	91	Telnet
54	NetBIOS-IP	73	rlogin	92	TFTP
55	Network-Config-Ports	74	RPC	93	Timbuktu
56	NFS	75	rsh	94	TimeServer
57	NNTPNews	76	RTSP	95	Unknown_TCP
58	NortonAntiVirus	77	SAP	96	Unknown_UDP
59	NortonGhost	78	SecureWeb	97	UpdateDaemon
60	NTP	79	SIP	98	VNC
61	OpenNap	80	SMS	99	Web-Port
62	OpenWindows	81	SMTP	100	WebFileTransfer
63	Oracle	82	SNA	101	WebMediaAudio
64	PCAnywhere	83	SNMP-Ports	102	WebMediaDocuments
65	PeerEnabler	84	Squid	103	WebMediaVideo
66	POP	85	SSDP	104	Webmin
67	POP-port	86	SSH	105	WindowsFileSharing
68	PostgreSQL	87	SSL-Shell	106	XFER
69	PPTP	88	StreamingAudio	107	XWindows
70	Printer	89	SunRPC	108	Yahoo

In a real situation we will most likely not have that information available. One way to guess the application used to generate the flow would be through port numbers. Further details about how this has been provided in the implementation section.

All the applications that have been presented in the dataset are discovered and a one hot vector is generated for each flow. The one will represent the application that was used to generate the flow. For the duration of the interval all of these one hot vectors are summed and divided by the number of flows in that interval. Therefore, we end up with a vector of ratios between [0, 1]. Each column represents the ratio of flows during that interval which were using that application.

6.2.1.2 Protocols

Similar to the case with application type we generate a vector of ratio for each interval. Each column is a value between [0, 1]. They show the ratio of flows during the interval which used a particular protocol. The protocols could be of different layers, for example the following protocols could be considered for a dataset (TCP, UDP, ICMP).

The following were the protocols found in the ISCX2012 dataset (Table 5):

Table 5 Different protocols used in the ISCX2012 dataset

1	NA
2	icmp_ip
3	igmp
4	ip
5	ipv6icmp
6	tcp_ip
7	udp_ip

6.2.1.3 Unique Local and Remote IPs

These are two predictors each indicating the total ratio of unique local and remote IPs to the overall unique IPs used during that interval. For example, consider the following flows during a time interval (Table 6):

Table 6 Sample flows

Flow number	Source	Destination
1	192.168.1.1	141.48.75.41
2	192.168.1.1	141.48.75.41
3	171.465.485.45	184.48.45.14
4	192.168.1.2	171.465.485.45
5	184.48.45.14	192.168.1.2
6	192.168.1.2	192.168.1.1
7	192.168.1.2	184.48.45.14

The unique local IPS are:

- 192.168.1.2
- 192.168.1.1

The unique remote IPS are:

- 141.48.75.41
- 171.465.485.45
- 184.48.45.14

Therefore, the ratio of unique local IPs will be: $^2/_5 = 0.4$

And the ratio of unique remote IPs will be: $^3/_5 = 0.6$

6.2.1.4 Direction

These are a set of four predictors determining the direction of the flows:

- R2L are connections initiated remotely and contacting local nodes.
- L2R are connections initiated locally and contacting remote node.
- R2R are connections initiated remotely and contacting remote nodes
- L2L are connections initiated locally and contacting local nodes.

6.2.1.5 Packet Count

This is the average packet count per flow during the interval.

6.2.1.6 Bytes Transferred

This is the average bytes transferred per flow during the interval

6.2.2 Machine Learning Models

One of the questions that needs to be answered is that which type of machine learning model will perform best for this type of problem. In this research 3 different machine learning models where used:

- Logistic regression
- Linear Discriminant Analysis
- Quadratic Discriminant Analysis

6.2.3 Intervals

It was mentioned that the data points where grouped in intervals and a new set of features were generated for each interval. The larger the interval the more temporal data we will be capturing. On the other hand, smaller intervals will capture finer details about the flow.

Three different intervals were considered

- 1 second intervals
- 5 second intervals
- 10 second intervals

6.2.3.1 Feature Selection

In each model forward selection was performed for predictor selection. Generally, in forward selection the predictors are chosen based on the best accuracy obtained from the training data. However, in the initial stages of running the models it was seen that the predictors fit too closely to the training data using this method. Therefore 2 different methods of feature selection where considered:

1- Accuracy of the model is selected based on the training set.

2- Accuracy of the model is selected based on the validation set.

However, the second method must be used with caution as it might also cause overfitting. Therefore, we also put aside a third dataset as a test set.

6.2.3.2 Low Count of Attack Records

While the number of attack flows was comparable with normal traffic, however since some of the attack flows occurred in bursts, the number of records containing attack flows ended up being very low. To overcome this one solution was to copy attack records based on the number attack flows it is representing. Therefore 2 different models where considered:

1- Attack records are not repeated

2- Attack records are repeated by the amount of attack flows it represents

6.2.3.3 Sensitivity and Specificity

As mentioned above after generating data records from the input flows, the number of records containing attacks will be very small relative to the total number of records. In our datasets this was something around 99 normal records for every attack record. Assume a model where it always classifies points as normal traffic. Such a model will have an accuracy of 99%. This is clearly not correct.

While copying attack records as explained above does help, in this research we also consider another method of alleviating this issue. In evaluating the models in forward selection two different methods have been considered:

1- The total accuracy is calculated
2- The attack accuracy is calculated

6.2.3.4 Binary or Multiple Classes

Two different methods for categorizing data points have been considered in this research:

1- Binary: A record either does contain an attack flow (which is assigned to category 1), or does not contain an attack flow (which is assigned category 0)
2- Multi Class: Records that do not contain attack traffic are assigned category 0. However, records that do have attack flows are assigned a label based on the type of attack:
    - Brute Force (Day 2) : 1
    - Infiltrating network from inside: 2
    - HTTPS attack: 3
    - Botnet DDOS: 4
    - Brute Force (Day 6): 5

- Brute Force SSH 6.

6.2.3.5 Naming convention

The naming convention used in this research is as follows (Table 7)

_ _ _ _ I _ E _ S _ C _

Table 7 Naming convention

Model Name	Description
M **********	An M at the start of the model name indicates that the different attacks types had been differentiated in the analysis. A nonexistent M indicates that all attack types had been assigned to the same class.
*LDA********	Linear Discriminant Analysis
*LOG********	Logistic regression
*QDA********	Quadratic Analysis
****I1******	The model considers 1 second intervals
****I5******	The model considers 5 second intervals
****I10******	The model considers 10 second intervals
******E1****	In calculating errors in the forward selection process, the error is calculated only based off the attack records. In other words, the accuracy shows how many attack records were missed.

Model Name	Description
******E0****	In calculating errors in the forward selection process all records are taken into account. Therefore, false alarms and missed attacks will both contribute to the error value
********S1**	In calculating errors in the forward selection process, the validation set is used.
********S0**	In calculating errors in the forward selection process, the training set is used.
**********C1	Due to the fact that attack flows are bursty, the number of attack records will be far less than the number of normal records. In this model each attack record is copied by the amount of attack records it is representing
**********C0	No copying of attack records is performed

A total of 24 different logistic models have been considered (Table 8):

Table 8 The different logistic models tested

Name	Type	Binary Category	Interval	Evaluation based on attack records	Evaluation based on validation set	Copied attack records
LOGI1E0S0C0	Logistic	T	1	F	F	F
LOGI1E0S0C1	Logistic	T	1	F	F	T
LOGI1E0S1C0	Logistic	T	1	F	T	F
LOGI1E0S1C1	Logistic	T	1	F	T	T

Name	Type	Binary Category	Interval	Evaluation based on attack records	Evaluation based on validation set	Copied attack records
LOGI1E1S0C0	Logistic	T	1	T	F	F
LOGI1E1S0C1	Logistic	T	1	T	F	T
LOGI1E1S1C0	Logistic	T	1	T	T	F
LOGI1E1S1C1	Logistic	T	1	T	T	T
LOGI5E0S0C0	Logistic	T	5	F	F	F
LOGI5E0S0C1	Logistic	T	5	F	F	T
LOGI5E0S1C0	Logistic	T	5	F	T	F
LOGI5E0S1C1	Logistic	T	5	F	T	T
LOGI5E1S0C0	Logistic	T	5	T	F	F
LOGI5E1S0C1	Logistic	T	5	T	F	T
LOGI5E1S1C0	Logistic	T	5	T	T	F
LOGI5E1S1C1	Logistic	T	5	T	T	T
LOGI10E0S0C0	Logistic	T	10	F	F	F
LOGI10E0S0C1	Logistic	T	10	F	F	T
LOGI10E0S1C0	Logistic	T	10	F	T	F
LOGI10E0S1C1	Logistic	T	10	F	T	T
LOGI10E1S0C0	Logistic	T	10	T	F	F
LOGI10E1S0C1	Logistic	T	10	T	F	T
LOGI10E1S1C0	Logistic	T	10	T	T	F
LOGI10E1S1C1	Logistic	T	10	T	T	T

Due to instability of the logistic regression method, Multi category was not considered for this method.

A total of 96 different discriminant analysis models have been considered (Table 9):

Table 9 The different linear discriminant and quadratic discriminant analysis models tested

Name	Type	Binary Category	Interval	Evaluation based on attack records	Evaluation based on validation set	Copied attack records
LDAI1E0S0C0	LDA	T	1	F	F	F
LDAI1E0S0C1	LDA	T	1	F	F	T
LDAI1E0S1C0	LDA	T	1	F	T	F
LDAI1E0S1C1	LDA	T	1	F	T	T
LDAI1E1S0C0	LDA	T	1	T	F	F
LDAI1E1S0C1	LDA	T	1	T	F	T
LDAI1E1S1C0	LDA	T	1	T	T	F
LDAI1E1S1C1	LDA	T	1	T	T	T
LDAI5E0S0C0	LDA	T	5	F	F	F
LDAI5E0S0C1	LDA	T	5	F	F	T
LDAI5E0S1C0	LDA	T	5	F	T	F
LDAI5E0S1C1	LDA	T	5	F	T	T
LDAI5E1S0C0	LDA	T	5	T	F	F
LDAI5E1S0C1	LDA	T	5	T	F	T
LDAI5E1S1C0	LDA	T	5	T	T	F
LDAI5E1S1C1	LDA	T	5	T	T	T
LDAI10E0S0C0	LDA	T	10	F	F	F

Name	Type	Binary Category	Interval	Evaluation based on attack records	Evaluation based on validation set	Copied attack records
LDAI10E0S0C1	LDA	T	10	F	F	T
LDAI10E0S1C0	LDA	T	10	F	T	F
LDAI10E0S1C1	LDA	T	10	F	T	T
LDAI10E1S0C0	LDA	T	10	T	F	F
LDAI10E1S0C1	LDA	T	10	T	F	T
LDAI10E1S1C0	LDA	T	10	T	T	F
LDAI10E1S1C1	LDA	T	10	T	T	T
MLDAI1E0S0C0	LDA	F	1	F	F	F
MLDAI1E0S0C1	LDA	F	1	F	F	T
MLDAI1E0S1C0	LDA	F	1	F	T	F
MLDAI1E0S1C1	LDA	F	1	F	T	T
MLDAI1E1S0C0	LDA	F	1	T	F	F
MLDAI1E1S0C1	LDA	F	1	T	F	T
MLDAI1E1S1C0	LDA	F	1	T	T	F
MLDAI1E1S1C1	LDA	F	1	T	T	T
MLDAI5E0S0C0	LDA	F	5	F	F	F
MLDAI5E0S0C1	LDA	F	5	F	F	T
MLDAI5E0S1C0	LDA	F	5	F	T	F
MLDAI5E0S1C1	LDA	F	5	F	T	T
MLDAI5E1S0C0	LDA	F	5	T	F	F
MLDAI5E1S0C1	LDA	F	5	T	F	T

Name	Type	Binary Category	Interval	Evaluation based on attack records	Evaluation based on validation set	Copied attack records
MLDAI5E1S1C0	LDA	F	5	T	T	F
MLDAI5E1S1C1	LDA	F	5	T	T	T
MLDAI10E0S0C0	LDA	F	10	F	F	F
MLDAI10E0S0C1	LDA	F	10	F	F	T
MLDAI10E0S1C0	LDA	F	10	F	T	F
MLDAI10E0S1C1	LDA	F	10	F	T	T
MLDAI10E1S0C0	LDA	F	10	T	F	F
MLDAI10E1S0C1	LDA	F	10	T	F	T
MLDAI10E1S1C0	LDA	F	10	T	T	F
MLDAI10E1S1C1	LDA	F	10	T	T	T
QDAI1E0S0C0	QDA	T	1	F	F	F
QDAI1E0S0C1	QDA	T	1	F	F	T
QDAI1E0S1C0	QDA	T	1	F	T	F
QDAI1E0S1C1	QDA	T	1	F	T	T
QDAI1E1S0C0	QDA	T	1	T	F	F
QDAI1E1S0C1	QDA	T	1	T	F	T
QDAI1E1S1C0	QDA	T	1	T	T	F
QDAI1E1S1C1	QDA	T	1	T	T	T
QDAI5E0S0C0	QDA	T	5	F	F	F
QDAI5E0S0C1	QDA	T	5	F	F	T
QDAI5E0S1C0	QDA	T	5	F	T	F

Name	Type	Binary Category	Interval	Evaluation based on attack records	Evaluation based on validation set	Copied attack records
QDAI5E0S1C1	QDA	T	5	F	T	T
QDAI5E1S0C0	QDA	T	5	T	F	F
QDAI5E1S0C1	QDA	T	5	T	F	T
QDAI5E1S1C0	QDA	T	5	T	T	F
QDAI5E1S1C1	QDA	T	5	T	T	T
QDAI10E0S0C0	QDA	T	10	F	F	F
QDAI10E0S0C1	QDA	T	10	F	F	T
QDAI10E0S1C0	QDA	T	10	F	T	F
QDAI10E0S1C1	QDA	T	10	F	T	T
QDAI10E1S0C0	QDA	T	10	T	F	F
QDAI10E1S0C1	QDA	T	10	T	F	T
QDAI10E1S1C0	QDA	T	10	T	T	F
QDAI10E1S1C1	QDA	T	10	T	T	T
MQDAI1E0S0C0	QDA	F	1	F	F	F
MQDAI1E0S0C1	QDA	F	1	F	F	T
MQDAI1E0S1C0	QDA	F	1	F	T	F
MQDAI1E0S1C1	QDA	F	1	F	T	T
MQDAI1E1S0C0	QDA	F	1	T	F	F
MQDAI1E1S0C1	QDA	F	1	T	F	T
MQDAI1E1S1C0	QDA	F	1	T	T	F
MQDAI1E1S1C1	QDA	F	1	T	T	T

Name	Type	Binary Category	Interval	Evaluation based on attack records	Evaluation based on validation set	Copied attack records
MQDAI5E0S0C0	QDA	F	5	F	F	F
MQDAI5E0S0C1	QDA	F	5	F	F	T
MQDAI5E0S1C0	QDA	F	5	F	T	F
MQDAI5E0S1C1	QDA	F	5	F	T	T
MQDAI5E1S0C0	QDA	F	5	T	F	F
MQDAI5E1S0C1	QDA	F	5	T	F	T
MQDAI5E1S1C0	QDA	F	5	T	T	F
MQDAI5E1S1C1	QDA	F	5	T	T	T
MQDAI10E0S0C0	QDA	F	10	F	F	F
MQDAI10E0S0C1	QDA	F	10	F	F	T
MQDAI10E0S1C0	QDA	F	10	F	T	F
MQDAI10E0S1C1	QDA	F	10	F	T	T
MQDAI10E1S0C0	QDA	F	10	T	F	F
MQDAI10E1S0C1	QDA	F	10	T	F	T
MQDAI10E1S1C0	QDA	F	10	T	T	F
MQDAI10E1S1C1	QDA	F	10	T	T	T

6.2.3.6 Training, Validation and Test Set

The ISCX2012 dataset was split into two parts. Days one to six and day seven. Day seven was used as the test set. Days one to six were used as the training and validation sets. The generated

data points where randomly split into two groups. One of these were used as the training set and the other was used as the validation set.

After obtaining the best model we also tested the model on the Darpa dataset. In order to perform this, test the Darpa dataset was split into two random parts. One was used for training and the other was used for validation.

6.3 Surge Test

One of the goals was to find a model that performs well when there are variations in network traffic. In other words, the model must perform well if there are sudden increases in network traffic due to unpredictable events. As an example, consider a university campus network where all grades are released on the same day. On that day there will probably be a surge in network traffic. However, an NIDS model trained to protect that campus will most likely not have seen data from that day.

In order to test how well the model works, surges were applied to the network traffic and tested with the best model obtained from among the 120 models tested. The following steps were performed to apply surges to the dataset:

- All 7 days of traffic were considered for this test
- The following load ratios were considered: 120%, 140%, 180%, 220%, 260%, 300%, 400%, 500%, 600%, 700%, 800%, 900%, 1000%
- The total number of clean records was obtained.
- The final number of clean records was determined using the equation: $Res = Per \times \frac{cleancount}{100}$
- A total of $Res/1000$ random records were selected.

- The table is sorted based on start time. 1000 records starting at each of the records started at the time of the records selected in step 5 are selected.
- Another random record was selected.
- The records selected in step 6 were copied and shifted starting at the start time of the record selected in step 7.

Using the method specified above we were able to increase the amount of regular traffic while preserving its temporal pattern.

The data points where then generated and split into two random batches for training and validation.

6.4 Implementation

Due to the format of the data the project has two different implementations. One for the ISCX 2012 dataset and another for the Darpa dataset.

6.4.1 ISCX2012 Dataset

6.4.1.1 Phase 1, Obtaining the Dataset

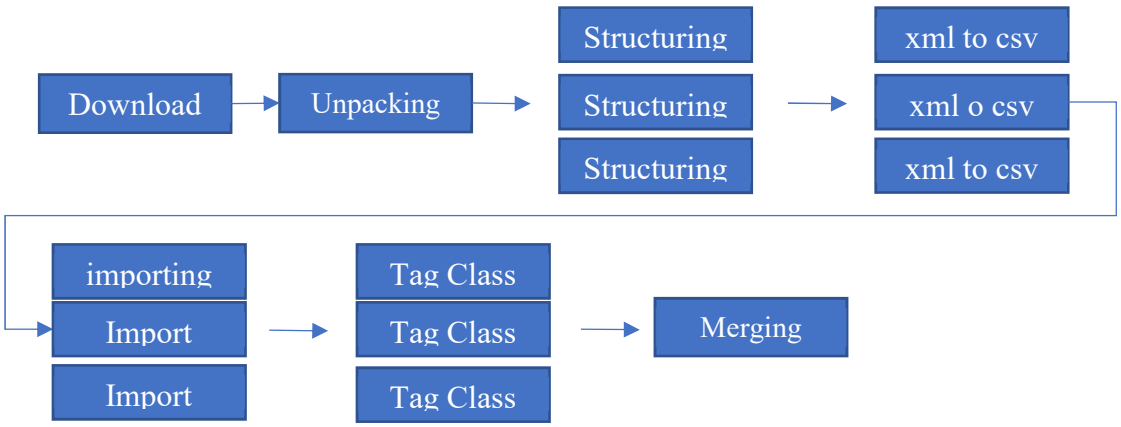

Figure 9 Obtaining the dataset

Download: The dataset was provided by the University of New Brunswick. The content was downloaded from the website.

Unpacking: The dataset contains a zip file with 25 different files. The zip file was unpacked. The files in the zip file included:

- readme.txt
- 12 xml files
- 12 xsd files

The extra files were deleted and only the xml files were kept.

Structuring: The 12 xml files were labeled data for the 7 days of attack. Some days contained several files. Directories were created for each day and the xml files for each day were placed in each folder.

XML to CSV: A custom script for importing csv to MySQL was implemented. In order to be able to reuse this script all xml files were converted to csv files.

6.4.1.2 Phase 2, Setting Up the Database

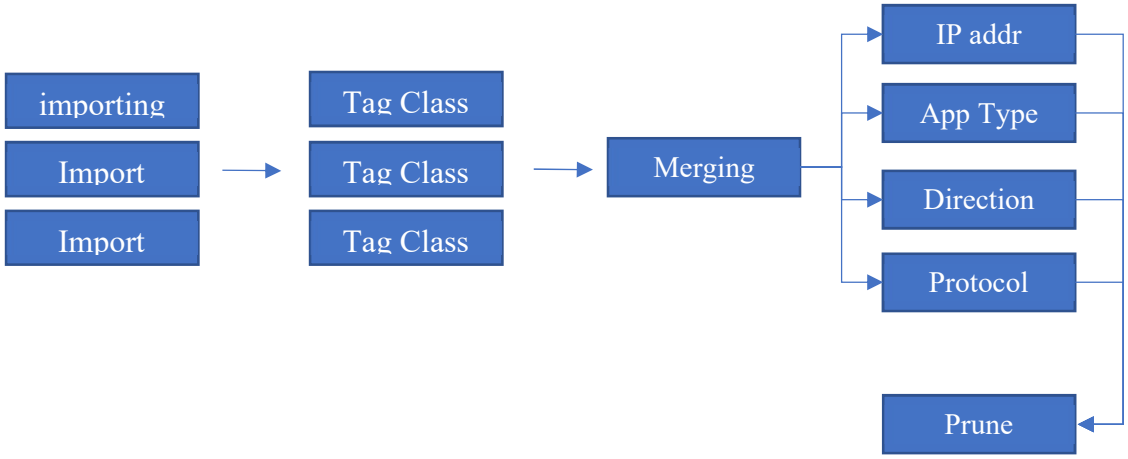

Figure 10 Setting up the database

Import: The csv files in each directory are uploaded to the MySQL database. One table is created for each day. Therefore, for directories with more than one csv file they are all uploaded to the same table.

Tag Class: Each day consisted of a different type of attack. A new column was created in each table and if the row was an attack, the day number is put in that column. This was a way of indicating the type of attack that is occurring on that day.

Merging: After tagging the attacks class the tables were all merged into a single table. This was because during the research a set of modules were written that perform common tasks. In order for these modules to be reusable in the different parts of the research the data has to be in a specific

format for that particular module. The parts explained in the next phase requires all the data to be in a single table.

IP Addr, App type, protocol, direction: At this point these values are all text values. They needed to be converted to numeric values. A new table is created for each them. Each table consists of two columns:

- id
- value

The distinct values for each column are extracted and inserted into the new table. The text values are then replaced by the ids in the associated table.

Prune: In order to keep day 7 data for testing all records belonging to day 7 are removed

6.4.1.3 Phase 3, Data Generation

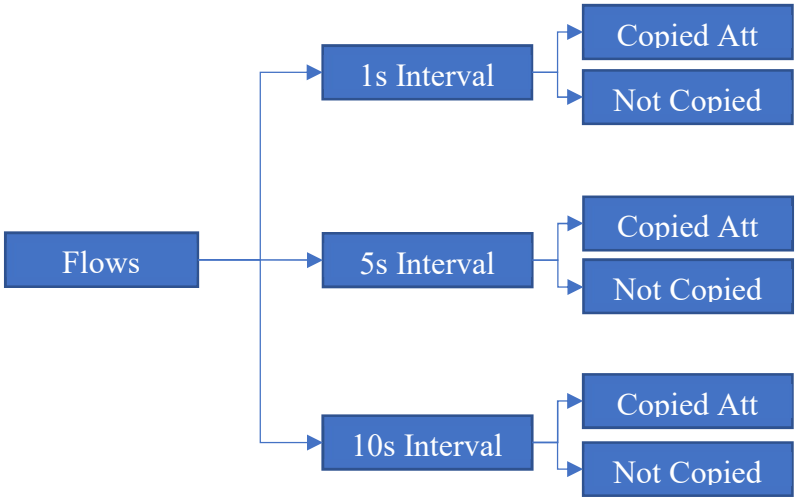

Figure 11 Data generation

3 different intervals were considered in this research:

- 1 second
- 5 seconds
- 10 seconds

For each interval 2 different methods for generating the data points were considered:

- The data points that contain attacks were copied multiple times. Once for every attack record in that interval.
- The data points were not copied

Overall 6 files were generated.

6.4.1.4 Phase 4, Running the Machine Learning Models

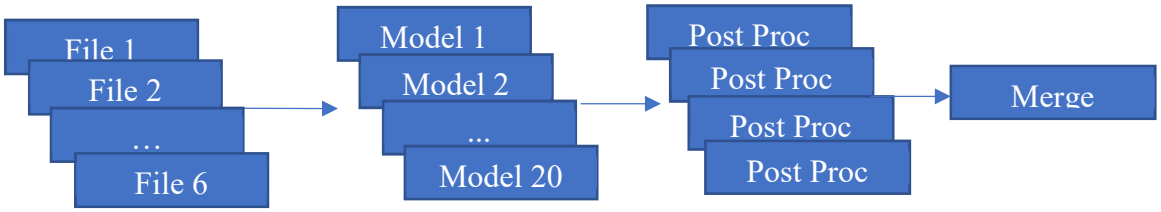

Figure 12 Running the machine learning models

Models: Each of the input files are put through 20 different machine learning models for a total of 120 different models. The program written to perform the machine learning models would output the results on the console.

Post Proc: The outputs of the models were generated in a way that would make debugging easiest. However, they were not in the best format for performing comparisons. A set of VBA scripts were written to post process the output from each of the individual files and output the results in a table that could be used for comparison.

6.4.1.5 Phase 5, Unseen Traffic and Attack Evaluation

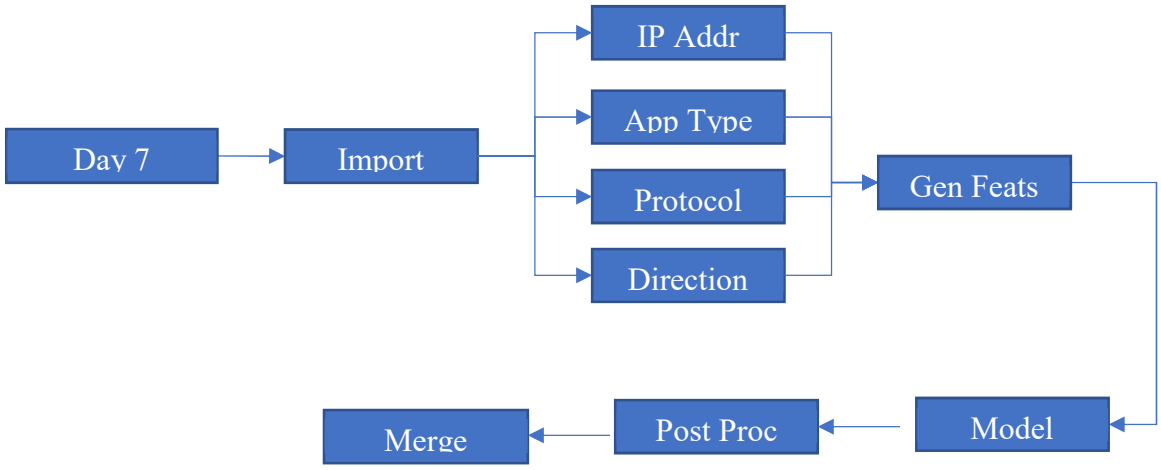

Figure 13 Test set

We also test the model on unseen data and attacks using the day 7 data. Most of the steps are similar to the steps performed before.

Import: The day 7 csv file is imported into a new table in the MySQL database.

IP Addr, App Type, Protocol, Direction: In the previous part a new table was created for each of these four columns and text values for the columns where replaced with numeric values based on their Ids in these tables. In this part we don't generate the tables again but rather use the existing tables to replace the textual values of these four columns with numeric values.

Gen Feats: The required feature files are generated.

Model: The top 20 models are run based on the generated files.

Post Proc: Similar to the previous phase the output data from the machine learning models were post processed by a VBA script

Merge: The results were merged into a table for comparison.

6.4.1.6 Phase 6, Surge Test

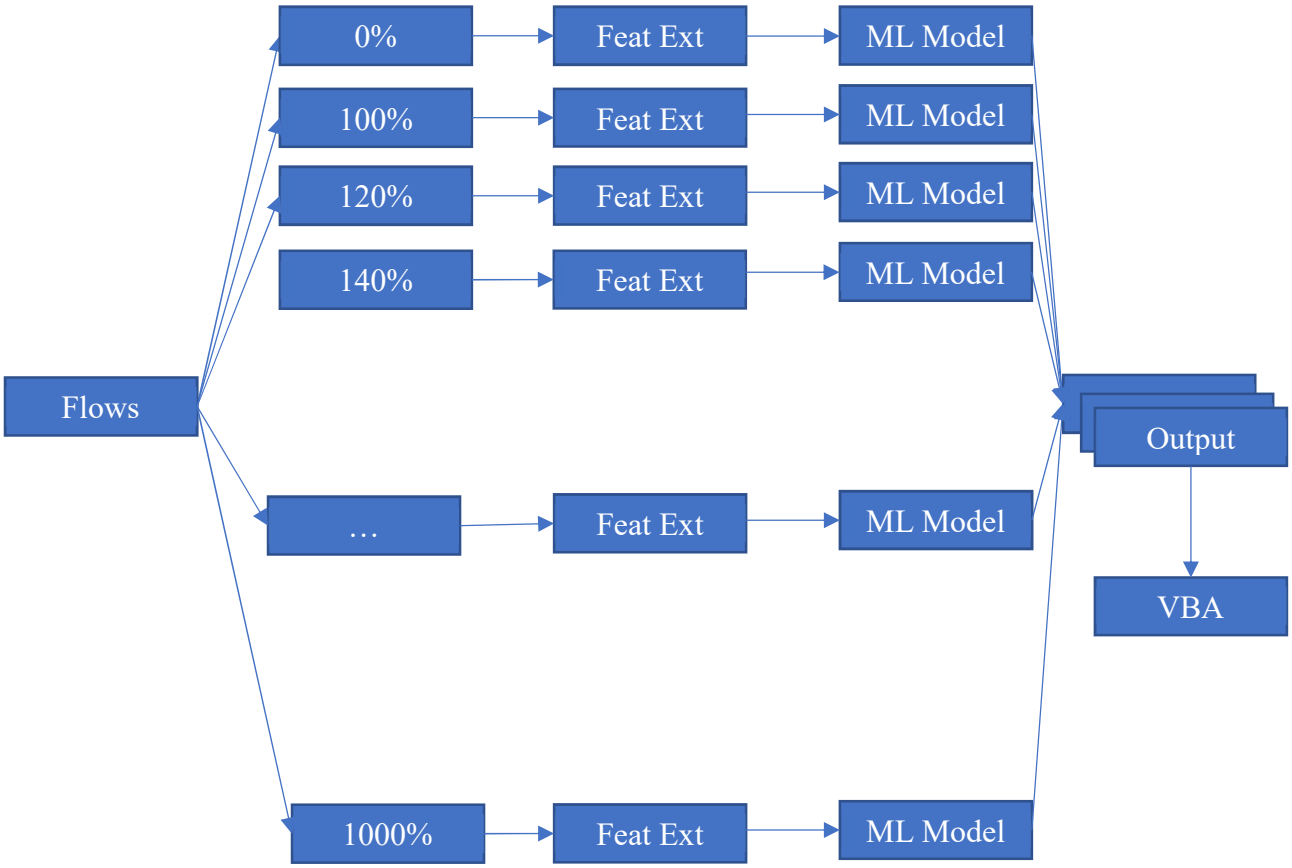

Figure 14 Surge test

As mentioned one of the requirements of any generated model is to perform well under surges in network traffic. In this phase we apply surges to the network traffic and reapply the best machine learning model obtained in the previous sections.

Step 1: A new table is generated for each surge value. 0% would be an exact replicate of the current flow table, while 100% would a table with twice as much regular traffic as the current flow table.

Step 2, Feat Ext: The required features are generated from the newly generated flow tables.

Step 3, ML Model: The output files are put through the machine learning model.

Step 4, VBA: The results are post processed and merged into a table for analysis.

6.4.2 Darpa Dataset

The purpose of the Darpa dataset is to prove the generality of the model and to show that the model is not only effective on one dataset. It also allows us to compare our results with previous work in this field. If the top model obtained using the ISCX dataset proves effective, then we have been able to prove the effectiveness of the model.

The main issue with the Darpa dataset is that the extracted features do not give us all the predictors we require in order to run our model. Therefore, we need to manually extract the features from the binary pcap files and match the flows with the labeled files. Below I have explained the steps required to achieve this.

6.4.2.1 Phase 1, Obtaining the Dataset

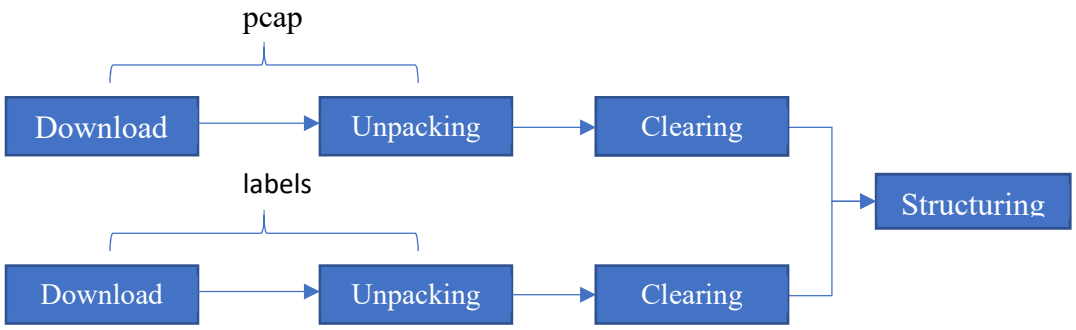

Figure 15 Obtaining the dataset

The Darpa Dataset consists of 7 weeks of data. In each week data had been collected for 5 days. There was a total of 35 days of collected data. The data can be found on the MIT Lincoln labs website. In order to perform the model over this dataset both the pcap files along with the labels were needed. One issue with the data was that it was scattered on multiple pages over the website.

The labels were also in an entirely different location. A group of scripts were written to download and structure the data in an easier to user format.

Download: Each of the 35 different pcap files where separately located and downloaded into a central location. The same was done for the labels.

Unpacking and Clearing: Each pcap file and label was zipped along with a series of extra files that were not required. After extracting each zip file, the extra content was removed.

Structuring: The data was then placed in a structured directory format. 7 folders one for each week. In each of these folders there were 5 additional folders one for each day. In each of those folders there was 1 pcap file and one label file.

6.4.2.2 Phase 2, Setting Up Database

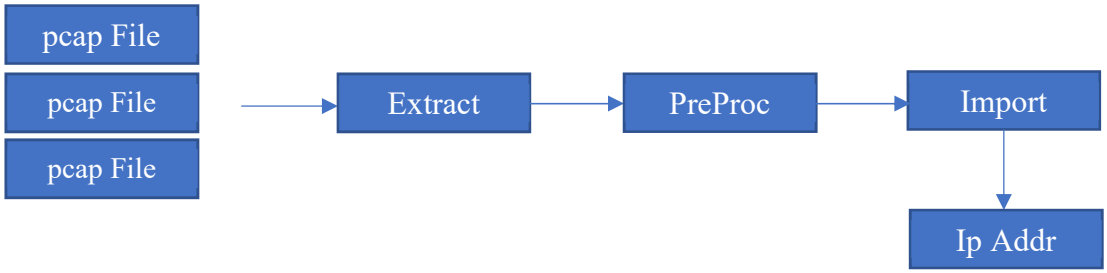

Figure 16 Setting up the database

Unlike the ISCX Dataset where the flows were provided to us, in this dataset we need to generate the flows ourselves.

Extract: Pcap files consist of binary data. While they are a very compact form of storing data, we need to extract human readable values from these files. In this first stage certain feature that we are interested in are extracted from these files and outputted into a csv file. Originally a program

was written in C to perform. Later it was realized that tshark (a command line version of wireshark for linux) is also able to do this and it was used.

PreProc: Corrupt data in any of the network layers will result in invalid data appearing in the output columns of the previous stage. In this stage the checksums extracted and computed from the previous stage are compared, both in the ethernet layer and in the network layer. If there are discrepancies, the payloads are nulled.

Import: The data obtained in the previous stage is inserted into a MySQL database. The data from all the Pcap files will appear as one big table.

IP Addr: In this stage an Ip address table is created. All the unique Ip addressed that appear in the Pcap files are extracted and inserted into this table. The packet tables ip address fields are then linked back to this table.

6.4.2.3 Phase 3 Flow Generation:

Packets alone do not provide much information about the traffic. In this stage all the flows are extracted and inserted into a new table. This table is linked back to the packet table to be able to later obtain aggregated features such as the number of packets or bytes transferred by the flow.

In order to generate flows 5 features of the packets are considered

- Source IP
- Destination IP
- Source Port
- Destination Port
- Protocol

If the packet only contains IVP4 layer data, the port values would be set to negative 1. Below we present the algorithm used for generating the flows. The algorithm has a simple implementation therefore it was chosen. Later we present another algorithm which is scalable and can be used in a deployment version of this model (Figure 17).

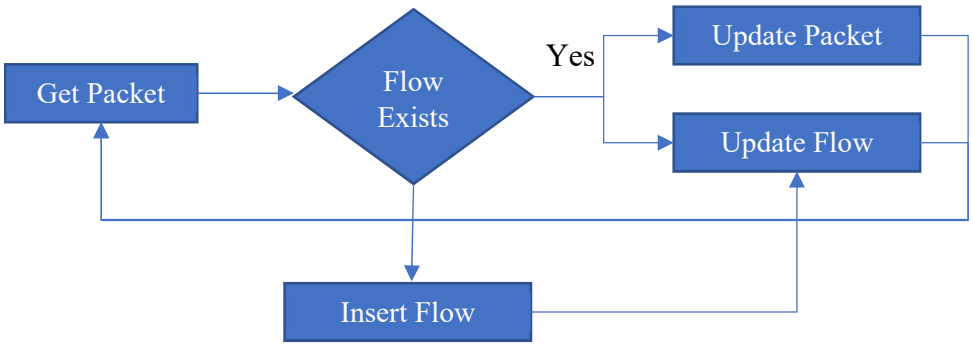

Figure 17 Flow generation

Get Packet: The packet table is sorted by the time column. The first packet that has not been assigned a flow is selected.

Flow Exists: The flow table is checked if a matching flow exists for the packet. In order for a matching flow to exists the following conditions must be met:

1 – The 5 values mentioned above must match (IP address source and dest, port source and dest and protocol)

2 -The last packet seen by the flow should not have been more than 60 seconds ago.

Update Packet: The packets flow column is update to match the id of the associated flow in the flow columns

Update Flow: The end time of the flow is updated in the flow table. This is used for matching the second criterion in the flow exists section.

Insert Flow: A new record is inserted into the flow table. The start and end times are set to the time of the packet.

6.4.2.4 Phase 4, Matching Labels

In the previous section we were able to generate flows, which can then be used to generate the required features for our machine learning model. The issue is that the extracted labels are not labeled. In order to label them we need to match them with the provided xml file that contains the tags for the flows (Figure 18).

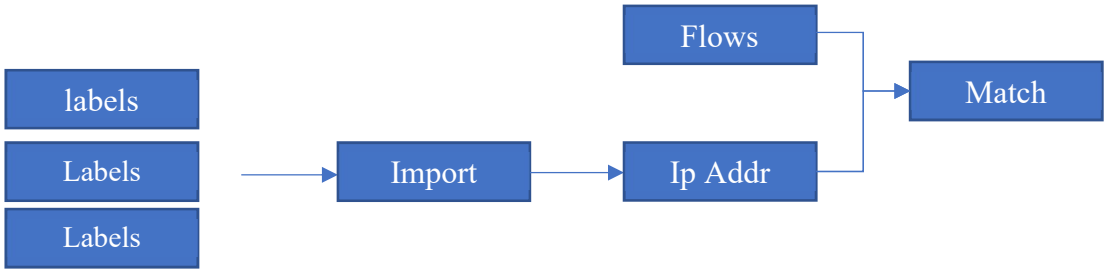

Figure 18 Maching labels

Import: The labeled xml files are imported into the MySQL database. They are all combined into a single table.

IP Addr: One of the tags in the xml file is the source and destination ip address. As we already have a table with a list of all Ip addresses, the Ip addresses in this table are matched with the ones in the previous table.

Match: The labeled flows are matched with the flows obtained using the Pcap file by looking at:

- source and destination ip address

- source and destination port number
- start day and time of the flow

6.4.2.5 Phase 4, Post Process

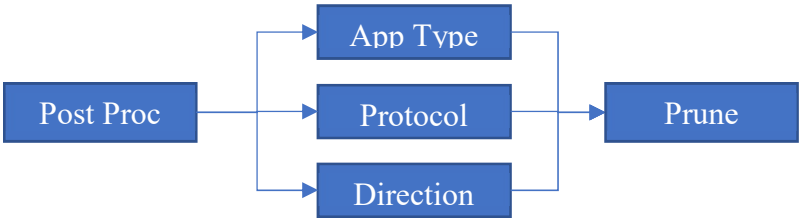

Figure 19 Post Process

PostProc: The model requires the total bytes and packets transferred during the flows. As the flow table and the packet table are linked and indexed, this information is obtained by aggregating the information in the packet table and applying it to the flow table.

App Type: Unlike the previous sections were the App Type was provided, this is however not always the case. A good indicator of the application that was used to generate the flow are the port numbers in the connections. Although this is not always an exact one to one match, however it can be said that there is some correlation between port numbers and the application used in the connection.

In every connection there is normally two ports. One on the receiving side and another on the client side. The client side is normally some random value, it is usually the receiving side that has a well-known port number. The following procedure was used for generating the App Type column:

1- If the flow is not transport layer, then the app type is set to -1.

2- As the better-established port numbers are below 1024 we only consider these. If both port numbers are larger than 1024, we set the application type to 1025.
3- If both port numbers are below 1024 we choose the smallest (although this would be a very unusual case as normally there is only one side uses a privileged port number).
4- If one port number is larger than 1024 and the other is smaller, we set the app type as the smaller port number. So for example a flow might have an app type of 80, 22, 443, …

After obtaining the app type of each flow, the unique values are obtained and put in a separate table. The ids of these records are then replaced by the app types in the original table.

Protocol: For transport layer flows this column would be the transport layer protocol. For other flow types this would be set as NA. Similar to the app type column explained above a new table is generated and the unique values are inserted into this table. The ids are then replaced by the protocols in the original flow table.

Direction: In our research we consider a node as local if it belongs to any of the following subdomains:

- 10.0.0.0 - 10.255.255.255
- 172.16.0.0 - 172.31.255.255
- 192.168.0.0 - 192.168.255.255

All other nodes are considered remote. Depending on the direction of the flow, it will be labeled as either LL, RL, LR or RR.

Prune: The different modules in this project were written at different times and they expect different format for their input data. In this stage we rename the columns in the database to match the format expected by the other modules in the program.

6.4.2.6 Step 5, Applying Surge, Generating Features and and Running Model

This is similar to phase 5 of the ISCX dataset. Surges are applied, features are generated and the machine

6.5 Deployment and Architecture

The steps outlined in the previous sections are the steps taken in the research phase of the project. Once the best model is obtained we need a scalable way to apply the model in real time to incoming network traffic. We propose the following architecture for the actual deployment (Figure 20).

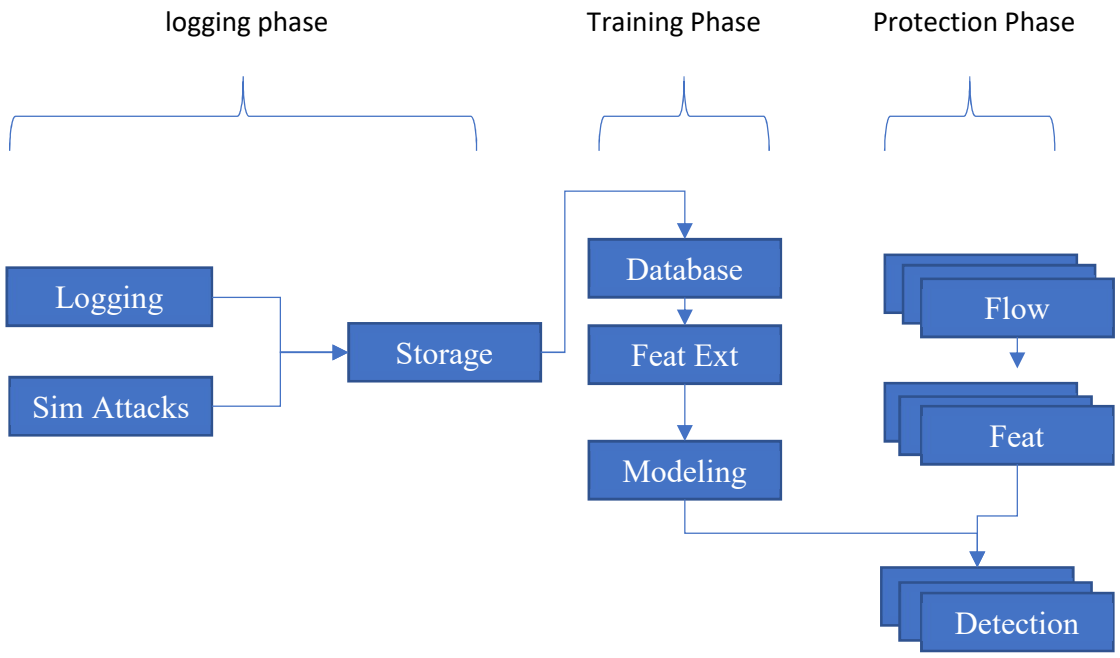

Figure 20 Deployment Architecture

logging phase: In this phase the network traffic is logged along with controlled labeled attacks. This information will be stored in Pcap files along with a detailed list of attacks that have taken

place during this period to use as labels. This phase would normally last a couple of weeks. No real-time response is necessary at this stage.

Training phase: In this phase the information gathered during the logging phase is extracted from the Pcap files, inserted into the database and structured. Then a model is built similar to what was done in this project. Again, this stage does not require any real-time response either. Even if retraining is required on regular periods this is still feasible as the training the machine learning models did not take considerable amount of time in our research.

Protection phase: This is the only phase that requires real time response. As traffic is captured the flows are to be extracted in real time. This can either be done by a variant of the program that we have implemented in this research, or using already available hardware that generate flow in real time. After the flow has been generated the rest of the process is scalable and can be done on parallel processors or even clusters of computers. A separate process is assigned to each flow to generate the required features. The features are then put inside the trained model and the result is output. Again, as the model is already generated this would take very little computational power and is scalable.

6.6 Results

6.6.1 ISCX 2012 Dataset

The results have been brought in the table below:

- Model: The name of the model. Details can be found in the sections above
- Total Accuracy: The accuracy of the model using all the records
- Attack Accuracy: The accuracy of the model only considering attack records

- False Alarms: the ratio of false alarms
- Predictors: This is a list of predictors used in the model. They are displayed in the order selected by the forward selection model. In the ISCX2012 dataset the predictors were as follows:
    - 1-108: Applications (Table 4)
    - 109-114: protocols (Table 5)
    - 115-116: Unique local and remote nodes
    - 117-120: Directions
    - 121: packet count
    - 122: byte count

Some rows have an NA. This is caused by the instability and the inability of the model to converge using the provided data and model (Table 10).

Table 10 Results

Model	Total Accuracy	Attack Accuracy	False Alarms	Predictors
LOGI1E0S0C0	0.9893	0.8752	0.0037	119, 120, 117, 16, 86, 66, 26, 30, 114, 6, 35, 91, 76, 28, 92
LOGI1E0S0C1	0.9774	0.9420	0.0073	86, 117, 118, 35, 16, 66, 91, 24, 29
LOGI1E0S1C0	0.9894	0.8760	0.0037	119, 120, 117, 16, 86, 66, 26, 30, 35, 114, 91, 46, 6, 44, 80
LOGI1E0S1C1	0.9777	0.9417	0.0070	86, 117, 118, 35, 16, 116, 66, 9
LOGI1E1S0C0	0.9896	0.8685	0.0037	119, 35, 120, 117, 26, 16, 25, 115, 91, 74, 54, 96, 28, 81, 22
LOGI1E1S0C1	0.9790	0.9424	0.0059	117, 118, 86, 16, 66, 35, 96, 91, 74, 30, 25
LOGI1E1S1C0	0.9896	0.8723	0.0036	119, 35, 120, 117, 16, 26, 25, 66, 91, 74, 96, 11, 105, 49
LOGI1E1S1C1	0.9790	0.9424	0.0059	117, 118, 16, 86, 66, 35, 96, 91, 74, 30, 25

Model	Total Accuracy	Attack Accuracy	False Alarms	Predictors
LOGI5E0S0C1	0.9519	0.8635	0.0190	26, 86, 118, 11, 35, 30, 66, 81, 8, 62
LOGI5E0S1C0	0.9754	0.8351	0.0052	35, 91, 86, 96, 117, 118, 26, 16, 66, 30, 54, 116, 102, 122
LOGI5E0S1C1	0.9519	0.8635	0.0190	26, 86, 118, 11, 35, 30, 66, 81, 8, 62
LOGI5E1S0C0	0.9696	0.1853	0.0041	116, 35, 119, 11, 6, 91, 74
LOGI5E1S0C1	0.9661	0.8751	0.0073	117, 118, 66, 16, 35, 11, 30, 91, 96, 33, 92, 74
LOGI5E1S1C0	0.9698	0.1856	0.0040	116, 35, 11, 119, 28, 91, 74, 105, 48
LOGI5E1S1C1	0.9664	0.8743	0.0069	117, 118, 16, 66, 35, 11, 91, 96, 28, 74, 116, 30, 86, 46
LOGI10E0S0C0	0.9650	0.4836	0.0040	119, 86, 91, 11, 116, 74
LOGI10E0S0C1	0.9448	0.8420	0.0225	118, 86, 11, 66, 35, 96, 60
LOGI10E0S1C0	0.9655	0.4841	0.0040	119, 74, 86, 11, 91, 28
LOGI10E0S1C1	0.9602	0.8564	0.0100	25, 117, 119, 78, 35, 118, 11, 91, 74, 30, 81, 105, 95
LOGI10E1S0C0	0.9731	0.9363	0.0052	11, 35, 119, 120, 74, 91, 26, 117, 25, 105, 81, 95, 46, 96
LOGI10E1S0C1	0.9631	0.8561	0.0070	118, 117, 119, 16, 35, 11, 66, 30, 91
LOGI10E1S1C0	0.9737	0.9340	0.0050	119, 11, 35, 120, 26, 117, 25, 54, 95, 81, 91, 105, 46, 28, 42, 116
LOGI10E1S1C1	0.9640	0.8561	0.0062	118, 117, 119, 35, 16, 11, 91, 25, 26, 74, 86, 30, 78, 105, 54, 115, 81, 20, 96, 97
LDAI1E0S0C0	0.9682	0.9373	0.0221	26, 54, 66, 30, 105, 16, 116, 91
LDAI1E0S0C1	0.9481	0.9366	0.0339	26, 86, 119, 25, 116, 30, 78, 35, 121, 91, 101
LDAI1E0S1C0	0.9775	0.9629	0.0183	54, 66, 30, 105, 16, 91, 26, 119, 117, 118
LDAI1E0S1C1	0.9506	0.9369	0.0315	26, 86, 119, 25, 116, 91, 16, 35, 66, 11, 81
LDAI1E1S0C0	0.9866	0.9261	0.0085	86, 118, 117, 16, 26, 66, 105, 91, 74, 114, 95, 51
LDAI1E1S0C1	0.9728	0.9238	0.0056	117, 118, 13, 16, 35, 76, 91, 116, 74, 6, 5, 4, 121, 19, 41
LDAI1E1S1C0	0.9882	0.9315	0.0071	86, 118, 117, 16, 26, 119, 28, 120, 104

Model	Total Accuracy	Attack Accuracy	False Alarms	Predictors
LDAI1E1S1C1	0.9740	0.9371	0.0081	117, 118, 13, 16, 35, 76, 86, 110, 119, 33, 78
LDAI5E0S0C0	0.9637	0.9563	0.0210	54, 66, 30, 91, 16, 105, 119
LDAI5E0S0C1	NA	NA	NA	NA
LDAI5E0S1C0	0.9638	0.9272	0.0143	119, 120, 117, 16, 26, 118, 42
LDAI5E0S1C1	NA	NA	NA	NA
LDAI5E1S0C0	0.9669	0.0951	0.0069	118, 35, 11
LDAI5E1S0C1	0.9213	0.9070	0.0216	54, 116, 66, 86, 30, 35, 96, 16, 119, 102, 91, 90, 81, 60, 74
LDAI5E1S1C0	0.9741	0.8011	0.0075	118, 11, 35, 74, 116, 26, 117, 16, 96, 66, 95, 81, 78, 91, 60, 113
LDAI5E1S1C1	0.9417	0.9313	0.0162	54, 116, 66, 86, 30, 35, 96, 119, 16, 117, 11, 91, 120, 46, 93
LDAI10E0S0C0	0.9392	0.9613	0.0323	119, 120, 117, 118, 105
LDAI10E0S0C1	NA	NA	NA	NA
LDAI10E0S1C0	0.9671	0.9784	0.0139	117, 26, 66, 118, 16, 30, 54, 105, 91, 96, 25, 4, 114, 35, 78, 6
LDAI10E0S1C1	NA	NA	NA	NA
LDAI10E1S0C0	0.9624	0.3720	0.0076	11, 35, 118, 25, 60
LDAI10E1S0C1	0.9321	0.9324	0.0167	116, 86, 117, 35, 66, 33, 6, 119, 16, 74
LDAI10E1S1C0	0.9634	0.3759	0.0087	11, 118, 35, 119, 30, 81, 29
LDAI10E1S1C1	0.9321	0.9324	0.0167	116, 86, 117, 35, 66, 33, 6, 119, 16, 74, 29, 17
MLDAI1E0S0C0	0.9834	0.9542	0.0134	117, 118, 16, 26, 28, 119
MLDAI1E0S0C1	0.9468	0.9548	0.0404	118, 117, 119, 122, 24
MLDAI1E0S1C0	0.9761	0.9578	0.0208	117, 118, 16, 95, 26, 28, 25, 46, 91, 74, 51, 121
MLDAI1E0S1C1	0.9468	0.9548	0.0404	118, 117, 119, 122, 24
MLDAI1E1S0C0	0.9756	0.1894	0.0000	91, 74, 2, 51, 41
MLDAI1E1S0C1	0.9769	0.9528	0.0097	117, 118, 110, 16, 35, 11, 95, 91, 74, 13, 19, 26, 96
MLDAI1E1S1C0	0.9756	0.1894	0.0000	91, 74, 89, 51
MLDAI1E1S1C1	0.9756	0.9503	0.0103	118, 117, 110, 16, 35, 119, 115
MLDAI5E0S0C0	0.9652	0.9669	0.0206	117, 118, 91, 16, 28, 11, 46, 66, 30, 54, 114, 95
MLDAI5E0S0C1	0.6428	0.9958	0.3547	46
MLDAI5E0S1C0	0.9677	0.9583	0.0185	119, 91, 11, 95, 46, 120, 117, 26, 25, 96, 16, 118, 78, 81, 28, 6, 2
MLDAI5E0S1C1	0.6428	0.9958	0.3547	46

Model	Total Accuracy	Attack Accuracy	False Alarms	Predictors
MLDAI5E1S0C0	0.9755	0.8887	0.0075	11, 35, 118, 120, 26, 91, 25, 96, 74
MLDAI5E1S0C1	0.9226	0.8987	0.0152	116, 86, 117, 11, 118, 16, 35, 119, 84, 78
MLDAI5E1S1C0	0.9746	0.8371	0.0070	11, 35, 117, 120, 26, 16, 53, 4, 91
MLDAI5E1S1C1	0.9228	0.9071	0.0201	116, 86, 117, 118, 11, 35, 119, 43, 12
MLDAI10E0S0C0	0.9692	0.9744	0.0123	119, 91, 11, 46, 86, 120, 117, 26, 16, 30, 66, 54, 105, 115, 25
MLDAI10E0S0C1	NA	NA	NA	NA
MLDAI10E0S1C0	0.9584	0.9802	0.0237	117, 118, 26, 66, 91, 28, 11, 46, 95, 86, 30, 119
MLDAI10E0S1C1	NA	NA	NA	NA
MLDAI10E1S0C0	0.9648	0.3910	0.0079	11, 35, 120, 74, 119, 25, 91
MLDAI10E1S0C1	0.9087	0.9078	0.0215	54, 66, 116, 86, 35, 11, 30, 114, 119, 4, 16, 81, 91
MLDAI10E1S1C0	0.9641	0.4568	0.0083	11, 35, 118, 54, 91, 51, 119, 30, 114, 17
MLDAI10E1S1C1	0.9091	0.9095	0.0224	54, 66, 116, 86, 11, 35, 30, 114, 119, 16, 4, 81, 91, 6, 102, 74
MQDAI1E0S0C0	0.9707	0.9528	0.0265	117, 118, 122
QDAI1E0S0C0	0.8171	0.9853	0.1814	117, 96, 102, 66
QDAI1E0S0C1	0.3178	0.9977	0.6816	46
QDAI1E0S1C0	NA	NA	NA	NA
QDAI1E0S1C1	0.3178	0.9977	0.6816	46
QDAI1E1S0C0	0.9758	0.0420	0.0071	33, 28
QDAI1E1S0C1	0.9806	0.9564	0.0070	117, 86, 16, 118, 114, 11, 35, 66
QDAI1E1S1C0	0.9758	0.0421	0.0071	33, 28, 95
QDAI1E1S1C1	0.9821	0.9620	0.0071	117, 86, 16, 118, 114, 11, 35, 66, 60, 78, 104, 26, 116, 61, 110, 95, 46, 105
QDAI5E0S0C0	NA	NA	NA	NA
QDAI5E0S0C1	NA	NA	NA	NA
QDAI5E0S1C0	NA	NA	NA	NA
QDAI5E0S1C1	NA	NA	NA	NA
QDAI5E1S0C0	0.9725	0.0485	0.0112	49, 104, 95
QDAI5E1S0C1	0.9273	0.9076	0.0159	116, 86, 118, 117, 16, 53
QDAI5E1S1C0	0.9725	0.0485	0.0112	49, 104, 95
QDAI5E1S1C1	0.9273	0.9076	0.0159	116, 86, 118, 117, 16, 53
QDAI10E0S0C0	0.0497	1.0000	0.9503	107

Model	Total Accuracy	Attack Accuracy	False Alarms	Predictors
QDAI10E0S0C1	NA	NA	NA	NA
QDAI10E0S1C0	0.0497	1.0000	0.9503	107
QDAI10E0S1C1	NA	NA	NA	NA
QDAI10E1S0C0	0.9697	0.0325	0.0111	49, 95, 115, 114
QDAI10E1S0C1	0.9095	0.9190	0.0291	116, 86, 119, 20
QDAI10E1S1C0	0.9697	0.0325	0.0111	49, 95, 114, 115
QDAI10E1S1C1	0.9095	0.9190	0.0291	116, 86, 119, 20
MQDAI1E0S0C1	NA	NA	NA	NA
MQDAI1E0S1C0	0.9759	0.9538	0.0211	117, 118, 26
MQDAI1E0S1C1	NA	NA	NA	NA
MQDAI1E1S0C0	NA	NA	NA	NA
MQDAI1E1S0C1	0.9686	0.9615	0.0205	117, 118, 26
MQDAI1E1S1C0	NA	NA	NA	NA
MQDAI1E1S1C1	0.9686	0.9615	0.0205	117, 118, 26
MQDAI5E0S0C0	0.3990	0.9872	0.5980	118, 121, 122
MQDAI5E0S0C1	0.6418	0.9969	0.3564	122
MQDAI5E0S1C0	0.1631	0.9883	0.8358	96, 121, 122
MQDAI5E0S1C1	0.6418	0.9969	0.3564	122
MQDAI5E1S0C0	NA	NA	NA	NA
MQDAI5E1S0C1	0.9288	0.9242	0.0247	118, 119, 26, 78
MQDAI5E1S1C0	NA	NA	NA	NA
MQDAI5E1S1C1	0.9288	0.9242	0.0247	118, 119, 26, 78
MQDAI10E0S0C0	0.8786	0.9383	0.0874	30
MQDAI10E0S0C1	NA	NA	NA	NA
MQDAI10E0S1C0	0.8810	0.9507	0.0861	66
MQDAI10E0S1C1	NA	NA	NA	NA
MQDAI10E1S0C0	NA	NA	NA	NA
MQDAI10E1S0C1	0.8930	0.9107	0.0393	120, 119, 26, 25
MQDAI10E1S1C0	0.9558	0.0293	0.0144	96
MQDAI10E1S1C1	0.8930	0.9107	0.0393	120, 119, 26, 25

6.6.1.1 Model Evaluation

The best model is a model that has the best attack prediction along with the lowest false alarms. In order to find the best model, we assign 2 numbers to each model:

1- First Number: We order the models from best model in predicting attacks to worst. The number will be the models rank in the list

2- Second number: We order the models in order of lowest false alarms to highest. The number will be the models rank in this list.

The best models will be the models where the sum of the 2 numbers above is least. The top 20 models are printed below (Table 11):

Table 11 Top 20

Model	Attack Accuracy	Attack Ranking	False Alarms	False Alarm Ranking	Total Ranking
QDAI1E1S0C1	0.9564	23	0.0070	21	44
QDAI1E1S1C1	0.9620	17	0.0071	27	44
LOGI1E1S0C1	0.9424	33	0.0059	15	48
LOGI1E1S1C1	0.9424	34	0.0059	16	50
LOGI10E1S1C0	0.9340	43	0.0050	11	54
LOGI10E1S0C0	0.9363	42	0.0052	12	54
LOGI1E0S1C1	0.9417	36	0.0070	20	56
MLDAI10E0S0C0	0.9744	14	0.0123	45	59
LDAI10E0S1C0	0.9784	13	0.0139	47	60
LOGI1E0S0C1	0.9420	35	0.0073	29	64
LDAI1E1S0C1	0.9238	52	0.0056	14	66
MLDAI1E1S0C1	0.9528	30	0.0097	38	68
LDAI1E1S1C0	0.9315	46	0.0071	24	70
LOGI1E0S0C0	0.8752	66	0.0037	5	71
LOGI1E0S1C0	0.8760	65	0.0037	6	71
LOGI1E1S1C0	0.8723	69	0.0036	3	72
MLDAI1E1S1C1	0.9503	32	0.0103	40	72
LDAI1E0S1C0	0.9629	16	0.0183	56	72
LDAI1E1S1C1	0.9371	39	0.0081	34	73
MLDAI1E0S0C0	0.9542	27	0.0134	46	73

The results of applying the top model to the data in day 7 can be seen below (

Table 12):

Table 12 Day 7 Results

Model	Total Accuracy	Attack Accuracy	False Alarms
QDAI1E1S0C1	0.9535	0.8302	0.0030

6.6.1.2 Interpretation of the Results

Interval size: Figure 22 shows thatFigure 1 most of the top 20 models had the 1s interval size as their interval size. This shows that following sharp changes in network traffic rather than an averaged effect over an interval was better for detecting the attacks in the dataset. The intuition behind this can be seen in Figure 21.

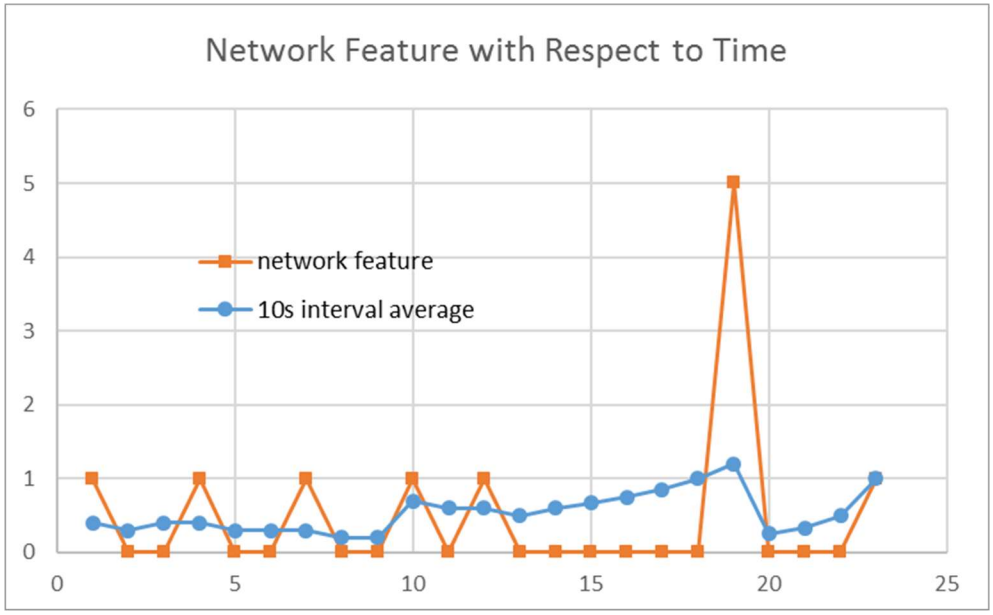

Figure 21 Effects of different interval sizes

While at the time t=19 there was an anomaly, but by averaging the effect over 10 second intervals we have basically faded out the anomaly. Although in our current dataset using a small interval has improved performance this may not always be the case. As smaller intervals will also catch a lot of noise.

The figures below show the distribution of the parameters selected by the top 20 models

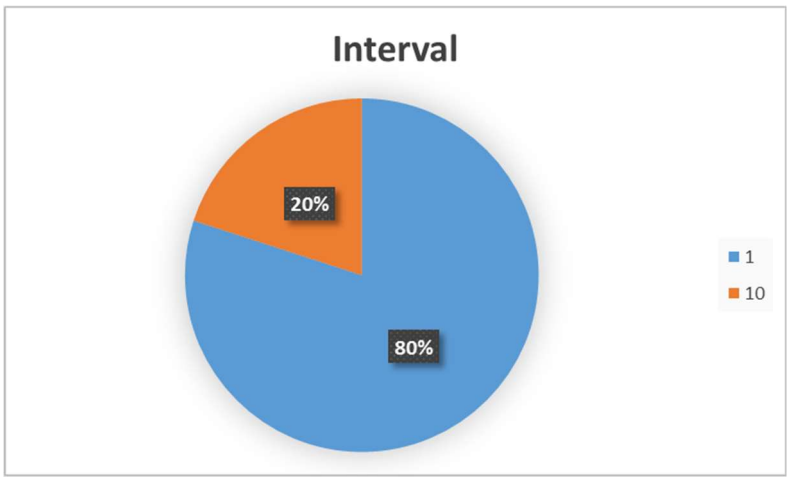

Figure 22 Ratio of interval size selected by the top 20 models

Copying Attack Records: Due to the fact that the number of attack records were far less than the number of normal records one of the ideas that were tried was to copy the attack records by the number of attack flows it represents as it was mentioned in the previous sections, in order to generate data points there is a sliding window that slides over 1 second at a time. In each interval we consider the flows that start during that time period. I would show the intervals that contain attack flows multiple times, based on the number of attack flows that it contains. As it can be seen in Figure 23 there was a 50/50 split in the top 20 models choosing between copying and not copying. This shows that by selecting enough features we can obtain good accuracy even when the number of attack records is much less than the number of normal records

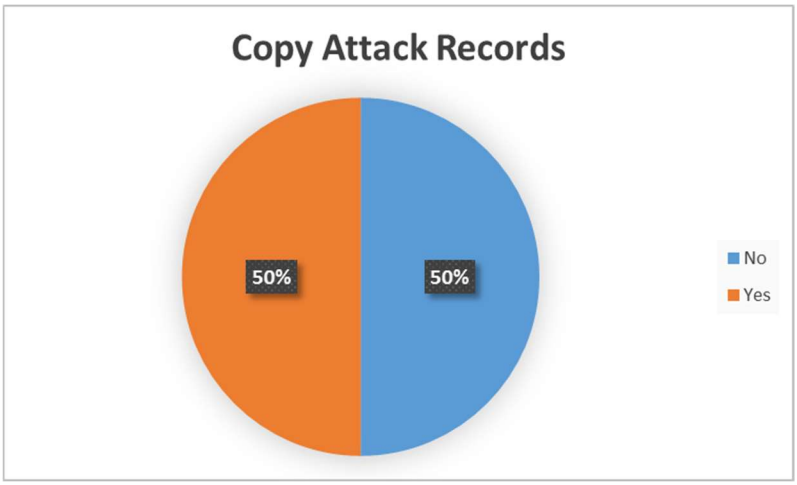

Figure 23 Ratio of top 20 models that considered copying the attack records

Evaluation Based on Attack Records: In the feature selection phase there were 2 steps. In the first step each of the predictors are used in a single predictor model and the error is calculated. Then the best among all these predictors is chosen. 2 different approaches were considered in this step for calculating the error. In one approach the error was calculated based off all the records. In the other approach the error was calculated only based off attack records. Figure 24 shows that the top 20 models leaned towards the second approach. This was mainly because we had much less attack records than normal records. By calculating the error based off attack records more emphasis is placed on these records.

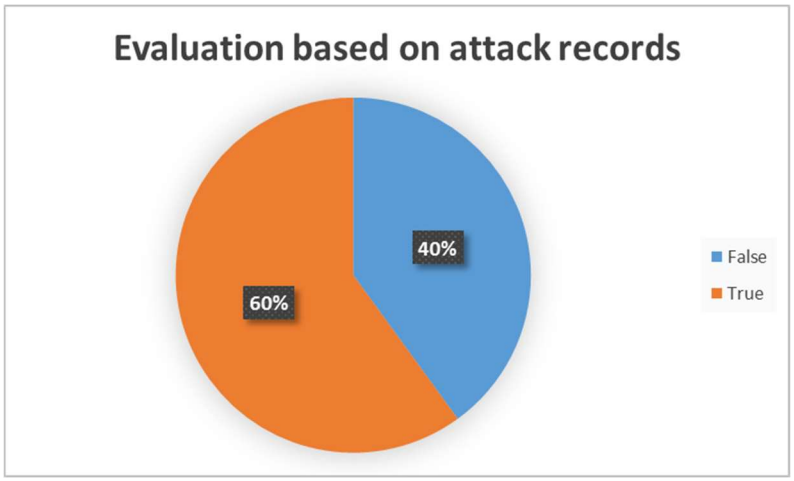

Figure 24 Ratio of the top 20 models that the feature evaluation was based on the attack records

Evaluation Based on Validation Set: Similar to the case of "Evaluation based on attack records" this model was involved with the method of calculating the error term in the first step of the feature selection phase. The usual method for calculating the error term is based off the training set. However, another method was also considered were the error term is calculated based off the validation set. This may cause some overfitting, however since the models were validated based off a third dataset, this was not of much concern. Figure 25 shows that the top 20 models showed a 50/50 split between the 2 methods, indicating that both methods perform as well.

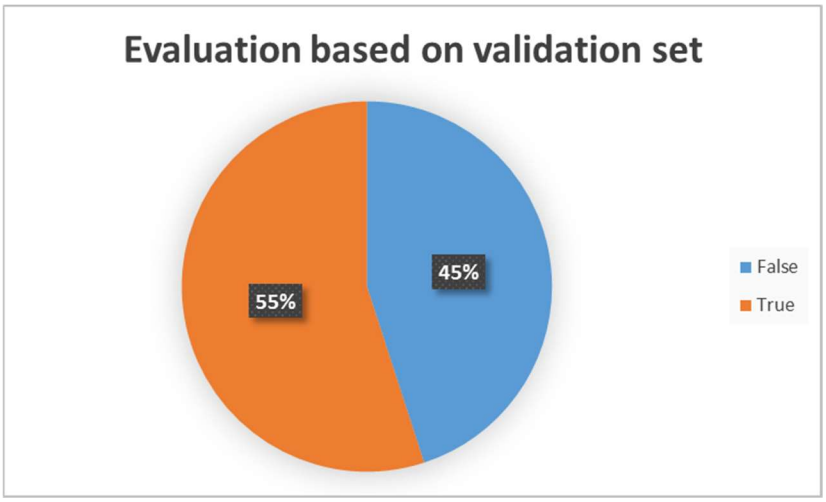

Figure 25 Ratio of the top 20 models that the feature evaluation was based on the validation set

Top 20 predictors:

In Figure 26 I have plotted the top predictors selected by the models

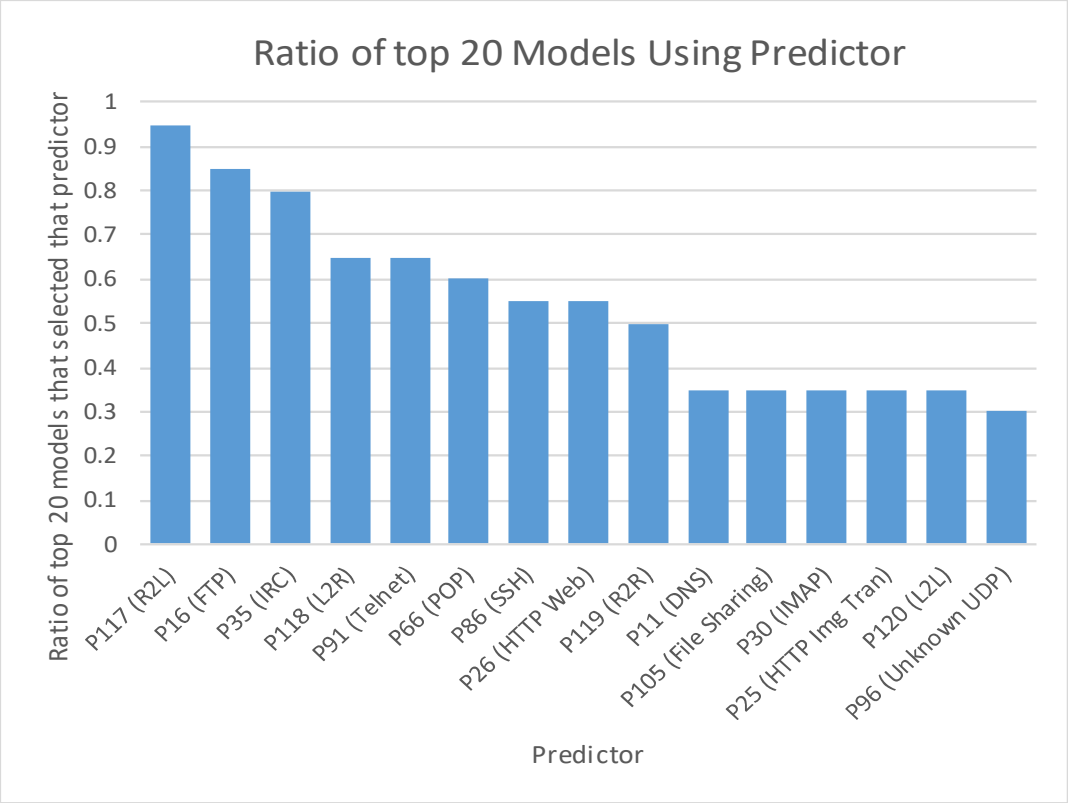

Figure 26 Predictors selected in the top 20 models

117: The predictor that appeared most in the top 20 models was predictor 117. This predictor showed the ratio of remote to local connection during that interval. By ratio we are referring the ratio of remote to local connection over the sum of:

- Remote to remote
- Local to local
- Local to remote
- Remote to local

This does fall in line with the fact that during DDOS attacks there will be a lot of connections initiated with the local nodes from remote addresses

16 (FTP), 35(IRC Chat), 91 (Telnet), 66 (POP), 86(SSH), 26(HTTP Web), 11(DNS), 30(IMAP), 25(HTTP Image Transfer): These were also among the top 15 predictors used by the

top 20 models. The machine learning models detected that monitoring the traffic from these applications is critical as they pose the most risk. FTP and SSH could be used for brute force attacks. IRC, DNS, POP, IMAP for DDOS attacks. Telnet and HTTP for custom made attacks.

118: This was the 4th most common predictor among the top 20 models. Predictor 118 is the ratio of local to remote connections initiated during that time interval. As most the attacks in the data sets were remote to local, predictor 118 was a good indicator that the connection is not an attack

199: This was the 9th most common predictor among the top 20 models. Predictor 119 is the ratio of remote to remote connections initiated during that time interval. While a remote to remote connection is a very suspicious connection but due to the fact that our training dataset did not label any of the remote to remote connection as attacks, similar to 118 this predictor was an indicator that the connection is not an attack.

120: This was the 2nd last most common predictor among the top 20 models. Predictor 120 is the ratio of local to local connections initiated during that time interval. An increase in this parameter would indicate that a lot of activity is happening inside the network. If this predictor raises beyond a certain amount, then it could be an indicator of an attack.

Other Predictors: Asides from the predictors above there were several other predictors that were not among the top 15 predictors selected. Among these predictors there was the ip protocol. Initially it was assumed that this predictor would have an impact on our models, but results show otherwise. This was mainly due to the fact that attacks can happen using any type of protocol (TCP, UDP, …). Therefore, keeping track of the protocol is not very useful in detecting attacks.

Another predictor that was initially assumed to have an impact on our models was the ratio of connection per unique IP. In other words, an average fan-in and fan-out over all nodes active during

that interval. After further investigation the reason why, these predictors were not considered important became apparent. The attacks were mainly from a large number of node to a large number of nodes. Therefore, the average fan-in and fan-out per active node isn't a very large number.

The last 2 predictors that were not considered important by the model, was the average number of packets per connection and the average bytes per connection. The reason for this was because the attack connection had similar packet and traffic patterns to normal connections.

6.6.2 Surge Test

After running the model on each of the ratios above the results have been plotted below (Figure 27):

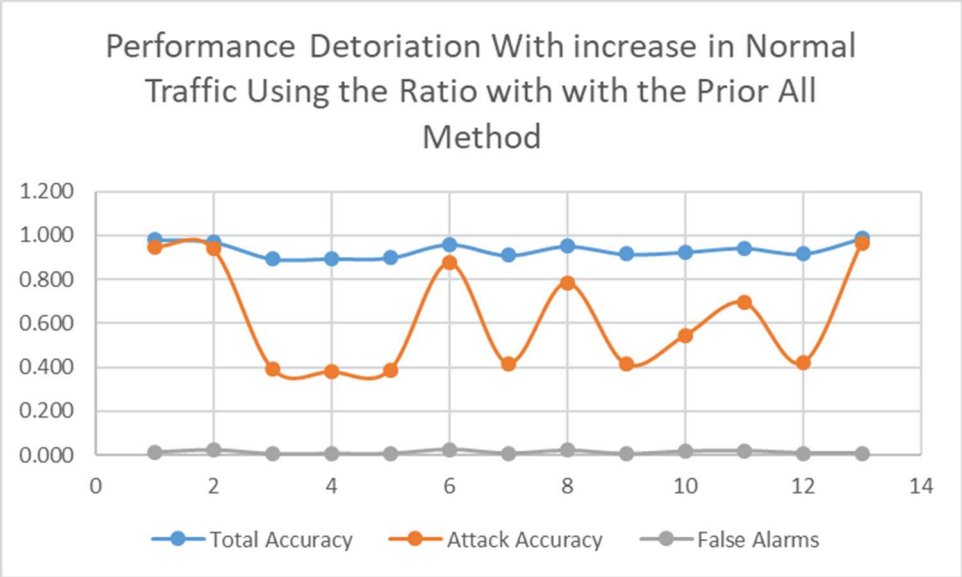

Figure 27 Performance deterioration with increase in normal traffic using the ratio with the prior all method

It can be seen that increasing normal flow causes the results to become very unstable and highly correlated with the split. In order to overcome this several other methods are tested and plotted below (Figure 28):

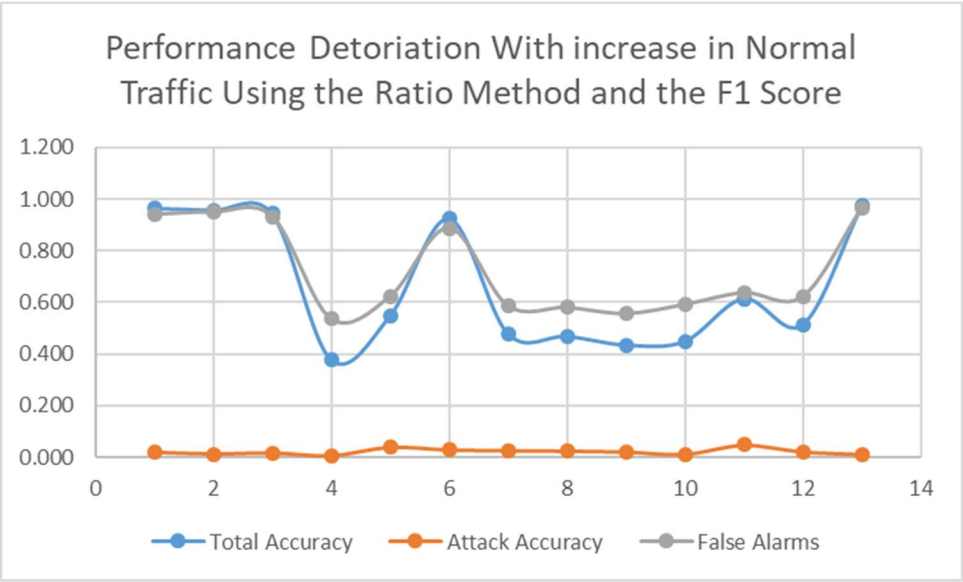

Figure 28 Performance deterioration with increase in normal traffic using the ratio method and the F1 score

Using the F1 score did not improve the instability. Using full flow counts did improve the attack accuracy at the cost of false alarms (Figure 29).

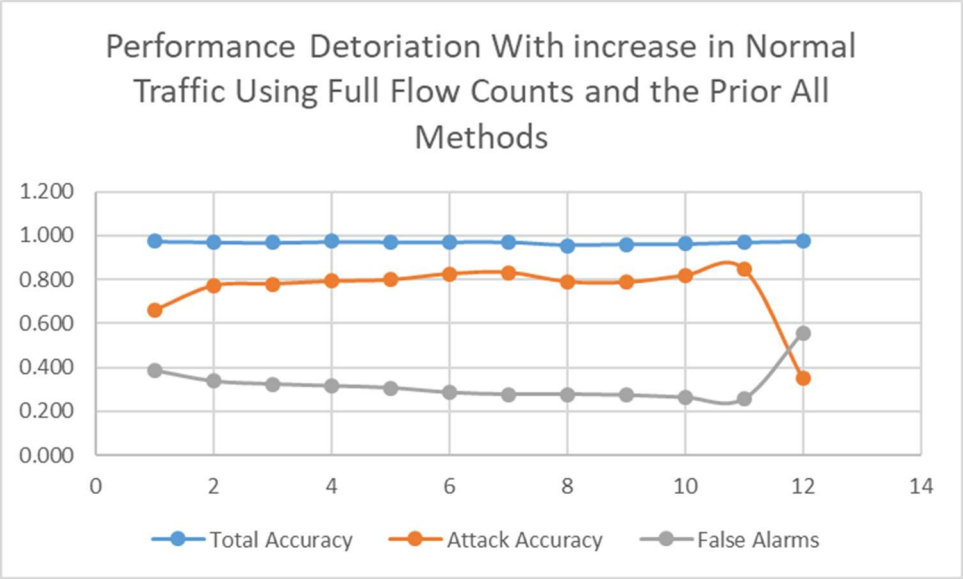

Figure 29 Performance deterioration with increase in normal traffic using full flow counts and the prior all methods

By using the F1 score and full flows we managed to find highly stable results even in the presence of increase traffic (Figure 30).

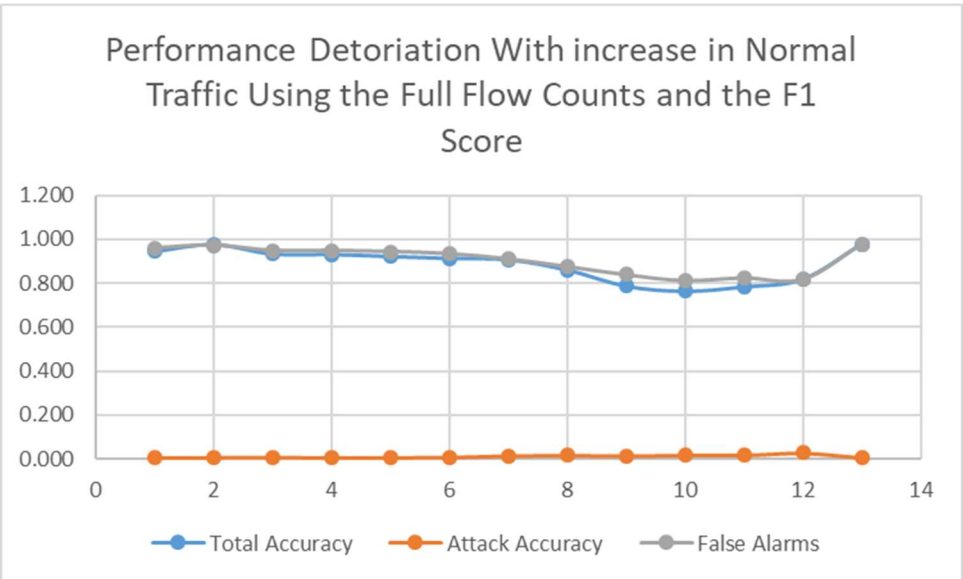

Figure 30 Performance deterioration with increase in normal traffic using the full flow counts and the f1 score

6.6.3 Comparison with Other Papers

As different papers have used different metrics for their results, there needs to be a way to display them using the same metric. In the equations below, I have assumed that TP, TN, FP, FN are ratios:

Equation 18

$$Total\ Accuracy = TP + TN$$

Equation 19

$$False\ alarm\ rate = FP$$

Equation 20

$$Attack\ Accuracy\ = \frac{TP}{TP + FN}$$

Equation 21

$$Precision = \frac{TP}{TP + FP} \Rightarrow TP \times Precision + FP \times Precision = TP \Rightarrow FP$$

$$= \frac{TP(1 - Precision)}{Precision}$$

Equation 22

$$Recall = \frac{TP}{TP + FN} \Rightarrow TP \times Recall + FN \times Recall = TP \Rightarrow FN = \frac{TP(1 - Recall)}{Recall}$$

Equation 23

$$Total\ Accuracy = TP + TN \Rightarrow TN = Total\ Accuracy - TP$$

For simplicity we will denote:

$Total\ Accuracy : A$

$Precision: P$

$Recall: R$

Equation 24

$$TP + TN + FN + FP = 1 \Rightarrow TP + TP + A - TP + \frac{TP(1-R)}{R} + \frac{TP(1-P)}{P} = 1$$

$$\Rightarrow TP\left(1 - 1 + \frac{1}{R} - 1 + \frac{1}{P} - 1\right) = 1 - A \Rightarrow \boxed{TP = \frac{1-A}{\frac{1}{R} + \frac{1}{P} - 2}}$$

Using the equations above all results will be converted into the same basis (Table 13).

Table 13 Comparison with other work

Paper	Total Accuracy	Attack Accuracy	False Alarms
This Research	0.982	0.989	0.026
[19]	0.76	0.75	0.09
[20]	0.96	0.98	0.064
[21]	0.957	0.39	0.007

7 Semi-Supervised Model of Network Under Attack (Part 3)

An perfect model would be a fully unsupervised model that only requires the regular network traffic to train on and would be able to detect attacks without being trained on them. Although we don't look into fully unsupervised models in this research, however in this part of the research we test semi supervised models on two different datasets. The datasets we tested our model on were the ISCX2012 and ISCX2014 dataset.

7.1 Dataset

Two different datasets were used in this part of the research:

- The ISCX2012 dataset
- The ISCX2014 dataset

7.1.1 ISCX2012 Dataset

The details of this dataset have been provided in chapter 6.

7.1.2 ISCX2014 Dataset

The ISCX2014 dataset consists of a series of botnet attacks. The following botnets have been used in the dataset: following botnets have been used in the dataset:

- Neris
- Rbot
- Virut
- NSIS
- SMTP Spam

- Zeus
- Zeus Control
- UDP Storm
- Tbot
- Zero Access
- Weasel
- Smoke Bot
- ISCX IRC Bot
- Menti
- Sogou
- Murlo
- Blackhole
- Osx_trojan
-

Unlike the 2012 dataset where the flows have already been extracted, the 2014 dataset only provides a set of pcap files and the flows and all other information need to be extracted. The only information provided by the dataset is the list of malicious IP addresses. After the flows have been generated the flows that originate or terminate in one of the malicious IPs are tagged as attacks.

7.2 Model

The model consists of two stages, the training and the detection stage.

7.3 Training Phase

In this stage, the model is trained based of regular network traffic. The model does not require any form of labeled data sets for this training, therefore it is considered unsupervised. The outcome of this stage are a set of clusters, (or to be more precise a set of cluster centroids). The training phase proceeds as follows:

7.3.1 Step 1, Obtaining Data Points

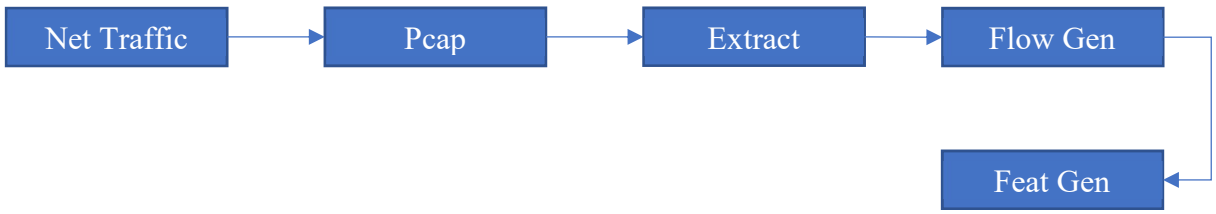

Figure 31 Obtaining the data points

As network traffic arrives they are captured and converted into Pcap files. As Pcap files are in binary format the relevant information needs to be extracted from them and imported into a database for easier access. This part was explained in Phase 2 of the Darpa dataset of chapter 6.

After the extracted features have been uploaded in the database flows need to be extracted from them. This was explained in Phase 4 of the Darpa dataset of chapter 6.

While flow data provides useful information about a single flow however it lacks the required information to capture temporal data. In order to better capture temporal data, we generate a new set of predictors from the flow information.

In order to generate these new predictors, we consider intervals of T seconds. For each interval of time we end up with one vector of predictors. The intervals are overlapping considering a granularity of 1 second. For example, considering 10 second intervals, the first vector of predictors

will be for the time interval 0~10, the second vector of predictors will be for the time interval 1~11 and so on.

In our model we consider seven different intervals:

- 1 second intervals
- 5 second intervals
- 10 second intervals
- 20 second intervals
- 30 second intervals
- 60 second intervals
- 120 second intervals

The predictors generated in each interval are the same as those explained in chapter 6.

7.3.2 Step 2, Clustering

In step 1 for each interval a vector is obtained. The elements of this vector have been discussed above. Figure 32 shows the vector along with where each predictor is coming from. The first N columns are the values associated with the protocol. The next N columns are the values associated with the applications. The next 2 columns are the number of unique local and remote nodes. The next 4 columns are the values associated with direction. The last 2 columns are for the packet and byte count.

Figure 32 Data points

Protocol					App Type					Nodes		Direction				Packet Count	Byte Count

Using the kmeans++ algorithm the vectors are clustered. After obtaining these clusters we can proceed to the detection phase.

7.4 Detection Phase

Once we have trained our model we can use it to detect attacks. The detection is performed as follows.

7.4.1 Step 1, Obtaining Data Points

This step is similar to the training phase with one difference that for the actual use case it would be performed in an online manner. As network traffic is logged, the predictors are immediately extracted.

7.4.2 Step 2, Cluster Assignment

As each data point is obtained it is assigned to one of the clusters obtained in the training phase. Each data point is assigned to the cluster for which it has the smallest Euclidean distance with the clusters centroid.

7.4.3 Step 3, Attack Detection

As each data point is generated, and assigned to a cluster it is put to two tests. If either one of these tests pass the point is considered to be an anomaly.

7.4.3.1 Test 1

In order to perform the test two parameters, need to be determined first:

σ: Each of the clusters obtained in the training phase has a size (the number of points in the cluster). This is the variance in size between the different clusters.

c: This is a tuning parameter that needs to be determined.

If a point falls in a cluster with less than σc points then it is considered an anomaly. The reason for considering this test is to consider the possibility of anomalies existing in the original training data set. Should this happen assuming that the number of anomalies is low, this tests leaves room for those clusters generated by the anomalies to be recognized as anomalous.

7.4.3.2 Test 2

In order to perform the test two parameters, need to be determined first:

σ': Each point in a cluster has a distance from the centroid. This is the variance of all such values.

c': This is a tuning parameter that needs to be determined.

If a point falls a distance greater than $\sigma' c'$ from the centroid of the cluster, it will be considered an anomaly.

7.5 Implementation

The model was evaluated on both the ISCX2012 and ISCX2014 datasets. In each case the following steps where performed:

1- The data points were generated.

2- The normal and attack data points where separated.

3- 10,000 random data normal and attack data points where selected.

4- Both the clean and attack data points where split into k equal and random.

5- A K fold cross validation is performed. K-1 of the data sets are used for training and one of the datasets is used for validation. For training only, the clean data points are considered. For testing both attack and clean data points are considered.

6- An average is obtained of the K trial

The above is run once for each of the configurations below:

1- The threshold value Is changed between 2 to 14 with 0.25 increments.

2- The clusters count is changed from 1 to 1000 with 10 increments.

3- The interval values, 1, 5, 10, 15, 20, 30, 60 and 120 seconds are considered.

In total over 153,000 simulations were run for each of the datasets.

The results of the best models were taken and applied to the test set.

7.6 Results

Before explaining how the model is evaluated two things need to be defined:

Attack Flow: An attack is defined as series of flows originating from one or more malicious nodes during adjacent time intervals. All such flows are considered a single attack.

Normal Flow: All other flows are considered normal.

Two different metrics are considered for the evaluation of the models, Attack accuracy and False positives. Attack accuracy is defined as:

$$Attack\ Accuracy = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

A good model generally performs best in both metrics. However, one generally comes at the expense of the other. To compare the different models, we set a threshold for the false alarm rate and find the best attack accuracy. The results have been plotted below (Figure 33):

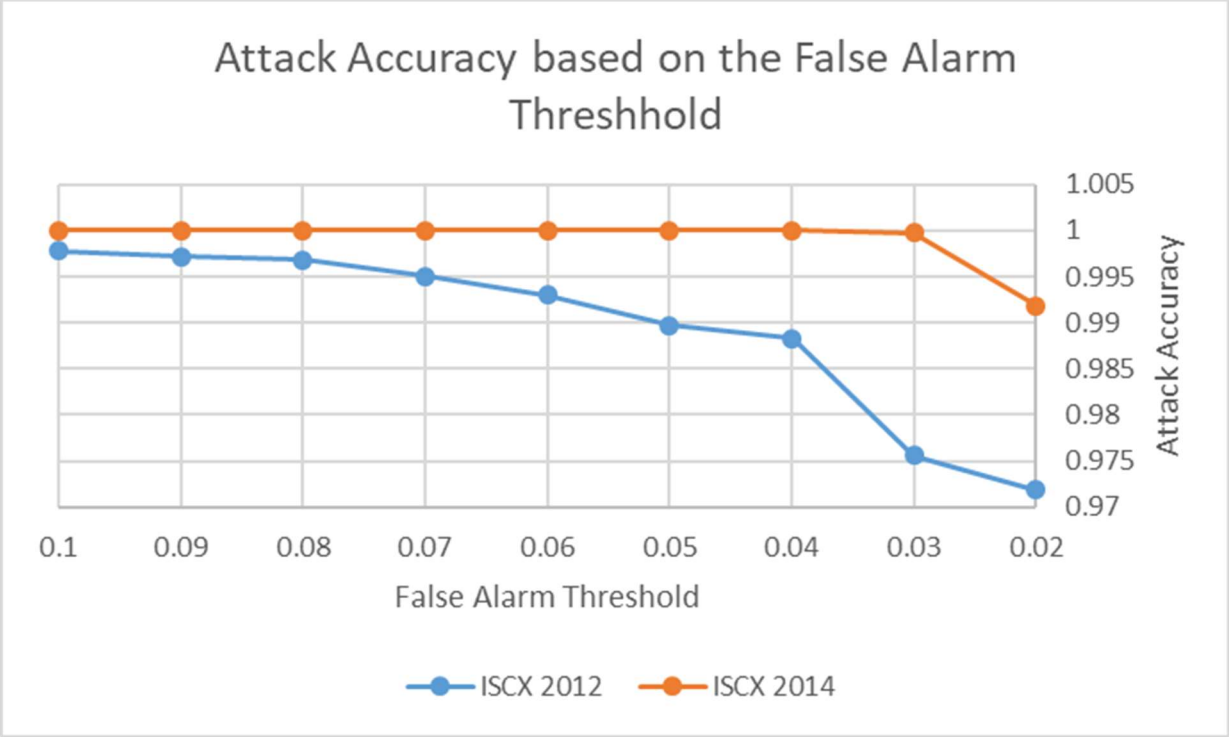

Figure 33 Results

The results have also been brought in table format below (Table 14, Table 15):

Table 14 ISCX2014 top models based on threshold

ISCX 2014 Dataset					
False Alarms Threshold (%)	**Sigma**	**clusters**	**Interval**	**Attack Accuracy**	**False Alarms**
0.1	7.25	581	120	1.000	0.037
0.09	7.25	581	120	1.000	0.037
0.08	7.25	581	120	1.000	0.037
0.07	7.25	581	120	1.000	0.037
0.06	7.25	581	120	1.000	0.037
0.05	7.25	581	120	1.000	0.037
0.04	7.25	581	120	1.000	0.037
0.03	7.75	761	120	1.000	0.029
0.02	10	621	120	0.992	0.020

Table 15 ISCX2012 top models based on threshold

ISCX 2012 Dataset					
False Alarms Threshold (%)	**Sigma**	**clusters**	**Interval**	**Attack Accuracy**	**False Alarms**
0.1	10.75	781	120	0.998	0.099
0.09	12	661	120	0.997	0.090
0.08	13.25	681	120	0.997	0.080
0.07	12.5	791	60	0.995	0.070
0.06	10.5	611	20	0.993	0.060
0.05	12	651	20	0.990	0.049
0.04	14	721	20	0.988	0.040
0.03	10	781	5	0.976	0.030
0.02	12.75	651	5	0.972	0.020

8 Conclusion and Future Work

In chapter 5 we used a statistical model in detecting network attacks. There were a number of issues with this model that motivated the research towards machine learning models.

- In the model we are specifically looking at connection counts. The issue with such an approach is that we are not taking any of the other predictors into account. There might be other features that could also help in detecting attacks.
- The model was evaluated on a relatively old dataset. The network traffic was much less complex than today's traffic. Due to the simplicity in the network traffic there were very few states which made the model work well. Had we used a dataset with more recent network traffic there would have been a much larger variety in states. Such a variety in states would results in a large number of transitions not being observed in the initial training phase and hence would results in false alarms.
- The model only takes into account a single node. Some attacks are not visible by just observing a single attack and would require looking at the entire network.

In chapter 6 we used machine learning models in detecting attacks. Our model performed well both using the old KDDCup99 Dataset and using the newer ISCX2012 dataset. Our model also proved effective under surge conditions of up to 1000% more traffic.

In chapter 7 we study a semi-supervised machine learning model. The model was tested on both the ISCX2012 and ISCX2014 dataset. What we learned was that there is a major trade off between the false alarm rate and the attack accuracy. Depending on how much false alarm rate we can tolerate we will get more or less accurate results

There is a lot of room for future research in this field:

- Future research can study the impact of malicious data being in the training set used for the semi supervised model and to see if the model can clearly separate those data points
- Research can also be done to see the impact of surges on the semi-supervised model.
- There is still a lot of room left for finding a fully unsupervised model that can detect unseen attacks in the network.

Appendix A, Interpreting the Results of the Semi-Supervised Model

In order to better understand how the model is working I have plotted the values of some of the clusters. In each of the 13 selected clusters I have extracted the data points that belong to it and plotted the results for the data points below

A.1 Applications

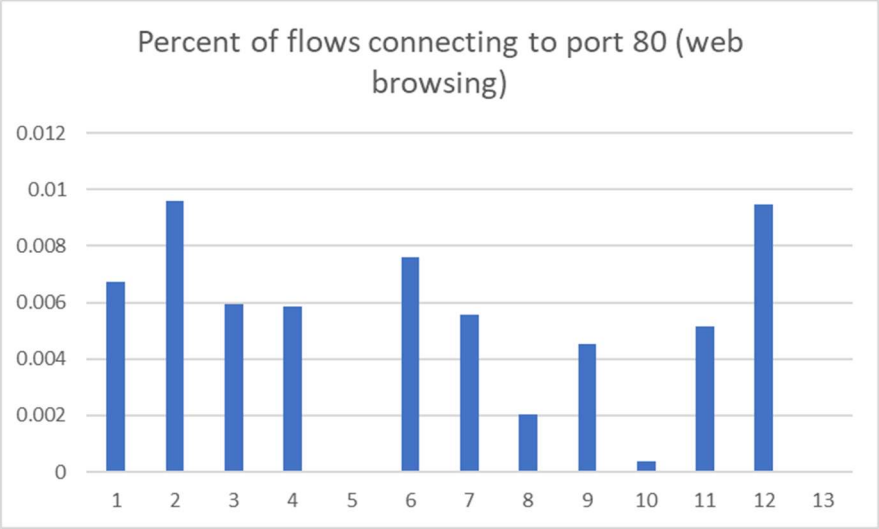

Figure A1 Percent of flows connecting to port 80

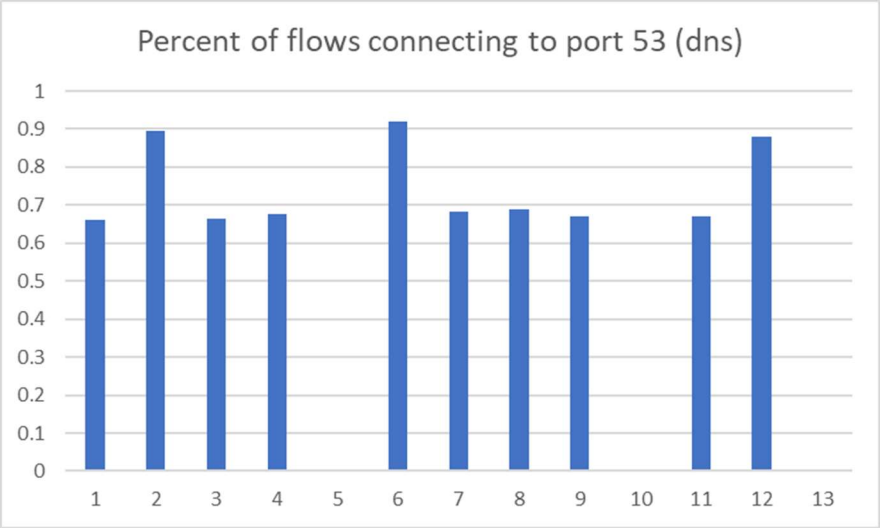

Figure A2 Percent of flows connecting to port 80

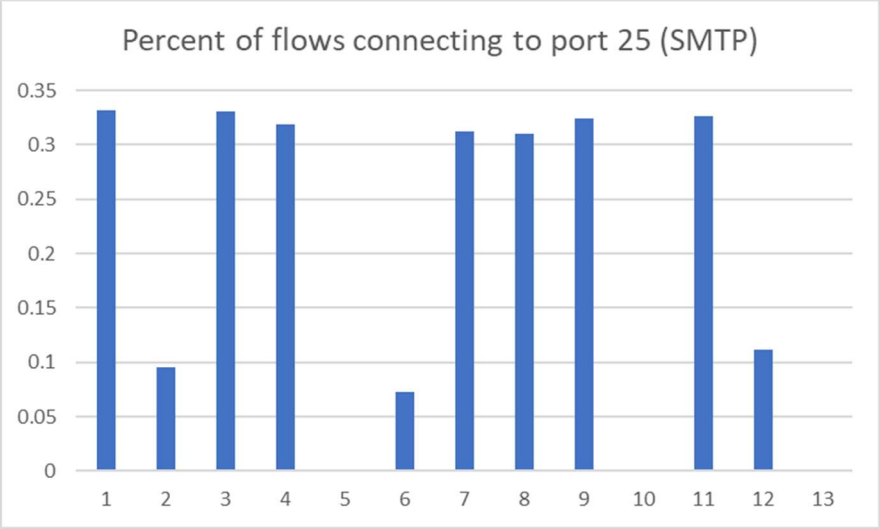

Figure A3 Percent of flows connecting to port 25

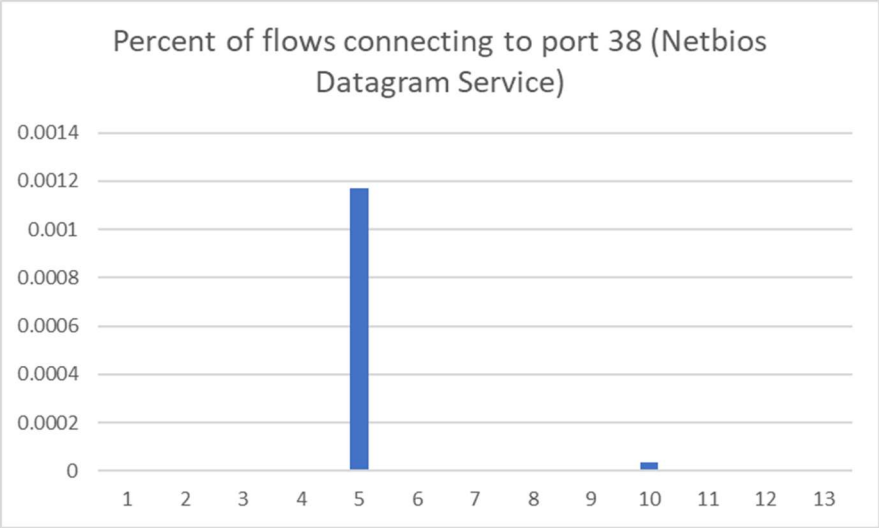

Figure A4 Percent of flows connecting to port 38

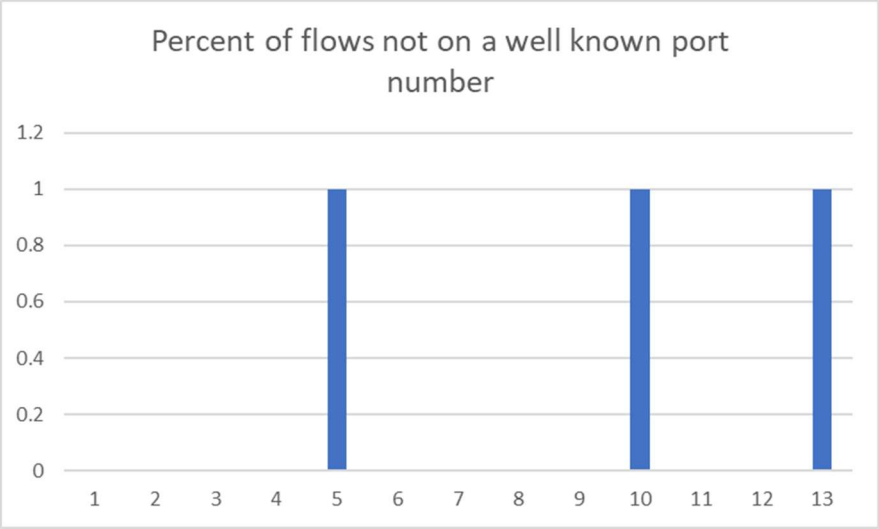

Figure A5 Percent of flow not on a well known port number

A.2 Protocols

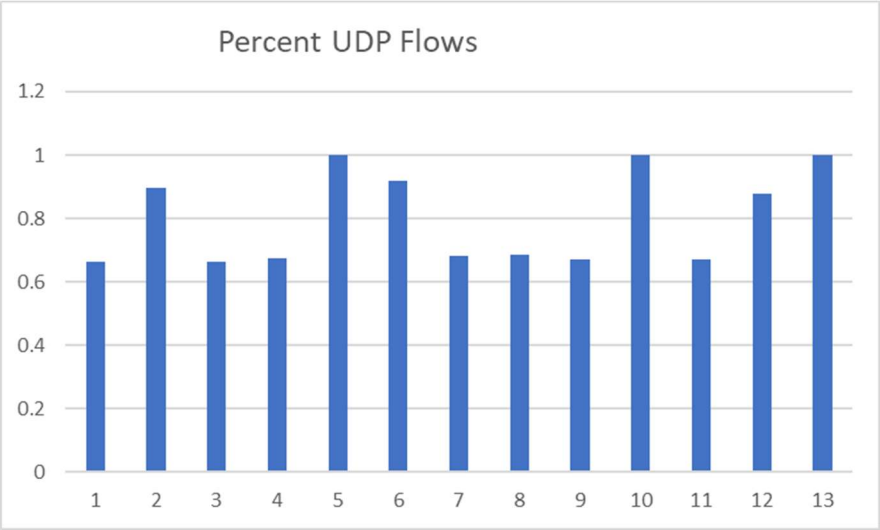

Figure A634 Percent UDP flows

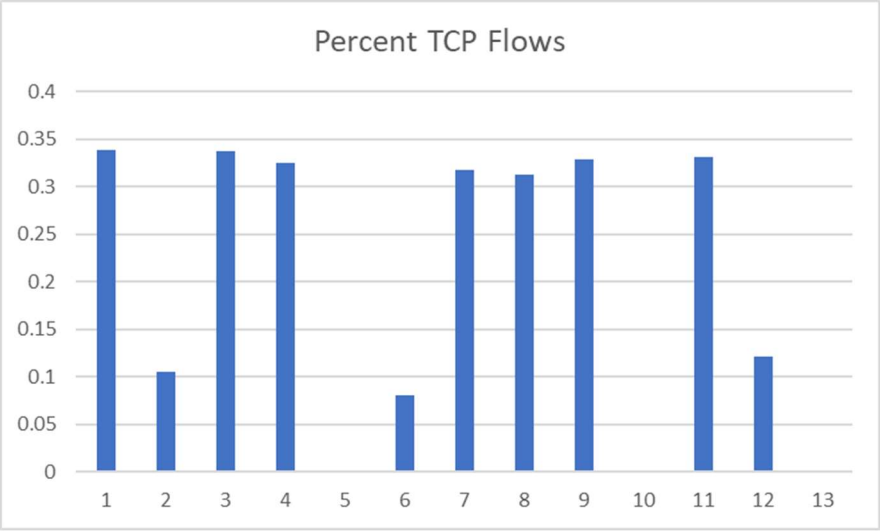

Figure A7 Percent TCP Flows

A.3 Local Vs Remote Nodes

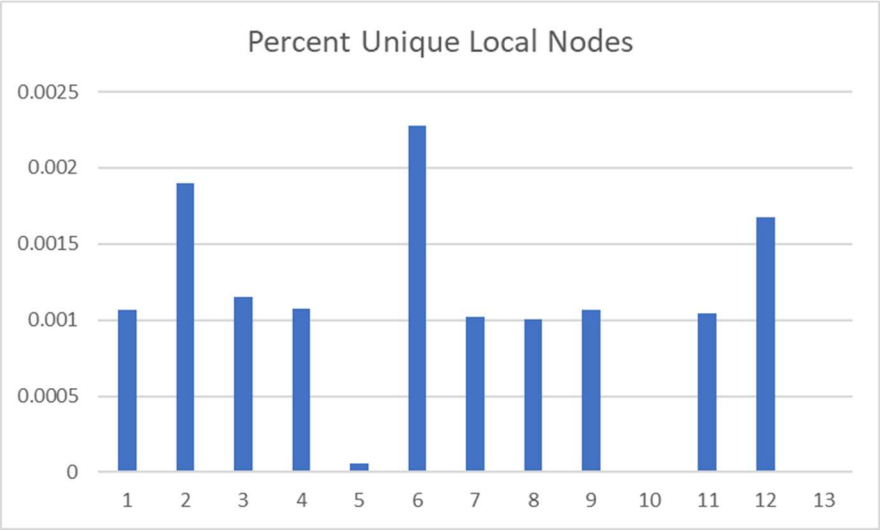

Figure A8 Percent Unique Local Nodes

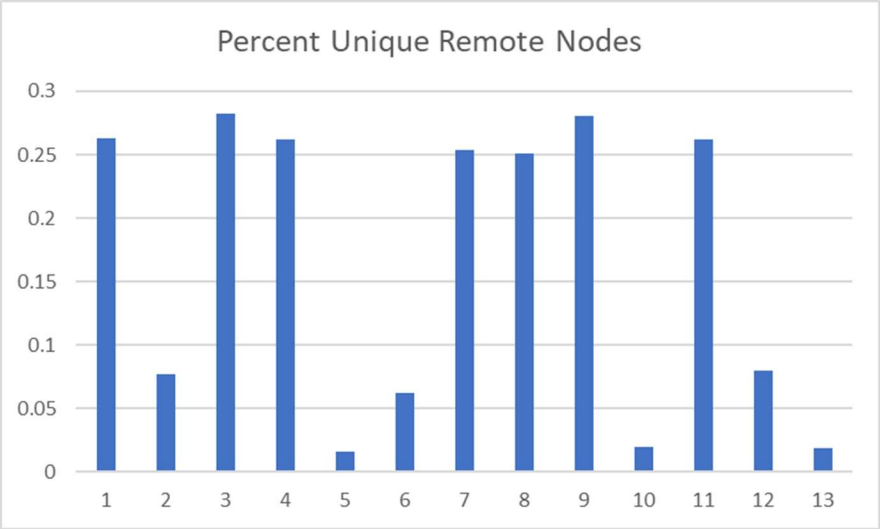

Figure A9 Percent Unique Remote Nodes

A.4 Direction

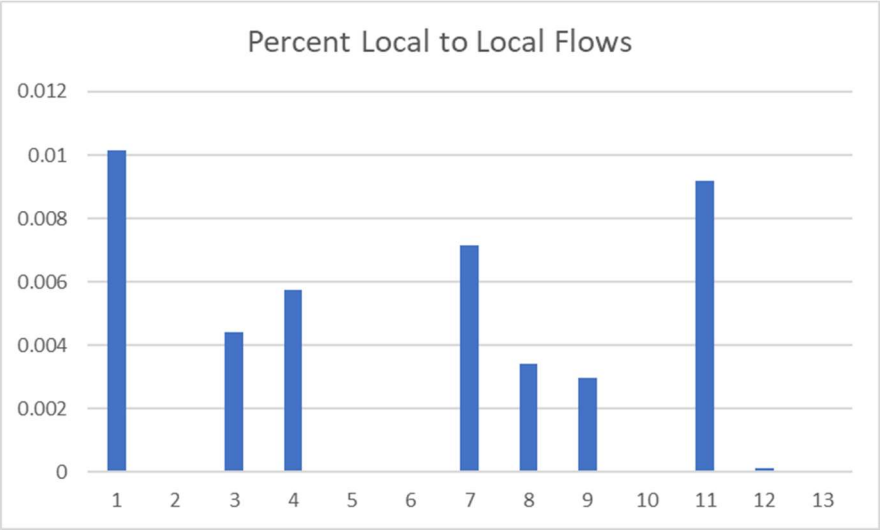

Figure A10 Percent local to local flows

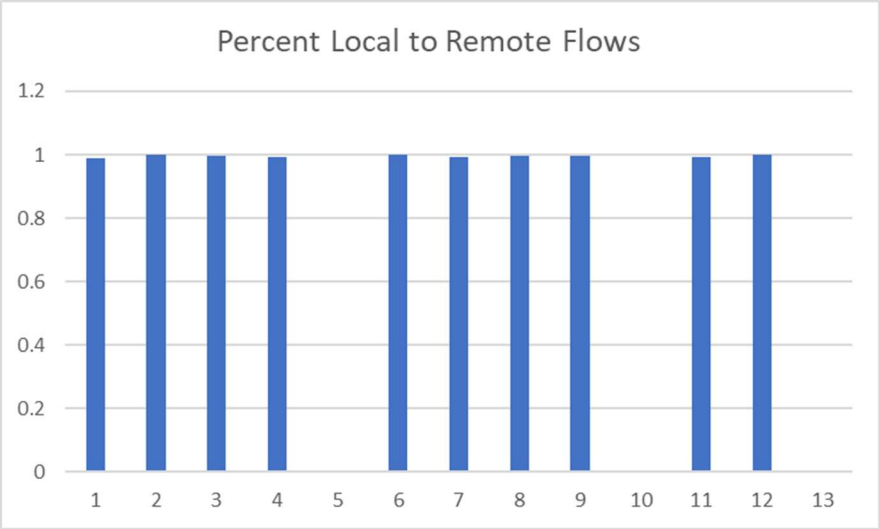

Figure A11 Percent local to remote flows

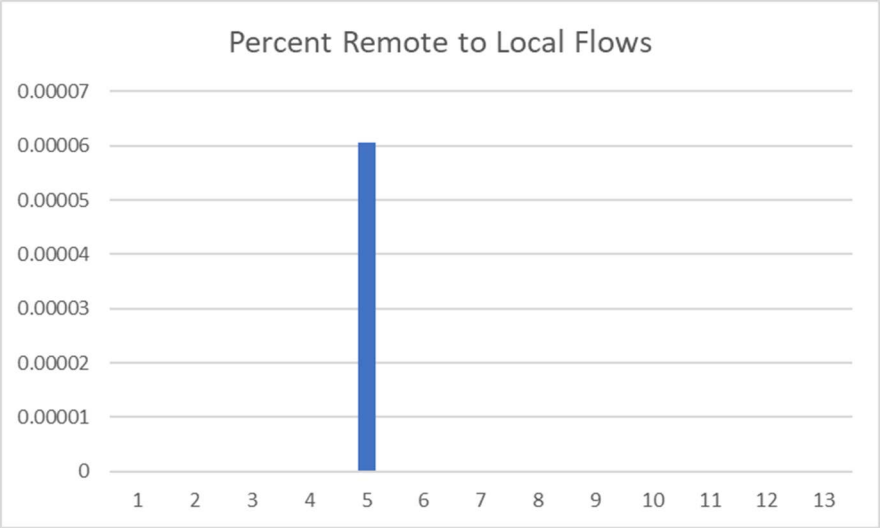

Figure A12 Percent remote to local flows

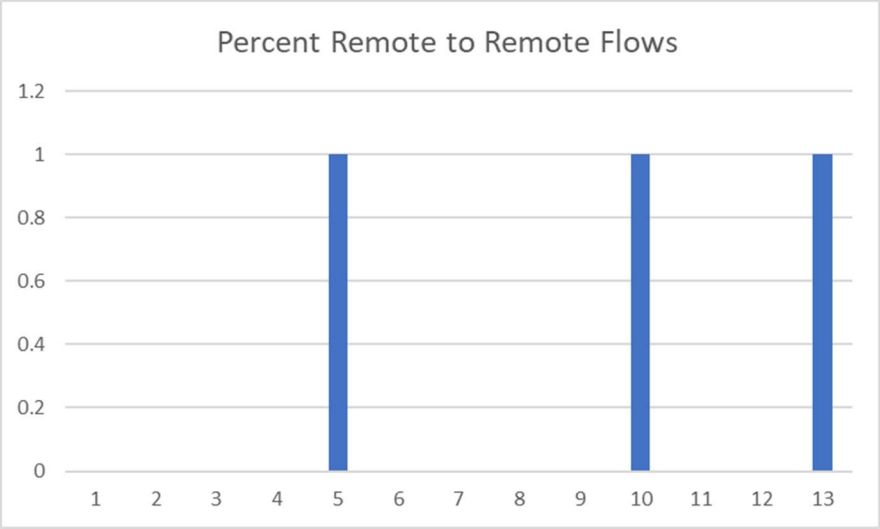

Figure A13 Percent remote to remote flows

A.5 Other

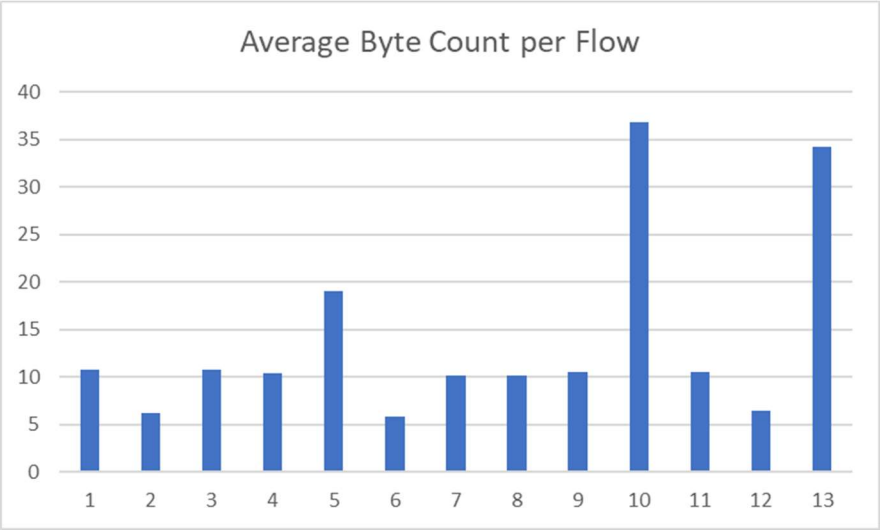

Figure A14 Average Byte Count per Flow

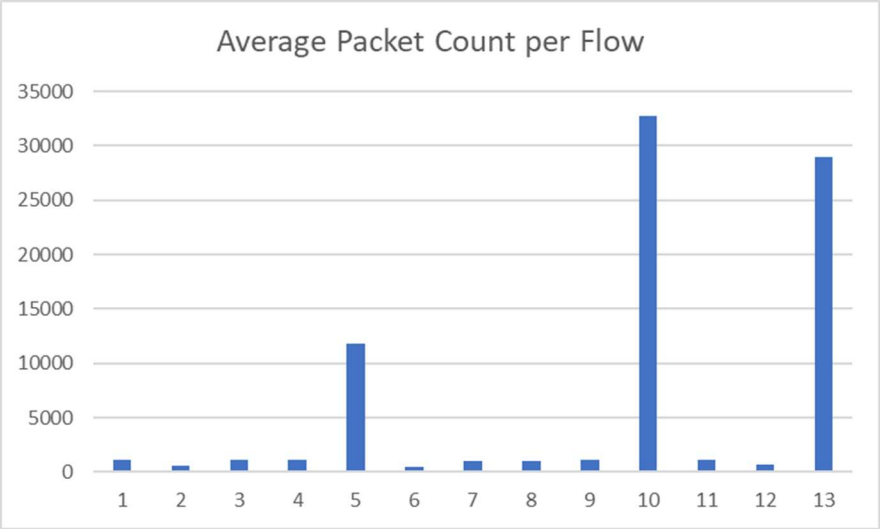

Figure A15 Average packet count per flow

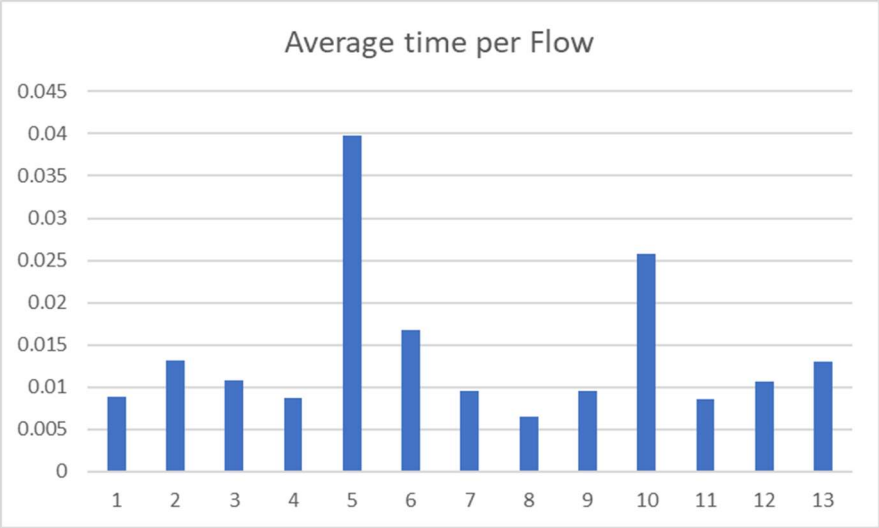

Figure A16 Average time per flow

References

[1] M. I. o. Technology, "Lincoln Labratory," 2000. [Online]. Available: https://www.ll.mit.edu/ideval/data/. [Accessed 30 11 2017].

[2] U. o. N. Brunswick, "University of New Brunswick," 2013. [Online]. Available: http://www.unb.ca/cic/research/datasets/ids.html. [Accessed 30 11 2017].

[3] J. Ericson, Hacking: The Art of Exploitation, No Starch Press, 2008.

[4] Carnegie Mellon University, "Software Engineering Institute," 24 September 1997. [Online]. Available: http://www.cert.org/historical/advisories/CA-1996-01.cfm. [Accessed 30 11 2017].

[5] H. S.H.C, A. R.B and G. M.A.H.A, "Detecting TCP SYN Flood Attack based on Anomaly Detection," in *Second International Conference on Network Applications, Protocols and Services (NETAPPS 2010)*, Kedah, Malaysia, 2010.

[6] G. B.B, J. R.C and M. Manoj, "Defending against Distrubuted Denial of Service Attacks," *Information Security Journal: A Global Perspective,* pp. 224-247, 2009.

[7] G. James, D. Witten, H. Trevor and R. Tibshirani, An Introduction to Statistical Learning with Applications in R, Springer, 2013.

[8] A. David and V. Sergei, "k-means++: the advantages of careful seeding," in *SODA '07 Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, New Orleans, Louisiana, 2007.

[9] P. Garcia-Teodoro, J. Diaz-Verdjo, G. Macia-Fernandez and E. Vazquez, "Anomaly-based network intrusion detection: techniques, systems and challenges," *Journal of Computers & Security,* 2008.

[10] J. Jyothsna, V. V Rama Prasad and K. Munivara Prasad, "A Review of Anomaly Based Intrusion Detection Systems," *International Journal of Computer Applications,* 2011.

[11] C.-W. Hsu, C.-C. Chang and C.-J. Lin, "A Practical Guide to Support Vector Classification," 2016.

[12] T. Zhang, R. Ramakrishnan and M. Livny, "BIRCH: An Effecient Data Clustering Method for Very Large Databases," in *Proceeding SIGMOD '96 Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, 1996.

[13] A. P. Muniyandi, R. Rajeswari and R. Rajaram, "Network Anomaly Detection by Cascading K-Means Clustering and C4.5 Decision Tree Algorithm," *Journal of Procedia Engineering,* 2012.

[14] B. Hssina, A. Merbouha, H. Ezzikouri and M. Erritali, "A comparative study of decisino tree ID3 and C4.5," *Internation Journal of Advanced Cmoputer Science Applications,* 2014.

[15] E. Eskin, A. Arnold, M. Prerau, L. Portnoy and S. Stolfo, "A geometric framework for unsupervised anomaly detection: detecting intrusions in unlabeled data," Columbia University, New York, 2002 .

[16] G. Sanjay, N. Harsha and C. Alok, "MAFIA: Effecient and Scalable Subspace Clustering for Very Large Data Sets," Center for Parallel and Distributed Computing, Evanston, 1999.

[17] Canadian Institute for Cybersecurity, "Intrusion detection evaluation dataset," Brunswick, University of New, 2013. [Online]. Available: http://www.unb.ca/cic/research/datasets/ids.html. [Accessed 30 11 2017].

[18] A. Shiravi, H. Shiravi, M. Tavallaee and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers and Security,* vol. 31, no. 3, pp. 357-374, 2012.

[19] T. A. Tang, L. Mhamdi and D. McLernon, "Deep learning approach for Network Intrusion Detection in Software Defined Networking," in *International Conference on Wireless Networks and Mobile Communications (WINCOM)*, Fez, Morocco, 2016.

[20] A. Sultana and M. Jabbar, "Intelligent network intrusion detection system using data mining techniques," in *2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, Bangalore, India, 2016.

[21] Shi-JinnHorng, Ming-YangSu and Yuan-HsinChen, "A novel intrusion detection system based on hierarchical clustering and support vector machines," *Expert Systems with Applications an International Journal,* vol. 38, no. 1, pp. 306-313, 2011.

[22] W. Yassin, N. I. Udzir, Z. Muda and M. N. Sulaiman, "Anomaly based intrusion detection through k-means clustering and naives bayes classification," *Proceedings of the 4th international conference on Computing and informatics,* 2013.

[23] Y.-D. Lin, P.-C. Lin, S.-H. Wang and Y.-C. Lai, "PCAPLib: A system of extracting, classifying and anonmyizing real packet traces," *IEEE systems journal,* 2014.

[24] N. Ye, "A markov chain model of temporal behavior for anomaly detectoin," *proceedings of the 2000 ieee,* 2000.

[25] D. Ourston, S. Matzner, W. Stump and B. Hopkins, "Applicatoins of Hidden Markov Models to Detecting Multi-Stage Network Attacks," *Proceedings of the 36th Hawaii International Conference on System Sciences,* 2003.

[26] K. Leung and C. Leckie, "Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters," in *Proceeding ACSC '05 Proceedings of the Twenty-eighth Australasian conference on Computer Science - Volume 38*, 2005.

[27] B. Monowar, D. Bhattacharyya and J. Kalita, "An Effective Unsupervised Network Anomaly Detection Method," in *ICACCI '12 Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, Chennai, India, 2012.

[28] L. Portnoy, E. Eskin and S. Stolfo, "Intrusion detection with unlabeled data using clustering," in *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001*, 2001.

[29] W.-H. Chen, S.-H. Hsu and H.-P. Shen, "Application of SVM and ANN for intrusion detection," *Journal of Computers and operations research,* 2005.

[30] G. Conti, B. Sergi, A. Shubina, R. Ragsdale, M. Supan, A. Lichtenber and P.-A. Robert, "Automated mapping of large binary objects using primitive fragment type classification," *Journal of Digital Investigation,* 2010.

[31] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals," *Elsevier Journal of Computers and Security,* 2013.

[32] S.-B. Cho and H.-J. Park, "Efficient anomaly detection by modeling privilege flors using hidden markov model," *Elesevier Journal of Computer and Security,* 2003.

[33] Y. Bouzida, F. Cuppens, N. Cuppens-Boulahia and S. Gombault, "Efficient Intrusion Detection Using Principal Component Analysis," Departement RSM GET/ENST Bretagne, Cedex, France, 2010.

[34] D. Brauckhoff, T. Bernhard, A. Wagner, M. May and L. Anukool, "Impact of packet sampling on anomaly detection metrics," in *Proceeding IMC '06 Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, 2006.

[35] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Journal of Expert Systems with Applications,* 2009.

[36] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications,* 2012.

[37] J. Mai, C.-N. Chuah, A. Sridharan, T. Ye and H. Zang, "Is Sampled Data Sufficient for Anomaly Detection," in *Proceeding IMC '06 Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, 2006.

[38] G. Prasanta, B. Bhogeswar and B. Dhruba, "Network Anomaly Detection Using Unsupervised Model," *2013 International Conference on Network Security and Cryptography,* no. 1, pp. 19-30, 2011.

[39] I.-C. Hsieh, L.-P. Tung and B.-S. Paul Lin, "On the classification of mobile broadband applications," in *2016 IEEE 21st International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD)*, Hsinchu, Taiwan, 2016.

[40] M. Dell Amico, M. Filippone, P. Michiardi and Y. Roudier, "On user Availability Prediction and Network Applications," *IEEE/ACM Transactions on Networking,* 2015.

[41] N. Duffield, C. Lund and M. Thorup, "Properties and prediction of flow statistics from sampled packet streams," At&T Labs - Research, NJ, USA, 2002.

[42] T. T. Nguyen, G. Armitage, P. Branch and S. Zander, "Timely and Continuous Machine-Learning-Based Classification for Intervative IP Traffic," *IEEE/ACM Transactions on Networking,* 2012.

[43] A. Shiravi, H. Shiravi, M. Tavallaee and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Elsevier Journal of Computers and Security,* 2012.

[44] B. Sangster, T. O'Connor, T. Cook, R. Fanelli, E. Dean, W. Adams and C. Morrel, "Towards Instrumenting Network Warfare Competitiions to Generate Labeled Datasets," United States Military Academy, West Point, New York, 2009.

[45] X. Sha, D. Quercia and M. Dell'Amico, "Trend Makers and Trend Spotters in a Mobile Application," *Proceeding CSCW '13 Proceedings of the 2013 conference on Computer supported cooperative work,* 2013.

[46] P. Casas, J. Mazel and P. Owezarski, "Unsupervised network Intrusion Detection Systems: Detecting the Unknown without knowledge," *Journal of Computer Communications,* vol. 35, no. 7, p. 772–783, 2011.