2018-07-30

Distributed Denial of Service Attack Detection Using a Machine Learning Approach

Gupta, Animesh

UNIVERSITY OF CALGARY

Distributed Denial of Service Attack Detection Using a Machine Learning Approach

by

Animesh Gupta

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA

AUGUST, 2018

Abstract

A distributed denial of service (DDoS) attack is a type of cyber-attack in which the perpetrator aims to deny the services on a network/server by inundating the traffic on the network/server by superfluous requests which renders it incapable to serve requests from legitimate users. According to Corero Network Security (A DDoS protection and mitigation provider), in Q3 2017, organizations around the world experienced an average of 237 DDoS attack attempts per month, which averages to 8 DDoS attacks every day. This was a 35% increase over Q2 that year and a staggering 91% increase over Q1. According to another research by Incapsula, a DDoS attack costs an average of $40,000 per hour to businesses. There are commercially available software which detect and mitigate a DDoS attack, but the high cost of these software makes them hard to afford for small and mid-scale businesses. The proposed work aims to fill this gap by providing real time open-source robust web application for DDoS attack prediction which can be used by small to mid-scale industries to keep their networks and servers secure from malicious DDoS attacks.

A Machine Learning approach is used to employ a window-based technique to predict a DDoS attack in a network with a maximum accuracy of 99.83%, if the recommended combination of feature selection and classification algorithm is chosen. The choice of both feature selection and classification algorithm is left to the user. One of the feature selection algorithms is the novel Weighted Ranked Feature Selection(WRFS) algorithm which performs better than other baseline approaches in terms of accuracy of detection and the overhead to build the model. Once the selection is made, the web application connects to the socket and starts capturing and classifying real-time network traffic. After the capture is stopped, information about attack instances (if any), number of attack packets, confusion matrix is rendered to the client using dynamic charts. The trained model used for classifying real-time packets is optimized and uses only enough attributes from the incoming packet which are necessary to successfully predict the class of that packet with high accuracy.

Acknowledgement

I would like to express my gratitude to Dr. Reda Alhajj and Dr. Jon Rokne for their valuable feedback and unending inspiration over the past two years. I am grateful for all the constructive criticism I received from them during my briefings and seminars. Their professional guidance has been extremely helpful towards my research and life in general. I am indebted to Dr. Reda for showing confidence in me by accepting me to his research group and encouraging me to pursue a research topic of my liking. I am also very thankful to Dr. Rokne for never saying no to any of my request, however frequent they were!

I am grateful to all my lab mates who started as colleagues but soon became very good friends. I feel thankful to Manmeet for always listening patiently to my research ideas and plans without ever losing patience or interest and for the numerous trips we had together. I am also obliged to Coskun and Alper for their words of wisdom and always being there to debug my dirty code. Coskun has also been a dear friend and a constant pillar of support during my lows. I am also thankful to the other members of the lab for maintaining a healthy atmosphere conducive to both research and fun.

None of this would have been possible without my family. I would like to thank my parents for their guidance and the confidence they have instilled in me. My brother has always been an inspiration for me with his sense of discipline and maturity.

Table of Contents

List of Tables

List of Figures

List of Symbols

ACK	Acknowledgment-packet of the TCP handshake
ACM DL	The Association for Computing Machinery Digital Library
ARPANET	The ARPA (Advanced Research Projects Agency) Network
AVG	Short for average
CPU	Central processing unit
CUSUM	Cumulative sum
DARPA	The Defense Advanced Research Projects Agency
DBSCAN	Density-based spatial clustering of applications with noise
DDoS	A distributed denial-of-service
DHCP	Dynamic host configuration protocol
DMZ	Demilitarized zone, a network segment
DNS	The domain name system
DoD	The Department of Defense
DoS	A denial-of-service or a denial of service
FN	False negative
FP	False positive
FPR	False positive rate
GET	An HTTP GET-request
HIDS	A host-based intrusion detection system
HTML	Hypertext mark-up language
HTTP(S)	Hypertext transfer protocol, HTTPS over

ICMP	The internet control message protocol
IDS	An intrusion detection system
IEEE	The Institute of Electrical and Electronics Engineers
IP	Internet protocol, IPv4 and IPv6
IPS	Intrusion prevention system
IRC	Internet relay chat, an instant messaging service
ISBN	International standard book number
KDD	Knowledge discovery from data
LOIC	Low Orbit Ion Cannon
ML	Machine Learning
M.Sc.	Master of Science
NAT	Network address translation
NIDS	Network intrusion detection system
NN	Neural network
OSI	Open Standards Interconnection
PCA	Principal component analysis
PCAP	Packet capture –file
POST	An HTTP POST-request
Ph.D.	A Doctor of Philosophy
SDN	Software-defined network
SNA/IP	Systems network architecture over internet protocol
SSH	Secure shell

SSL	Secure sockets layer
SVM	Support vector machines
SYN	Synchronize-packet of the TCP handshake
SYN-ACK	Synchronize-acknowledgment-packet of the TCP handshake
TCP	Transmission control protocol
TCP/IP	Transmission control protocol over internet protocol
TFN2K	The Tribe Flood Network
TN	True negative
TP	True positive
TPR	True positive rate
TTL	Time to live
UDP	The user datagram protocol
VPN	Virtual private network
WRFS	Weighted Ranked Feature Selection

Section 1: Introduction

1.1 Background and Problem Definition

The first known Denial of Service (DoS) attack goes back to the year 1974 courtesy of a 13-year old high school student who had recently learnt about a command which could be run on CERL'S PLATO terminal. PLATO was first of its kind computerized shared learning system. The command 'ext' short for external was used to communicate with external devices but if a system was not connected to an external device, then the command 'ext' would force the system to shut down. David learnt of this flaw and sent the 'ext' command to systems at CERL causing 31 users to log off simultaneously. Eventually, the acceptance of the 'ext' command from a remote system was disabled, fixing the problem.

Since then, DoS elevated to become distributed DoS or DDoS and has become infamous for the most destructive kind of cyber-attack. Because of the nature of a DDoS attack, it is very hard to mitigate as it penetrates right through the open ports on the firewall and leads to both financial loss and the loss of reputation for a company. Almost all the major technology companies have been a target of a DDoS attack at some point in their history. Due to the high impact of such attacks, it is a constant cause of concern for people responsible for cyber-security.

This research was done with the desire to create a highly efficient comprehensive DDoS detection application for small to mid-scale companies using which they can detect a DDoS attack on a network with a high accuracy.

In the past, various approaches have been used to detect a DDoS attack. Two of the most common categories of defence mechanisms were Signature based and Anomaly based approach. A Signature based DDoS detection tries to detect a DDoS attack by maintaining a database of signatures of past attacks and comparing the signature of an incoming attack with the signatures already present in the database and finally employing the defense for that attack

signature. Clearly, detecting a new kind of attack was impossible using the Signature based approach. On the other hand, Anomaly based DDoS detection techniques tried to detect a DDoS attack by setting a pre-defined threshold and comparing the attack pattern with that threshold. False Positives were just too high for an Anomaly based DDoS detection approach. The latest trend in detecting a DDoS attack is using ML techniques which are both fast and accurate in detecting a DDoS attack.

1.2 Motivation

The motivation for this work was to employ a Machine Learning technique to detect a DDoS attack in a computer network using a small subset of attributes. Since it is important to detect a DDoS attack as soon as possible, fewer number of attributes allows fast processing of network packets to classify them as either an attack or a normal packet. This proposed approach can detect a DDoS attack with 99.83% accuracy using only 5 attributes.

1.3 Overview of the proposed system

A network packet carries a lot of information such as source IP, destination IP, source bytes, payload, duration, flag, etc. The Knowledge Discover Dataset (KDD) Cup from 1999 extract 41 different categories of data for a network packet; but not all attributes are equally important to detect a DDoS attack. In this research, we use some widely-used filter-based feature selection algorithms to sort the real-time network packet features from most important to least important. A novel Weighted Ranked Feature Selection (WRFS) is then employed to create a final sorted list of features based on the weighted ranks of features given by other feature selection algorithms. The ranked features from all the subset selection algorithms including WRFS is fed incrementally to different classification algorithms most commonly employed for DDoS detection. The accuracy, precision and recall is calculated for each run. It was observed after experimentation that when the top 5 features are selected using WRFS and classified using Random Forest, the accuracy of prediction is 99.83 % with more than 99% precision and recall.

The test bed setup is done in a Virtual Private Network (VPN) environment which was setup within the University of Calgary network to withhold the attack packets from spreading into the network. Real-time network traffic containing attack and normal packet instances was captured through the socket. After that, 28 features relevant for this problem were then extracted, normalized and stored. This real-time dataset is then used to create windows of 100 packets each in real time. A sliding window mechanism is then used to classify every window as either an attack window or a normal window based on a pre-defined threshold value.

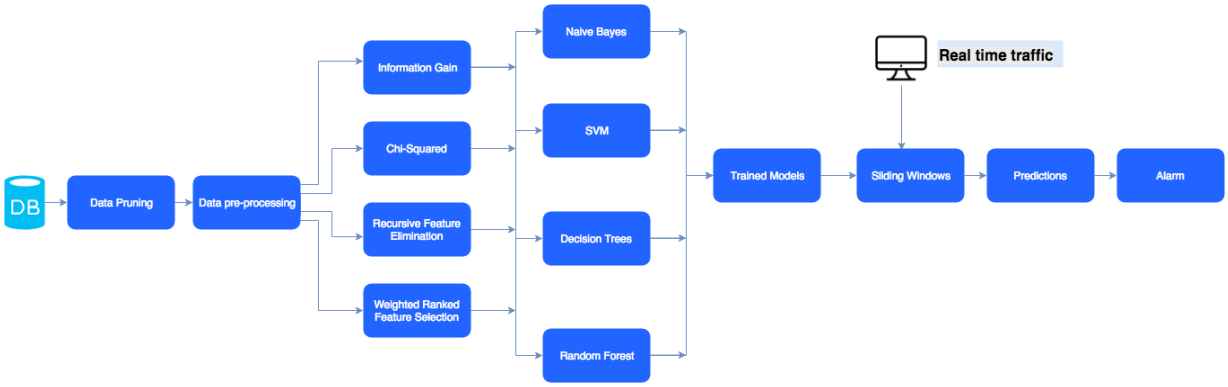

Figure 1. 1 *Overview of the proposed DDoS detection tool*

1.4 Contributions

The main contributions of this work are summarized below:

1. Explored the correlation between four different feature selection algorithms and four classification algorithms by using the sorted list of features by each feature selection method and measuring the accuracy using each classification algorithm by incrementally adding a feature to the list of features used for classification.
2. A novel feature selection method called 'Weighted Ranked Feature Selection' (WRFS) is proposed in Chapter 5. Using this feature selection method, the number of attributes required to detect a DDoS attack with high accuracy reduces by an average of 4, for different classification algorithms.
3. A sliding window-based approach is used to classify windows as attack or normal windows after real-time capture of network packets.

4. A web application is designed which allows the Start and Stop Capture functionality to the user and shows a classification summary of the captured packets using dynamic visualizations.

1.5 Organization of the thesis

This thesis is divided into seven chapters to transition the reader from understanding about DDoS attacks and introducing the proposed methodology and the results obtained. Chapter 2 on related work is divided into three topics based on the different categories of DDoS detection algorithms. Chapter 3 talks about Network Security and its importance. Chapter 4 discusses Distributed Denial of Service (DDoS) attacks and their architecture. Chapter 5 begins with the discussion of the Dataset used, followed by the environment setup and the proposed methodology. Chapter 6 summarizes the experiments, results and simulations of DDoS detection using the proposed approach. Chapter 7 outlines the conclusions and the future work.

Section 2: Related Work

Since the introduction of Machine Learning for DDoS detection, majority of the proposed algorithms can be categorized into four broad techniques – Clustering, Classification, Statistics, and Hybrid. There are several approaches which use algorithms from either one of these four classes for DDoS detection.

2.1 Clustering based techniques

In the 2014 paper 'A proactive DDoS Attack Detection Approach Using Data Mining Cluster Analysis' by Wesam Bhaya and Mehdi Manaa, a hybrid approach called centroid-based rules is proposed to detect and prevent a real-world DDoS attack using unsupervised k-means data mining clustering techniques with proactive rules method. The 'CAIDA DDoS Attack 2007 Dataset' and 'The CAIDA Anonymized Internet Traces 2008 Dataset' are used in this research. The first dataset contains normal packets with no attack instances whereas the second dataset contains packets with attack instances only. To create a more real-life scenario, one million packets are then chosen from each dataset randomly to create the final dataset which is then normalized before being used for experimentation and testing. The proposed 'Proactive DDoS Attack Detection System' is shown in Figure (2.1) below.

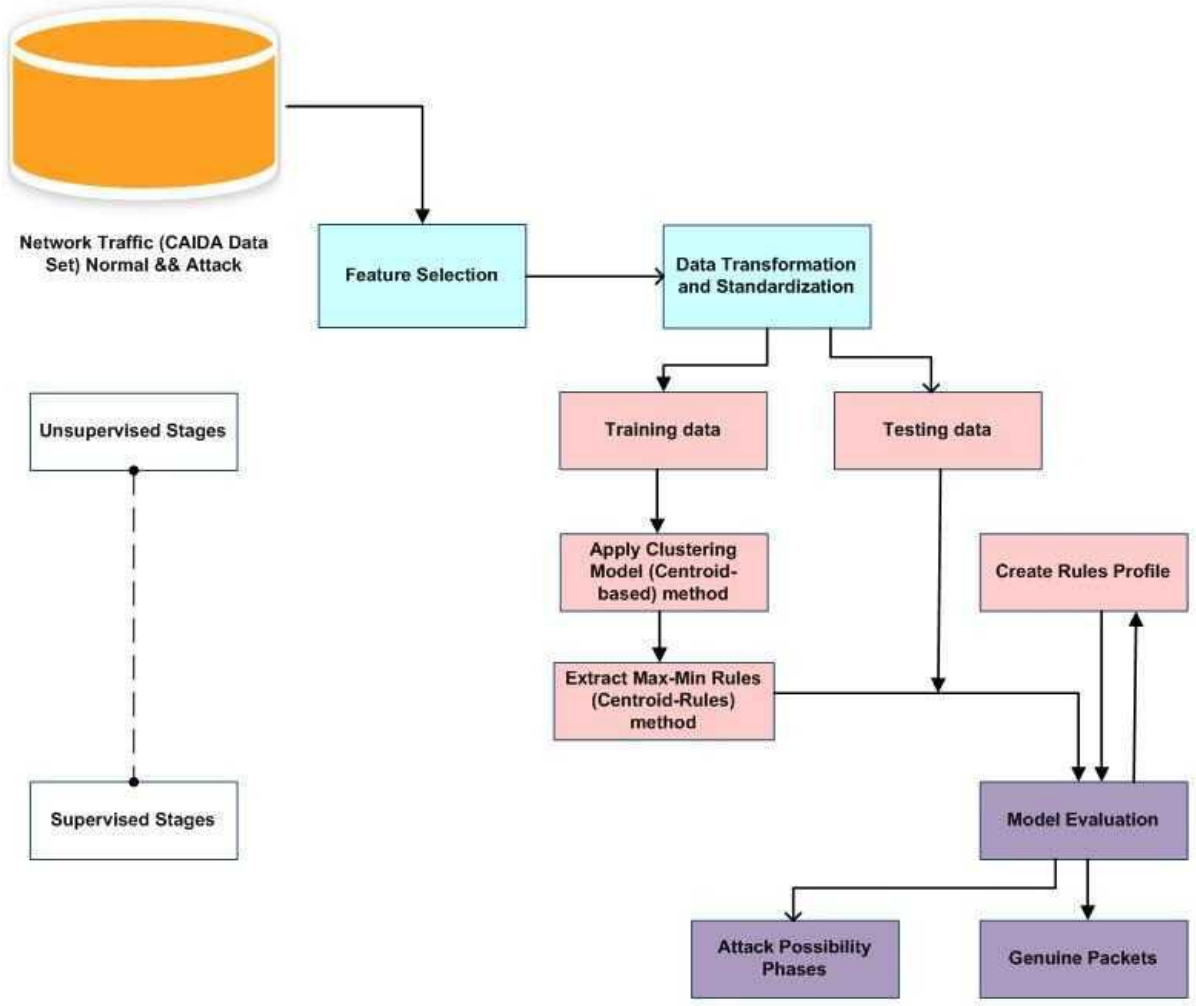

Figure 2. 1 *A proactive DDoS Detection System (Adopted from* ***[1]****)*

For the first step which is feature selection, six features are chosen from experience. These are Time, Source IP, Destination IP, Source Port, Destination Port, and Protocol. The data is then transformed and standardized using the Shannon's entropy method [2], [3] . Next, the data is divided into training and testing data using the 70-30 split. In the training phase, k-means clustering algorithm is used to form centroids. Max-Min rules are then created after extracting the max-min data points for each cluster based on the number of centroids. The noise and outlier points are handled using the shrink factor(s=0.85) which shrinks any points lying outside the range of max-min points and brings it within the range. Figure (2.2) below shows the

accuracy measures of the proposed approach compared to the baseline Centroid-based method.

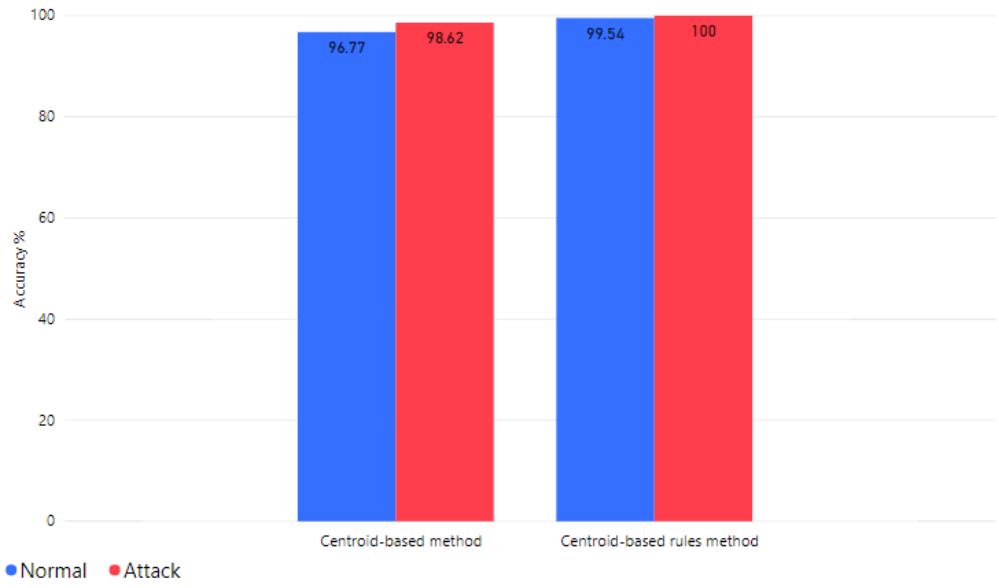

Figure 2. 2 *Accuracy comparison*

In another clustering based approach, Xi Qin et al. in their work 'DDoS Attack Detection Using Flow Entropy and Clustering Technique' propose a novel entropy based DDoS attack detection approach by constructing entropy vectors of different features from traffic flows, modelling normal packets using clustering analysis algorithms, and then detecting deviations from the created models. The proposed approach differs from other comparable approaches by dynamically setting the threshold value based on the traffic models. The dataset used is created using a traffic collection procedure. Entropy is used to construct the required features from the collected packets. The selected features are destination address, destination port, source address, packet size, and flow duration. Next, in the training phase, clustering is used for modelling normal patterns of behavior and for determining the detection threshold. K-means is chosen as the clustering algorithm. The following steps are then followed to detect a DDoS attack:

- For on-line traffic flows to be detected in a unit time, calculate the value of entropy and get entropy vector X in pre-process module.

- Calculate the distances between X and all cluster centres Ci and record the results as di. Select the smallest distance dt = min{di}, and then assign the sample X to this corresponding cluster.
- Compare dt to the radius rt. If dt ≤ rt, the sample X is judged as normal data, then we save X and update the normal model when the new normal data reaches a certain amount. Otherwise, DDoS attacks would be considered occurred.

DF-Rate which is defined as the ratio of the detection rate and the false positive rate is used as a metric to compare the results. Figure (2.3) shows the results of their approach compared with a baseline entropy-based clustering approach.

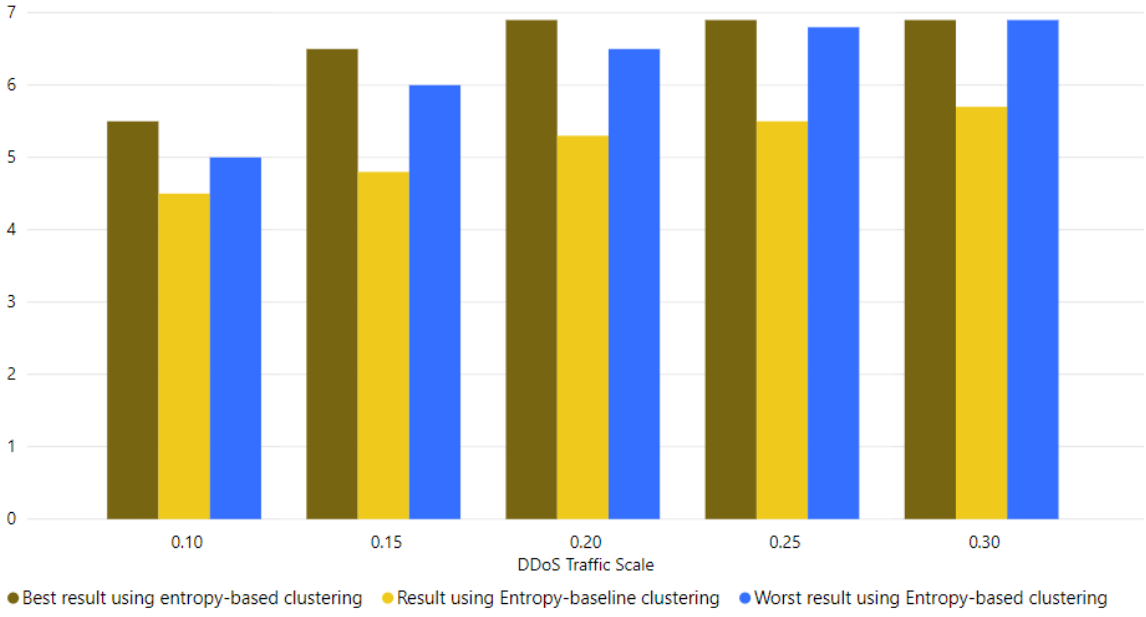

Figure 2. 3 *Comparison of proposed approach with baseline approach*

2.2 Statistics based techniques

It is possible to detect a DDoS attack by measuring the statistical fields of incoming network packets. Attributes such as source IP address, destination IP address and packet rate are generally very good measures of detecting a DDoS attack. There are a few other derived fields, the most common of them being entropy, which are also used in conjunction with independent attributes to successfully detect a DDoS attack. Ease of implementation and fast computation

of these techniques are the reasons why statistical approaches have been widely used in this field.

In the 2016 paper 'A Novel Measure for Low-rate and High-rate DDoS Attack Detection using Multivariate Data Analysis', Nazrul Hoque et al. propose a statistical approach to DDoS detection. A statistical measure called Feature Feature Score(FFSc) is introduced for multivariate data analysis to distinguish the attack traffic from legitimate traffic. If an attack is generated from a botnet, then the attack traffic has strong correlation among its samples because the bot-master uses the same attack statistics during attack generation [4]. Therefore, a correlation measure is proposed to distinguish attack packets from regular packets. On the other hand, if the attacker generates attack traffic very similar to normal network traffic, a correlation measure may not distinguish the difference between normal and attack traffic. So, multiple network traffic features are analyzed in such a way that change in an individual feature value may reflect the overall change in the network traffic sample.

CAIDA and DARPA datasets, two common datasets for DDoS research, are used for experimentation. The feature selection step extracts and calculates the entropy of source IPs, variation of source IPs and packet rate. The entropy of Source IPs is calculated using Equation (1).

$$H(X) = -\sum_{i}^{n} P(x_i) \log_2 P(x_i) \quad \textbf{(1)}$$

Here X is the random variable for Source IPs and n is the count of distinct Source IPs. The variation of source IPs is then defined as the rate of change of IP address w.r.t time in a traffic sample. Finally, packet rate is calculated as the total number of packets transmitted in 1 second. Windows are then created of packets captured in 1 second and then the extracted/calculated features are used to compute the FFSc score using equations (2-5).

$$FFoR(O_i^{f_i}) = \sum_{j=1 \& i \neq j}^{n} (|f_i - f_j|) \quad \textbf{(2)}$$

$$AFFoR(O_i) = \frac{\sum_{j=1}^{n}(FFoR(O_i^{f_j}))}{n} \quad \textbf{(3)}$$

$$Dev(O_i^{f_j}) = |AFFoR(O_i) - FFoR(O_i^{f_j})|, \forall j = 1, 2, \cdots n \quad \textbf{(4)}$$

$$FFSc(O_i) = \frac{(O_i \times Dev(O_i)^{\intercal})}{(mean(O_i) + mean(Dev(O_i)))} \quad \textbf{(5)}$$

Here, equation (2) calculates the Feature Feature ordered Relation(FFoR) for a feature f_i with all other features of an object O_i. Equation (3) then calculates the average FFoR value(AFFoR) of an object O_i for all the features. The Devian vector(Dev) of an object O_i is defined in Equation (4) as the absolute difference between the FFoR values of an object and its corresponding AFFoR value. Finally, the FFSc score of an object O_i is calculated using Equation (5). Using the FFSc for all the objects, a normal profile is created which stores the average FFSc score (M_{FFSc}) and the range of FFSc scores (N_{range}). Upon capturing of real-time traffic, the same features used before are extracted and the FFSc score is calculated for the captured packet instances (C_{FFSc}). A dissimilarity value is then calculated using equation (6) below.

$$Dis\_HBK = \frac{|C_{FFSc} - M_{FFSc}|}{N_{range}} \quad \textbf{(6)}$$

If the Dis HBK value is greater than a user defined threshold an alarm is generated. Using the CAIDA dataset, the method gives 100% detection accuracy for the threshold value between 1 and 1.3. However, detection accuracy degrades gradually when the threshold is less than 0.5 and greater than 1.3. Similarly, in DARPA dataset, the method gives 100% detection accuracy and high detection accuracy for threshold value of between 0.1 to 2 whereas the accuracy gradually decreases as the threshold value increases. It was concluded that the ideal threshold range is between 0.05 to 0.8 to achieve high detection accuracy for both DARPA and CAIDA datasets.

Another Statistics based novel DDoS detection approach was proposed by İlker Özçelik et al. in their work 'CUSUM-Entropy: An Efficient Method for DDoS Attack Detection'. The novelty here

was to perform additional signal processing on the entropy of the packet header field to improve detection efficiency. For a dataset X, with a finite number of independent symbols from 1 to n, the entropy is calculated and normalized using Equations (7-8).

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i \quad \textbf{(7)}$$

$$H_N(X) = \frac{H(X)}{\log n} \quad \textbf{(8)}$$

In this work, the entropy of the source IP address is used as a measure to detect a DDoS attack. Initially, wavelet transform is used to filter out the long-term variations of the observed entropy values to reduce the number of false alarms. A ten-step wavelet decomposition was performed to filter out the tenth level low-pass components.

The cumulative sum approach(CUSUM) used in this work was first proposed by Blazek et al. [5]. The idea behind the approach was to compare the current entropy average of observations with the long-term average. If the current average increases faster than the long-term average, then the CUSUM coefficient also increases and if it increases beyond a pre-defined threshold, then a DDoS attack is said to have occurred. Equation (9) describes the basic CUSUM process.

$$S[t] = \max\{0, (S[t-1] + H[t] - m[t])\};\ S(0) = 0 \quad \textbf{(9)}$$

S[t-1] – Old CUSUM value

H[t] – Entropy value at time t

m[t] – Long term average of CUSUM input

The long-term average m[t] is calculated using Equation (10)

$$m[t] = \varepsilon\, m[t-1] + (1-\varepsilon)H[t];\ m[0] = 0 \quad \textbf{(10)}$$

(ε) – Long term averaging memory; $0 < \varepsilon < 1$

Now, to reduce the high frequency noise, the entropy value (H[t]) is low-pass filtered using local averaging memory (α) in Equation (11).

$$\tilde{H}[t] = \alpha\, H[t] + (1-\alpha)\tilde{H}[t-1];\ \tilde{H}[0] = 0 \qquad \textbf{(11)}$$

Finally, equation (11) is substituted in Equation (9) and an algorithm correction variable C is added to form Equation (12).

$$\check{S}[t] = \max\{0, (\check{S}[t-1] + \tilde{H}[t] - m[t] - C)\};\ \check{S}(0) = 0 \qquad \textbf{(12)}$$

In Equation (12), C is multiplication of m[t] and correction parameter (ce) which forces the CUSUM coefficient values to 0 by adding more weight to long term average, (m[t]).

Figure (2.4) below shows the detection efficiency of the proposed CUSUM algorithm compared to the baseline Source IP based entropy approach.

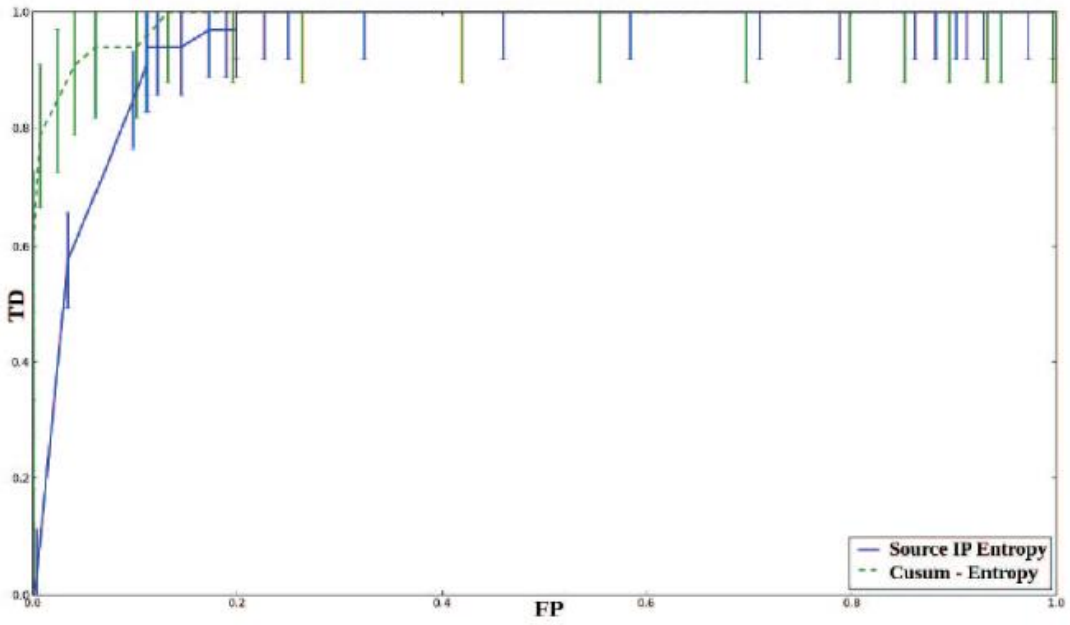

Figure 2. 4 *Detection efficiency of CUSUM - entropy approach and detection approach using entropy of source IP address with 95% confidence. Solid line: detection approach using entropy of source IP address. Dashed line: CUSUM - entropy approach (Adopted from* ***[6]****)*

The proposed modification of the CUSUM algorithm is shown to improve the detection efficiency of a DDoS attack with low false positive rates.

2.3 Hybrid Techniques

A Hybrid approach to detect a DDoS is one which uses a Statistical concept for attribute selection and then uses a Machine Learning algorithm for predicting a DDoS attack. One such hybrid approach is discussed in the paper 'Detecting Distributed Denial of Service Attacks through Inductive Learning'. The authors Sanguk Noh et al. propose a network traffic analysis mechanism by computing the ratio of number of TCP flags to the total number of TCP packets. Based on the calculation of the TCP flag rates, state action rules are compiled (using ML) by linking the TCP flag rates with the presence or absence of a DDoS attack.

The basis of the proposed approach is the differences between the rates of TCP flags to detect a DDoS attack. The proposed method is called the Traffic Rate Analysis (TRA) and calculates the TCP flag rate and protocol rate. Only TCP packets are retained from the captured TCP, UDP and ICMP packets. Next, amongst the selected TCP packets, the payload is filtered out and the TCP header is retained. The six possible flags in a TCP header are SYN, FIN, RST, ACK, PSH, and URG flags. If any of these flags are set, the agent counts and sums it up. The first metrics TCP flag rates are then calculated using Equation (13).

$$R_{td}[Fi] = \frac{total\ number\ of\ a\ flag\ (F)\ in\ a\ TCP\ header}{total\ number\ of\ TCP\ packets}\ (inbound)$$
$$R_{td}[Fo] = \frac{total\ number\ of\ a\ flag\ (F)\ in\ a\ TCP\ header}{total\ number\ of\ TCP\ packets}\ (outbound) \quad \textbf{(13)}$$

t_d – Sampling period

F – One of the six TCP flags; SIN, FIN, RST, ACK, PSH, URG

A protocol rate is defined as the ratio of total number of TCP, UDP or ICMP packets to the total number of IP packets.

The second and final stage of this work is to employ a packet collecting agent and an adaptive reasoning agent that analyses network traffic, detects a DDoS attack using a Machine Learning algorithm and finally issues an alarm in case of a DDoS attack. The complete set of complied rules for the alarming agents is constructed using three ML algorithms – C4.5 [7], CN2 [8] and

Bayesian classifier [9]. Figure (2.5) below summarises the performance of the proposed algorithm (TRM) for the three different classifiers used.

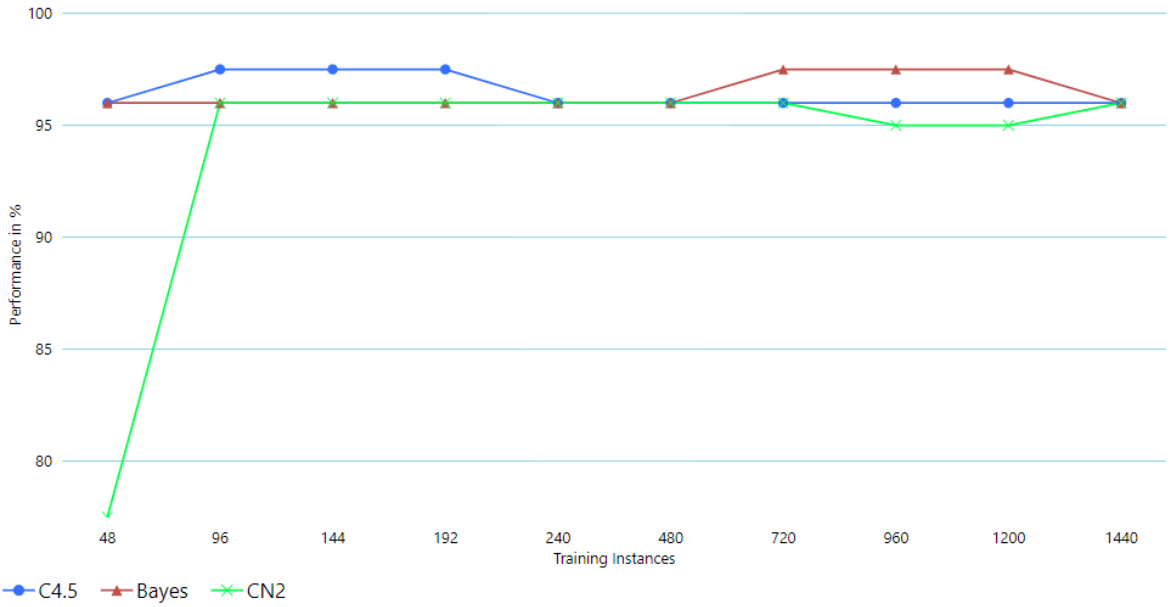

Figure 2. 5 *DDoS detection performance in terms of accuracy using the compiled ruled of TRA for the C4.5, Bayes and CN2 classifiers*

2.4 Classification based techniques

Machine Learning techniques including both classification and clustering have recently gained popularity as defence used against DDoS attacks. Apart from being faster, these methods are significantly more accurate than traditional methods used in detecting a DDoS attack. In the 2016 paper, 'Analysing Feature Selection and Classification Techniques for DDoS Detection in Cloud' by Opeyemi Osanaiye et al., the authors have analysed different feature selection methods and ML classification algorithms to establish a correlation between them. The objective of this work is to identify a feature selection method which when coupled with a ML algorithm can achieve a higher DDoS detection rate. The KDD Cup 1999 Dataset [10] containing 41 feature sets is used for experimentation and testing.

In the data-processing phase, filter based Feature Selection methods are used to extract the most important features from the set of all features. The four Feature Selection methods used are Information Gain, Gain Ratio, Chi-Squared and ReliefF.

IG is measured by a reduction in the uncertainty of identifying the class attribute when the value of the feature is unknown [11]. The uncertainty is measured using Entropy. For a variable X, Entropy can be calculated using Equation (14).

$$H(X) = -\sum_{i} P(x_i) log_2(P(x_i)) \quad \textbf{(14)}$$

Here, P(x_i) is the prior probabilities of X. After another attribute Y is observed, the Entropy changes and is now given using Equation (15) below.

$$H(X/Y) = -\sum_{j} P(y_j) \sum_{i} P(x_i|y_j) log_2 \left(P(x_i|y_j)\right) \quad \textbf{(15)}$$

where P($x_i|y_i$) is the posterior probability of **X** given the values of **Y.** Information Gain can now be defined as the amount by which the Entropy of X decreases with the addition of Y and is calculated using Equation (16).

$$IG(X/Y) = H(X) - H(X|Y) \quad \textbf{(16)}$$

The Information Gain (IG) value is now calculated for every feature using Equation (16) and the values are then sorted to select the most important features.

The next Feature Selection method implemented was Gain Ratio, which is a slight modification of the Information Gain method. Gain Ratio was introduced as a remedy to improve IG technique that tends to exhibit a bias towards features with a large diversity value [12] and can be calculated using Equation (17).

$$Gain\ Ratio\ (y,x) = \frac{Information\ Gain(y,x)}{Intrinsic\ Value(x)} \quad \textbf{(17)}$$

Here, the Intrinsic Value (x) is $-\sum \frac{|S_i|}{|S|} * \log_2 \frac{|S_j|}{|S|}$ where |S| is the number of possible values feature x can take and $|S_i|$ is the actual values taken by feature x.

The third Feature Selection method used is Chi-Squared which is used to test the independence of two variables. A high score indicates a strong dependent relationship. Equation (18) shows the calculation of Chi-square for a variable.

$$\chi^2(r, c_i) = \frac{N[P(r, c_i)P(\tilde{r}, \tilde{c}_i) - P(r, \tilde{c}_i)P(\tilde{r}, c_i)]^2}{P(r)P(\tilde{r})P(c_i)P(\tilde{c}_i)} \quad \textbf{(18)}$$

N: The whole dataset

r: Presence of the feature

$\tilde{r}$: Absence of the feature

c_i: class

P(r,c_i): Probability that feature r occurs in class c_i

P($\tilde{r}$,c_i): Probability that feature r does not occur in class c_i

P(r, $\widetilde{c_i}$): Probability that feature r occurs in a class not labelled c_i

P($\tilde{r}$, $\widetilde{c_i}$): Probability that feature r does not occur in a class not labelled c_i

P(r): Probability that feature r appears in the dataset

P($\tilde{r}$): Probability that feature r does not appear in the dataset

P(c_i): Probability that a dataset is labelled to class c_i

P($\widetilde{c_i}$): Probability that a dataset is not labelled to class c_i

ReliefF feature selection method evaluates a feature's worth by continuously sampling instances to distinguish between the nearest hit and nearest miss (nearest neighbour from same class and from different class) [13]. The attribute evaluator appends a weight to each feature according to its ability to distinguish among the different classes. Weights of features that exceed the user-defined threshold are selected as key features [14]. The top 14 features returned by each of these algorithms are selected for the next classification stage, although it is not clear how they came up with the number 14.

Finally, different classification algorithms are applied on the sorted list of features and the accuracy results are shown in Table (2.1).

Classifier	NIL	IG	Gain Ratio	Chi-Squared	ReliefF
Bayes Net	96.56%	93.76%	95.78%	93.76%	92.33%
DT	99.01%	99.13%	99.04%	99.13%	97.00%
J48	99.56%	99.67%	99.56%	99.67%	99.08%
Naive bayes	89.59%	89.45%	88.41%	89.45%	90.47%
One R	96.28%	96.28%	96.28%	96.28%	91.45%
Random Forest	99.81%	99.76%	99.78%	99.76%	99.36%

Table 2. 1 *Detection Accuracy with different classifiers*

The time taken to build the different models is shown in Table (2.2).

Classifier	NIL (seconds)	IG (seconds)	Gain Ratio (seconds)	Chi-Squared (seconds)	ReliefF (seconds)
Bayes Net	10.30	0.28	0.26	0.27	0.40
DT	15.06	3.82	3.78	3.61	5.22
J48	2.59	1.04	0.85	0.74	0.89
Naive bayes	0.14	0.12	0.16	0.11	0.12
One R	0.34	0.18	0.05	0.21	0.18
Random Forest	13.27	6.47	7.63	6.16	7.15

Table 2. 2 *Time to build models*

It was therefore concluded that the chi-squared feature selection method and J48 classification algorithm shows a high correlation and forms the most efficient pair to detect a DDoS attack.

Another unique ML based approach to detect DDoS attacks was proposed by Zecheng He et al. in their work 'Machine Learning Based DDoS Attack Detection from Source Side in Cloud'. The idea behind this approach is to use the statistical information from the cloud server's hypervisor and the information from virtual machines to detect a DDoS attack. This was done to prevent the network packets from being sent out to the outside world. Statistical features

of various kinds of attacks in the proposed framework, including DDoS attacks-flooding, spoofing and brute force attacks are also analysed.

The architecture of the proposed system is shown in Figure (2.6) below where an attacker rents multiple virtual machines (VM) and turns them into botnets. To monitor the activity on the virtual machines, a Virtual Machine Manager (VMM) stands between the VMs and the routers. The information gathered by the VMM from the VMs is fed to a ML engine which is responsible for detecting malicious activity. If suspicious behaviour is detected across multiple VMs, it is concluded that there might be an ongoing DDoS attack and the network connection of all those VMs is cut off.

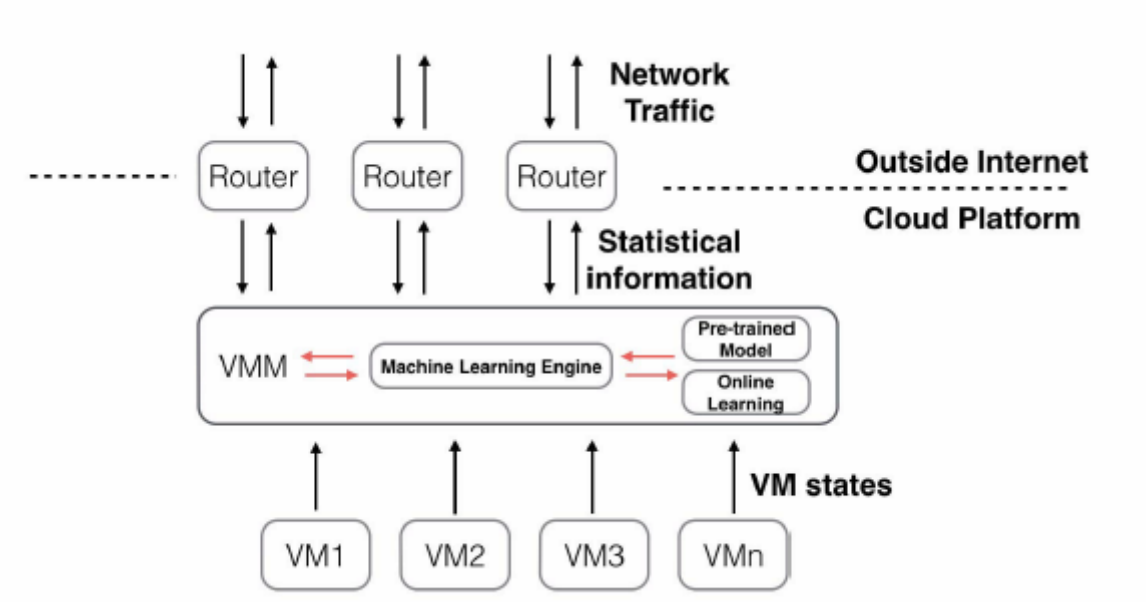

Figure 2. 6 *Architecture of the proposed system (Adopted from* ***[15]****)*

The VMs are programmed to simulate normal and attack traffic pattern and the data used for training the model is collected from the network packages coming in and going out of the attacker virtual machine(s) for 9 hours. Four different kinds of attacks are programmed to randomly start and end. The performance is measured using Accuracy, confusion matrix metrics and the F1-score for 9 classifiers. The results are shown in Table (2.3) below.

Model	Accuracy (%)	FP (%)	FN (%)	Precision (%)	Recall (%)	F1-Score
Decision Tree	99.07	0.06	0.01	99.95	98.33	0.99
Guassian EM	66.53	13.17	50.37	81.94	49.63	0.62
K-means (unsupervised)	87.76	0.44	22.05	99.54	77.95	0.87
LR	97.77	0.37	3.82	99.68	96.18	0.98
Naive Bayes	98.47	3.07	0.27	97.51	99.73	0.99
Random Forest	99.53	0.00	0.09	100.00	99.12	1.00
SVM Linear Kernel	99.73	0.04	0.44	99.94	99.56	1.00
SVM Poly Kernel	99.13	0.40	1.27	99.66	98.73	0.99
SVM RBF Kernel	98.15	3.78	0.24	96.93	99.76	0.98

Table 2. 3 *Joint detection results of three virtual machines*

In the multiple hosts monitoring experiment, it was shown that all machine learning algorithms got better results than in the single host monitoring experiment. The highest 0.9975 F1-Score and 99.73% accuracy using SVM was achieved with a linear kernel. Also, four algorithms (SVM with Linear and Poly kernels, Decision Tree and Random Forest) achieve accuracy greater than 99%.

Section 3: Network Security

3.1 What is Network Security?

Security is "the quality or state of being secure—to be free from danger." [16] In other words, security is the absence of threat. Network security also falls under this definition and can specifically be defined as the absence of threat in a computer network. It is achieved by designing and following a set of policies and rules to protect the integrity of a computer network and the data stored or transmitted within that network. An effective network security measure should be robust and thwart any threat aimed towards the network. A strong network security in place ensures the peace of mind of people within that network and in turn leads to a safe work environment.

Enforcers of a secure network aim towards achieving Confidentiality, Integrity and Availability (CIA) of a network and systems within that network. The three components of a CIA triad are:

1. Confidentiality – Protecting information and assets from unauthorised users
2. Integrity – Ensuring that information and assets is modified by authorised users only
3. Availability – Ensuring that information and assets is available to authorised users when needed

The CIA triad is discussed in the IT Security Policy document which is the principle document for network security and outlines the rules to ensure the security of the assets including information of an organisation. Ensuring that the CIA triad is met is often an important step towards designing a secure network.

In the next sub-sections, I will discuss about the network security terminology, followed by implementing network security in the different layers of an OSI model and finally a summary of this chapter.

3.2 Network Security Terminology

Within the security community, some words have specific meanings, whereas other words commonly associated with computer security have virtually no meaning [Krawetz 2007, 31]. Common security vocabulary [Schneider1999] includes the following:

Vulnerability: A defect or weakness in the feasibility, design, implementation, operation, or maintenance of a system [Krawetz 2007, 31]. No system is immune to vulnerabilities but a counter measure must be in place for every threat associated with the vulnerabilities.

Threat: An adversary who is capable and motivated to exploit a vulnerability [Krawetz 2007, 31]. A threat should always be taken seriously because if a threat translates into an attack, it often costs the company in both reputation and finances.

Attack: The use or exploitation of a vulnerability. This term is neither malicious nor benevolent. A bad guy may attack a system, and a good guy may attack a problem. [Krawetz 2007, 31].

Attacker: The person or process that initiates an attack. This can be synonymous with threat [Krawetz 2007, 31]. An attacker exploits the vulnerability of a system and tries to target that using the appropriate attack tools and techniques.

Exploit: The instantiation of a vulnerability; something that can be used for an attack. A single vulnerability may lead to multiple exploits, but not every vulnerability may have an exploit (e.g., theoretical vulnerabilities) [Krawetz 2007, 31].

Target: The person, company, or system that is directly vulnerable and impacted by the exploit. Some exploits have multiple impacts, with both primary (main) targets and secondary (incidental) targets [Krawetz 2007, 31].

Attack vector: The path from an attacker to a target. This includes tools and techniques [Krawetz 2007, 31]. Many companies require a high level of security with passwords (i.e. requiring people to use lower case, upper case, numeric and special characters), making them difficult to remember. Therefore, many people write their passwords on a piece of paper, exposing an alternative attack vector to acquire a password to the system (Krawetz 2007, 74).

Defender: The person or process that mitigates or prevents an attack [Krawetz 2007, 31]. Nowadays, many companies have an automated system in place between the inside company network and the world-wide web. This system needs to be robust and efficient in detecting and thwarting any kind of cyber-attack.

Compromise: The successful exploitation of a target by an attacker [Krawetz 2007, 31]. A compromised system or a network is one which is either taken down by the attacker or rendered useless definitely or indefinitely for legitimate users of that network or service.

Risk: A qualitative assessment describing the likelihood of an attacker/threat using an exploit to successfully bypass a defender, attack a vulnerability, and compromise a system [Krawetz 2007, 31]. Risk analysis should be done at every layer of the network architecture and appropriate measures should be in place to avoid any possible attack.

3.3 Implementing Network Security

The starting point of network security should be understanding the OSI or "Open System Interconnection" model. It is a standard for worldwide communication that defines a networking framework for implementing protocols in seven layers [17]. An OSI model breaks down the network into easily understood components that can be secured individually [17]. Figure (3.1) below shows the seven layers of an OSI model. The information flows from one layer to another and each layer implements its own set of protocols to make that transfer possible. The information flow starts from the Application layer and flows from layer to layer until it reaches the final Physical layer from where it reaches the destination. Upon reaching the destination, information flows from the Physical layer to the Application layer to deliver the information to the user at the destination.

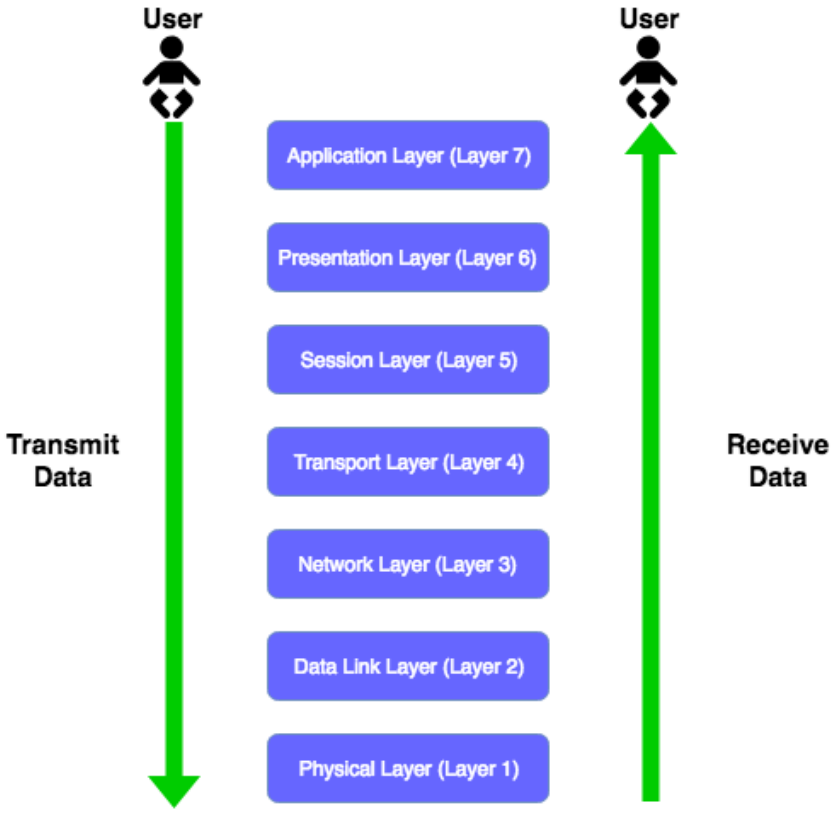

Figure 3. 1 *The OSI model*

Each layer in the OSI model is independent of the other layers and can only communicate with the layer above or below that layer. Although security vulnerabilities can creep in any of these layers, but here we will only discuss about the vulnerabilities in the Network Layer and the security measures that need to be taken.

In the book, "Top-Down Network Design" by Priscilla Oppenheimer, she discusses about the following steps towards achieving a secure network:

1. **Identify network assets.** Network assets can include hosts' operating systems, applications, routers, switches and network data that traverses the network. It is important to identify all such assets in the network under consideration and identifying the risks in the case of these assets being sabotaged or inappropriately accessed [18].
2. **Analyze security risks.** Risks can range from hostile intruders to untrained users who download Internet applications that have viruses. Hostile intruders can steal data, change data, and cause service to be denied to legitimate users [18].

3. **Analyze security requirements and trade-offs.** The security requirements, in general, involve the implementation of the CIA triad discussed in section 3.1. The Confidentiality, Integrity and Availability of assets should be the baseline security requirement for an organisation. Achieving security often involves trade-offs in terms of CPU power, network performance, network redundancy, etc.
4. **Develop a security plan.** A *security plan* is a high-level document that proposes what an organization is going to do to meet security requirements [18]. The plan is often based on the goals of an organisation after analysing the network assets and risks. A security plan should reference the network topology and include a list of network services that will be provided (for example, FTP, web, email, and so on). This list should specify who provides the services, who has access to the services, how access is provided, and who administers the services [18].
5. **Develop a security policy.** A security policy is a formal statement of the rules by which people who are given access to an organization's technology and information assets must abide [18]. A security policy can differ from one organisation to another but often has the basic items in addition to the organisation specific items.
6. **Develop security procedures.** Security procedures implement security policies. Procedures define configuration, login, audit, and maintenance processes. Security procedures should be written for end users, network administrators, and security administrators. Security procedures should specify how to handle incidents (that is, what to do and who to contact if an intrusion is detected) [18].
7. **Maintain security.** Finally, it is important to make sure that all the above steps are enforced by scheduling audits reading audit logs, responding to incidents, reading current literature and agency alerts, performing security testing, training security administrators, and updating the security plan and policy [18].

3.4 Summary

This chapter summarized the concepts of Network Security including the terms and terminologies which will be used in the subsequent chapters. The next chapter discusses about the Distributed Denial of Service attacks from the network security point of view.

Section 4: Distributed Denial of Service Attacks

4.1 What is a DDoS attack?

A Distributed Denial of Service attack is a co-ordinated attack by a malicious user(s) on a resource by inundating it with continuous high-rate legitimate *request* packets in a very short duration of time which ultimately takes down the resource and renders it useless for legitimate users. It is an attack by multiple sources on a single target system. This makes a DDoS attack deadly and difficult to mitigate. To use a popular metaphor, DDoS is considered a weapon of mass destruction on the Internet [16]. The simplest type of distributed denial of service (DDoS) is a Smurf attack. In this attack, one host generates many echo requests to many hosts across the network. Each echo request specifies a forged sender—the target of the DDoS. The result is a bombardment of echo replies sent to the target node. A large enough attack can cripple large networks due to the high volume of echo replies [19].

DDoS attacks are the most difficult to defend and unfortunately there are no standard defence mechanisms that organisations can deploy to defend against a DDoS attack. This is largely due to the fact that DDoS attacks try to mimic regular traffic but increased exponentially. Some of the world's largest DDoS attacks of the past decade are shown in Table (4.1) along with their rate of attack.

Organisation	Description	When?	Peak Traffic
GitHub	Web-based hosting service for version control	February 28, 2018	1.35 Tbps
GitHub	Web-based hosting service for version control	March 25, 2015	Unreported
BBC	British public service broadcaster	December 31, 2015	602 Gbps
CloudFlare	A US based company providing network security	February 11, 2014	~400 Gbps

	services including DDoS mitigation		
Spamhaus	A non-profit organisation to track email spammers and spam related activity	March 19, 2013	~300 Gbps

***Table 4. 1** Biggest DDoS attacks in terms of peak traffic rate*

4.2 Types of DDoS attacks

The flooding of a target system in a DDoS attack can be done by one of the following ways:

HTTP Flood. In this type of attack, the attacker exploits the HTTP GET or POST request to attack the server or application. An HTTP Flood uses less bandwidth and is most effective when the request packet can force the target to send back maximum resources possible.

UDP Flood. The attacker floods the target with User Datagram Protocol (UDP) packets at random ports of a random host. This forces the victim to constantly check for applications listening on those ports but since no applications are found, it responds with 'Destination Unreachable' packet causing the exhaustion of resources.

ICMP Flood (Ping). An ICMP flood aims to overwhelm the target with ICMP request (ping) packets without waiting for a reply.

SYN Flood. A SYN Flood attack exploits the three-way handshake protocol of a TCP connection. In a three-way handshake, a SYN request is answered by a SYN-ACK from the host and finally an ACK from the requester. Attackers continuously send SYN requests without responding to the victim's SYN-ACK or by using spoofed IP addresses to send a SYN request. Either way, the handshake remains incomplete and eventually exhausts more and more resources at the victim's.

Ping of Death. In a ping of death attack, IP protocols are manipulated to send malicious packets to the target. Ping of death was popular two decades ago but is not as effective as other attacks right now.

Slowloris. A Slowloris attack is aimed at a web server in which an attacker uses minimal resources to attack a system by requesting a connection with the target and as soon as

the connection is established, the attacker tries to keep the connection open for as long as possible and sends bogus HTTP packets to exhaust the web server.

NTP Amplification. In a NTP Amplification attack, the perpetrator uses UDP packets to target the publicly available Network Time Protocol server, a protocol used to synchronise computer clocks. It is an amplification attack because the query-to-response ratio is such attacks can be anywhere between 1:20 – 1:200 or even more.

Zero-day DDoS attacks. "Zero-day" is a term used for all unknown or new attacks. These attacks exploit vulnerabilities for which no defence mechanism exists yet.

4.3 Architecture of a DDoS attack

There are four broad components in a DDoS architecture – Attacker, Controller, Bots or zombies and a target. This is shown in Figure (4.1) below. The two components in the middle, Controllers and Botnets makes it a distributed attack. The perpetrator of a DDoS attack (Attacker) aims to disrupt the services at the Target machine. For this the attacker uses controllers (Handlers) to infect many computers (Botnets) to aid the attacker to carry out a DDoS attack. Handlers are computers which issue instructions to the zombies about how or when to attack the victim's servers to cripple it. Botnets can be voluntary but, in most cases, botnets have no idea that they are being used to accelerate a DDoS attack. For this reason, botnets are also commonly referred as zombies. Upon being infected, the botnets start sending bogus requests to the target machine. In some cases, the attacker programs the virus in a way that upon reaching the botnets, apart from sending high-rate traffic to the target, botnets also infect other systems to make more zombies for the attack. This exponentially increases the power of a DDoS attack and has the capability to bring down the victim server in less than a few minutes.

It is almost impossible to track the original source of a DDoS attack because of the presence of unsuspecting botnets between the attacker and the target. Also, the botnets use spoofed IP addresses to send traffic to the target which makes it difficult to track the botnets in the first place.

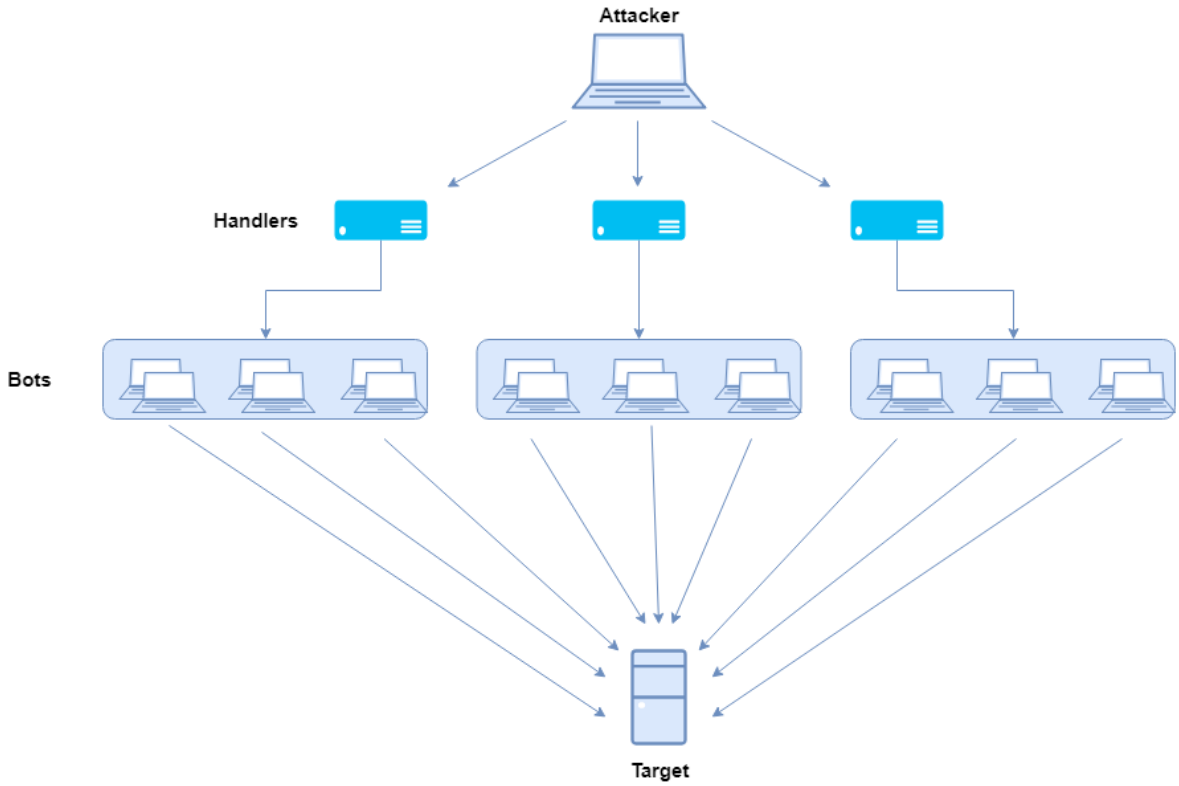

Figure 4. 1 *Architecture of a DDoS Attack*

4.4 Summary

This chapter gave an introduction of DDoS attacks with some of the biggest DDoS attacks seen, the several types of DDoS attacks and the architecture of a DDoS attack. The next chapter discusses the methodology of the proposed DDoS detection tool.

Section 5: Methodology

5.1 Dataset

The KDD Cup 1999 dataset contains a standard set of data to be audited, including a wide variety of intrusions simulated in a military network environment. Since 1999, KDD Cup 99 dataset has been the most wildly used data set for the evaluation of anomaly detection methods [20]. There are two versions of this dataset. The full dataset contains around 500 million packets with 41 features for each packet and categorized into normal or the kind of attack present in that packet. The second version, which is 10% of the original dataset, contains approximately half a million rows with the same structure as the full dataset.

Out of the 41 feature sets, 9 features are **Basic features** of individual TCP connections, 13 are **Content features** within a connection which are suggested by domain knowledge and the remaining attributes are **Traffic features** computed using a two-second time window. The description and category of each of these features is shown in Table (5.1) below. Basic features encapsulates all attributes that can be extracted from a TCP/IP connection [20]. Traffic Features include those that are computed with respect to a window interval and is divided into two groups [20]:

1. "same host" features: Examine only the connections in the past 2 seconds that have the same destination host as the current connection, and calculate statistics related to protocol behavior, service, etc [20].
2. "same service" features: Examine only the connections in the past 2 seconds that have the same service as the current connection [20].

Feature Name	Description	Category
duration	length (number of seconds) of the connection	Basic
protocol_type	type of the protocol, e.g. tcp, udp, etc.	Basic
service	network service on the destination, e.g., http, telnet, etc.	Basic
src_bytes	number of data bytes from source to destination	Basic

dst_bytes	number of data bytes from destination to source	Basic
flag	normal or error status of the connection	Basic
land	1 if connection is from/to the same host/port; 0 otherwise	Basic
wrong_fragment	number of ``wrong'' fragments	Basic
urgent	number of urgent packets	Basic
hot	number of ``hot'' indicators	Content
num_failed_logins	number of failed login attempts	Content
logged_in	1 if successfully logged in; 0 otherwise	Content
num_compromised	number of ``compromised'' conditions	Content
root_shell	1 if root shell is obtained; 0 otherwise	Content
su_attempted	1 if ``su root'' command attempted; 0 otherwise	Content
num_root	number of ``root'' accesses	Content
num_file_creations	number of file creation operations	Content
num_shells	number of shell prompts	Content
num_access_files	number of operations on access control files	Content
num_outbound_cmds	number of outbound commands in an ftp session	Content
is_hot_login	1 if the login belongs to the ``hot'' list; 0 otherwise	Content
is_guest_login	1 if the login is a ``guest'' login; 0 otherwise	Content
count	number of connections to the same host as the current connection in the past two seconds	Traffic
serror_rate	% of connections that have ``SYN'' errors	Traffic
rerror_rate	% of connections that have ``REJ'' errors	Traffic

same_srv_rate	% of connections to the same service	Traffic
diff_srv_rate	% of connections to different services	Traffic
srv_count	number of connections to the same service as the current connection in the past two seconds	Traffic
srv_serror_rate	% of connections that have ``SYN'' errors	Traffic
srv_rerror_rate	% of connections that have ``REJ'' errors	Traffic
srv_diff_host_rate	% of connections to different hosts	Traffic

Table 5. 1 *Features of KDD Dataset*

Unlike most of the DoS and Probing attacks, the R2L and U2R attacks don't have any intrusion frequent sequential patterns. This is because the DoS and Probing attacks involve many connections to some host(s) in a very short period of time; however, the R2L and U2R attacks are embedded in the data portions of the packets, and normally involves only a single connection [20]. Content features are used to detect such attacks by looking for suspicious behaviour in the data portion. Since we are concerned with detecting only DDoS attacks, we do not use the Content features and only use the Basic and Traffic features to base our models.

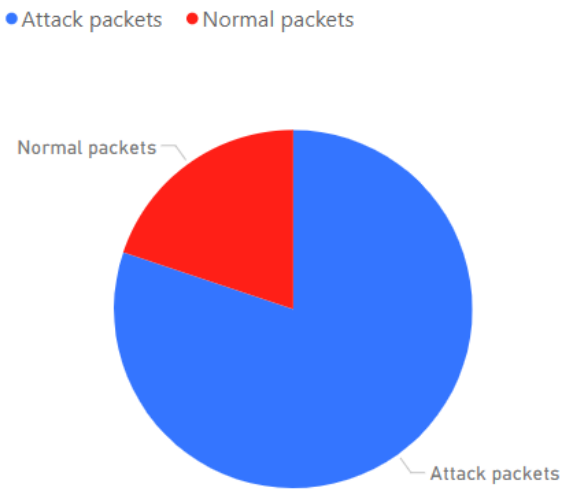

Figure 5. 1 *Distribution of packets in 10% KDD Dataset*

5.2 Data Pre-Processing

Out of the 28 remaining features, *protocol type, service and flag* had categorical values whereas all the other features have numeric values. To be able to apply feature selection in the next step to select the most important features, the features with categorical values are converted to numeric values. For each such feature, the distinct values are identified for the all the entries in that column and replaced with numeric values using simple integer assignment from 1 onwards. The reference table for this conversion is shown in Table (5.2) below.

protocol type	**tcp**:1, **udp**: 2, **icmp**: 3
service	**http**: 1, **smtp**: 2, **finger**: 3, **domain_u**: 4, **auth**: 5, **telnet**: 6, **ftp**: 7, **eco_i**: 8, **ntp_u**: 9, **ecr_i**: 10, **other**: 11, **private**: 12, **pop_3**: 13, **ftp_data**: 14, **rje**: 15, **time**: 16, **mtp**: 17, **link**: 18, **remote_job**: 19, **gopher**: 20, **ssh**: 21, **name**: 22, **whois**: 23, **domain**: 24, **login**: 25, **imap4**: 26, **daytime**: 27, **ctf**: 28, **nntp**: 29, **shell**: 30, **IRC**: 31, **nnsp**: 32, **http_443**: 33, **exec**: 34, **printer**: 35, **efs**: 36, **courier**: 37, **uucp**: 38, **klogin**: 39, **kshell**: 40, **echo**: 41, **discard**: 42, **systat**: 43, **supdup**: 44, **iso_tsap**: 45, **hostnames**: 46, **csnet_ns**: 47, **pop_2**: 48, **sunrpc**: 49, **uucp_path**: 50, **netbios_ns**: 51, **netbios_ssn**: 52, **netbios_dgm**: 53, **sql_net**: 55, **vmnet**: 56, **bgp**: 57, **Z39_50**: 58, **ldap**: 59, **netstat**: 60, **urh_i**: 61, **X11**: 62, **urp_i**: 63, **pm_dump**: 64, **tftp_u**: 65, **tim_i**: 66, **red_i**: 67
flag	**SF**: 1, **S1**: 2, **REJ**: 3, **S2**: 4, **S0**: 5, **S3**: 6, **RSTO**: 7, **RSTR**: 8, **RSTOS0**: 9, **OTH**: 10, **SH**: 11

Table 5. 2 *Conversion table for categorical variables to numerical values*

As discussed in Section 4.2, there could be different types of DDoS attacks and the class variable in the KDD Cup 1999 dataset stores information about the type of attack for each

packet. This is irrelevant for this research and therefore the class variable for each packet is modified as either “Attack” or “Normal” packet. Figure (5.1) the number of attack and normal packets in the 10% KDD Dataset.

To avoid the influence of features with high values over features with low values, the dataset is normalised using min-max normalisation to fall in 0-1 range.

$$n_i = \frac{o_i - \min(o)}{\max(o) - \min(o)} \quad \textbf{(19)}$$

Here o_i is the old value for a feature of a packet, min(o) is the minimum value across all the packets for that feature, max(o) is the maximum value across all the packets for that feature and n_i is the new normalised value. After normalization, the dataset falls within a 0-1 range and is ready to apply statistical operations which are the basis of any feature selection method. In the next sub-section, four different feature selection algorithms are discussed which are used to generate a sorted list of features from most important to least important.

5.3 Feature Selection

5.3.1 Information Gain

Information Gain (IG) is a common filter-based feature selection technique used in Machine Learning for subset selection. Information gain (IG) measures the amount of information in bits about the class prediction, if the only information available is the presence of a feature and the corresponding class distribution [21]. The main idea behind Information Gain is to measure the reduction in uncertainty while detecting a class variable if other feature(s) is known. With Information Gain, it is easy to differentiate between important features from non-important ones. This simplifies subset selection which in turn speeds up the classification process. The uncertainty is measured in entropy for distributions, sample entropy or estimated model entropy for datasets [22] where the entropy of a variable X [23] is defined as:

$$H(X) = -\sum_i P(x_i) \log_2\big(P(x_i)\big) \quad \textbf{(20)}$$

Here, $P(x_i)$ is the values of prior probabilities of variable X when considered independently. Now, the entropy of X changes when the value of another variable (let's say Y) is known beforehand and this can be defined as:

$$H(X|Y) = -\sum_j P(y_j) \sum_i P(x_i|y_i) \log_2\left(P(x_i|y_i)\right) \quad \textbf{(21)}$$

Here, $P(x_i|y_i)$ is the posterior probabilities of X given the values of Y. Information Gain is the gain in information after the entropy of X decreases upon knowing Y. It is defined as:

$$IG(X|Y) = H(X) - H(X|Y) \quad \textbf{(22)}$$

The information gain value returned for each feature is shown in Table (5.3) below.

Feature Name	Information Gain value	Feature Name	Information Gain value
src_bytes	0.6441664342020751	**dst_host_srv_count**	0.1695377427553092
srv_count	0.3474585661130222	**srv_rerror_rate**	0.0025543147200079996
rerror_rate	0.0013186625033869692	**dst_host_srv_diff_host_rate**	0.2669981620089094
urgent	4.787317054821827e-05	**dst_host_same_src_port_rate**	0.38310560332101895
dst_host_same_srv_rate	0.15567769993712754	**flag**	0.06222867599868387
duration	0.05289744658954354	**dst_host_count**	0.29194148615709103
srv_serror_rate	0.06612773328520494	**protocol_type**	0.30406555638179383
wrong_fragment	0.0008351242334735387	**count**	0.0027905981063345298

service	0.5709958288855335	**land**	4.75207351996465e-05
serror_rate	0.06231071022063339	**same_srv_rate**	0.08316473254702994
dst_host_rerror_rate	0.009665623509596988	**dst_host_diff_srv_rate**	0.16198961298925585
dst_host_srv_serror_rate	0.07999604826493611	**dst_host_srv_rerror_rate**	0.02439268384550275
diff_srv_rate	0.08259260969651527	**srv_diff_host_rate**	0.15743838301513258
dst_bytes	0.5343595652422289	**dst_host_serror_rate**	0.07429505823794769

Table 5. 3 *Information Gain Values*

Upon sorting the features from highest to lowest value where high IG value means more important feature and a low value means less importance of a feature towards predicting the class variable.

The feature ranking shown in Table (5.4) below is based on Equations (20-22) and ranks the feature based on their IG values.

Rank	Feature Name	Rank	Feature Name
1	land	15	diff_srv_rate
2	urgent	16	same_srv_rate
3	wrong_fragment	17	dst_host_same_srv_rate
4	rerror_rate	18	srv_diff_host_rate
5	srv_rerror_rate	19	dst_host_diff_srv_rate
6	count	20	dst_host_srv_count
7	dst_host_rerror_rate	21	dst_host_srv_diff_host_rate
8	dst_host_srv_rerror_rate	22	dst_host_count
9	duration	23	protocol_type

10	flag	24	srv_count
11	serror_rate	25	dst_host_same_src_port_rate
12	srv_serror_rate	26	dst_bytes
13	dst_host_serror_rate	27	service
14	dst_host_srv_serror_rate	28	src_bytes

Table 5. 4 *Ranked feature list according to the Information Gain values*

5.3.2 Chi-Squared

The chi-squared (χ^2) statistic is used to test the independence of two variables by computing a score to measure the extent of independence of these two variables [22]. With respect to feature selection, (χ^2) measures the independence of features with respect to the class [22]. (χ^2) begins with an initial assumption of independence between the features and the class. A high (χ^2) value for a feature is indicative of a strong correlation between the feature and the class. Chi-squared [22] is defined as:

$$\chi^2(r, c_i) = \frac{N[P(r,c_i)\mathrm{P}(\tilde{r},\tilde{c_i}) - \mathrm{P}(\mathrm{r},\tilde{c_i})\mathrm{P}(\tilde{r},c_i)]^2}{P(r)\mathrm{P}(\mathrm{r})\mathrm{P}(c_i)\mathrm{P}(\tilde{c_i})} \quad \textbf{[23]}$$

N: The whole dataset

r: Presence of the feature

$\tilde{r}$: Absence of the feature

c_i: class

P(r,c_i): Probability that feature r occurs in class c_i

P($\tilde{r}$,c_i): Probability that feature r does not occur in class c_i

P(r, $\tilde{c_i}$): Probability that feature r occurs in a class not labelled c_i

P($\tilde{r}$, $\tilde{c_i}$): Probability that feature r does not occur in a class not labelled c_i

P(r): Probability that feature r appears in the dataset

P($\tilde{r}$): Probability that feature r does not appear in the dataset

P(c_i): Probability that a dataset is labelled to class c_i

$P(\widetilde{c_i})$: Probability that a dataset is not labelled to class c_i

Like Information Gain, the Chi-Squared statistic is implemented and Table (5.5) shows the chi-squared values for every feature in our dataset. Again, a higher value means more dependence with the class variable and hence more importance. Table (5.6) sorts the features in descending order from most important to least important.

Feature Name	Chi-Squared value	Feature Name	Chi-Squared value
src_bytes	139785916.64967358	**srv_rerror_rate**	3370.779298718246
srv_count	32880138.155221865	**dst_host_srv_diff_host_rate**	84030.81966669411
rerror_rate	2919.9551572747096	**dst_host_same_src_port_rate**	44829.28024073678
urgent	2107816.575280261	**flag**	33838.866274725486
dst_host_same_srv_rate	1828.2455777968153	**dst_host_count**	3673139.9442167776
duration	71831510.27078745	**protocol_type**	51599.73007758231
srv_serror_rate	21748.8025789898	**land**	635162.936035931
wrong_fragment	8363.867767256423	**same_srv_rate**	6011.5147828457
service	413748.4809911481	**dst_host_diff_srv_rate**	16767.43578945106
serror_rate	21679.78240237042	**dst_host_srv_rerror_rate**	12637.074832932985
dst_host_rerror_rate	11569.502023477318	**count**	2677.5778616786006
dst_host_srv_serror_rate	24599.847492818524	**srv_diff_host_rate**	55811.621720508316

diff_srv_rate	12193.341832957834	**dst_bytes**	882933994.7439191
dst_host_srv_count	115409.27095554316	**dst_host_serror_rate**	24103.666110168026

Table 5. 5 *Chi-Squared values*

Rank	Feature Name	Rank	Feature Name
1	dst_host_same_srv_rate	15	flag
2	count	16	dst_host_same_src_port_rate
3	rerror_rate	17	protocol_type
4	srv_rerror_rate	18	srv_diff_host_rate
5	same_srv_rate	19	dst_host_srv_diff_host_rate
6	wrong_fragment	20	dst_host_srv_count
7	dst_host_rerror_rate	21	service
8	diff_srv_rate	22	land
9	dst_host_srv_rerror_rate	23	urgent
10	dst_host_diff_srv_rate	24	dst_host_count
11	serror_rate	25	srv_count
12	srv_serror_rate	26	duration
13	dst_host_serror_rate	27	src_bytes
14	dst_host_srv_serror_rate	28	dst_bytes

Table 5. 6 *Ranked feature list according to the Chi-Squared values*

5.3.3 Recursive Feature Elimination (RFE)

The next feature selection technique chosen was the Recursive Feature Elimination (RFE) technique. RFE starts with a full set of features and recursively considers smaller and smaller features by pruning the least important feature in every step until the required number of features is reached.

In step 1 of the algorithm, it fits the model to all the features, 28 in our case. The model then ranks the importance of each feature in terms of predicting the value of the class variable. Every feature is given a score from 1 till the number of features (S1>S2>….>Sn). Now, for each subset size S_i where *i* varies from 1 till n, the algorithm retains the i most important features and re-trains the model using those features. The model performance is re-evaluated and the features are ranked again based on this new model. This is done till a stopping condition is reached or when i reaches n. The final ranking returned by the algorithm lists the features from most important to least important and is afterwards used as an input to the classifiers. The pseudo code of the algorithm shown below clearly outlines every step of the RFE algorithm.

1.1 Use the training set to train the model with all features
1.2 Calculate the model performance
1.3 Note the feature ranking or importance
1.4 **for each** subset size S_i where i = 1…S **do**
1.5 Keep the S_i most importance features
1.6 Train the model using S_i features
1.7 Calculate the model performance
1.8 Calculate the feature ranking again
1.9 **end**
1.10 Calculate the performance profile over S_i
1.11 Determine the appropriate number of features to use
1.12 Use the RFE model returned by the optimal number of features S_i

Recursive Feature Elimination Pseudo-code

Recursive Feature Elimination does not consider all the possible subsets of the features but in most real-world problems searching over all possible subsets of features is not feasible and in those case RFE acts as a good alternative. The RFE algorithm is implemented using the scikit-learn RFE library and the sorted list of features from most important to least important features is shown in Table (5.7) below.

Rank	Feature Name	Rank	Feature Name
1	diff_srv_rate	15	flag
2	same_srv_rate	16	dst_host_diff_srv_rate
3	dst_host_srv_serror_rate	17	count
4	srv_serror_rate	18	service
5	rerror_rate	19	dst_host_srv_count
6	srv_rerror_rate	20	srv_count
7	protocol_type	21	dst_host_rerror_rate
8	dst_host_serror_rate	22	dst_host_count
9	wrong_fragment	23	src_bytes
10	dst_host_same_src_port_rate	24	srv_diff_host_rate
11	dst_host_srv_diff_host_rate	25	urgent
12	dst_host_same_srv_rate	26	dst_bytes
13	dst_host_srv_rerror_rate	27	land
14	serror_rate	28	duration

Table 5. 7 *Ranked feature list based on Recursive Feature Elimination*

5.3.4 Weighted Ranked Feature Selection (WRFS)

Weighted Ranked Feature Selection (WRFS) is a novel feature selection technique implemented for this project and aimed at improving the ranked list returned by a feature selection technique. If the important features needed for classification are ranked high in the list, the classifier can predict the value of the class variable using relatively less number of features and this can help to improve the detection time for a classifier. This is significant in any classification problem especially the DDoS detection problem in which the time of detection is of utmost importance. In WRFS, for each feature selection technique, a weight is assigned to the features based on the rank of that feature in the sorted list returned by that algorithm. Then, for every distinct feature, the weights assigned to that feature by all the previous three feature selection algorithms are summed to produce the final weight for that feature. These weights are used to produce a final ranked list of features.

The input of the WRFS algorithm is the sorted list of features from IG, Chi-squared and RFE feature selection techniques. In Step 1.3 of the pseudo code below, three empty dictionaries are initialised to store the weights of the features. In steps 1.4-1.6, we iterate through each distinct feature starting from the first feature till the 28^{th} feature and for each feature, its weight is determined based on the rank of that feature in the sorted IG, Chi2 and RFE list. These lists are sorted in a way that the most important feature appears first and the least important feature appears last. Since we have three weight values corresponding to the three lists, we store the weights as the value in the dictionary corresponding to that feature name (key). Now, for every feature, we have one additional entry in all the three dictionaries. After we have 28 values corresponding to every feature in all the three dictionaries, in step 1.7-1.9, we iterate through the list of features again and this time, the weights of that feature from all the three dictionaries is aggregated to get the final weight value for that feature. This is stored in the WRFS_dict dictionary. Finally, in steps 1.10-1.11, the WRFS_dict is sorted based on the values and only the keys are then returned as the final list of sorted features using WRFS from most important to least important.

1.1 **Input:** sorted_IG_list, sorted_chi2_list, sorted_RFE_list
1.2 **Output:** sorted_WRFS_list
1.3 Initialise 3 empty dictionaries, IG_dict, Chi2_dict, RFE_dict to store the weights of features
1.4 **for each** distinct feature
1.5 Get the rank of that feature from the sorted list and assign it as the weight of that feature in the dictionary, e.g sorted_gain_dict[sorted_gain_list[i]] = i
1.6 **end**
1.7 **for each** distinct feature
1.8 Get the weight of that feature from all three dictionaries-IG_dict, Chi2_dict, RFE_dict and sum it up to get the final weight of that feature in WRFS_dict
1.9 **end**

1.10 Sort the WRFS_dict from highest weight value to lowest weight value

1.11 Get the keys of the sorted WRFS_dict and return it as sorted_WRFS_list

Based on the algorithm above, the results are shown in Table (5.8) as the sorted list of features returned by WRFS. It was later observed from the classification accuracy results that the features returned by WRFS led to high accuracy of DDoS detection using considerably less number of features than IG, Chi-squared and RFE.

Rank	Feature Name	Rank	Feature Name
1	dst_bytes	15	flag
2	src_bytes	16	serror_rate
3	srv_count	17	dst_host_rerror_rate
4	dst_host_count	18	dst_host_serror_rate
5	service	19	dst_host_srv_serror_rate
6	duration	20	dst_host_same_srv_rate
7	srv_diff_host_rate	21	dst_host_srv_rerror_rate
8	dst_host_srv_count	22	srv_serror_rate
9	dst_host_srv_diff_host_rate	23	count
10	dst_host_same_src_port_rate	24	diff_srv_rate
11	urgent	25	same_srv_rate
12	land	26	wrong_fragment
13	protocol_type	27	srv_rerror_rate
14	dst_host_diff_srv_rate	28	rerror_rate

Table 5. 8 *Ranked feature list based on Weighted Ranked Feature Selection*

5.4 Classification

Naïve Bayes, SVM, Decision Trees and Random Forest are used as the classification algorithms to create models used for predicting a DDoS attack in the network. 20-fold cross validation is used for all the algorithms except SVM where a 10-fold cross validation is used because of reasons discussed in section 6.1. For every classifier, four models are created corresponding to

the four different feature selection algorithms discussed in section 5.3. The number of features used is different for every model and is optimized based on the accuracy results. The optimized models are then stored in pickles which are later used to predict a DDoS attack in a real-time network traffic.

5.4.1 Naïve Bayes

Naïve Bayes is a common classifier used in many Machine Learning problems. It is based on Bayes theorem, which helps us to define the probability of an event based on some prior knowledge of certain conditions associated with that event. Naïve Bayes classifiers work based on the assumption that the features are independent of each other and this is the reason why they are called "Naïve" classifiers.

Bayes' theorem is stated mathematically as the following equation [24]:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad \textbf{[24]}$$

A and B are some events and P(B) is not 0. P(A|B) is a conditional probability of A given that the event B is true. Similarly, P(B|A) is a conditional probability of B given A is true. P(A) and P(B) are the independent probabilities of A and B without observing each other.

From a machine learning perspective, given a problem instance to be classified, represented by a vector $\mathbf{X} = \{\mathcal{X}_1, \dots, \mathcal{X}_n\}$ representing *n* independent features, Naïve Bayes assigns to this instance probabilities

$$P((C_k \mid \mathcal{X}_1, \dots, \mathcal{X}_n)) \quad \textbf{[25]}$$

for each of K possible outcomes or classes C_k [25]. Based on Equation (24), we can now write equation (25) as:

$$P(C_k \mid \boldsymbol{X}) = \frac{P(\mathbf{X} \mid C_k)\, P(C_k)}{P(\boldsymbol{X})} \quad \textbf{[26]}$$

After transforming Equation (26) based on the assumption that each feature $\mathcal{X}_i$ is conditionally independent of every other feature $\mathcal{X}_j$, we get Equation (27).

$$
\begin{aligned}
p(C_k \mid x_1, \ldots, x_n) &\propto p(C_k, x_1, \ldots, x_n) = \\
&= p(C_k)\, p(x_1 \mid C_k)\, p(x_2 \mid C_k)\, p(x_3 \mid C_k)\, \cdots \\
&= p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)\,.
\end{aligned}
\qquad \textbf{[27]}
$$

where $\propto$ defines proportionality.

Some of the attributes in the KDD Dataset have continuous values. We use the Gaussian Naïve Bayes classifier to account for continuous values. Gaussian Naïve Bayes assumes that the values are distributed according to a Gaussian distribution and uses the mean and variance of each attribute in each class and is shown in Equation (28).

$$
p(x = v \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}}\, e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}} \qquad \textbf{[28]}
$$

Scikit-learn's GaussianNB library is used to implement Gaussian Naïve Bayes. The sorted list of features (from most important to least important) returned by the feature selection techniques is used as input for the Naïve Bayes classifier. Since our aim is to find the minimum number of features which can detect a DDoS attack with high accuracy, we run the classifier 28 times, incrementally adding a feature from the sorted list in every iteration. So, iteration 1 runs with just 1 feature (the most important feature) and trains the model using 20-fold cross validation and returns the accuracy of detection. The second iteration runs with 2 most important features and returns the accuracy. The last iteration runs with all the 28 features. The accuracy results are shown in section 6. Here is the pseudo code of the classifier:

1.1 **for each** feature selection technique
1.2 **for** num_features in range(1,28)
1.3 Import the pre-processed dataset file in a Pandas dataframe
1.4 Prune the dataframe according to the num_features
1.5 Train the model using 20-fold cross validation
1.6 Use the trained model to predict the class variable values for the testing data
1.7 Return Accuracy
1.8 **end**

1.9 **end**

5.4.2 SVM for Binary Classification

Another commonly used Machine Learning classifier to detect a DDoS attack is SVM or Support Vector Machines. The aim of SVM is to orientate a hyperplane in such a way as to be as far as possible from the closest members of both classes [26].

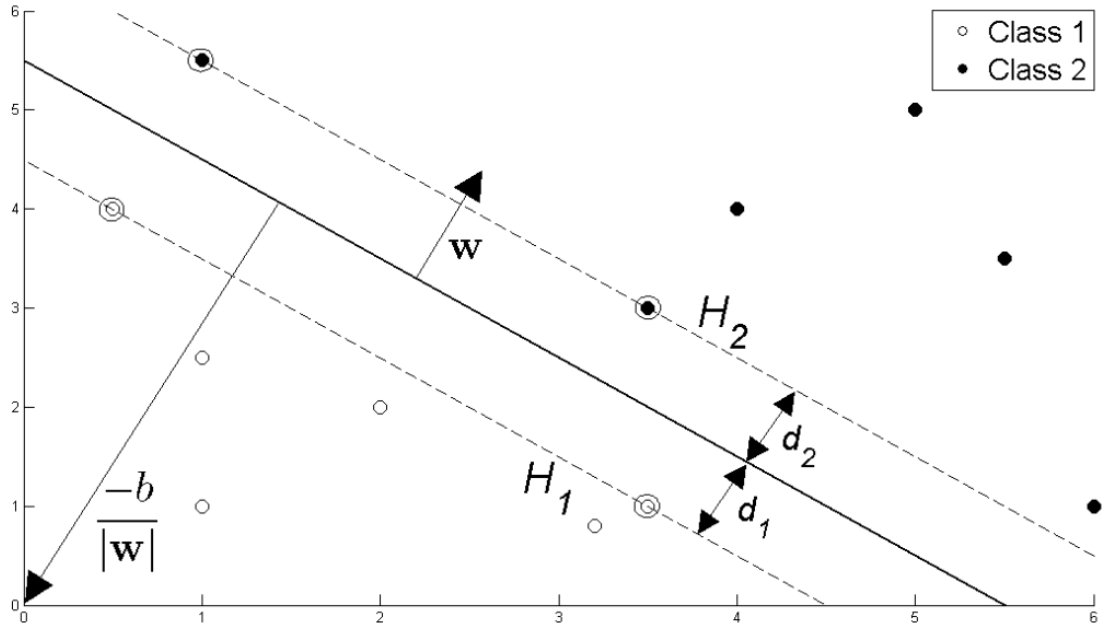

Figure 5. 2 *Support Vectors (Adopted from* ***[26]****)*

Referring to Figure (5.2) above, implementing SVM boils down to selecting the variables **w** and b so that our training data can be described by [39]:

$$x_i.w + b \geq +1 \quad for \; y_i = +1 \qquad \textbf{[29]}$$

$$x_i.w + b \; +1 \leq -1 \; for \; y_i = -1 \qquad \textbf{[30]}$$

d_1 is defined as the distance from H_1 to the hyperplane and d_2 from H_2 to it. The hyperplane's equidistance from H_1 and H_2 means that $d_1 = d_2 - a$ which is also known as the SVM's margin. To orient the hyperplane to be as far from the Support vectors as possible, this margin is maximized using some vector geometry and Quadratic Programming optimization. The final equation translates to:

$$\max_{\alpha} \left[\sum_{i=1}^{L} \alpha_i - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha} \right] \quad \text{s.t.} \quad \alpha_i \geq 0 \ \ \forall_i \quad \text{and} \quad \sum_{i=1}^{L} \alpha_i y_i = 0$$

and running a QP solver returns α whereas Equations 31 and 32 can be used to find **W** and b.

$$\mathbf{w} = \sum_{i=1}^{L} \alpha_i y_i \mathbf{x}_i \qquad 31$$

$$b = \frac{1}{N_s} \sum_{s \in S} (y_s - \sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s) \qquad 32$$

Variables **w** and b define the separating hyperplane's optimal orientation and hence the Support Vector Machine.

5.4.3 Decision Tree

To add a diversity of models to test our sorted lists returned by the feature selection methods in the previous step, we decided to add a Decision Tree classifier which is one of the widely-used algorithms in classification and regression problems. A decision tree creates a tree like structure where each node of the tree represents a feature, each link represents a decision (rule) and each leaf represents a possible outcome. Compared to other classifiers, Decision Tree is the easiest to understand and interpret. The basic steps while implementing a Decision Tree are:

1. Identify the best attribute and position it at the root of the tree.
2. Split the training dataset into subsets in a way that each subset only contains data with the same value from an attribute.
3. Repeat step 1 and 2 until we find the leaf node for each branch of the tree.

Now, to predict the value of a class variable using a Decision Tree, we start from the root of the tree and compare the values stored in the root attribute with the values of the instance's attribute. Based on the results of this comparison, we choose a branch to follow and move on to the next node in the tree. This process is repeated until the leaf node is reached with the predicted value of the class variable.

5.4.4 Random Forest

The last classifier used to create the models was Radom Forest. A Radom Forest is basically an ensemble of many decision trees working together and trained using the *bagging* method. The *bagging* method is based on an underlying theory that the combination of multiple models increases the effectiveness of the resulting model. Radom Forest can be used for both classification and regression problems but in our study, we have used it for classification since we are dealing with a binary class variable. Figure (5.3) [27] shows a decision tree with two trees. The results from the two trees are combined by the ensemble approach employed by the Random Forest to produce the result.

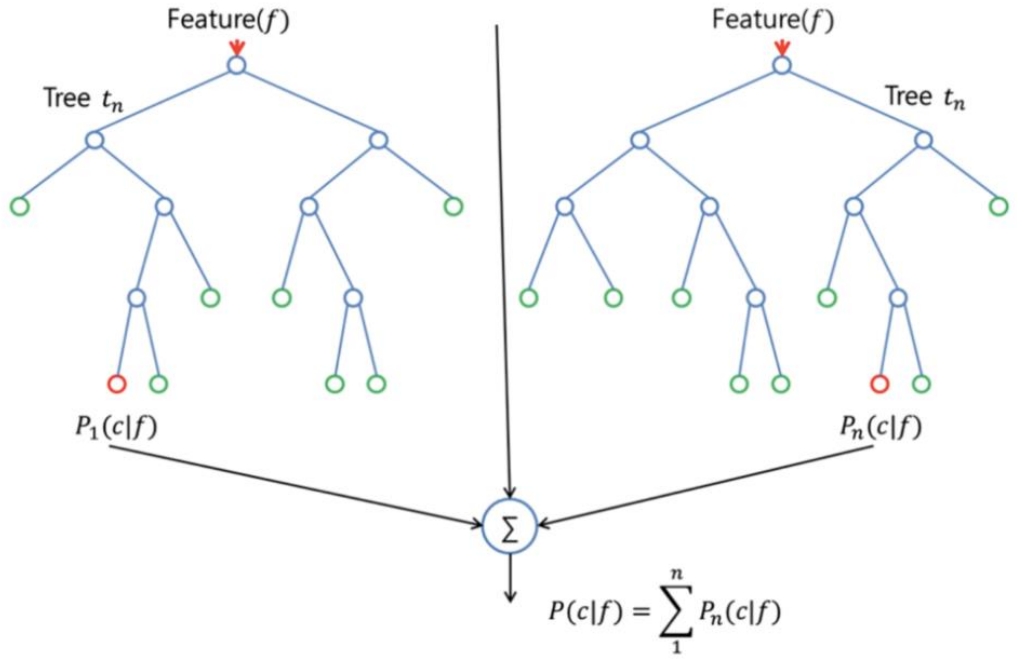

Figure 5. 3 *Decision Tree with two trees (Adopted from* ***[27]****)*

A Random Forest has almost the same hyper parameters as a Decision Tree in addition to the hyper parameters of a bagging classifier which controls the ensemble of trees. Instead of searching for the best feature while splitting a node, it searches for the best feature among a random subset of features. This process creates a wide diversity, which generally results in a better model [27].

Section 6: Results and Simulations

6.1 Accuracy Results

The sorted lists returned by the four classifiers are used to incrementally feed the attributes to Naïve Bayes, SVM, Decision Trees and Random Forest classifiers. Through Table (6.1) till Table (6.4), we show the accuracy results obtained for every feature selection algorithm. The first column, '# features' is the number of features taken from the most important to the least important feature for accuracy calculations. The bold value in every column is the one where the accuracy reaches the maximum and does not fall after that. The number of features corresponding to the bold value is then used to create the model for that classifier using those many number of features. For e.g. in Table (6.1) below, if we look at the Naïve Bayes column, we find that the bold value is 98.36099847 and the number of features corresponding to that value is 24. Using this, we then create a Naïve Bayes model using the top 24 features returned by the Information Gain list and use that model to do predictions on real-time network traffic with an accuracy of 98.36099847%.

First, the features selected using Information Gain feature selection technique are used as input to four classification models – Naïve Bayes, SVM, Decision Tree and Random Forest. The accuracy results with Naïve Bayes show an erratic accuracy pattern. The accuracy of detection starts with ~76% using only one feature but this is most likely due to underfitting. This is soon proved when the input number of features increase and the accuracy starts dropping and drops to a low of ~19% before reaching a superficial constant value of ~37% for majority of the input sets of features. The highest accuracy achieved using Naïve Bayes is ~98% using 24 most important features. With these results, Naïve Bayes when working with the sorted list returned by Information Gain cannot be classified as a good classifier as the number of features needed to achieve good accuracy is far too many. SVM is known to be an exhaustive classifier and for this project, SVM would not converge with the regular 10% KDD dataset, therefore the size of the dataset was reduced for SVM to 10,000 packets. SVM performs significantly better than Naïve Bayes with a constant accuracy of ~77% before reaching ~99% with the top 23 features.

Although SVM performs better than Naïve Bayes but it still uses 23 features which is not a significant reduction in the number of features and defeats the purpose of feature selection. Decision Tree and Random Forest were implemented next and accuracy results were seen to be very close to each other. Both these algorithms out-perform Naïve Bayes and SVM in terms of reaching and maintaining a relatively high accuracy percentage with the incremental increase of features. The maximum accuracy reached is ~99% with the top 23 features. Table (6.1) and Figure (6.1) summarize these results in a table and a line chart which shows the accuracy change for all the four algorithms with the increase in number of features used for classification.

# Features	Naïve Bayes	SVM(10k), k=5	Decision Trees	Random Forest (10)
1	76.68390804	77.86001	80.30906492	80.30906492
2	78.52776052	77.86001	80.30906492	80.30906492
3	19.94537724	77.86001	80.30906492	80.30906492
4	19.94537724	77.86001	80.24995799	80.22951348
5	19.94537724	77.86001	80.34044018	80.33234334
6	19.94537724	77.86001	80.49104086	80.44225741
7	19.94537724	77.86001	81.15457654	81.18615398
8	19.87776864	77.86001	81.87296813	81.86628859
9	77.03146462	77.86001	84.04898924	84.03380761
10	33.04903431	77.86001	84.08522207	84.07874457
11	37.07950146	77.86001	84.19331395	84.16133129
12	37.10217256	77.86001	84.27994981	84.26051726
13	37.3215969	77.86001	84.84328413	84.88720933
14	37.30803459	77.86001	85.24974384	85.24427839
15	37.09569504	77.86001	86.18492756	86.19079765
16	41.2115202	77.86001	86.20597924	86.18452259
17	41.30665804	77.86001	89.88962569	89.89023294
18	41.32001777	77.86001	93.64083814	93.64306435
19	41.27811668	77.86001	93.65379313	93.70945842
20	41.27811666	77.86001	93.92240189	93.99061709
21	41.42041862	77.86001	97.38981358	97.42280801
22	41.76696346	56.94247376	97.41511606	97.43090484
23	45.03464493	**99.43001**	**99.64697869**	**99.68786778**
24	**98.36099847**	99.28001	99.6919159	99.72207633
25	98.40755524	99.28001	99.70689472	99.7710621
26	96.50886191	99.28001	99.69191579	99.77288412

27	95.84632453	99.28001	99.72855358	99.78685105
28	76.60600997	99.28001	99.78583869	99.81640469

Table 6. 1 *Accuracy (%) results for different classifiers based on the list returned by Information Gain feature selection technique*

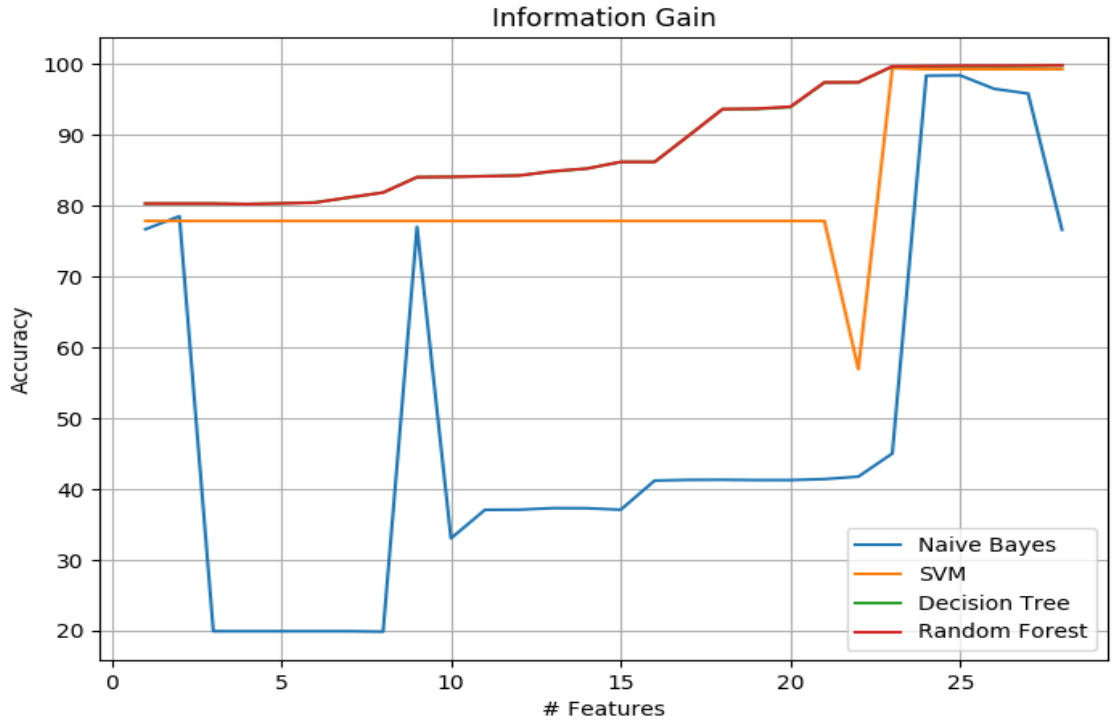

Figure 6. 1 *Accuracy variation for different classifiers with the number of features used from the sorted Information Gain list*

When features were selected using Chi-Squared feature selection technique, we observed that the number of features needed to reach a good detection accuracy drops to 18 features compared to an average of 23 features required to achieve the same accuracy when the features were selected using Information Gain. Naïve Bayes performs poorly again and as can be seen in Figure (6.2) shows an erratic behaviour with the incremental increase in the number of features. Eventually, it reaches a high of ~98% with 25 features but a drop is also observed when the input features increase further. SVM shows a steady pattern with the accuracy remaining at ~77% until 16 features after which it increases to ~99% for 17 features and stays at that point till the end. Decision Tree and Random Forest show a similar pattern again with

the accuracy values following a very similar pattern with the incremental increase in the number of features. To conclude, top 25, 17, 16 and 16 features are respectively selected for Naïve Bayes, SVM, Decision Trees and Random Forest to create the models and are stored in pickles to be used in the testing stage. With these results, we observed a significant improvement in terms of number of features selected using Chi-Squared Table (6.2) compared to Information Gain Table (6.1).

# Features	Naïve Bayes	SVM(10k), k=5	Decision Trees	Random Forest (10)
1	80.30906492	77.86001	85.18435927	85.18435927
2	80.28639318	77.86001	85.15844946	85.16472445
3	76.27481497	77.86001	85.81591235	85.80599372
4	76.27785129	77.86001	85.87947254	85.88028221
5	34.92709601	77.86001	87.9391047	87.91623117
6	41.51535503	77.86001	88.12634309	88.11419785
7	41.56960384	77.86001	88.54879566	88.54636659
8	41.45361564	77.86001	88.4915105	88.54231815
9	41.29208372	77.86001	88.71680499	88.80506053
10	41.18824177	77.86001	88.92873983	88.9078904
11	41.58822591	77.86001	89.17670473	89.21759332
12	41.59369138	77.86001	89.24957596	89.2805461
13	41.56899602	77.86001	89.62243224	89.64753238
14	41.51677137	77.86001	89.93011067	89.93942206
15	41.71089269	77.86001	89.94853077	89.91654856
16	41.65522692	77.76036482	**99.39598048**	**99.36379584**
17	41.69206751	**99.67007996**	99.49091559	99.4688516
18	41.70137882	99.68007996	99.47897136	99.48868748
19	41.72465723	99.68007996	99.57653683	99.64029897
20	42.72259257	99.69001	99.69495207	99.71053856
21	42.71287633	99.69001	99.72956585	99.77774225
22	42.72279496	99.69001	99.71984961	99.7431282
23	42.70579161	99.69001	99.71843309	99.77753982
24	46.43256401	99.69001	99.68685522	99.78644647
25	**98.3769896**	99.28001	99.75952506	99.7807786
26	98.38387195	99.28001	99.77632598	99.78604124
27	77.46225052	99.28001	99.78664889	99.80466432
28	76.60600997	99.28001	99.7771346	99.8202505

Table 6. 2 *Accuracy (%) results for different classifiers based on the list returned by Chi-Squared feature selection technique*

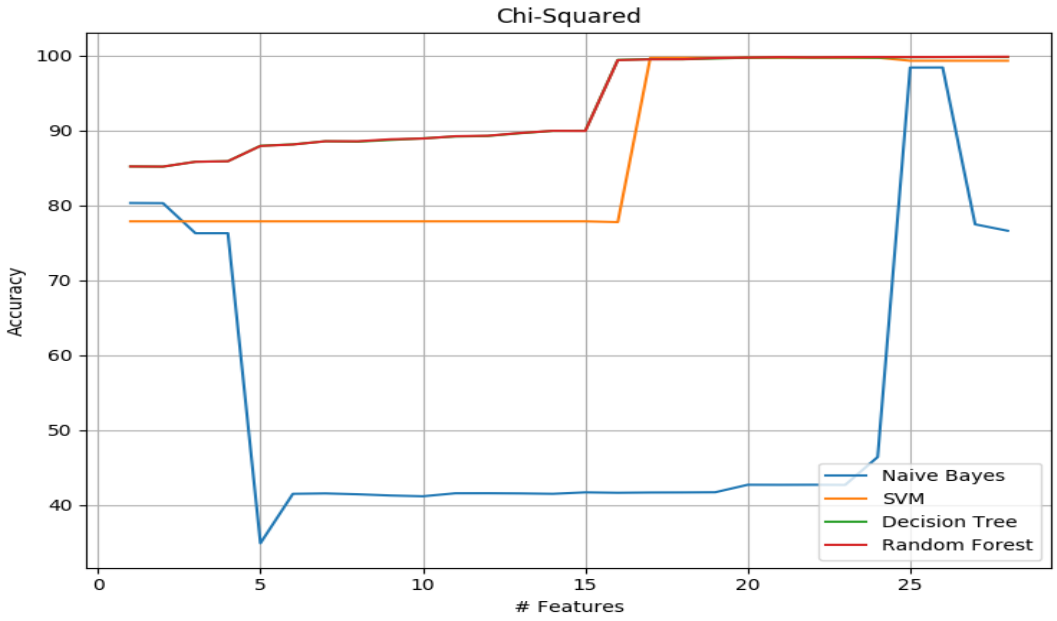

Figure 6. 2 *Accuracy variation for different classifiers with the number of features used from the sorted Chi-Squared list*

Features selected using Recursive Feature Elimination (RFE), used an average of 12 features to detect an attack with a high accuracy. Naïve Bayes performed poorly as expected and showed erratic results before reaching a high accuracy of ~98% using top 20 features but this does not last long as the accuracy drops again to ~76% as the number of features are increased further. SVM performs better using RFE as it reaches a high accuracy of ~99% using only 7 features and the accuracy does not drop further with the increase in the number of features until all the features are fed to the model. Decision Tree and Random Forest also perform better using features from RFE as they reach a high accuracy of ~99% using the top 11 features without any drop in accuracy as more features are seen. The top 20, 7, 11 and 11 features are respectively selected for Naïve Bayes, SVM, Decision Trees and Random Forest to create the models and stored in pickles. This is a further improvement in terms of average number of features used to achieve high accuracy when compared to the results from Chi-Squared feature selection (Table 6.2).

# Features	Naïve Bayes	SVM(10k), k=5	Decision Trees	Random Forest (10)
1	80.32525768	77.86001	80.53962227	80.54124163
2	80.91632672	77.86001	80.67908934	80.68374503
3	41.27265198	77.86001	81.73835546	81.74260631
4	41.21718919	77.86001	81.92255727	81.92660569
5	41.21496263	77.86001	82.84073859	82.83810711
6	41.21273605	77.86001	82.86280247	82.86179037
7	98.16950959	**99.68007996**	98.75673245	98.75288645
8	42.26572981	99.68007996	98.74438488	98.74519455
9	41.53883513	99.68007996	98.92555082	98.93283792
10	41.53397702	99.68007996	98.85612337	98.86644684
11	41.53438186	99.68007996	**99.15226234**	**99.1609666**
12	41.49652914	99.67007996	99.16663489	99.17837527
13	41.48883715	99.67007996	99.38889269	99.41500492
14	41.50341156	99.67007996	99.39557257	99.40650321
15	41.71838224	99.67007996	99.40427662	99.43909302
16	41.7266815	99.67007996	99.39172656	99.43281797
17	41.7238476	99.67007996	99.57856078	99.62795136
18	41.69409171	99.65009495	99.7307809	99.7751107
19	41.67810048	99.69001	99.76276277	99.77207439
20	**98.38872976**	99.28001	99.71236036	99.76600165
21	98.37010703	99.28001	99.71397949	99.76660904
22	98.38447917	99.28001	99.76863374	99.78705363
23	76.3037953	99.28001	99.78118382	99.80425936
24	76.30318804	99.28001	99.77956447	99.80000859
25	76.30318804	99.28001	99.7809814	99.82227492
26	76.39771863	99.28001	99.78624359	99.82106031
27	76.39771863	99.28001	99.78280241	99.80021098
28	76.60600997	99.28001	99.78280241	99.82207204

Table 6. 3 *Accuracy (%) results for different classifiers based on the list returned by Recursive Feature Elimination feature selection technique*

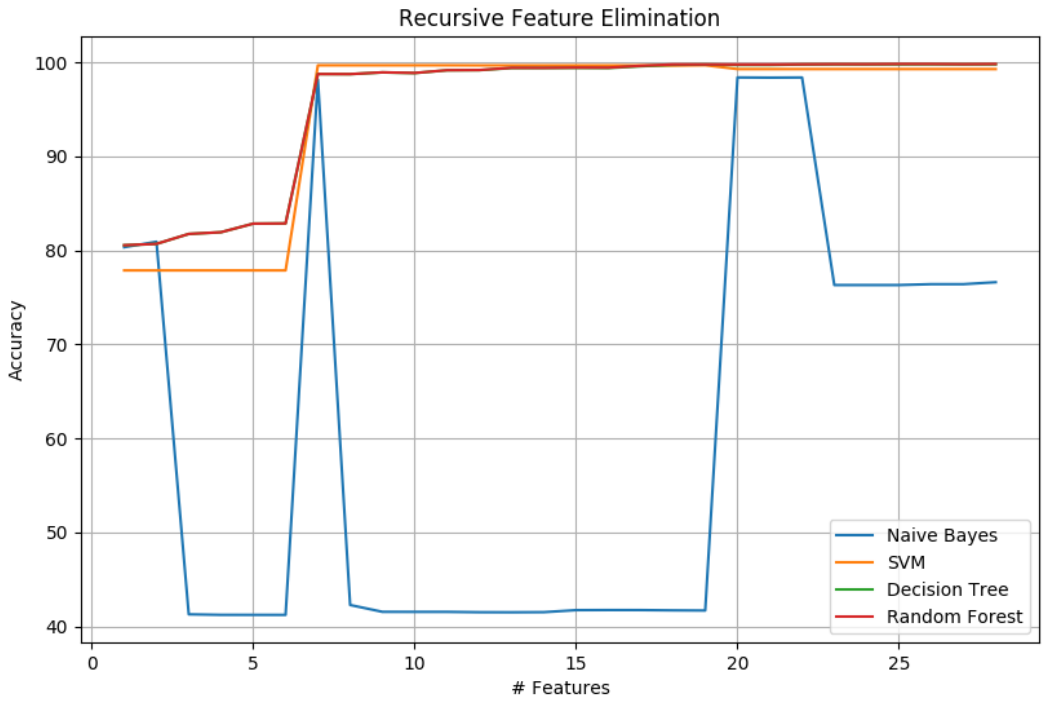

Figure 6. 3 *Accuracy variation for different classifiers with the number of features used from the sorted Recursive Feature Elimination list*

Finally, Table (6.4) and Figure (6.4) show the results obtained when the order of features is selected using the novel Weighted Ranked Feature Selection (WRFS) algorithm.

# Features	Naïve Bayes	SVM(10k), k=5	Decision Trees	Random Forest (10)
1	80.79143385	99.01021489	96.9942554	96.99992314
2	16.67607741	99.94001	98.75146827	98.75713602
3	76.27444422	99.52001	98.80855098	98.75956503
4	76.2040018	**99.42001**	**99.7983896**	99.71134855
5	76.20056065	99.42001	99.76903835	**99.83077662**
6	75.32913865	99.42001	99.76843108	99.81255877
7	75.32873381	99.42001	99.77086017	99.79515059
8	**76.57686135**	99.28001	99.72835175	99.77632542
9	76.57686135	99.28001	99.73381762	99.78199321
10	76.57908798	99.28001	99.74940364	99.79697236
11	76.57908798	99.28001	99.74677218	99.76377536
12	76.57908798	99.28001	99.75142782	99.80081836
13	76.5879945	99.28001	99.7404971	99.78138592
14	76.5879945	99.28001	99.74920115	99.82612093

15	76.60277123	99.28001	99.76539468	99.79434087
16	76.60317607	99.28001	99.75345203	99.78644646
17	76.60317607	99.28001	99.73806801	99.82106047
18	76.60479544	99.28001	99.75952541	99.80425952
19	76.60479544	99.28001	99.75810839	99.80851053
20	76.60459302	99.28001	99.75102368	99.81377329
21	76.60459302	99.28001	99.75972777	99.80810542
22	76.60520028	99.28001	99.74980905	99.81761934
23	76.60499786	99.28001	99.74373604	99.80405708
24	76.60499786	99.28001	99.7532498	99.81984593
25	76.60600997	99.28001	99.73280528	99.84879166
26	76.60600997	99.28001	99.77895682	99.82591815
27	76.60600997	99.28001	99.78098059	99.81620224
28	76.60600997	99.28001	99.75871471	99.81903612

Table 6. 4 *Accuracy (%) results for different classifiers based on the list returned by Weighted Ranked Feature Selection technique*

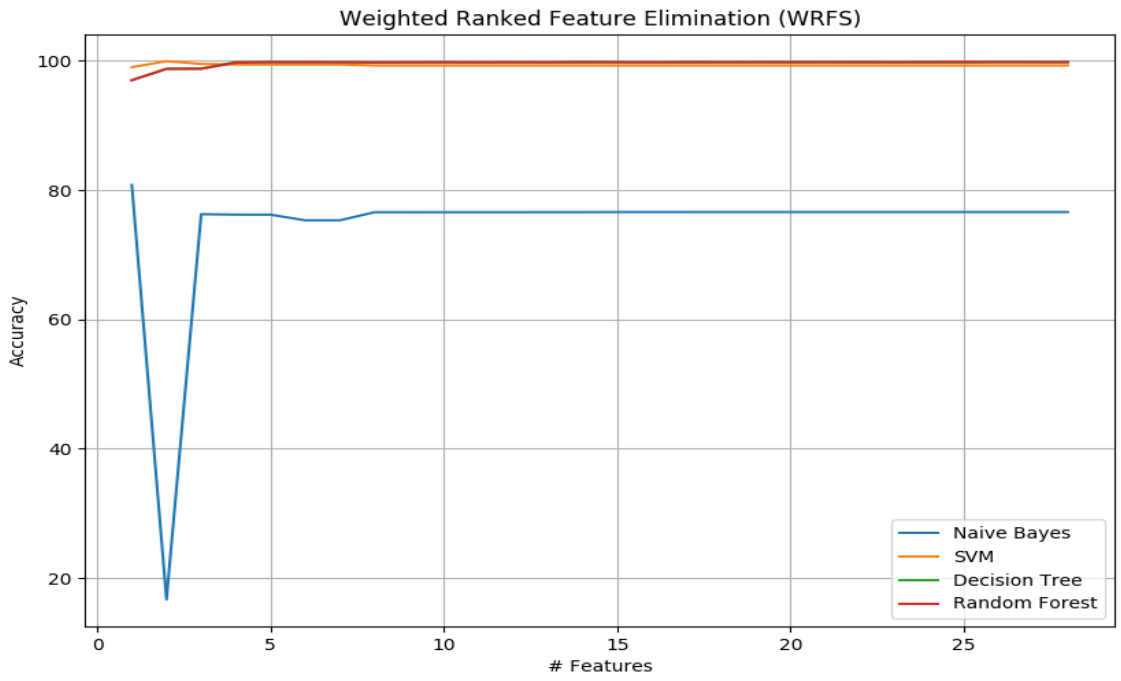

Figure 6. 4 *Accuracy variation for different classifiers with the number of features used from the sorted WRFS list*

The results show that when features are sorted using the WRFS algorithm, we only need the top 8 features to detect a DDoS attack using Naïve Bayes with ~76.58% accuracy. This is a significant improvement compared to the previous three approaches where the number of features required were 24, 25 and 20 to detect an attack using the Naïve Bayes classifier. The top 4 features are required to detect an attack with high accuracy using SVM. In the previous three approaches, the number of features chosen for classification were 23, 17 and 7 for list of features returned by Information Gain, Chi-Squared and RFE respectively. Using the WRFS list, only top 4 features again are required to detect an attack using the Decision Tree classifier. 23, 16, 11 are the number of features chosen for the Decision Tree classifier when the feature selection is done using Information Gain, Chi-Squared and RFE respectively. Finally, WRFS also performs better when the classification is done using Radom Forest. The accuracy results show that we only need the top 5 features from the list returned by WRFS to achieve an accuracy of ~99.83%, an improvement from 23, 16 and 11 features needed to achieve the same accuracy percentage when the feature selection is done using Information Gain, Chi-Squared and RFE respectively.

Table (6.5) and Table (6.6) show the precision and recall values and the confusion matrix values for all the 16 models created using the optimized number of features from the sorted lists returned by each feature selection technique. The optimized number of features is shown alongside each classification algorithm based on the accuracy results from before.

Feature Selection	Classification	Precision	Recall
Information Gain	Naïve Bayes (24)	98.186	98.441
	SVM (23)	-	-
	Decision Tree (23)	99.779	99.733
	Random Forest (23)	99.919	99.906
Chi-Squared	Naïve Bayes (25)	97.288	97.944
	SVM (17)	-	-
	Decision Tree (16)	99.510	99.332
	Random Forest (16)	99.373	99.311

RFE	Naïve Bayes (20)	97.391	97.975
	SVM (7)	-	-
	Decision Tree (11)	99.169	99.357
	Random Forest (11)	99.243	99.429
WRFS	Naïve Bayes (8)	70.984	61.528
	SVM (4)	-	-
	Decision Tree (4)	99.775	99.807
	Random Forest (5)	99.927	99.428

Table 6. 5 *Precision and Recall values for models created using the optimized number of features*

Feature Selection	Classification	TN	FP	FN	TP
Information Gain	Naïve Bayes (24)	**8945**	**136**	**253**	**15367**
	SVM (23)	-	-	-	-
	Decision Tree (23)	**9039**	**42**	**18**	**15602**
	Random Forest (23)	**9057**	**24**	**17**	**15603**
Chi-Squared	Naïve Bayes (25)	**8961**	**120**	**436**	**15184**
	SVM (17)	-	-	-	-
	Decision Tree (16)	**8978**	**103**	**27**	**15593**
	Random Forest (16)	**8991**	**90**	**30**	**15590**
RFE	Naïve Bayes (20)	**15211**	**409**	**130**	**8951**
	SVM (7)	-	-	-	-
	Decision Tree (11)	**15493**	**127**	**44**	**9037**
	Random Forest (11)	**15500**	**120**	**35**	**9046**
WRFS	Naïve Bayes (8)	**3672**	**11948**	**41**	**9040**
	SVM (4)	**1965**	**35**	**0**	**0**
	Decision Tree (4)	**15588**	**32**	**18**	**9063**
	Random Forest (5)	**15607**	**13**	**4**	**9077**

Table 6. 6 *Confusion Matrix*

For features sorted using information gain, the confusion matrix values reflect a high percentage of True Positive and True Negative values and a very small fraction of False Positives and False Negatives. This is true for Naïve Bayes, Decision Trees and Random Forest. The confusion matrix values for Decision Tree and Random Forest are observed to be very close. This is consistent with the accuracy results of these classifiers as shown in Table (6.3) where the variation of accuracy follows closely with the change in the number of features and can leads to a conclusion that Decision Tree and Random Forest perform very closely for our problem statement and are a good choice of classifier for early detection of a DDoS attack. The confusion matrix values corresponding to SVM is left blank because the Scikit-learn's confusion matrix function needs more than one value to unpack and calling that function with SVM throws that error. This is because SVM is run only on 10,000 network packets compared to half a million packets for Naïve Bayes, Decision Trees and Random Forest and this is not diverse enough data for the confusion matrix function and therefore it fails to unpack.

Similarly, when features are sorted using Chi-Squared, the results are quite like the results obtained from Information Gain. SVM fails to unpack again and Decision Tree and Random Forest show a comparable measure of the confusion matrix values with low FP, FN and high TP and TN. When the features sorted using RFE are used for classification by the four classifiers, Naïve Bayes shows high TN and relatively less TP compared to the previous two feature selection techniques. The values for FP and FN remain low. The metrics for Decision Tree and Random Forest again show a similar trend but in the reverse order. This time, the TN are high compared to the TP. Finally, features sorted using WRFS perform poorly for Naïve Bayes classifier with very high FP and 3672 and 9040 TN and TP respectively. This time SVM unpacks but the results returned cannot be used to make any conclusion because the FN and TP are 0. This is most likely because of the pruned dataset used for training the SVM model. Decision Tree and Random Forest show comparable performance again but with a significant improvement in terms of low FP and FN.

Based on the confusion matrix, we can say that Random Forest or Decision Tree should be used as the classifier if the features are sorted using the WRFS technique.

6.2 Simulations

To test the proposed model on real-time network traffic, a simulation environment was set-up within the University of Calgary network. The proposed application captures real-time network traffic and predicts the presence or absence of an attack based on the user-selected model through a web application interface.

The set-up consists of a 64-bit, 12GB RAM, Ubuntu machine, a TP-Link AC3150 Gigabit Router, MacBook Air Laptop and another 64-bit, 8GB Windows machine to send attack traffic. The Router is connected to the University network using a LAN cable and the Ubuntu Machine, MacBook Air and Windows Machine is connected to the subnetwork created by the router through Ethernet or wirelessly. The set-up is shown in Figure (6.5). This was done to create a Virtual Private Network (VPN) to contain the attack simulations within the sub-network. Now, each of the three machines are assigned an internal IP address by the router. This is shown in Table (6.7).

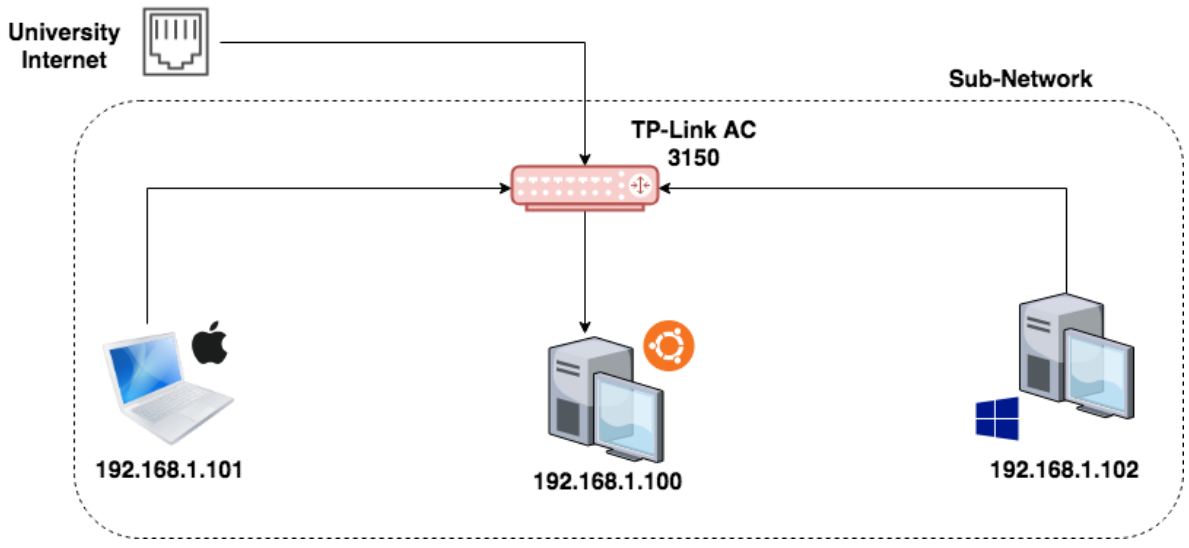

Figure 6. 5 *Simulation test-bed setup*

Host and Bots	Assigned IP
Ubuntu Machine (Host)	192.168.30.100
MacBook Air (Bot 1)	192.168.30.101
Windows Machine (Bot 2)	192.168.30.102

Table 6. 7 *Internal IP addresses assigned by the router to the machines used in the simulation*

The Ubuntu Machine is the host running the DDoS detection tool whereas the other two machines (bots) run a software called Low Orbit Ion Cannon (LOIC) to send UDP/TCP/HTTP attack traffic to the Ubuntu Machine. Low Orbit Ion Cannon (LOIC) is an open-source network stress testing and denial-of-service attack application, written in C#. LOIC was initially developed by Praetox Technologies, but was later released into the public domain, [28] and now is now hosted on several open source platforms. [29] [30]. LOIC can be used to perform both DoS and DDoS attack (depending on the number of people using the software to target the same server) on a target by flooding it with TCP, UDP or HTTP packets with the aim of disrupting the services. The LOIC application is shown in Figure (6.6).

Figure 6. 6 *Low Orbit Ion Cannon (LOIC) application*

The DDoS detection application running on the host captures the incoming traffic at 192.168.30.100 from the socket and processes the packets to extract the 28 features which

are consistent with the dataset used to train the models. The incoming packets and packet rate is also tracked using Wireshark. The incoming packets after processing are also stored in a continuously updating csv file. The proposed application reads from the csv file and uses a sliding window technique to make windows of 100 packets each in real-time. Each window is pre-processed to convert categorical variables to numeric values and then the window is normalized based on the same max-min rule used during training. The next step depends on the user-selection from the web-application.

The home page of the web-application is shown in Figure (6.7). The user can select the feature selection technique and classification algorithm to use and the application picks the pickle of the trained model for this combination. All the models are optimized based on the number of features which gives the maximum accuracy results for the selected feature selection technique. After the selection is made, the user can click on the 'Start Capture' button which starts capturing packets in real-time. As discussed above, the sliding windows of 100 packets each are formed and normalized. Based on the user selection, the corresponding pickle is chosen which starts predicting every packet, for every window.

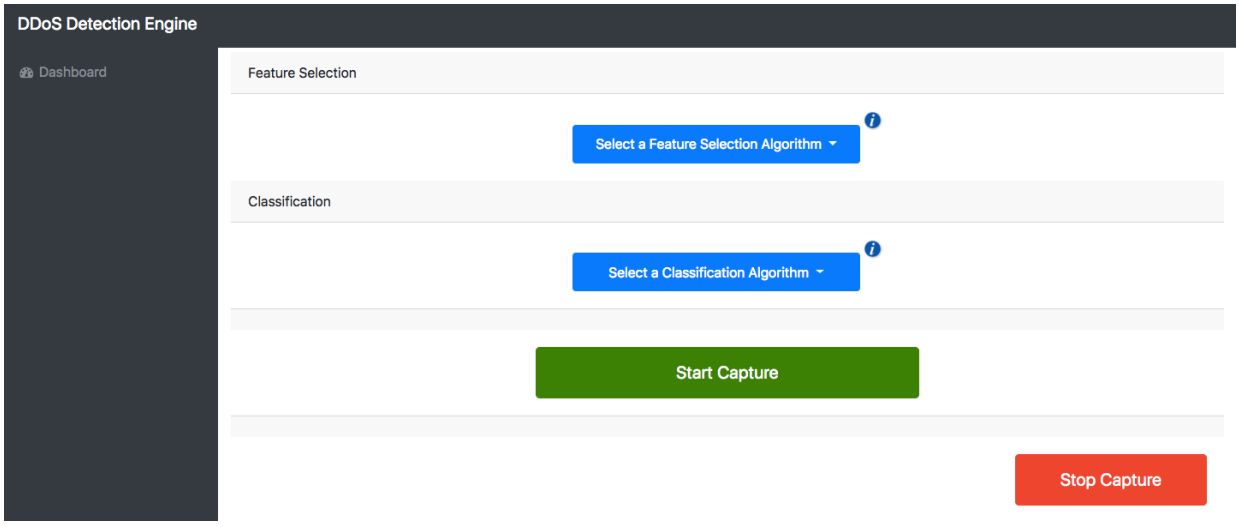

Figure 6. 7 *Homepage of the web-application*

Each window is categorized as either an attack window or a normal window based on a pre-defined threshold of 50 packets. This was done to prevent false positives which can be

unusually high in case of a DDoS attack because of the nature of these attacks. If more than 50 packets in a window are categorised as attack packets, then the entire window is declared as an attack window. But having an attack window or two is not a string indication of an attack because this could be due to surge traffic as well. Therefore, another threshold is defined which is based on the number of windows which needs to be categorized as attack windows for the system to declare an attack. This threshold value is based on common trends in DDoS attacks over the past 10 years. A typical DDoS attack lasts between 10-20 minutes with an average traffic of 200-300 Gbps during that time. Based on this information, the second threshold was chosen to be a time-based threshold in which if the windows are categorised as attack windows for more than 10 minutes, then the application concludes an attack. The attack statistics in the form of a line chart showing attack instances, bar chart showing the number of attack packets and normal packets and a pie chart showing the confusion matrix values are shown to the user through the web application.

As part of the simulations, attack traffic mixed with normal traffic was sent to the Ubuntu machine at 192.168.30.100 running the DDoS detection tool. 12 hours of regular traffic followed by 9 hours of attack traffic and again followed by 12 hours of normal traffic was simulated. The As part of the attack traffic, the LOIC tool was set to send HTTP packets with 10 threads at the maximum rate possible. The result of this simulation instance is shown in Figure (6.8-6.9).

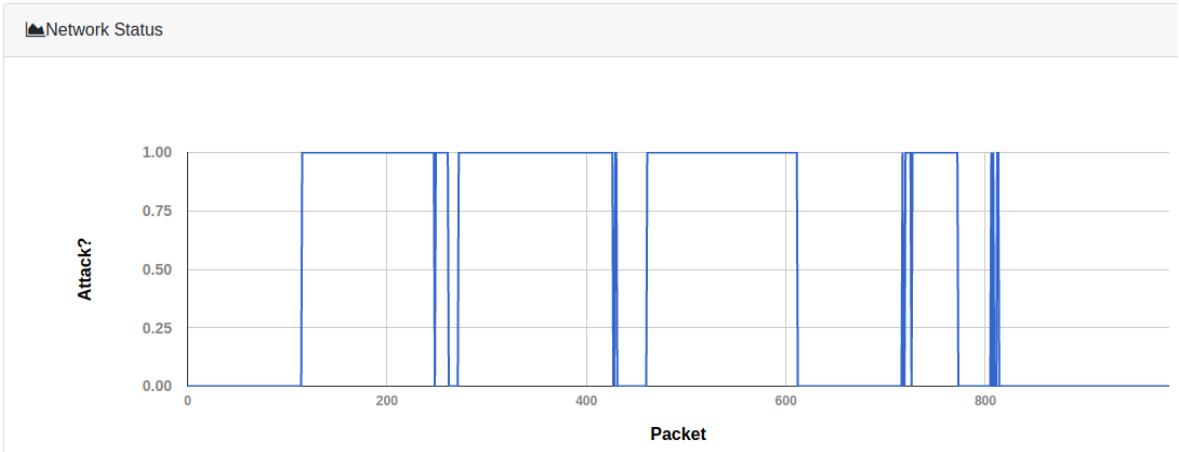

Figure 6. 8 *Attack instances in the 24-hour simulation period*

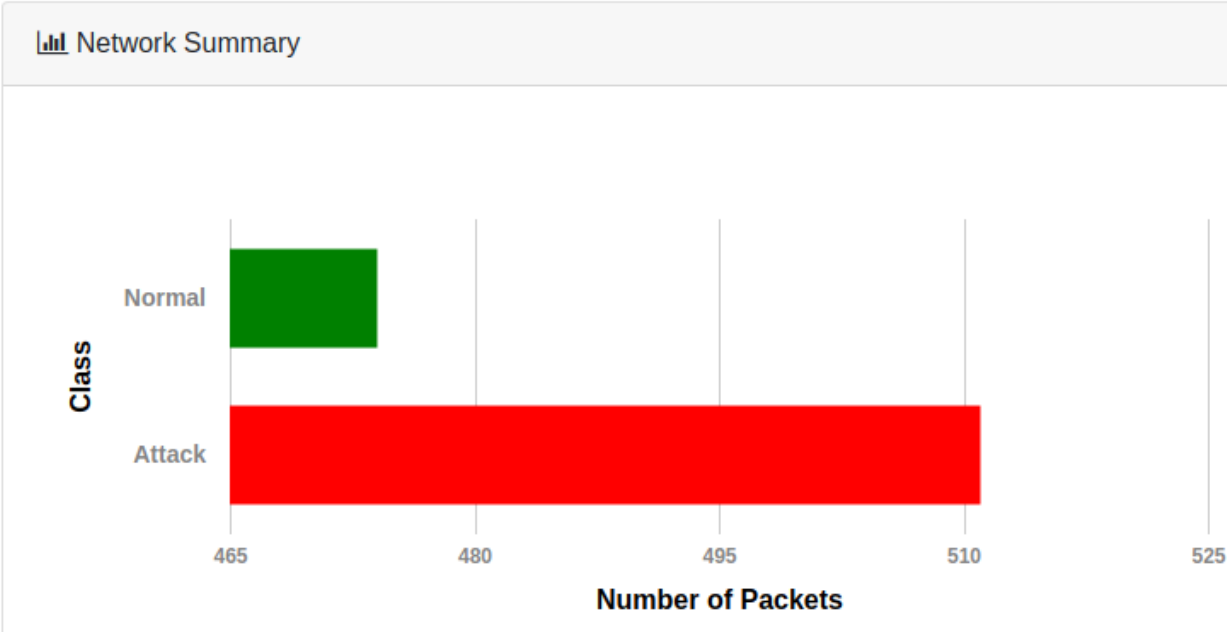

Figure 6. 9 *Number of attack packets vs normal packets during the simulation time-frame*

The x-axis of the line chart shows the incoming window with increasing time and the y-axis takes the value 0 in case the window does not have an attack and 1 in case of an attack. We can see that during normal activity, the proposed and suggested model predicts normal traffic with only a few misclassifications and predictions show a sudden surge in traffic starting from packet 150,239 onwards till packet 1,156,982. This was only categorized as attack after observing consistent high traffic for 10 consecutive minutes. The line chart drops again from packet 1,156,982 onwards. This is consistent with the traffic sent. We can also see through the bar chart that there were 5,132,451 incoming attack packets and 2,043,010 normal packets. This proves the efficiency and robustness of detecting a DDoS attack using as few attributes as possible without compromising in the efficiency of detection.

6.3 Comparison with baseline approaches

In this section, a comparison study is done by comparing the proposed approach with other classification-based DDoS detection techniques. Some of these baseline approaches are also discussed in the related work section of this thesis. The comparison is done based on the

number of features used by a model to detect a DDoS attack while maintaining a high accuracy of detection. The three commonly used datasets used by researchers working on DDoS detection are KDD Cup Dataset, CAIDA dataset and DARPA dataset. Each of these datasets have different number of attributes based on the level at which information is extracted form a network packet. The number of attributes in each of the datasets is shown in Table (6.8).

Dataset	Number of features
KDD Cup 1999 Dataset [20]	41
CAIDA DDoS Attack 2007 Dataset [31]	6
DARPA Intrusion Detection Dataset [32]	6

Table 6. 8 *Number of features in Datasets widely used for DDoS detection*

Majority of the researchers chose one of these three datasets to tackle the highly pervasive problem of DDoS attacks. The choice of dataset usually depends on the research question and the level of information needed because CAIDA provides a very broad overview of a network packet such as source IP, destination IP, protocol, etc. whereas KDD Cup dataset provides a drill-down view of a packet such as source bytes, urgent packets, duration, etc.

Classification accuracy is defined as a percentage and is the number of correctly classified packets from the total number of packets. It is represented in terms of TP and TN as shown in Equation (29).

$$\text{Classification Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} * 100\ \% \qquad \textbf{[29]}$$

Table (6.9) shows comparison of classification accuracy and the number of features used across different models using a filter-based feature selection technique and a classifier. Our best model which uses the novel WRFS feature selection technique with the Random Forest classifier is used for comparison with the baseline approaches. To keep the comparison ground fair, we have only used the approaches which use the KDD'99 dataset to build their models.

#	Approach	Classifier	Number of features	Classification Accuracy (%)
1	Information Gain	Random Forest	23	99.68
2	Chi-Squared	Decision Tree	16	99.39
3	CFS [33]	GA	8	76.2
4	CSE [33]	GA	15	75.6
5	CFS, CONS and INTERACT [34]	HNB_PKI_INT	7	93.72
6	New Medoid Clustering Algorithm [35]	k-Medoid	41	96.38
7	Gradual feature removal [36]	Cluster based, Ant Colony, SVM	19	98.62
8	Linear correlation based FS [37]	C4.5	17	99.1
9	EMFFS [38]	J48	13	99.67
10	Proposed approach using WRFS	Random Forest	**5**	**99.83**

Table 6. 9 *Performance comparison on Accuracy and Number of Features for different approaches which use the KDD'99 dataset*

Upon comparison with similar research on DDoS detection using the KDD'99 dataset, it was observed that the proposed approach which uses only 5 features from the sorted list of features returned by WRFS is able to detect a DDoS attack with an accuracy of 99.83% which is highest amongst all the other approaches. It is also important to note that the accuracy of the proposed model does not fall below 99.83% even when the number of features are

incrementally increased, unlike some of the models implemented in this research where the classification accuracy shows an erratic pattern before reaching a steady value (Figure (6.1-6.3)).

The three categories of classifiers used for the comparison study are Genetic Algorithms (GA), Classification and Clustering-based techniques and we observed that the Genetic Algorithm based classifiers implemented in [33] perform poorly and achieve an accuracy of only ~76%, which is the lowest amongst all. Clustering based approaches achieve a high accuracy of ~98% but this comes at an expense of using more number of features. Finally, classification based approaches are seen to perform the best with an average accuracy of ~99% and also using relatively less number of features from the dataset.

Section 7: Conclusion and Future Work

In this thesis, we proposed an approach to detect a DDoS attack using a Machine Learning approach. The proposed approach was also tested on real-time network traffic to corroborate the performance of the models to detect a DDoS attack using few number of attributes from the network packet. Four different feature selection algorithms were implemented including the novel Weighted Ranked Feature Selection (WRFS). The sorted list of features returned by each of these algorithms were cross coupled with four classification algorithms. Each classifier was trained and tested on every list. Features were incrementally increased starting from 1 till 28 and accuracy was noted. This allowed us to find out the perfect balance between the number of features used and the accuracy of detection of a DDoS attack. The proposed models were stored in pickles and tested on real-time network traffic through a simulation environment which was set-up within the University of Calgary network. The proposed model performed as expected and used only a fraction of the attributes from a network packet to detect the simulated DDoS attack with a ~99.8% accuracy of detection.

A comprehensive tool should have a two-fold objective – detection and mitigation of a DDoS attack. In the past decade, the power of a DDoS attack has increased exponentially and has forced organisations to use third party DDoS detection and mitigation services. Handling a DDoS in house is not feasible for an organisation just because of the amount of resources required to set-up such a system. Therefore, upon detection of a DDoS attack, the organisation re-routes the entire traffic to an external server with a different organisation which is equipped with DDoS mitigation capabilities. The job of the mitigation service is to mitigate any attack by either distributing the incoming traffic or by dropping malicious packets and allowing the regular traffic to flow through.

As part of the future work of this thesis, a mitigation engine is planned which would complete our system and can provide a holistic approach towards detecting and mitigating a DDoS attack. The existing system would work as-is and detect a DDoS attack but upon detection of

an attack, the entire traffic would be re-routed to another application on an external server. This application would receive all the packets and try to mitigate the attack by keeping the target server up at all times without the disruption of service for legitimate users. This can be done by dropping all the packets which have been categorised as attack packets and forwarding only the normal packets to the target server. This may cause some delays on the server side to serve the request but can defeat the attacker by keeping the server up and running.

A feedback loop which would re-train the model as regular intervals is also planned. The new incoming traffic will be used to re-train the model on existing and new data and the old pickles will be updated by new pickles especially after an attack instance is observed. This will ensure that the proposed application can update itself with the change in trends of DDoS attacks.

Bibliography

[1] M. E. M. Wesam Bhaya, "A Proactive DDoS Attack Detection Approach," *Journal of Next Generation Information Technology,* pp. 36-47, 2014.

[2] A. K. P. Devi, "A Security framework for DDoS Detection in MANETs," *Telecomminication and Computing,* pp. 325-333, 2013.

[3] M. James, "Data Clustering Using Entropy Minimization".

[4] D. K. B. J. K. K. N. Hoque, "Botnet in DDoS Attacks: Trends and Challenges," *IEEE Communications Surveys and Tutorials,* 2015.

[5] H. K. B. R. a. A. T. R. B. Blazek, "A novel approach to detection of denial-of-service attacks via adaptive sequential and batch-sequential change-point detection methods," *IEEE Systems, MAN, and Cybernetics Information Assurance and Security Workshop,* pp. 220-226, June 2001.

[6] R. R. B. İlker Özçelik, "Cusum - entropy: an efficient method for DDoS attack detection," *4th International Istanbul Smart Grid Congress and Fair (ICSG),* 2016.

[7] J. Quinlan, "C4.5: Programs for Machine Learning," *Morgan Kaufmann Publishers,* 1993.

[8] P. a. N. T. Clark, "The CN2 Induction Algorithm," *Machine Learning Journal 3(4),* pp. 261-283, 1989.

[9] R. S. J. a. C. P. Hanson, *Bayesian Classification Theory. Technical Report,* 1991.

[10] "kddcup99.html," [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. [Accessed 15 September 2017].

[11] N. M. B. Agarwal, "Optimal feature selection for sentiment analysis," *14th International Conference on Computational Linguistics and Intelligent Text Processing, Samos, Greece,* pp. 13-24, 2013.

[12] S. S. A. S. Z. Baig, "GMDH-based networks for intelligent intrusion detection," *Engineering Applications of Artificial Intelligence, 26(7),* pp. 1731-1740, 2013.

[13] A. A. M. J. H. S. M. Moradkhani, "A hybrid algorithm for feature subset selection in high-dimensional datasets using FICA and IWSSr algorithm," *Applied Soft Computing,* pp. 119-135, 2015.

[14] R. P. M. Y. a. N. J. R. Miao, "The dark menace: Characterizing network based attacks in the cloud," *ACM Conference on Internet Measurement Conference,* pp. 169-182, 2015.

[15] T. Z. R. B. L. Zecheng He, "Machine Learning Based DDoS Attack Detection From Source Side in Cloud," *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud),* 2017.

[16] H. M. Michael Whitman, Principles of Information Security, Course Technology; 4 edition, 2011.

[17] “understanding-security-osi-model-377,” 21 March 2018. [Online]. Available: https://www.sans.org/reading-room/whitepapers/protocols/understanding-security-osi-model-377.

[18] P. Oppenheimer, Top-Down Network Design, 3rd Edition, Cisco Press, 2010.

[19] N. Krawetz, Introduction to Network Security, Charles River Media, 2007, p. 31.

[20] E. B. W. L. a. A. A. G. Mahbod Tavallaee, “A Detailed Analysis of the KDD CUP 99 Data Set,” *IEEE Symposium on Computational Intelligence in Security and Defense Applications,* 2009.

[21] G. K. N. V. C. Danny Roobaert, “Information Gain, Correlation and Support Vector Machines,” *Springer, Studies in Fuzziness and Soft Computing,* pp. 463-470, 2006.

[22] K.-K. R. C. M. D. Opeyemi Osanaiye, “Analysing Feature Selection and Classification Techniques,” *Southern Africa Telecommunication Networks and Applications Conference (SATNAC),* September 2016.

[23] H. L. L. Yu, “Feature selection for high-dimensional data: A fast correlation-based filter solution,” *Twentieth International Conference on Machine Learning (ICML-2003),* pp. 856-863, 2003.

[24] A. Stuart and K. Ord, Kendall's Advanced Theory of Statistics, Distribution Theory, Wiley; 6 edition , 2010.

[25] M. N. D. V. S. Murty, Pattern Recognition, An Algorithmic Approach, Springer-Verlag London, 2011.

[26] T. Fletcher, “Support Vector Machines Explained,” 2008. [Online]. Available: https://cling.csd.uwo.ca/cs860/papers/SVM_Explained.pdf.

[27] N. Donges, 20 March 2018. [Online]. Available: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd.

[28] Praetox, “Low Orbit Ion Cannon,” 2010.

[29] “LOIC | Free Security & Utilities software,” 17 11 2014. [Online].

[30] “NewEraCracker/LOIC · GitHub,” 22 11 2013. [Online].

[31] “The CAIDA UCSD "DDoS Attack 2007" Dataset,” 15 9 2017. [Online]. Available: http://www.caida.org/data/passive/ddos-20070804_dataset.xml.

[32] 15 9 2017. [Online]. Available: https://www.ll.mit.edu/ideval/data/.

[33] P. H. C. L. S Rastegari, “Evolving statistical rulesets for network,” *Applied Soft Computing 33,* pp. 348-359, 2015.

[34] T. M. S. S. L Koc, “A network intrusion detection system based,” *Expert Syst Appl 39(18),* p. 13492–13500, 2012.

[35] R. R. a. G. Sahoo, “A new clustering approach for anomaly intrusion detection,” *International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol.4, No.2,* March 2014.

[36] K.-K. R. C. H. A. J Peng, “Bit-level n-gram based forensic authorship analysis on social media: Identifying individuals from linguistic profiles,” *Netw Comput Appl. (Elsevier, 2016 in press),* 2016.

[37] A. H. T. K. S. B. H Eid, “Linear correlation-based feature selection for network intrusion detection model,” *Proceedings of the 1st International Conference on Advances in Security of Information and Communication Networks (SecNet),* pp. 240-248, 2013.

[38] H. C. K.-K. R. C. A. D. X. a. M. D. Opeyemi Osanaiye, “Ensemble-based multi-filter feature,” *EURASIP Journal on Wireless Communications and Networking,* 2016.

[39] M. James, “Data Clustering Using Entropy Minimization”.