Detecting DDoS Attacks Using a Hybrid Model

A Thesis Proposal
Presented to
the Faculty of the College of Computer Studies
De La Salle University Manila

In Partial Fulfillment
of the Requirements for the Degree of
Master of Science in Computer Science

by

Caychingco, Jedidiah

Gregory Cu
Adviser

October 2, 2017

Abstract

A Distributed Denial of Service (DDoS) attack can disrupt and damage businesses by preventing legitimate users from accessing its resources. Some estimate their losses to be at 500$ per minute of DDoS. Being able to detect these attacks can allow security analysts to apply the proper techniques in order to mitigate it. With this, this study aims to use a hybrid method in order to detect DDoS attacks. During the first stage, a machine learning algorithm is first used to differentiate normal and attack traffic. If the traffic has been deemed to be part of a DDoS attack, it is passed to the second stage. The second stage involves using another machine learning algorithm in order to determine whether it is part of a low rate or high rate DDoS attack. The created model will be evaluated using accuracy, precision, recall, f-score, and the Kappa statistic.

Keywords: DDoS detection, Anomaly detection, Supervised machine learning, Hybrid model

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ACK	Acknowledge
ANFIS	Adaptive Neuro-Fuzzy Interface System
ANN	Artificial Neural Network
AR	Auto Regressive
CAIDA	Center for Applied Internet Data Analysis
CUSUM	Cumulative Sum
DARPA	Defense Advanced Research Projects Agency
DDoS	Distributed Denial of Service
DoS	Denial of Service
DT	Decision Tree
FCM	Fuzzy C-means clustering
FIN	Finish
FN	False Negative
FP	False Positive
FTP	File Transfer Protocol
Gbps	Gigabits per second
GMM	Gaussian Mixture Model
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
KDD	Knowledge Discovery and Data Mining
KNN	K-Nearest Neighbours
LGP	Linear Genetic Programming

MARS	Multivariate Adaptive Regression Splines
MLP	Multilayer Perceptron
NB	Naive Bayes
NS2	Network Simulator Version 2
NS3	Network Simulator Version 3
R2L	Root to Local
RUDY	R-U-Dead-Yet
SA	Simulated Annealing
SOM	Self Organising Map
SVM	Support Vector Machines
SYN	Synchronise
SYN-ACK	Synchronise-Acknowledge
TCP	Transmission Control Protocol
TN	True Negative
TP	True Positive
U2R	User to Root
UDP	User Datagram Protocol

Chapter 1

Research Description

1.1 Overview of the Current State of Technology

There are several attacks that can cause great damage to networks such as: Man in the Middle, Session Hijacking, Cross Site Scripting, and Denial of Service (DoS). The last attack, DoS, is considered to be the most deadly among them (Rao & Rao, 2011).

A DoS is a type of attack that aims to consume a large amount of resources from the victim in order to prevent legitimate users from being able to use the victim's services (Siris & Papagalou, 2006; C. Li et al., 2015; Liu, 2009). Attackers make use of different methods in order to make services unavailable. For example, there are attacks catered towards: congesting network resources, exhausting CPU memory, making databases inaccessible, and so on. There are also different methods to conduct these attacks (Liu, 2009). Furthermore, DoS attacks are popular since it is easy to carry out. The attacker merely needs to generate enough traffic in order to overwhelm the server (Hinchliffe, 2016). The effect of this attack can range from the users of the victim's website to experience minor inconveniences to great financial losses for companies that rely on online availability when conducting businesses (Peng et al., 2004).

A Distributed Denial of Service, or DDoS, is a variation of the DoS attack. A DDoS attack relies on numerous compromised hosts performing a DoS attack on a single victim (K. Li et al., 2009).

Arbor networks surveyed different service providers as well as people from the enterprise, government, and educational sector on October 2016. They were

able to get 356 responses overall. They conducted this survey with the intention identifying the trends in the threats facing the different organisations. Among their results, they reported an increase in the frequency and strength of DDoS attacks. For service providers, the largest attack in 2016 peaked at 800 Gigabits per second (Gbps), a 60% increase from the 500 Gbps peak that occurred on 2015. A third of their respondents stated that attacks that reach 100 Gbps, while an eighth reported attacks that are above 200 Gbps. The report also states that 53% of respondents experience at least 51 DDoS attacks per month, a 44% increase from 2015 (Arbor Networks, 2017).

With regards to the Philippine setting, at least 68 government web sites were targeted by a DDoS attack on July 12, 2016. The attack continued and surged again on the following day. Among the websites affected were: The Department of National Defense, Metro Manila Development Authority, and Bangko Sentral ng Pilipinas (Central Bank of the Philippines) (Sta. Ana, 2017).

Arbor networks also surveyed how DDoS attacks damage their respondents. The most common response at 48% was reputation damage. With regards to the cost of damages, almost 60% of the respondents stated that the downtime caused by DDoS attacks cause them at least 500$ per minute (Arbor Networks, 2017).

Despite of the damages caused by DDoS attacks, it is difficult to detect them when they occur. This is due to its two characteristics. The first is that it is difficult to differentiate between DDoS attack traffic and normal traffic. The second is that the source of the traffic is difficult to discover due to the distributed nature of the attack (You, 2007).

There are two types of DDoS attacks: a low rate DDoS attack, and a high rate DDoS attack. They are similar to different characteristics of normal traffic. A low rate attack is similar to legitimate traffic while a high rate attack is similar to a flash crowd (Bhuyan et al., 2014).

Over the years, different researches have made use of different types of DDoS detection approaches. Their approaches consist of the following: non-hybrid approaches (C. Li et al., 2015; Siris & Papagalou, 2006; Korczynski et al., 2011), hybrid approaches (Barati et al., 2014; Chen et al., 2013; Hussain et al., 2015), a stacking approach (Seo et al., 2005; Depren et al., 2005), and a combination of approaches (Kumar & Selvakumar, 2013; Boroujerdi & Ayat, 2013). These defences focus on detecting different types of DDoS attacks using different thresholds, machine learning algorithms, and features. An in depth analysis of the related researches can be found on Chapter 2.

Non-hybrid approaches involve using a single classifier in detecting DDoS at-

tacks. Conversely, hybrid approaches involves applying classifiers in sequence of one another in detection. The idea of ensemble classifiers is to build a predictive model that integrates multiple base models with the intention of improving the overall performance. Some examples of ensemble classifiers are boosting, bagging, and stacking. Boosting is a general method that aims to improve the performance of a weak learner. It works by training a weak learner on different subsets of the training data. The classifiers produced by the weak learners are then combined in order to achieve a greater accuracy compared to an individual weak learner. In bagging, each classifier is trained with different subsets of the dataset. An instance is classified according to the most common prediction of the classifiers. Stacking involves training a meta classifier from the outputs of the base classifiers. Using a combination of approaches refer to those that cannot be placed under the previous categories since they combine different types of approaches instead of using only one. A more detailed explanation of the different types of classifiers can be found on Chapter 3.

However, these researches have limitations with regards to their current relevance and the adaptability of their methods. For example, Hoque et al. (2016) and Siris & Papagalou (2006) only made use of statistical methods. Using machine learning algorithms may be able to perform better since they are more adaptive. Hussain et al. (2015); Boroujerdi & Ayat (2013) uses the NSL KDD dataset. But due to the dataset's age, it may no longer be an accurate representation of current network attacks. Moreover, the majority of related research do not detect more than 1 general category of DDoS attacks. Lastly, the majority of approaches use a non-hybrid approach. Using a hybrid, stacking, or a combination of approaches would aid in classification by allowing each machine learning algorithm to compensate for each other's weaknesses.

1.2 Problem Statement

There are a few studies that try to classify both high rate and low rate attacks. However, they only use statistical methods. These approaches are not as adaptive as machine learning algorithms. Lastly, the majority of these studies make use of old datasets, which may no longer be able to represent the current internet situation.

1.3 Research Objectives

1.3.1 General Objective

To build a hybrid model that is able to differentiate normal traffic from DDoS traffic and identify the type of DDoS attack as either low rate or high rate.

1.3.2 Specific Objectives

1. To study the different methodologies in detecting low and high rate DDoS attacks
2. To gather low and high rate DDoS datasets as well as a dataset of normal traffic
3. To preprocess the datasets to make them compatible with one another
4. To identify and make use of the best combination of features that can be used to detect normal and attack traffic
5. To identify and make use of the best combination of features that can be used to detect low and high rate DDoS attacks
6. To construct and evaluate the hybrid model

1.4 Scope and Limitations

Only researches that detect low rate and high rate DoS or DDoS attacks have been studied. This includes studies that detect a specific type of DDoS attack instead of its general category. For example, studies that aim to detect SYN flood attacks were also studied. In addition, researches that detect different types of attacks were also studied as long as low rate and high rate DoS or DDoS attacks are among what they detect.

The datasets used were based on the ones used by prior research. The only datasets that were used were the NSL KDD dataset and the CAIDA Internet Traces 2013 dataset. The NSL KDD dataset contains 4 attacks: DoS, R2L, U2R, and probes. However, only the DoS attacks was used. The CAIDA dataset was used for normal data. Since a dataset for low rate DDoS data was not obtained,

data in that category was obtained by creating a network environment and simulating the attacks there. The network environment is implemented virtually. It is composed of the following components: attacking hosts, a switch, a router, sniffers, and a victim server.

The features available in the NSL KDD Dataset will be the ones to be extracted on the other datasets. If a feature cannot be extracted from the other datasets, it will be removed from the NSL KDD dataset. This is done so that the datasets will be compatible with one another. Afterwards, the datasets will be partitioned into three parts: testing, training, and validation. Lastly, since there are different instances of each class available, each class would be sampled in such a way that the machine learning algorithm would process an equal number of instances for each class (i.e. there is an equal number of normal and attack instances, and an equal number of low rate and high rate instances).

The features used will be based on or similar to the features used by the previous researches. Experiments using the different features and machine learning algorithm will be used to determine the best features.

Only supervised machine learning algorithms will be used in creating the hybrid model. The machine learning algorithms that are considered are: Naive Bayes, Support Vector Machines, Artificial Neural Network, K-Nearest Neighbours, and Decision Tree. The hybrid model created can only classify the traffic as being normal or an attack during its first stage. If the classification is an attack, the hybrid model's second stage can only classify the attack traffic as either part of a high rate DDoS attack, or part of a low rate DDoS attack. The hybrid model will be evaluated using only: accuracy, precision, recall, f-score, and the Kappa statistic.

1.5 Significance of the Study

Many organisations greatly rely on their networks but DDoS attacks can greatly affect them. Detecting the type of DDoS attack that is occurring can allow security analysts to apply the proper mitigation techniques to it. This ultimately reduces the damages caused by the attack. This work is significant since experiments will be performed with various combinations of different features and classifiers to determine which features work well with which classifiers in detecting normal and attack data, and low and high rate attacks. Lastly, conducting this research will give a basis on whether the hybrid approach is effective or not in comparison to the previous works.

1.6 Research Methodology

1. **Review of related literature.** This step involved analysing several materials such as journal papers and articles related to the detection of low rate and high rate DDoS. Priority in analysis was given to material that was created up to five years ago in order to determine the current state of technology. This step was performed in order to gain knowledge of the trends and techniques related to the field. Afterwards the material was grouped together depending on the relationship of their methodologies. Furthermore, the following characteristics form the materials were also taken note of: the methodology's motivation, the features used, the datasets used, and the accuracy or performance.

2. **Dataset selection.** This phase involved determining the datasets used by the related literature and analysing why it was used. The datasets were selected based on: the features they contain, their date of creation, and how easily experiments using it can be compared with other researches. If an appropriate dataset is not found for a particular category, it would be created using a network simulation instead.

3. **Dataset preprocessing and feature extraction.** Since multiple datasets are used, this phase involves preprocessing the datasets so that they will be compatible with one another. Since the NSL KDD dataset is already processed and that the other datasets are packet capture files, the features from the other datasets will be extracted to make them compatible with the NSL KDD dataset. In the case that a feature from the NSL KDD dataset cannot be extracted from the other datasets, that feature would be removed from the NSL KDD dataset to ensure compatibility.

4. **Feature set and classifier selection.** This stage involves selecting the feature set and both the initial and the final classifier.

5. **Testing and evaluation.** This is the final stage and it involved performing tests on the classifiers in order to determine their overall performance. The classifiers were tested using different DDoS scenarios from the datasets and evaluated using the following criteria: accuracy, precision, recall, and f-score.

1.7 Calendar of Activities

Table 1.1 shows a Gantt chart containing the calendar of the activities in this thesis. In the column pertaining to a month, each asterisk (*) represents

approximately a week's worth of activity with regards to the activity on the left-most column. A hyphen (-) represents a week in which little or no work will be done with that activity. If the space does not contain any markings, it represents a scenario in which no activity of the sort will be done.

Table 1.1: Gantt Chart

Activities	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Review of related literature	-***	***-	*-*-	*-*-	*-*-	*__		
Dataset selection		-*-*	-*-*					
Feature set extraction			_**	-*-*				
Classifiers creation				*-*-	****			
Testing and evaluation					-*-*	****	*-*-	
Documentation	-***	***-	-*-*	*__	*-*-	****	****	*__

Chapter 2

Review of Related Literature

2.1 Data Analysis

The most significant challenge in evaluating an intrusion detection algorithm is the lack of appropriate public DDoS datasets. The reason for the lack of those datasets is that deep inspection of network traffic reveals sensitive information such as: confidential communication, the user's network access patterns, and so on. Having those information breached can have a disastrous effect on both the organisation and their service providers (Kumar & Selvakumar, 2013; Boroujerdi & Ayat, 2013). The datasets discussed below are the ones that are commonly used by the related literature.

2.1.1 Datasets

2.1.1.1 DARPA

The goal of the Defense Advanced Research Projects Agency (DARPA) 1998 Intrusion Detection System Evaluation was to provide and distribute the first standard corpus for evaluating intrusion detection systems. The data was created through the simulation of the local area network traffic that can be found at a United States air force facility. The simulation took nine weeks in total. Seven weeks were used to create the training dataset, and the remaining two weeks were used to create the testing dataset (Kendall, 1999).

The simulation network consists of two Ethernet segments connected to each

other through a router, and eleven machines. An illustration of the network topology is shown on Figure 2.1. The router is labelled "CISCO". The items to the left of the router represents the outside of the network, while the items to the right of the router represent the inside of the simulation network and are part of the fabricated eyrie.af.mil domain. The outside of the network contains 5 machines: a machine that generates both background traffic and automated attacks, a web server, a sniffer, and two machines that generate non-automated attacks. The inside of the network contains 6 machines: a background traffic generator, a sniffer, and victims that run different operating systems. The different operating systems of the victims are: Solaris 2.5, SunOS 4.1.4, Linux 4.2, and a Linux 5.0 that can dynamically change its IP address(Kendall, 1999).

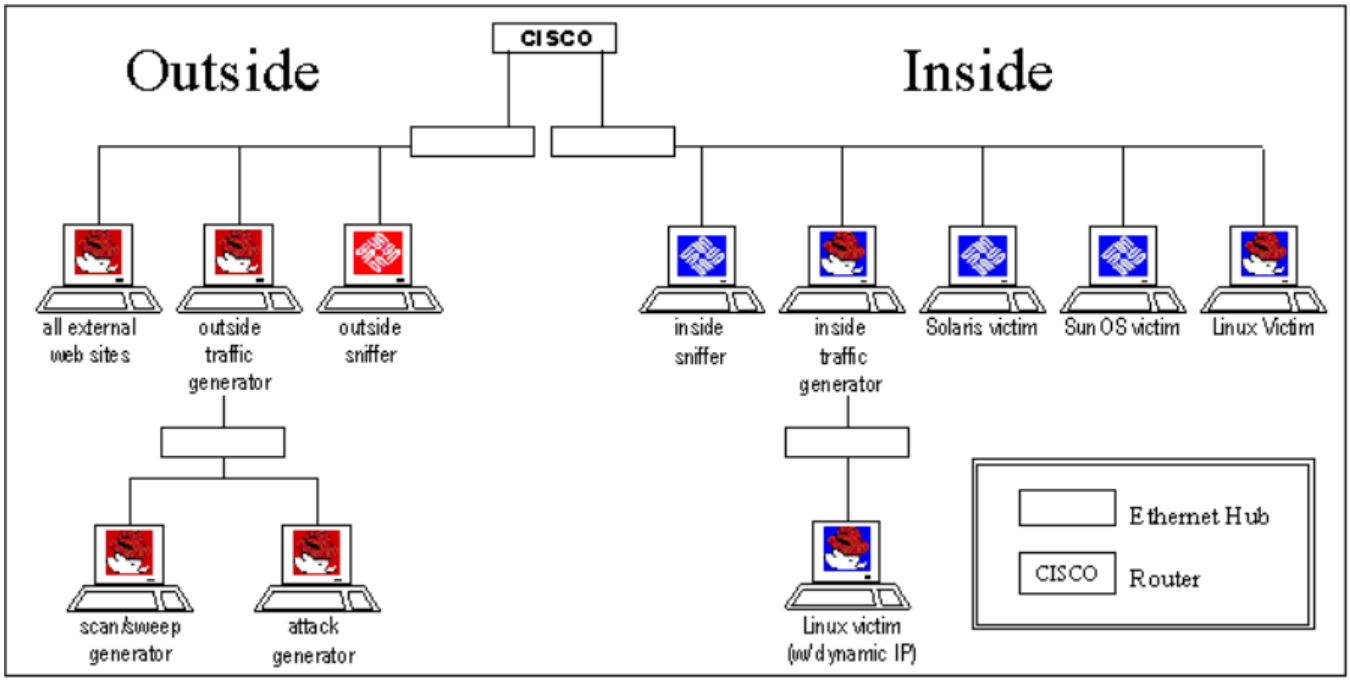

Figure 2.1: DARPA 1998 network topology

The simulation lasted for nine weeks. The first seven weeks are the training dataset while the remaining two weeks are the testing dataset. The data contains both normal traffic as well as attacks. The primary services and protocols used are: HTTP, SMTP, Pop3, FTP, IRC, Telnet, X, Sql/ Telnet, DNS, Finger, SNMP, and Time (Richard P. Lippmann et al., 1998).

With regards to the creation of background traffic, different software tools such as traffic generators and schedulers have been used. These tools function in real time. Furthermore, they also use tools that can analyse the sniffing and audit data to verify that the system ran correctly and label each TCP/IP session. Different scenarios were also created with regards to the sending and processing of

mail, FTP sessions, and Telnet sessions. The mail consist of nonsense messages, FTP sessions consist of downloading files, while within the Telnet session, different programs would be written and ran (Richard P. Lippmann et al., 1998).

The attacks in the dataset fall under four categories: denial of service (e.g. SYN flood), user to root (e.g. password guessing), remote to local (e.g. buffer overflow), and probes (e.g. port scans). Denial of service (DoS) is an attack that make exhausts the victim's resources in order to deny access from legitimate features. User to Root (U2R) is an attack in which the attacker starts with normal level access to the system, and by exploiting the system's vulnerabilities, gains root level access to the system. Root to Local (R2L) is an attack in which attackers without an account on a remote machine exploits vulnerabilities in order to gain access as a local user on that machine. Probes are attempts to gather information about a network of computers. The attacks were designed to be able to be detected by using tcpdump, BSM audit data, or both. Different scenarios were created with the execution of these attacks with some attacks extending over many days. In addition, different attacks were performed in a more stealthy manner. However, no stealthy attacks were performed for R2L and DoS (Richard P. Lippmann et al., 1998).

Here are the DoS attacks found in the dataset as well as their descriptions. The following attacks consist of high rate DDoS attacks (Lincoln Laboratory, 1999)

The Back attack consists of an attacker submitting requests with URL's containing numerous front slashes. As the server tries to process these, it will slow down and becomes unable to process other requests.

The Land attack occurs when an attacker sends a spoofed SYN packet. In this packet, the source address is the same as the destination address.

The Neptune attack is the same as the SYN flood attack. This attack involves creating a connection with the victim. However, the final ACK packet of the attacker is not sent during the three way handshake. This makes the victim allocate resources in maintaining the connection. Once the resources have been used, new connections may no longer be made.

Pod refers to the Ping of Death attack. It involves sending a large ping packet to the target. Some systems are not able to handle oversized ping packets and they may react in unpredictable ways because of this.

The Smurf attack involves the attacker sending an ICMP echo request packet to the broadcast address of an intermediary network with the source IP address spoofed to be that of the victim's IP address. Since the packet is sent to the

broadcast address of the intermediary network and that the source IP address of the packet is spoofed, the hosts of the intermediary network will send the ICMP echo reply packet to the victim. The amount of amplification this attack provides depends on the number of hosts in the intermediary network.

The Teardrop attack involves sending numerous fragmented packets to the target. The fragmented packets are created in such a way that the victim would be unable to reassemble them.

A table containing all the specific attack types, and features in the dataset can be found on Appendix A.

Another DARPA dataset is the Defense Advanced Research Projects Agency (DARPA) LLDoS dataset. It is a commonly used DDoS dataset. It has two scenarios. Scenario one (LLDOS 1.0) contains a DDoS conducted by a novice attacker. The attack is composed of 5 phases: performing an IP sweep of the target, determining the hosts that run the Solaris Sadmind application, installing the DDoS tool, and performing the DDoS attack. The second scenario (LLDOS 2.0.2) contains a DDoS attack performed by a more stealthy attacker compared to the previous scenario. This scenario also contains 5 phases similar to the first scenario (Lincoln Laboratory, 2000).

Despite its popularity, the usage of DARPA datasets has been criticised and is considered to be outdated. Yan & Zhang (2013) argued that evaluating intrusion detection systems with those datasets may give an inaccurate representation of their performance given the onset of more recent attacks, attacks against different types of machines or infrastructure due to the dataset's age.

2.1.1.2 KDD CUP

The Knowledge Discovery and Data Mining (KDD) CUP 99 dataset was prepared by Stolfo et al. (2000). It is one of the most widely used datasets for evaluating anomaly detection methods and is based from the DARPA 1998 dataset. It contains the same attacks as the DARPA 1998 dataset. Their difference is that the KDD dataset already has its features extracted. The dataset contains a training and testing set. They have a separate probability distribution of attack types. Specifically, the training set contains 22 attack types while there are an additional 15 attack types that are only present in the testing set. This dataset is an imbalanced dataset. Its training set has the following distribution ratio: 19.69% Normal, 79.24% DoS, 0.83% Probe, 0.23% R2L, 0.01% U2R (UCI Knowledge Discovery, 1999).

A connection is a sequence of TCP packets starting and ending at some well defined times. Within it, data flows to and from a source and target IP address under some well defined protocol. Each connection is labelled as either normal, or as an attack, with one specific attack type. Each connection record consists of about 100 bytes. Each record in the dataset pertains to a connection between two network hosts.

The dataset contains 41 features and can be generalised into three groups: basic TCP features, traffic features, content features, and time-based traffic features. The time-based features were defined by Stolfo et al. (2000) and examine connections in the past two seconds. The item examined depends on the category: either same host, or same service. Same host features only examine connections that have the same destination host as the current connection while same service features only examine connections that have the same service as the current connection.

Despite of the numerous features and attacks present in the KDD dataset, it also has its downsides. Tavallaee et al. (2009) states that the KDD CUP 99 dataset inherently contains problems. The first is the presence of redundant records (78% in the training set and 75% in the testing set). These cause bias in the learning algorithms and prevents infrequent records from being learned. And the second is with regards to the lack of difficulty with the dataset. The authors have used seven machine learning algorithms trained in three different ways from three different subsets of the training dataset and have found that the accuracy of all the algorithms in classifying a record correctly is 93%.

2.1.1.3 NSL KDD

With the aforementioned problems being present in the KDD dataset, Tavallaee et al. (2009) addressed them by creating a new dataset called the NSL KDD. The new dataset contains the same attacks and features as the KDD dataset. The first change was that both the training and testing dataset in the NSL KDD do not contain redundant records. In order to increase the difficulty of the dataset, the authors have included a greater number of commonly misclassified records compared to records that were correctly classified by all of their machine learning algorithms. These changes caused the accuracy of their machine learning algorithms to greatly vary. This produces a better metric for comparison compared to the original KDD dataset. Lastly, the number of records in the NSL KDD dataset is far fewer compared to the ones present in the KDD dataset. The extent of the dataset reduction allows for experiments to be ran on the full NSL KDD dataset; thus removing the need of sampling. This change the makes evaluation between

different works using the dataset more consistent and easily comparable.

2.1.1.4 CAIDA

The Center for Applied Internet Data Analysis (CAIDA) datasets is another popular dataset used by several researchers in evaluating their DDoS detection approaches CAIDA (2017). CAIDA provides numerous datasets with each containing a specific category and purpose. Moreover, they are also performing an ongoing collection of data.

The CAIDA UCSD DDoS attack 2007 contains approximately an hour of anonymised DDoS traces from August 4, 2007. The attack attempts to block access to the targeted server by consuming computing resources and by consuming all of the bandwidth of the network connecting the server to the internet. Only attack traffic to the victim and the responses from the victim are included in this dataset. Furthermore, non-attack traffic was removed as much as possible (CAIDA, 2007).

The CAIDA UCSD Conficker dataset contains three days of the Conficker worm. The samples occurred a month apart. The first day occurred on November 21, 2008 and contains the onset of the Conficker A infection. The second day occurred on December 21, 2008. Only Conficker A was active at this date. The third, and final day occurred on January 21, 2009 and contains both Conficker A and B (CAIDA, 2008).

The CAIDA UCSD Anonymised Internet Traces 2013 contains anonymised passive traces from 2013. This dataset is useful for research regarding the characteristics of internet traffic. This includes: application breakdown, security events, topological distribution, and flow volume and duration. The traces were taken from CAIDA's equinix-chicago and equinix-sanjose passive monitors on high-speed internet backbone links. Each monitor has two machines with each machine having an Endace network card. The network card was use to record the traces in a pcap file format. The IP addresses present in the files were anonymised and replace with a synthetic IP address. However, each actual IP address is replaced with the same synthetic IP address. Lastly, the payload has also been removed from the captured traces (CAIDA, 2013).

2.1.2 Simulated Data

2.1.2.1 Network Simulator

The Network Simulator Version 2 (NS2) is an event driven simulation tool tat is useful in studying the dynamic nature of communication networks. It allows the user to simulate wired and wireless network functions and protocols. Furthermore, the structure of NS2 is modular and is flexible. This allowed it to gain popularity with several researches using it in their simulations (ns2blogger, 2017).

Network Simulator Version 3 (NS3) is a discrete-event network simulator that is used for research and educational purposes. It was built from scratch in order to replace NS2. Furthermore, NS3 is not backwards compatible with NS2 (nsnam, 2017).

2.1.3 Simultaneous usage of a combination of datasets

There are also researches that simultaneously make use of more than one dataset. A common approach is by using a dataset containing only normal data, and another dataset containing only malicious data. They then train their machine learning algorithms with these datasets.

2.2 DDoS detection

2.2.1 Non-hybrid

Xiang et al. (2011) proposed a method that can detect low rate DDoS attacks. The method uses two information metrics: generalised entropy, and information distance. The usage of these information metrics was motivated by how they can perform better compared to Shannon's entropy and the Kullback-Leibler divergence method. Their information metrics increase the gap between normal and attack traffic. Adjusting the order of the information metrics allows them to adjust the gap created. Properly tuning the order can allow them to detect low rate attacks early on while maintaining a low false positive value. They evaluated their approach using normal data from DARPA and DDoS data from CAIDA. They compute the probability distribution of the source IP addresses of the normal and attack datasets. They then applied their information metrics on the probability

distributions.

Hoque et al. (2016) aims to detect both low and high rate DDoS attacks in their approach. To do this, they made use of two new statistical measures. The first is called Dis_HBK and it aims to differentiate DDoS and normal traffic. The second is called AI_HBK and that aims to differentiate low rate and high rate DDoS attacks. The approach was motivated by how the different statistical measures that can detect anomalies produce high correlation values even with dissimilar objects. To do this they assume that normal network traffic follows specific patterns considering all attributes and that its traffic features are closely correlated. They also assume that attack traffic has at least 1 attribute with a diverging pattern and that during flooding, abrupt changes occur in at least one feature. Their approach has two main categories. The first aims to detect if there is a DDoS attack or not. If there is an attack, the second module aims to determine if it is a low rate or a high rate attack. They use different feature sets for each category. For determining the presence of a DDoS attack, they use the source IP's entropy and variation, and the packet rate. For determining the type of attack occurring, they use the same features as the previous category but with the addition of the number of distinct source IPs. To detect the presence of an attack, they first create a baseline of the normal traffic split into multiple 1-second time windows. The baseline is created by taking the average value the features from the time window. After creating the baseline, they deem data from another 1 second time window to be part of a DDoS attack if the difference between the average of its features and the baseline exceed a threshold alpha. In order to classify the traffic as part of a low rate or high rate attack, they compute its attack intensity. If the value of the attack intensity is less than 0, the traffic is part of a low rate attack. On the other hand, if it is greater than 0, the traffic is part of a high rate attack. The authors state assumptions and proofs in order to justify having a threshold value of 0. They evaluate their approach using the CAIDA DDoS 2007 dataset, and the DARPA dataset. They also evaluate their approach using the datasets after being normalised using z-score normalisation. They change the value of the alpha threshold in order to determine what produces the best accuracy. They were able to obtain an accuracy of 97.53% to 100% depending on the dataset and what they aim to detect.

The work of Bhuyan et al. (2014) focuses on detecting low-rate DDoS attacks. They were motivated by how there are only few researches that focus on detecting low-rate DDoS attacks and that current detection methods are more concentrated on packet-level datasets. Their approach consists of using generalised entropy in order to measure the information gain using the following features: source IP, destination IP, and protocol. They state that there is a low-rate attack if up to 1000 attack packets per seconds are sent to the victim. In order to test their

approach, they used the DARPA dataset for normal data, and the CAIDA DDoS 2007 dataset for low-rate DDoS attacks. They also conduct experiments changing the threshold values in their approach.

The approach of C. Li et al. (2015) in detecting DDoS flooding relies on the observation that flooding traffic is persistent and synchronous while normal traffic is short-lived and non-synchronous. They create a pair containing the source and destination IP address for each packet; they do this without removing the duplicates. They then create a set containing the pairs collected in a given time slot. By performing an intersection operation on a set and the set in the next timeslot, a flow lasting two timeslots can be obtained. Continuing the intersection operation over several more timeslots will show the persistent synchronous flows since normal flows will be removed by the intersection operation. In order to detect a flooding attack, the number of synchronous long flows that exceed a threshold must occur for a set number of timeslots. They then experimented with the threshold values using the CAIDA Anonymised Internet Traces 2013 for normal traffic and the CAIDA 2007 dataset for DDoS traffic. The accuracy they achieved was 93.3%.

Siris & Papagalou (2006) used two anomaly detection algorithms namely: an adaptive threshold, and a Cumulative Sum (CUSUM) in order to detect SYN flood attacks. In addition, their method can be used with other flooding types. The adaptive threshold algorithm signals an alarm when the number of measurements exceed a threshold for a certain number of consecutive intervals. The threshold and the number of consecutive intervals is adaptively set using the recent measurements. Measurements depend on what needs to be detected. In the case of detecting SYN flood attacks, measurements pertains to the number of SYN packets. The CUSUM algorithm signals an alarm when the volume of measurements exceeding a threshold exceeds an aggregate volume threshold. A measurement refers to the number of SYN packets that were measured in a given time interval. They evaluated their approach using network traffic taken from DARPA. However, the attacks were generated synthetically. The advantage in doing so is that the researchers can control the characteristics of the attacks; allowing them to see the performance of their algorithm with different attack types. With this, they conducted experiments using high and low intensity attacks. Their results produced 100% accuracy with high intensity attacks. But the performance deteriorated for low intensity attacks. Another aspect of their approach is that the accuracy was achieved with low complexity. Lastly, the authors also state that despite their work focusing on SYN flood attacks, their algorithm can also be applied in detecting other types of flooding attacks.

The approach of C. Sun et al. (2009) in detecting SYN flood attacks relies on the the difference between the number of outgoing SYN-ACK packets from the

server to the client, and the number of incoming ACK packets from the client to the server during a three way handshake. They call the client ACK packet during a three way handshake "CliACK" in order to differentiate it from the other ACK responses during the TCP connection. The rationale for this approach is that there must only be one SYN-ACK packet and one ACK packet in a normal TCP connection. However, during a SYN flood attack, the CliACK packet is not sent to the server; resulting in a greater number of SYN-ACK packets compared to CliACK packets. In addition, they state that it is impossible to spoof the SYN-ACK and CliACK pair. In order to separate the CliACK from other ACK packets, they store the SYN-ACK and CliACK pairs in a counting Bloom filter. They implement their approach using the Auckland-VIII trace set since there exists some low rate SYN flood attacks in it. They then compare their approach along with others.

The approach of Neethu Raj et al. (2015) uses three nodes with each monitoring a different metric to detect SYN flood attacks. The metrics are: the SYN packets received per second, the number of SYN-ACK packets sent per second, and the SYN/FIN rate per second. This approach was done based on the following observations. First, there is an abnormal rate of SYN packets arriving to the victim during attacks. Second, during a three way handshake, once the server receives the initial SYN packet from the client, the server will reply with a SYN-ACK packet. If the server does not receive an ACK packet from the client after a certain period of time, it will resend a SYN-ACK packet and double the time interval before resending another packet. This is done until the maximum time out is set. If that period is reached, the connection with the client is terminated. Third, during normal connections, there must be a FIN packet for every SYN packet. Furthermore, they state that the false alarm rate is reduced by simultaneously monitoring three different metrics and analysing them using the same method. Afterwards, they then use an Auto Regressive (AR) model to predict the future values of each metric. If at least two of the three nodes report an abnormality, there is a SYN flood. To verify their claims, they created a simulation of normal and SYN flood traffic and applied their approach to it. Their results show that the approach can successfully detect SYN flood attacks.

Korczynski et al. (2011) focuses on detecting SYN floods and port scans. They do this by randomly sampling the incoming and outgoing TCP packets with the ACK flag turned on. Their approach was motivated by how a SYN packet with its ACK flag is turned on indicates that a legitimate connection has taken place. The system used to detect port scans consist of measuring the number of unacknowledged SYN packets from a certain client. If the number of unacknowledged segments is greater than a threshold, there is a port scan occurring. In order to detect SYN flood attacks, they measure the number of unacknowledged requests

to a certain IP. A SYN flood is present if the number of unacknowledged requests is greater than a threshold.

Sammany et al. (2007) used a backpropagation Multilayer Perceptron (MLP) in order to detect SYN floods, probes, and normal traffic. This approach was motivated by how previous research that used similar and the same approach performed well. They used the DARPA 1999 dataset in order to validate their work. From the dataset, they used 35 features that can be grouped into 3 categories: commands used, connection specifications, and connections to the same host in the last 2 seconds. The average accuracy of their approach is 93.43%

2.2.2 Hybrid

Hybrid classification involves applying classifiers in sequence of one another (Prabowo & Thelwall, 2009).

Abuadlla et al. (2014) used a flow based approach in detecting anomalous traffic. It was motivated by how payload-based systems were unable to function well at high speed networks and that encrypted protocols are challenging to manage. With this, their approach consists of using two Artificial Neural Networks (ANNs). The first identifies whether the current flow is anomalous or not. If it is anomalous, it is sent to the second neural network in order to classify its type. The features used differ for both neural networks. The features given to the first neural network are those that have the most influence on determining if a flow is anomalous or not. The majority of these features are extracted from the packets. The features given to the second relate to identifying the type of anomalous flow. These consist of the number of different flows between the attacker and the target. To validate their approach, they used a subset of the DARPA dataset and three neural network training algorithms in order to find out what combination works best as the anomaly detection and type classification neural network. The three training algorithms are: Resilient Backpropagation, Radial Basis Function net, and Levenberg-Marquardt. The most accurate training algorithm for the anomaly detection module was Levenberg-Marquardt with 94.2%, while the most accurate for type classification was Resilient Backpropagation with 99.4%.

Barati et al. (2014) created an anomaly based DDoS detection method. They used a genetic algorithm in order to decrease the number of features and select the ones that are the most descriptive of the dataset. Afterwards, an MLP is applied in order to improve the detection method. They used the CAIDA UCSD 2007 dataset for DDoS data and the CAIDA UCSD Anonymised Internet Traces 2013 for normal traffic. Network Measurement and Accounting System (Netmate) was

used in order to extract features from the datasets. Their approach was able to correctly classify instances more than 99.99% of the time.

Chen et al. (2013) uses an AR model in order to predict the future traffic rate. The choice of using an AR model instead of other time series models was made because the AR model was more suited to the bursty nature of networks. Furthermore, using the other time series models may result in a large prediction error. They determine that a DDoS attack is occurring if the difference between the predicted traffic rate and the actual rate exceeds a certain threshold. Afterwards, they train a Back Propagation ANN with the DDoS traffic in order to aid in classification. They used the DARPA 1998 and 1999 dataset in order to evaluate their work. Using the AR model by itself produced an accuracy of 71.84%. Moreover, using the ANN with the AR model increased their accuracy to 99.95%

The anomaly detection system of Lin et al. (2012) uses a Support Vector Machines (SVM), Decision Tree (DT), and Simulated Annealing (SA) in order to detect an attack. The research was motivated by how prior research did not simultaneously provide the feature selection and decision rules. The choice of algorithms that were used was motivated by the following: the SVM performs well in classifying anomalies, the DT can generate decision rules, and the SA can converge towards the global optima. They use the KDD dataset in order to evaluate their approach. Furthermore, they can detect attacks from the following types: DoS, R2L, U2R, and probes. Their methodology first uses the SA and the SVM in order to optimise the parameters and select the appropriate features. Afterwards, the SA and DT is used in order to increase the accuracy and construct the decision rules. The model is evaluated after this. If it passes a set threshold, it is deemed as the solution. If not, the authors repeat the process using the current solution but changing some features to be used. This iterative approach is used in order to improve classification and reach the acceptable threshold. They used 10-fold cross validation in order to evaluate their approach. This resulted with an accuracy of 99.96% and used 23 of the 41 features available in the dataset.

Hussain et al. (2015) used a two-step hybrid DDoS detection system. Their work combines an anomaly and misuse detection system. This approach was motivated by how combining the advantages of both detection systems will compensate for each other's weaknesses and will ultimately lead to a greater accuracy and smaller false positive value. The first step consists of an SVM in order to determine if there is an attack or not. This stage functions as the anomaly detection mechanism. If there is an attack, an ANN that functions as the misuse detection system will determine the attack type from the following: DoS, R2L, U2R, and probes. However, they only determine the general attack type and not the specific kind. For example, their system can identify if there is a DoS attack, but it will not be able to identify if it is a SYN flood or something else. In order to evaluate

their approach, they used the NSL KDD dataset with all 41 features. They experimented with different kernels to use with the SVM and different gamma values. They settled on using the Radial Basis Function kernel and a gamma of 0.00001. For the ANN, they use the Resilient Backpropagation algorithm to train it. They also vary the number of hidden layers and nodes in the hidden layers during their experiments. They also compared the performance of the two-step approach to using the SVM and ANN individually. The individual SVM has an accuracy of 95.72% while the individual ANN has an accuracy of 86%. Combining the two approaches in the hybrid model produced an accuracy of 99.95%.

Jawhar & Mehrotra (2010) created an anomaly detection system using a Fuzzy C-means clustering (FCM) algorithm and an MLP in order to detect the presence of anomalous traffic and identify its type. The FCM was used in order to separate normal traffic from anomalous ones. The algorithm used two clusters in which each one contains a particular traffic category. Afterwards, the anomalous data is given to an MLP in order to classify its type from the following: DoS, R2L, U2R, and probes. They used the KDD dataset in order to validate their approach. In addition, they compared their results with others. For the MLP, they used trial and error in order to determine the best number of nodes and hidden layers. The result was a 99.99% accuracy for the FCM, and a 100% accuracy.

2.2.3 Stacking

Stacking involves having multiple base classifiers which classify an instance. A meta classifier then uses the classifications given by the base classifiers in order to classify an instance (Rokach, 2010).

Cepheli et al. (2016) created a stacking classifier by using an anomaly and misuse detection module as its base classifiers. A multidimensional Gaussian Mixture Model (GMM) functions as the anomaly detection module while the Snort IDS (Roesch, 1999) functions as the misuse detection module. For their configurations, the parameters of the Gaussian model was estimated using Expectation Maximisation (EM). And Snort was configured using commonly applied rules. The output of both detection modules was given to a decision combiner. It is responsible for controlling and evaluating the outputs of both detection modules. The researchers used the following features: packet interarrival times, packet sizes, and protocol frequencies. Their work was evaluated using data from DARPA and a Turkish bank with True Positive (TP) and False Positive (FP) functioning as their accuracy metrics. With the DARPA dataset, using only the anomaly detection module only gave a TP of 92.1% and an FP of 1.8%. Using only the misuse detection module gave a TP of 64.7% and an FP of 13.2%. Using both modules

gave a TP of 98.7% and an FP of 0.73%. They achieved near perfect accuracy using the Turkish bank dataset. However, the DDoS attack was easy to detect using that dataset.

The work of Seo et al. (2005) created a stacking classifier by combining multiple SVM's with each being trained on a specific attack type. The attacks are categorised into three types: DoS attack, DDoS attack, and DrDos attack. This approach was inspired by how using a single SVM generated a large number of false positives in their previous work. The features fed to the SVM's are related to different traffic rates. Specifically, the rate of a certain TCP flag occurring in the traffic, and the rate of a certain protocol being used. Lastly, they compared the performance of their approach to that of using only a single SVM. The results show that the stacking approach created with multiple SVM's produced a lower false positive rate compared to using only a single SVM. Using the poly kernel, their multiple SVMs approach produced a false positive of 26.82%. Using the same kernel type with a single SVM gave a false positive of 40.15%.

The stacking approach of Depren et al. (2005) combines an anomaly detection system with a misuse detection system. Self Organising Map (SOM) acts as the anomaly detection system and is used to determine if there is an attack or not. DT functions as the misuse detection system and is used to determine the type of attack occurring. They identify the specific attack type instead of the attack's general category. The outputs of the SOM and the DT are then passed on as inputs to the Decision Support System (DSS). The DSS is responsible for interpreting its inputs in order to be able to give a report of the intrusion detection activity to the end user. The detection support system is configured with the following rules. If the DT reports an attack regardless of whether or not the SOM has reported one, the DSS reports the attack and its classification. And if only the SOM detects an attack, the DSS reports an unclassified attack. The authors use the KDD dataset to evaluate their work. They have also compared the performance of their hybrid approach with using the SOM and DT individually. They used the following features from the dataset: duration of the connection, protocol type (TCP, UDP, or ICMP), service type (FTP, HTTP, Telnet), connection status flag, total bytes sent to destination host, and the total bytes sent to source host. Their experiments gave the following results: the individual SOM has an accuracy of 98.96% while the individual DT has an accuracy of 99.61%. The hybrid approach has an accuracy of 99.9%.

2.2.4 Combination of approaches

Researches under this category are those that cannot be placed under the previous categories since they combine different types of approaches instead of using only one.

Kumar & Selvakumar (2013) used a combination of bagging and boosting in their ensemble algorithm called NFBoost. They applied NFBoost by using a number of Adaptive Neuro-Fuzzy Interface System (ANFIS) classifiers. The ANFIS classifier is a Sugeno type hybrid classifier that uses fuzzy rules, genetic algorithms, and neural networks. Its creation was motivated by the following reasons: fuzzy logic can interpret rules well but cannot automatically acquire those rules, neural networks can generalise the network well but is unable to interpret the rules, and by how genetic algorithms can produce the optimal solution but at the cost of high time complexity. Using the ANFIS provides the advantages of the previous methods. For their methodology, they first partition a dataset into k segments and an ANFIS is trained with each subset. After training, different weights are given to each ANFIS depending on its performance. In order to get the final classification, the fuzzy membership value of each classifier is multiplied with its weight. Those are then added together and divided by k. If the resulting value is less than 0.5, it is classified as normal. It is an attack otherwise. To validate their approach they used the following datasets: KDD, Conflicker, CAIDA 2007, UNINA, and SSE Lab. The features they used were the number of: UDP echo packet to a port, connections to the same host within a time window, ICMP echo replies from the same source. The other features used were the connections with SYN errors using the same service during the time window, variance of time difference between 2 consecutive packets, and the ratio of incoming and outgoing SMTP packets. Th previously stated features were extracted using Tcptrace. They conducted three experiments using datasets. The first experiment only used the KDD dataset. The second is a mixed dataset that used the Conficker, CAIDA 2007, and UNINA dataset. The final experiment was conducted in an emulated SSE testbed from the author's laboratory. The accuracy they obtained was 98.2% for the first experiment, 98.8% for the second, and 99.2% for the final one.

The ensemble approach of Boroujerdi & Ayat (2013) can be thought of as part hybrid and part boosting. For the hybrid approach, they used several ANFISes with each being trained using features selected by different machine learning algorithms using the NSL KDD dataset. The features of the first ANFIS was selected by an SVM, the second by Linear Genetic Programming (LGP), the third by Multivariate Adaptive Regression Splines (MARS), and CfsSubsetEval for the last classifier. The rationale of this approach was that using different features results in different performances in each classifier. This process in turn increases

the accuracy of the entire ensemble. In order to further increase accuracy, they employ a boosting technique in which the correct outputs of the first classifier is fed into the second, and so on. This reduces the processing time of the later classifiers compared to the former ones. In order to validate the effectiveness of their approach, they compared their method with several other machine learning methods namely: Random Tree, J48 Decision Tree, Naive Bayes, SVM, Random Forest, Naive Bayes Tree, and Multilayer Perceptron. Furthermore, they also used the KDD dataset in evaluation. Their approach performed better than the other machine learning methods on both datasets. Their approach was able to achieve an accuracy of 96.38% by using the NSL KDD dataset and 82.88% when using the KDD dataset.

2.2.5 Summary table of DDoS detection researches

A question mark (?) was put in the accuracy area if the research it pertains to does not provide an accuracy or if it is presented as a graph. It is also mentioned in that area if they did not use accuracy as their evaluation metric. For the sake of brevity, the word "source" refers to the source IP address while "destination" refers to the destination IP address in the features area.

Table 2.1: Non-hybrid DDoS detection researches

Author(s)	Data source	Features	Detection model	Detect what	Accuracy
(Xiang et al., 2011)	DARPA (Normal), CAIDA 2007 (DDoS)	Probability distribution of the source	Generalised entropy, Information distance	Low rate DDoS	?
(Hoque et al., 2016)	DARPA, CAIDA 2007	Source entropy and variation, Packet rate, Distinct sources	Dis_HBK, AI_HBK	Low and High rate DDoS	97.53%-100%
(Bhuyan et al., 2014)	DARPA (Normal), CAIDA 2007 (DDoS)	Source, Destination, Protocol, Alpha	Entropy, Thresholds	Low rate DDoS	?
(C. Li et al., 2015)	CAIDA 2013 (Internet traces), CAIDA 2007 (DDoS)	Thresholds	Long flows	DDoS flood	93.3%
(Siris & Papagalou, 2006)	DARPA	SYN packet rate	Adaptive threshold, CUSUM	SYN flood	100%
(C. Sun et al., 2009)	Auckland-VIII Trace set	Thresholds	Difference of SYN+ACK and CliACK packet rate	SYN flood	?
(Neethu Raj et al., 2015)	(From the authors)	SYN, SYN+ACK, and SYN/FIN rate	At least 2 rates are abnormal	SYN flood	?

(Korczynski et al., 2011)	(From the authors)	Thresholds	ACK rate going over a threshold	SYN flood, and Port scans	?
(Sammany et al., 2007)	DARPA 1999	Commands used, connection specifications, connections to the same host	Backpropagation MLP	SYN floods, and Probes	93.43%

Table 2.2: Hybrid DDoS detection researches

Author(s)	Data source	Features	Detection model	Detect what	Accuracy
(Abuadlla et al., 2014)	Subset of the DARPA dataset	Features extracted from the packet, and flow	2 ANNs. ANN 1: presence of anomalous data. ANN2: anomalous data's type.	DoS/DDoS, Port Scan, Land Attack, other attacks	Attack or not: 94.2%, What attack: 99.4%
(Barati et al., 2014)	CAIDA 2007, CAIDA 2013	Different statistical values form the packets	GA for feature selection. Feed the features to an ANN	DDoS	99.99%
(Chen et al., 2013)	DARPA 1998 and 1999	Network traffic flow rate	AR, ANN	DDoS	AR only: 71.84%, AR and ANN: 93.75%
(Lin et al., 2012)	KDD	23 features from the dataset	SVM, DT, SA	DoS, R2L, U2R, Probe	99.96%
(Hussain et al., 2015)	NSL KDD	41 features from the dataset	SVM: attack or not, ANN: attack type	DoS, R2L, U2R, Probe	SVM: 95.72% , ANN: 86%, Hybrid: 99.95%
(Jawhar & Mehrotra, 2010)	KDD	35 features from the dataset	FCM: Attack or not, Resilient Back Propagation MLP: Attack type	DoS, R2L, U2R, Probe	FCM: 99.99%, MLP: 100%

Table 2.3: DDoS detection researches that used Stacking

Author(s)	Data source	Features	Detection model	Detect what	Accuracy
(Cepheli et al., 2016)	DARPA, a Turkish bank	Packet interarrival times, Packet sizes, Protocol frequencies	GMM, Snort, Decision combiner	DDoS	Individual: TP: 64.7-92.1%, FP: 1.8-13.2%, Both: TP: 98.7%, FP: 0.73% (DARPA)
(Seo et al., 2005)	(From the authors)	TCP flags, Protocol, Request per connection, Simultaneous connections	Multiple SVMs (Multi-SVM), Single SVM	DoS, DDoS, DrDoS	Poly kernel FP: 26.82% (Multi-SVM), 40.15% (SVM)%
(Depren et al., 2005)	(From the authors)	6 features taken from the packet and flow	SOM: Attack or not, DT: Attack type	Attack type	Individual: 98.96%, Hybrid: 99.99%

Table 2.4: DDoS detection researches that used a combination of approaches

Author(s)	Data source	Features	Detection model	Detect what	Accuracy
(Kumar & Selvakumar, 2013)	KDD, Conficker, UNINA, SSE Lab, CAIDA 2007	Different statistical values from the packets	Majority voting of ensemble fuzzy neural network	DDoS	First: 98.2%, Mixed: 98.8%, SSE: 99.2%
(Boroujerdi & Ayat, 2013)	KDD, NSL KDD	Selected by different ML algorithms	Ensemble neural network	DDoS	KDD: 82.88%, NSL KDD: 96.38%

Chapter 3

Theoretical Framework

3.1 Different Types of DDoS Attacks

3.1.1 High Rate DDoS Attacks

These types of attacks involve overwhelming the target with packets with the intention of saturating its connection bandwidth or resources. Furthermore, some high rate attacks send malformed packets Radware (2017b).

An example of a high rate attack is the SYN flood. During a normal three way handshake, a client sends a server an initial SYN packet. Once the server receives the packet, it will reserve resources to track the TCP state. Afterwards, the server will reply with a SYN-ACK packet. The client will then reply with an ACK packet once it receives that. During a SYN flood attack, the client will not send the final ACK packet. This causes the server to continue using resources in maintaining the connection (C. Sun et al., 2009). Once the resource has been reserved, a new connection cannot use it until the resource has been freed (Q. Sun et al., 2009).

3.1.2 Low Rate DDoS Attacks

Low rate attacks are more effective in causing damages to legitimate flows and are more difficult to detect compared their high rate counterpart. Low rate attacks differ from high rate attacks since they exploit the vulnerabilities in the congestion control mechanism of TCP. Instead of continually sending network traffic, this type

of attack sends periodically pulsing data flows which can dramatically reduce the average rate of attack flows (Zhang et al., 2012). Low rate attacks can be performed using a single attacking host whereas high rate attacks rely on several hundred hosts in order to achieve the same effect (Kenig, 2013).

An example of this type of attack is the Slowloris. It holds HTTP connections open by periodically sending partial HTTP packets. In doing so, the connection is kept alive. By creating additional similar connections, the victim's resources will be used up in maintaining the connections (Kenig, 2013).

3.2 Tools in performing DDoS attacks

3.2.1 SlowHTTPTest

SlowHTTPTest is a highly configurable tool that simulates some Application Layer DoS attacks by prolonging HTTP connections in different ways. It is able to simulate the following attacks: Slowloris, R-U-Dead-Yet (RUDY), Apache Killer, and Slow Read (Shekyan, 2017).

RUDY functions by sending partial HTTP post requests with a long content-length header field. The information is sent one byte-sized packet at a time with an interval of a few seconds per between each byte. Sending numerous small packets at a slow rate creates a massive backlog of application threads for the victim. In addition, the long content-length header field prevents the server from closing the connection (Imperva Incapsula, 2017).

The Apache Killer involves the attacker sending a request to an Apache server to retrieve URL content with a large number of overlapping byte ranges. This causes the victim to run out of usable memory (Radware, 2017a).

Slow Read involves making a legitimate request to the victim. However, the attacker reads the responses slowly. The attacker does this by advertising a small receive window size (Shekyan, 2012).

3.3 Common Methods of Detecting DDoS Attacks

Liu (2009) states that there are three major ways of detecting attacks: pattern detection, anomaly detection, and a hybrid method that combines misuse and

anomaly detection.

3.3.1 Misuse Detection

Pattern or misuse detection uses a database of attack signatures to detect network attacks by matching the current input to the database (C. Li et al., 2015; Kumar & Selvakumar, 2013).

Misuse detection is generally performed by using one of the following methods: expert systems that contain a set of rules that describe attacks, signature verification wherein attack scenarios are converted into a sequence of audit events, Petri nets that graphically represent the known attacks, and state transition diagrams that represent attacks with a set of goals and transitions (Jean-Philippe, 2001).

However, this type of approach can only detect known attacks. With this, its database must stay current in order to maintain reliability (Liu, 2009). C. Li et al. (2015) adds that since DDoS attacks use legitimate packets to generate malicious traffic, misuse detection methods may not be suitable.

3.3.2 Anomaly Detection

An anomaly can be characterised as an old attack that has changed its pattern in order to avoid detection from misuse detection. An anomaly can also be a new form of attack (Kumar & Selvakumar, 2013).

Anomaly detection creates a baseline of normal network traffic. Once the baseline has been determined, it is periodically compared against the current state of the network. A network attack is detected once a certain threshold of difference is reached (C. Li et al., 2015; Siris & Papagalou, 2006). The challenge in this approach lies in lessening false alarms. With this, the baseline stating what is normal must be constantly updated and the threshold categorising anomalous traffic must be properly adjusted (Liu, 2009).

The usage of legitimate packets in a DDoS attack leads to a change in the traffic flow statistics. Those statistics can include: the packet type and size, the number of half open connections, the packet rate towards a particular application or port number. Therefore, the use of anomaly detection is suitable (Siris & Papagalou, 2006). C. Li et al. (2015) states that, due to the ability of anomaly detection to discover unknown attacks, many algorithms based on it have been proposed to detect DDoS flooding attacks.

Anomaly detection is generally performed by using one of the following methods: using thresholds, statistical measures, rule-based measures with expert systems, and non-linear algorithms such as Neural Networks (Jean-Philippe, 2001).

However, with this type of detection, there is a tradeoff in its ability to detect attacks and its ability to misclassify normal behaviour as attacks (C. Li et al., 2015).

3.3.3 Hybrid Detection

Liu (2009) states that hybrid detection systems can use the attacks detected using anomaly detection in order to update their database signatures for misuse detection. However, an attacker can bypass this system by characterising normal traffic as attacks.

3.4 Machine Learning Algorithms

3.4.1 Naive Bayes (NB)

Mitchell (1997) states that Naive Bayes (NB) classifier is a probabilistic classifier that applies the Bayes theorem. It assumes that attributes are independent of one another given the target value. Because of this assumption, the probability of classifying a set of attributes $a_1, a_2, ..., a_n$ as being a member of a certain class v_j is equal to the product of each attribute with respect to the class. The class with the highest probability is selected as the final classification.

3.4.2 Support Vector Machines (SVM)

Support Vector Machines (SVM) is a non-probabilistic and large-margin classifier. Its aim is to create a hyperplane that can separate instances of classes from each another. The hyperplane created should maximise the margin between the classes as much as possible. In the case that any hyperplane is unable to completely separate the classes from one another, the misclassification percentage should be minimised instead. A hyperplane is a generalisation of a plane in n-dimensions: in one dimension, it is a point; in two dimensions, it is a line; in three, it is a plane; it is known in further dimensions as a hyperplane. (Cortes &

Vapnik, 1995).

The SVM has good performance and is speedy with regards to classifying anomalies in intrusion detection (Lin et al., 2012). Furthermore, SVMs are scalable and are relatively insensitive to the number of data points (Horng et al., 2011; Peddabachigari et al., 2007).

A drawback of the SVM is that its training complexity is dependent on the size of the dataset with the complexity being at least quadratic to the number of data points (Horng et al., 2011).

An example of the SVM classifier is shown on Figure 3.1 (OpenCV, 2017). The circles and squares represent instances of different classes. They are mapped in two dimensions. The hyperplane created that separates both classes is also shown in the figure. There are no instances of either class within the margin of the optimal hyperplane. The instances that are on the border of the hyperplane are called the support vectors.

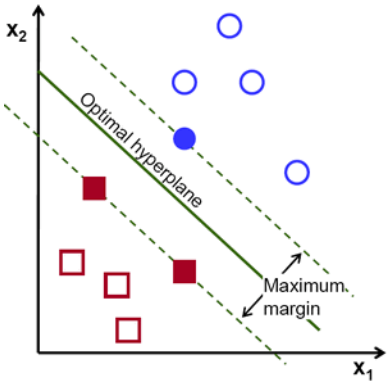

Figure 3.1: Support Vector Machine

3.4.3 Artificial Neural Networks (ANN)

Mitchell (1997) states that Artificial Neural Networks, or ANNs, were inspired by the observation that biological learning systems are built from complex interconnections of neurons. ANNs are built from an interconnected set of simple units wherein each units takes a number of inputs and produces a single output. The input and output types are both real-valued. The outputs may serve as inputs to other units. The structure of ANNs consist of an input layer that sends its outputs to a hidden layer. The hidden layer in turn will send its outputs to the output layer. The hidden layer received its namesake since their outputs are only visible within the ANN and is not available as part of the global output. Aside

from this, ANNs can also have many different structures such as: cyclic, acyclic, directed, or undirected. An illustration of an ANN can be found on Figure 3.2 (Stanford University, 2017).

Neural networks work well on noisy data. In addition, the speed that they can provide makes them suitable for real time intrusion detection (Peddabachigari et al., 2007).

However, the problem in this approach lies in its training phase. Its efficiency is heavily reliant on the correct configuration of its parameters. Moreover, this phase requires a large amount of data (Peddabachigari et al., 2007).

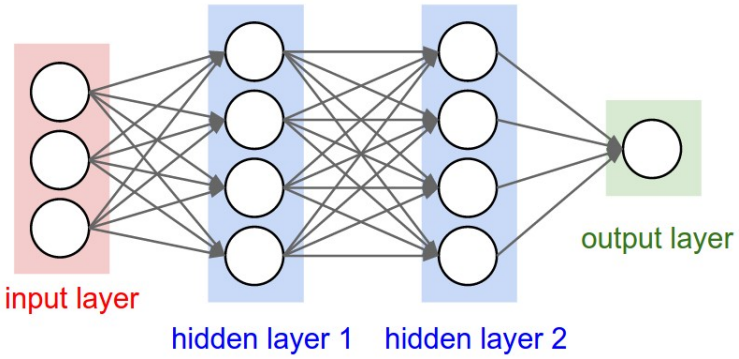

Figure 3.2: Artificial Neural Network

3.4.4 K-Nearest Neighbours (KNN)

According to Mitchell (1997), the K-Nearest Neighbours, or KNN, algorithm is a basic instance based method. It assumes that all instances correspond to points in an n-dimensional space. The nearest neighbours of an instance can be found using the Euclidean distance formula. Assume that x is denoted to be the instance to be classified. Its class is determined be the same as the most common class among the k instances closest to it. For example, assigning the value 1 to k will classify x with the same class as the instance closest to it. If the value 5 is assigned to k, the algorithm will get the classes of the 5 nearest neighbours from x. Afterwards, x will be classified using the most common class among the 5. Since the KDD algorithm equally considers all instances close to x as determined by k, a variant of the algorithm involves increasing the weights of instances that are closer to x compared to those that are further away. Since the variation has weights, the downside in including all training examples instead of specifying a number for k is reduced since distant instances will have little influence on x. An advantage in this approach is lessening the impact of isolated noisy instances. The disadvantage

of this approach is the increased time required to classify x. Another variation involves weighing each feature differently. This arose from how KDD considers all features when making a classification, This allows the algorithm to focus more on relevant features.

An example of the KDD algorithm is shown on Figure 3.3 (Numerical Algorithms Group, 2017). The squares and circles are instances of different classes. The triangle at the centre of the image indicates the instance that needs to be classified. The instances are in 2 dimensions and k is set to be 3. With this, the 3 nearest classes from the triangle are selected. Since within the 3 nearest classes there are more square classes than circle classes, the triangle instance would be classified as a square.

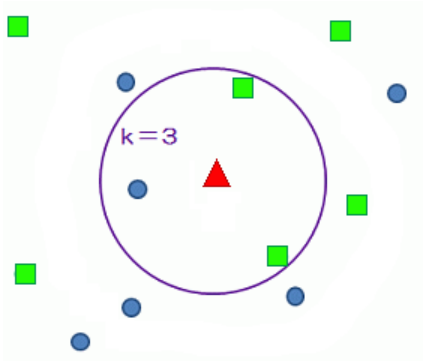

Figure 3.3: K-Nearest Neighbours

3.4.5 Decision Trees (DT)

The Decision Tree (DT) is a method for approximating functions that have a finite number of classes. The DT tolerant to noisy data and is capable of learning disjunctive expressions. Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute. Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. In order to classify an instance, the attribute of the root node is compared against it. The instance then moves down the tree branch that matches the value of the attribute. This process is repeated for the subtree rooted at the new node and continues until a leaf node is reached. The leaf node contains the class that will be assigned to the instance. The path from the root node to the leaf node shows the rules for classification. Decision trees are suited to problems with the following characteristics: instances are represented by attribute value pairs,

the output values are discrete, disjunctive descriptions may be required, and if the training data may contain errors and missing values.

Information gain is a measure on how well can a given attribute separate the training examples according to their target classification. The tree is constructed top down. The root node is selected by conducting a statistical test of all the features. The test uses entropy and it aims to determine how well can using only that feature classify the training examples. The best attribute is selected and used as the test at the root node. The branches of the root node is created for each possible value of the attribute. The node created at the branches is created in a similar way to the root node (Mitchell, 1997).

An example of the Decision Tree algorithm is shown on Figure 3.4 (Galea, 2016). When an instance needs to be classified, it starts at the root node of the tree and checks the value of the feature stated there, in this case outlook. Depending on the instance's outlook value (either sunny, overcast, or rain), it will move on to that branch and evaluate it once more. If the instance arrives at leaf node, the instance is then assigned the instance at that node.

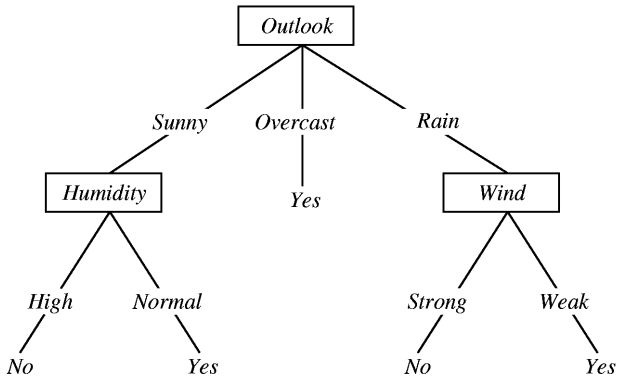

Figure 3.4: Decision Tree

3.4.6 Ensemble Classifiers

Ensemble classifiers generally contain the following components: a training set, a base inducer, a diversity generator, and a combiner. A training set is a labelled dataset consisting of a number of attributes and a single class attribute. A base inducer is an induction algorithm that obtains a training set and forms a classifier that represents the generalized relationship between the input attributes and the target attribute. The diversity generator is the component that is responsible for generating the diverse classifiers. The combiner is responsible for combining the classifications of the various base classifiers.

There are two types of frameworks for ensemble classifiers: dependent, and independent. The output of a classifier in a dependent framework is passed on as the input of the next classifier. On the other hand, the classifiers in an independent framework do not influence each other. Their outputs are instead combined afterwards (Rokach, 2010).

3.4.6.1 Boosting

Boosting is a dependent framework and is also the most well known model-guided instance selection. A model guided instance selection is an approach in which the classifiers that were constructed during previous iterations manipulate the training set the training set of the classifiers on the next iteration. The intention of boosting is to improve the performance of a weak learner. It works by training a weak learner on different subsets of the training data. The classifiers produced by the weak learners are then combined into a single composite strong classifier. The strong classifier achieves a greater accuracy compared to an individual weak learner (Rokach, 2010).

AdaBoost, or Adaptive Boosting, improves over the original boosting algorithm by focusing on misclassified instances. Specifically, the weights assigned to the data are initially uniform. After a weak learner classifies the data, the weights of misclassified instances are increased. This forces the algorithm to focus on the misclassified instances during the following iteration (Rokach, 2010).

3.4.6.2 Bagging

Bagging is the most well known independent framework. It aims to increase the overall accuracy by combining the outputs of its base classifier. In this approach, there are a number of base classifiers, with each being trained with a different subset of the dataset. The subsets may be disjointed or overlapping. In order to classify a new instance, each base classifier classifies that instance and the most predicted class is assigned to the instance. Moreover, since the base classifiers are independent of each other, their training can be parallelized (Rokach, 2010).

3.4.6.3 Stacking

Stacking is a technique that uses a meta classifier in order to achieve the highest generalization accuracy. This method attempts to identify the classifiers

that perform best from those that do not (Wolpert, 1992). Stacking is usually employed to combine models built by different algorithms. An instance in the training set is first classified by each of the base classifiers. After getting the classifications of the base classifiers, a meta dataset is created with each of its tuples corresponding to one in the training dataset. The class value of the tuples in the meta dataset is the same as the one in the training set. However, the attributes of each tuple in the meta dataset consists of the predicted classifications of the different models used in the stacking framework. The meta classifier is then created from meta dataset. This classifier combines the different predictions of the base classifier into a final one. With this, the meta classifier predications reflect the true performance of base level learning algorithms. Furthermore, the performance of stacking can be improved by using output probabilities for every class label from the base level classifiers. In such cases, the number of input attributes in the meta dataset is multiplied by the number of classes (Rokach, 2010).

3.4.7 Hybrid Classifiers

Hybrid classification involves applying classifiers in sequence of one another. An example of this is when the classification task is passed on to the next classifier if the initial classifier is unable to classify the data. This process is then repeated until the data is classified or until no other classifier exits (Prabowo & Thelwall, 2009).

3.4.8 Evaluation Metrics

3.4.8.1 True Positive (TP)

True Positive (TP) is the number of positive records that are classified as positive (Tang et al., 2016).

3.4.8.2 True Negative (TN)

True Negative (TN) is the number of negative records that are classified as negative (Tang et al., 2016).

3.4.8.3 False Positive (FP)

False Positive (FP) is the number of non-positive records that are classified as positive (Tang et al., 2016).

3.4.8.4 False Negative (FN)

False Negative (FN) is the number of non-negative records that are classified as negative (Tang et al., 2016).

3.4.8.5 Accuracy

Accuracy states how often the classifier is correct in classifying an instance (Boroujerdi & Ayat, 2013). It is calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

3.4.8.6 Precision

Precision is also known as the positive predictive (Alkasassbeh et al., 2016). It is a measure of how often a positive instance is actually a positive instance given the total number of items classified as positive (Boroujerdi & Ayat, 2013). It is calculated as:

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

3.4.8.7 Recall

Recall is also known as the positive sensitivity value. It is the ratio of the number of correctly classified positive instances out of the total number of positive instances (Alkasassbeh et al., 2016). It is calculated as:

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

Table 3.1: Degrees of agreement

Kappa	Agreement
0.00 âĂŞ 0.20	Slight agreement
0.21 âĂŞ 0.40	Fair agreement
0.41 âĂŞ 0.60	Moderate agreement
0.61 âĂŞ 0.80	Substantial agreement
0.81 âĂŞ 1.00	Almost perfect or perfect agreement

3.4.8.8 F-score

The F-score is also known as the F1 score and F-measure. It measures accuracy by considering both the precision and recall (Tang et al., 2016). It is calculated as:

$$F\text{-}score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.4}$$

3.4.8.9 Cohen's Kappa

Cohen's Kappa (Cohen, 1960) is a statistical measurement that measures the agreement between 2 raters. The categories for classification are mutually exclusive from each other. The degree of agreement in classification over the degree of agreement by chance is measured. Kappa is equal to the proportion of agreement observed between raters after adjusting for the proportion of agreement expected by chance. Another way of viewing the Kappa formula is that it is equal to the amount of improvement that raters showed above chance divided by the maximum amount of improvement that they could have shown (Wood, 2007). The kappa, K, can be defined as:

$$K = \frac{P_o - P_e}{1 - P_e} \tag{3.5}$$

P_o refers to the observed percentage of agreement among the raters. P_e refers to the percentage of agreement expected by chance if the raters were scoring randomly. A Kappa value of 1 indicates that the raters are in complete agreement with each other while a Kappa value of -1 indicates that the raters are in complete disagreement with one another. The degrees of agreement are shown on Table 3.1.

3.5 Feature Selection

The aim of feature selection is to select the most representative features of the given dataset (Barati et al., 2014). Using many features leads to higher accuracy but at the cost of increased computational complexity in terms of the time, memory, and disk usage. This ultimately makes the process very slow (Barati et al., 2014; Kumar & Selvakumar, 2013). If a real time system is needed, using less features is suitable (Kumar & Selvakumar, 2013).

There are three types of features: strongly relevant, weakly relevant, and irrelevant. Strongly relevant features are those that are always necessary in order to produce the optimal feature set. These features cannot be removed without affecting the original conditional class distribution. Weakly relevant features are those that are not always necessary to be members of the optimal feature set but may become necessary at certain conditions. Irrelevant features are those that are not needed. In addition, two features are redundant to each other if their values are completely correlated. An optimal feature set includes all strongly relevant features, a subset of the weakly relevant features, and no irrelevant feature (Yu & Liu, 2004; John et al., 1994).

Feature selection is broadly categorised into wrapper models and filter models. There are also main two ways used determine the optimal feature set: individual feature evaluation, and subset evaluation (Yu & Liu, 2004).

3.5.1 Wrapper Models

This type of feature selection uses the predictive accuracy of a learning algorithm in order to determine the quality of the selected features.

A problem with this approach is that the learning algorithm is executed with each feature set to be evaluated. This makes wrapper models computationally expensive and unable to scale to data with a large number of features Kohavi & John (1997).

3.5.2 Filter Models

This approach separates feature selection from classifier learning by selecting features independently from it. In this approach, the undesirable features are removed before the learning algorithm is used. Separating feature selection from

classifier learning allows filter models to be faster than wrapper models. This also allows filter models to be applied with datasets with a large number of features. Another advantage is that filter models can be used with any learning algorithm. In contrast, wrapper models need to be executed again when a different learning algorithm is used Kohavi & John (1997).

3.5.3 Individual Feature Evaluation

Individual feature evaluation ranks each feature by their importance in differentiating instances of different classes. In application, a subset of features is selected from the top of the ranking list. In addition, irrelevant features are easily removed since they will rank low on the feature ranking list. This subset approximates the set of relevant features. An advantage of this approach is its linear time complexity.

However, it is incapable of removing redundant features since their rank is likely to be close to one another. As long as features are deemed relevant to the class, they will be selected even they may be highly correlated to each other. For data with a large number of redundant features, this approach may be unable to produce an optimal result (Yu & Liu, 2004).

3.5.4 Subset Evaluation

Subset evaluation searches for a minimum subset of features that satisfy a certain threshold. This method evaluates each candidate subset by a certain metric and compare it to the current best subset. The candidate subset replaces the current best subset if the candidate performs better than it with respect to the metric. This process is repeated until a given stopping condition is satisfied. This method differs from individual feature selection by how the features are evaluated together instead of individually. This approach approximates the optimal feature set and can remove irrelevant features as well as redundant ones.

A problem in this approach lies in searching through the different combinations of feature sets. The different heuristic search strategies used in this approach still possess at least a quadratic time complexity. Thus, this method does not scale well with data with numerous features (Yu & Liu, 2004).

Chapter 4

Research Framework

4.1 Low rate DDoS attack simulation

4.1.1 Network topology

The low rate DDoS attack simulation network topology is shown on Figure 4.1. It is modelled after the DARPA 98 topology shown on Figure 2.1. The network topology was implemented as virtual machines on the same host machine. The network topology used is composed of one attacker and one victim. The attacking hosts and the server are connected to their own switch. A router is then used to connect the switches together. In addition a sniffer is placed on the attacker side to gather data.

Figure 4.1: Simulation network topology

4.1.2 Attack simulation specifics

The attacker is a Kali Linux machine (Kali by Offensive Security, 2017), the victim is an Apache server that is ran on a Linux desktop environment (Ubuntu Desktop, 2017), the sniffer is the Wireshark application (Wireshark, 2017) on the attacker machine, and the router is a separate command line interface Linux machine. The low rate attacks were performed by using the SlowHTTPTest tool (Shekyan, 2017).

The tests conducted using the tool consist of the following: 3 attack types (Slow body, Slow headers, Slow read), 2 HTTP verbs (Get, Post), and varying connection numbers. Each test was ran for 30 seconds. After all the iterations, they were conducted again, with the difference of increasing the victim server's strength to be able to handle more clients. The server's settings are as follows: default (about 300 clients at most), 1000 clients, and 3000 clients. The results are stored as pcap, csv, and html files. The pcap file is given by Wireshark, while both the csv and html files are given by SlowHTTPTest. This algorithm's pseudocode can be found on Algorithm 1. The algorithm has a complexity of $O(n^4)$ due to the nested loops needed to evaluate each combination of server setting, verb type, attack type, and connection count.

Algorithm 1 Low rate data gathering pseudocode

```
1:  procedure LOW RATE(C, n)                          ▷ C: Connection count limit
2:                                                    ▷ n: Amount to add to connection count
3:      verbs ← {GET, POST}
4:      attacks ← {Body, Headers, Read}
5:      results ← Array[verbs.length][attacks.length][C/n]
6:
7:      for each verb ∈ verbs do
8:          for each attack ∈ attacks do
9:              while connectionCount < C  do
10:                 Start Wireshark capture
11:                 add to results the evaluation using the loop's parameters
12:                 connectionCount+=n
13:             end while
14:         end for
15:     end for
16: end procedure
```

4.2 Data preprocessing

The NSL KDD dataset contains 4 attack types: DoS, R2L, U2R, and probes. However, the R2L, U2R, and probe attacks will be removed since they are not within the scope of this thesis.

The features extracted from the CAIDA and low rate attack dataset will be the ones that are available in the NSL KDD dataset. Since the packets in the CAIDA datasets has had their payload removed, features from the NSL KDD dataset will be removed if it cannot be extracted from the CAIDA and low rate attack dataset. This is done so that the datasets will be compatible with one another during the training phase.

Half of the testing dataset in the NSL KDD dataset will be used for testing phase while the remaining half would be used as the validation set. For the CAIDA and low rate dataset, 1/6 of the dataset will be used as the testing set, another 1/6 will be used as the validation set, and the remaining 4/6 will be used as the training set. The instances that will be in which dataset are chosen randomly but with respect to the ratio.

Since machine learning algorithms experience difficulties when they are trained with unbalanced datasets, instances from each dataset will be sampled in such a way that the training set will have the following distributions: normal 50%, low rate attacks 25%, high rate attacks 25%. An oversampling or undersampling technique is considered used to ensure this ratio. This ratio was created in order to ensure that the both classifiers will be trained with an equal number of instances of each class. With this, the first classifier will be trained using the entire dataset and the low and high rate attack instances will be relabelled as "attack". The second classifier will only be trained with attack data.

A dedicated training and testing set will be used instead of performing cross validation since the latter requires that the machine learning algorithm be trained multiple times when a feature needs to be evaluated. Doing so is a costly operation.

4.3 System Architecture

The system functions as an anomaly detection module. It is composed of two parts: the training phase, and the testing phase. The goal of the training phase is to find the optimal combination of machine learning algorithm and features. The testing phase involves classifying network traffic as normal, part of a high rate

DDoS attack, or part of a low rate DDoS attack.

4.3.1 Training Phase

The training phase flowchart is shown on Figure 4.2 and it involves the following processes: preprocessing the dataset depending on the module, and giving the preprocessed data to the appropriate module. There are two modules in this stage: the first module aims to determine if there is an attack or not, the second module aims to determine if there is a low rate or high rate DDoS attack. They are trained separately but in similar ways. The modules are trained separately in order to avoid false positives from the output of the first module being given as input to the second module. The modules are trained similarly in the sense that the same process is used in determining the optimal feature set and machine learning algorithm in each module.

They use the same dataset. It contains normal data, low rate DDoS attacks, and high rate DDoS attacks. However, they have different preprocessing steps. In the preprocessing step of the first module, both low rate and high rate attacks were relabelled to "attack". This was done so that the first module will be trained with binary data: either an attack or not. The preprocessing step of the second module involves removing normal data. This was done so that the second module will also be trained with binary data: either a low or high rate DDoS attack.

The training phase module flowchart module is shown on Figure 4.3. It is an iterative process and involves the following processes: feature selection given the preprocessed input, training a machine learning algorithm with the feature, evaluating the model produced, storing the evaluation results, and selecting the next combination of feature and algorithm. In order to determine the optimal feature set and machine learning algorithm, a combination of a wrapper model and individual feature evaluation will be used. A wrapper model was selected instead of a filter model since the performance of a feature is related to the learning algorithm used: the best features selected by a filter model may not perform well on a given learning algorithm. In addition, the feature selection method used in the filter model is another aspect to consider when the filter model is used. Individual feature evaluation is used since it is less costlier compared to feature subset evaluation. This approach was recommended by Arauzo-Azofra et al. (2011) in order to better standardise the evaluation of the machine learning algorithms and features with one another. The downside in this approach lies on the cost required to evaluate a feature since the learning algorithm needs to be ran each time to evaluate it. As for the evaluation of the model produced, it will be evaluated using accuracy, precision, recall, f-score, and the Kappa statistic.

These results will then be stored. The process repeats after this but a different combination of feature and algorithm is chosen. The pseudocode of this phase can be found on Algorithm 2.

The complexity of this phase is $O(n \times m)$. n indicates the number of features and m indicates the number of machine learning algorithms.

Algorithm 2 Training phase pseudocode

```
1: procedure TRAIN(ML, F)                    ▷ ML: machine learning algorithms
2:                                                          ▷ F: features
3:     results ← new Array[ML.length][F.length]
4:     for each ml ∈ ML do
5:         for each f ∈ F do
6:             add to results the evaluation of ml using f
7:         end for
8:     end for
9: end procedure
```

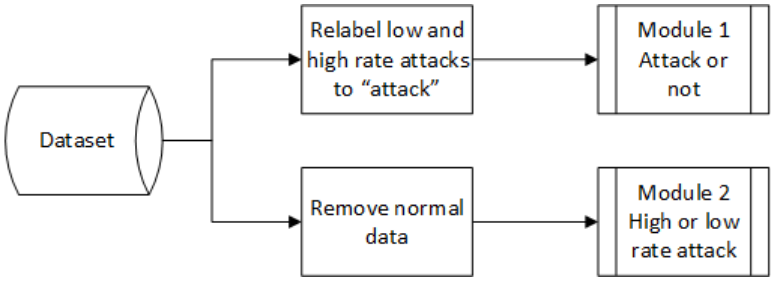

Figure 4.2: Training phase flowchart

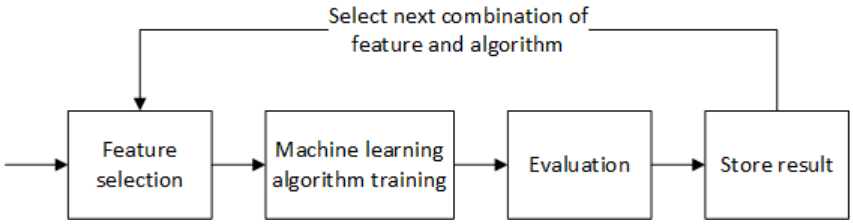

Figure 4.3: Training phase module flowchart

4.3.2 Testing Phase

The testing phase flowchart is presented in Figure 4.4. Network traffic is taken as input and then it is preprocessed. Features are then extracted form the

preprocessed data. The features to be extracted are the ones that were selected during the training phase. The features extracted in this phase are for both modules. Afterwards, the extracted features are given to the first machine learning algorithm. If the instance is classified as normal, the program will stop processing the current instance and start to process the next input. If the instance is classified is an attack, the attack traffic is given to the second machine learning algorithm. This algorithm classifies the instance either as a low rate or high rate DDoS attack. Once the final classification is made, the program will then execute the next input.

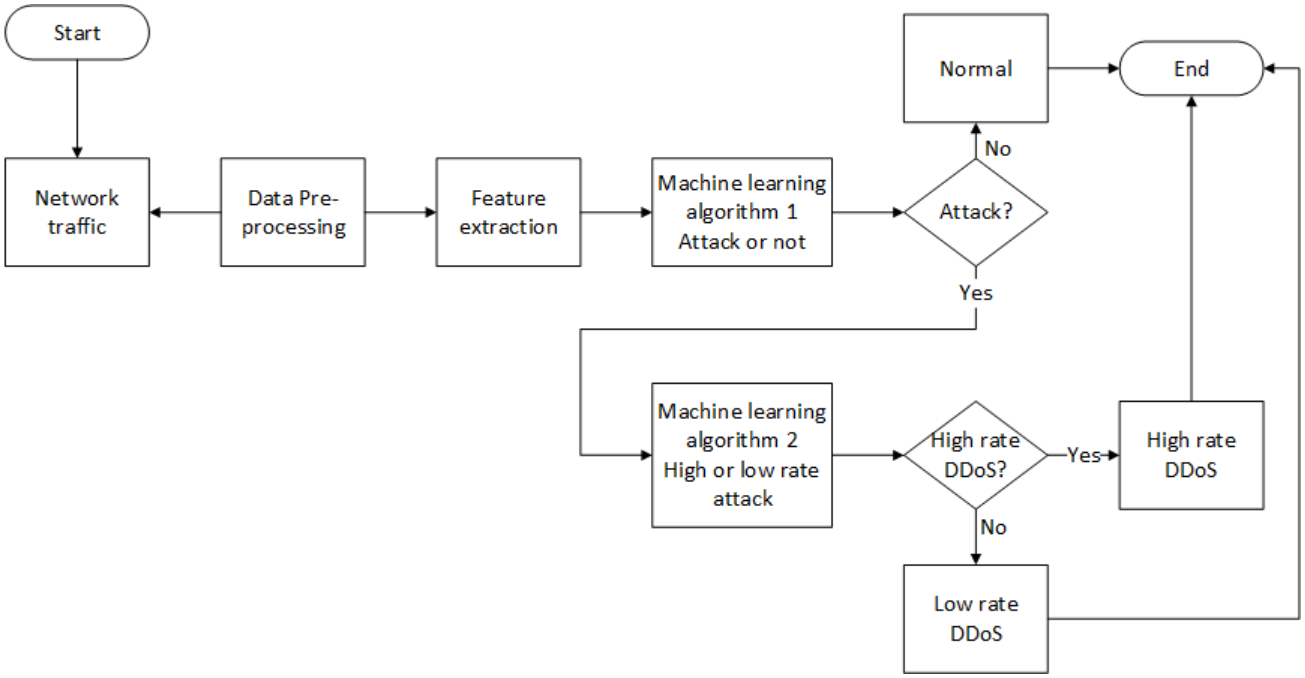

Figure 4.4: Testing phase flowchart

Appendix A

Attacks and Features in the KDD and NSL KDD Datasets

Table A.1: Attacks

Attack category	Specific attack name
DoS	Back
	Land
	Neptune
	Pod
	Smurf
	Teardrop
Probe	Ipsweep
	Nmap
	Portsweep
	Satan
R2L	Ftp_write
	Guess_passwd
	Imap
	Multihop
	Phf
	Spy
	Warezclient
	Warezmaster
U2R	Buffer_overflow
	Loadmodule
	Perl
	Rootkit

Table A.2: Basic features of individual TCP connections

Feature name	Description	Type
duration	Length (number of seconds) of the connection	Continuous
protocol_type	Type of the protocol, e.g. tcp, udp, etc.	Discrete
service	Network service on the destination, e.g., http, telnet, etc.	Discrete
src_bytes	Number of data bytes from source to destination	Continuous
dst_bytes	Number of data bytes from destination to source	Continuous
flag	Normal or error status of the connection	Discrete
land	1 if connection is from/to the same host/port; 0 otherwise	Discrete
wrong_fragment	Number of "wrong" fragments	Continuous
urgent	Number of urgent packets	Continuous

Table A.3: Content features within a connection suggested by domain knowledge

Feature name	Description	Type
hot	Number of "hot" indicators	Continuous
num_failed_logins	Number of failed login attempts	Continuous
logged_in	1 if successfully logged in; 0 otherwise	Discrete
num_compromised	Number of "compromised" conditions	Continuous
root_shell	1 if root shell is obtained; 0 otherwise	Discrete
su_attempted	1 if "su root" command attempted; 0 otherwise	Discrete
num_root	Number of "root" accesses	Continuous
num_file_creations	Number of file creation operations	Continuous
num_shells	Number of shell prompts	Continuous
num_access_files	Number of operations on access control files	Continuous
num_outbound_cmds	Number of outbound commands in an ftp session	Continuous
is_hot_login	1 if the login belongs to the "hot" list; 0 otherwise	Discrete
is_guest_login	1 if the login is a "guest"login; 0 otherwise	Discrete

Table A.4: Traffic features computed using a two-second time window

Category	Feature name	Description	Type
Same host	count	Number of connections to the same host as the current connection in the past two seconds	Continuous
	serror_rate	% of connections that have "SYN" errors	Continuous
	rerror_rate	% of connections that have "REJ" errors	Continuous
	same_srv_rate	% of connections to the same service	Continuous
	diff_srv_rate	% of connections to different services	Continuous
Same service	srv_count	Number of connections to the same service as the current connection in the past two seconds	Continuous
	srv_serror_rate	% of connections that have "SYN" errors	Continuous
	srv_rerror_rate	% of connections that have "REJ" errors	Continuous
	srv_diff_host_rate	% of connections to different hosts	Continuous

References

Abuadlla, Y., Kvascev, G., Gajin, S., & Jovanovic, Z. (2014). Flow-based anomaly intrusion detection system using two neural network stages. *Computer Science and Information Systems*, *11*(2), 601–622.

Alkasassbeh, M., Al-Naymat, G., Hassanat, A. B., & Almseidin, M. (2016). Detecting Distributed Denial of Service Attacks Using Data Mining Techniques. *International Journal of Advanced Computer Science & Applications*, *1*(7), 436–445.

Arauzo-Azofra, A., Aznarte, J. L., & BenÃŋtez, J. M. (2011). Empirical study of feature selection methods based on individual feature evaluation for classification problems. *Expert Systems with Applications*, *38*(7), 8170–8177.

Arbor Networks. (2017, January). *Arbor NetworksâĂŹ 12th Annual Worldwide Infrastructure Security Report Finds Attacker Innovation and IoT Exploitation Fuel DDoS Attack Landscape.* Retrieved 2017-06-06TZ, from https://www.arbornetworks.com/arbor-networks-12th-annual-worldwide-infrastructure-security-report-finds-attacker-innovation-and-iot-exploitation-fuel-ddos-attack-landscape

Barati, M., Abdullah, A., Udzir, N. I., Mahmod, R., & Mustapha, N. (2014). Distributed Denial of Service detection using hybrid machine learning technique. In *Biometrics and Security Technologies (ISBAST), 2014 International Symposium on* (pp. 268–273). IEEE.

Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2014). Information metrics for low-rate DDoS attack detection: A comparative evaluation. In *Seventh International Conference on Contemporary Computing, IC3 2014, Noida, India, August 7-9, 2014* (pp. 80–84). doi: 10.1109/IC3.2014.6897151

Boroujerdi, A. S., & Ayat, S. (2013). A robust ensemble of neuro-fuzzy classifiers for DDoS attack detection. In *Computer Science and Network Technology (ICCSNT), 2013 3rd International Conference on* (pp. 484–487). IEEE.

CAIDA. (2007). *The CAIDA UCSD "DDoS Attack 2007" Dataset.* Retrieved 2017-06-12TZ, from https://www.caida.org/data/passive/ddos-20070804_dataset.xml

CAIDA. (2008). *The CAIDA UCSD Network Telescope "Three Days Of Conficker.* Retrieved 2017-06-14TZ, from http://www.caida.org/data/passive/telescope-3days-conficker_dataset.xml

CAIDA. (2013). *The CAIDA Anonymized Internet Traces 2013.* Retrieved 2017-06-12TZ, from https://www.caida.org/data/passive/passive_2013_dataset.xml

CAIDA. (2017). *CAIDA: Center for Applied Internet Data Analysis.* Retrieved 2017-06-12TZ, from https://www.caida.org/home/

Cepheli, Ã., BÃijyÃijkÃğorak, S., & Karabulut Kurt, G. (2016). Hybrid intrusion detection system for ddos attacks. *Journal of Electrical and Computer Engineering*, *2016*.

Chen, Y., Ma, X., & Wu, X. (2013). DDoS Detection Algorithm Based on Preprocessing Network Traffic Predicted Method and Chaos Theory. *IEEE Communications Letters*, *17*(5), 1052–1054.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, *20*(1), 37–46.

Cortes, C., & Vapnik, V. (1995, September). Support-Vector Networks. *Mach. Learn.*, *20*(3), 273–297. doi: 10.1023/A:1022627411411

Depren, O., Topallar, M., Anarim, E., & Ciliz, M. K. (2005). An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert systems with Applications*, *29*(4), 713–722.

Galea, M. (2016). *Decision Trees with Kotlin.* Retrieved 2017-08-07TZ, from http://cloudmark.github.io/Decision-Trees/

Hinchliffe, E. (2016, October). *DDoS cyberattacks are strikingly simple.* Retrieved 2017-06-11TZ, from http://mashable.com/2016/10/21/ddos-attack-dyn-cyberattack/#1uXz5kcfyaq7

Hoque, N., Bhattacharyya, D. K., & Kalita, J. K. (2016). FFSc: a novel measure for low-rate and high-rate DDoS attack detection using multivariate data analysis. *Security and Communication Networks*, *9*(13), 2032–2041.

Horng, S.-J., Su, M.-Y., Chen, Y.-H., Kao, T.-W., Chen, R.-J., Lai, J.-L., & Perkasa, C. D. (2011). A novel intrusion detection system based on hierarchical clustering and support vector machines. *Expert systems with Applications*, *38*(1), 306–313.

Hussain, J., Lalmuanawma, S., & Chhakchhuak, L. (2015). A Novel Network Intrusion Detection System Using Two-Stage Hybrid Classification Technique. *IJCCER*, *3*(2), 16–27.

Imperva Incapsula. (2017). *R.U.D.Y. (R-U-Dead-Yet?).* Retrieved 2017-08-15, from https://www.incapsula.com/ddos/attack-glossary/rudy-r-u-dead-yet.html

Jawhar, M. M. T., & Mehrotra, M. (2010). Design network intrusion detection system using hybrid fuzzy-neural network. *International Journal of Computer Science and Security*, *4*(3), 285–294.

Jean-Philippe, P. (2001). Application of Neural Networks to Intrusion Detection. *SANS Institute*.

John, G. H., Kohavi, R., Pfleger, K., & others. (1994). Irrelevant features and the subset selection problem. In *Machine learning: proceedings of the eleventh international conference* (pp. 121–129).

Kali by Offensive Security. (2017). *Our Most Advanced Penetration Testing Distribution, Ever.* Retrieved 2017-09-04TZ, from https://www.kali.org/

Kendall, K. K. R. (1999). *A database of computer attacks for the evaluation of intrusion detection systems* (Unpublished doctoral dissertation). Massachusetts Institute of Technology.

Kenig, R. (2013, June). *Why Low & Slow DDoS Application Attacks are Difficult to Mitigate | Radware Blog.* Retrieved 2017-06-29TZ, from https://blog.radware.com/security/2013/06/why-low-slow-ddosattacks-are-difficult-to-mitigate/

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial intelligence*, *97*(1-2), 273–324.

Korczynski, M., Janowski, L., & Duda, A. (2011). An accurate sampling scheme for detecting SYN flooding attacks and portscans. In *Communications (ICC), 2011 IEEE International Conference on* (pp. 1–5). IEEE.

Kumar, P. A. R., & Selvakumar, S. (2013). Detection of distributed denial of service attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems. *Computer Communications*, *36*(3), 303–319.

Li, C., Yang, J., Wang, Z., Li, F., & Yang, Y. (2015). A Lightweight DDoS Flooding Attack Detection Algorithm Based on Synchronous Long Flows. In *Global Communications Conference (GLOBECOM), 2015 IEEE* (pp. 1–6). IEEE.

Li, K., Zhou, W., Li, P., Hai, J., & Liu, J. (2009). Distinguishing DDoS attacks from flash crowds using probability metrics. In *Network and System Security, 2009. NSS'09. Third International Conference on* (pp. 9–17). IEEE.

Lin, S.-W., Ying, K.-C., Lee, C.-Y., & Lee, Z.-J. (2012). An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection. *Applied Soft Computing*, *12*(10), 3285–3290.

Lincoln Laboratory. (1999). *Intrusion Detection Attacks Database.* Retrieved 2017-08-06TZ, from https://www.ll.mit.edu/ideval/docs/attackDB.html

Lincoln Laboratory. (2000). *DARPA Intrusion Detection Evaluation; MIT Lincoln Laboratory.* Retrieved 2017-02-08TZ, from https://www.ll.mit.edu/ideval/data/

Liu, S. (2009). Surviving distributed denial-of-service attacks. *IT professional*, *11*(5).

Mitchell, T. M. (1997). *Machine Learning* (1st ed.). New York, NY, USA: McGraw-Hill, Inc.

Neethu Raj, P., Babu, S. S., & Nishanth, N. (2015). A Novel Syn Flood Detection Mechanism for Wireless Network.

ns2blogger. (2017). *INTRODUCTION TO NS2.* Retrieved 2017-06-21TZ, from http://www.ns2blogger.in/p/introduction-to-ns2.html

nsnam. (2017). *What is ns-3.* Retrieved 2017-06-21TZ, from https://www.nsnam.org/overview/what-is-ns-3/

Numerical Algorithms Group. (2017). *(K Nearest Neighbor).* Retrieved 2017-08-07TZ, from http://www.nag-j.co.jp/nagdmc/knn.htm

OpenCV. (2017). *Introduction to Support Vector Machines âĂŤ OpenCV 2.4.13.3 documentation.* Retrieved 2017-08-07TZ, from http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

Peddabachigari, S., Abraham, A., Grosan, C., & Thomas, J. (2007). Modeling intrusion detection system using hybrid intelligent systems. *Journal of network and computer applications*, *30*(1), 114–132.

Peng, T., Leckie, C., & Ramamohanarao, K. (2004). Proactively detecting distributed denial of service attacks using source IP address monitoring. In *International Conference on Research in Networking* (pp. 771–782). Springer.

Prabowo, R., & Thelwall, M. (2009). Sentiment analysis: A combined approach. *Journal of Informetrics*, *3*(2), 143–157.

Radware. (2017a). *Apache Killer Attack - Apache Server Vulnerabilities.* Retrieved 2017-08-15TZ, from https://security.radware.com/ddos-knowledge-center/ddospedia/apache-killer/

Radware. (2017b). *High-rate attack.* Retrieved 2017-06-29TZ, from https://security.radware.com/ddos-knowledge-center/ddospedia/high-rate-attack/

Rao, S., & Rao, S. (2011). Denial of service attacks and mitigation techniques: Real time implementation with detailed analysis. *This paper is from the SANS Institute Reading Room site*.

Richard P. Lippmann, R. K. Cunningham, D. J. Fried, S. L. Garfinkel, A. S. Gorton, I. Graf, ... M. A. Zissman (1998, December). *MIT Lincoln Laboratory Offline Component of DARPA 1998 Intrusion Detection Evaluation.* Lincoln Laboratory. Retrieved 2017-09-16, from https://ll.mit.edu/ideval/files/LLab-1-Intro.ppt

Roesch, M. (1999). Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the 13th USENIX Conference on System Administration* (pp. 229–238). Berkeley, CA, USA: USENIX Association.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, *33*(1), 1–39.

Sammany, M., Sharawi, M., El-Beltagy, M., & Saroit, I. (2007). Artificial neural networks architecture for intrusion detection systems and classification of attacks. In *The 5th international conference INFO2007* (pp. 24–26).

Seo, J., Lee, C., Shon, T., Cho, K.-H., & Moon, J. (2005). A new DDoS detection model using multiple SVMs and TRA. In *Embedded and Ubiquitous ComputingâĂŞEUC 2005 Workshops* (pp. 976–985). Springer.

Shekyan, S. (2012, January). *Are you ready for slow reading?* Retrieved 2017-08-15TZ, from https://blog.qualys.com/securitylabs/2012/01/05/slow-read

Shekyan, S. (2017, August). *slowhttptest Application Layer DoS attack simulator.* Retrieved from https://github.com/shekyan/slowhttptest (original-date: 2015-03-15T00:57:46Z)

Siris, V. A., & Papagalou, F. (2006, May). Application of Anomaly Detection Algorithms for Detecting SYN Flooding Attacks. *Comput. Commun.*, *29*(9), 1433–1442. doi: 10.1016/j.comcom.2005.09.008

Sta. Ana, D. J. (2017, January). *At least 68 govt web sites attacked following UN Arbitral court ruling.* Retrieved 2017-06-06TZ, from http://web.archive.org/web/20170111234608/http://interaksyon.com/article/130387/at-least-68-govt-web-sites-attacked-following-un-arbitral-court-ruling

Stanford University. (2017). *CS231n Convolutional Neural Networks for Visual Recognition.* Retrieved 2017-08-07TZ, from http://cs231n.github.io/neural-networks-1/

Stolfo, S. J., Fan, W., Lee, W., Prodromidis, A., & Chan, P. K. (2000). Cost-based Modeling for Fraud and Intrusion Detection: Results from the JAM Project. In *In Proceedings of the 2000 DARPA Information Survivability Conference and Exposition* (pp. 130–144). IEEE Computer Press.

Sun, C., Hu, C., Tang, Y., & Liu, B. (2009). More accurate and fast SYN flood detection. In *Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on* (pp. 1–6). IEEE.

Sun, Q., Wang, S., Yan, D., & Yan, F. a. (2009). ARM-CPD: detecting SYN flooding attack by traffic prediction. In *Broadband Network & Multimedia Technology, 2009. IC-BNMT'09. 2nd IEEE International Conference on* (pp. 443–447). IEEE.

Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., & Ghogho, M. (2016, October). Deep learning approach for Network Intrusion Detection in Software Defined Networking. In *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)* (pp. 258–263). doi: 10.1109/WINCOM.2016.7777224

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A Detailed Analysis of the KDD CUP 99 Data Set. In *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications* (pp. 53–58). Piscataway, NJ, USA: IEEE Press.

Ubuntu Desktop. (2017). *Download Ubuntu Desktop | Download | Ubuntu.* Retrieved 2017-09-04TZ, from https://www.ubuntu.com/download/desktop

UCI Knowledge Discovery. (1999). *KDD-CUP-99 Task Description.* Retrieved 2017-08-08TZ, from https://kdd.ics.uci.edu/databases/kddcup99/task.html

Wireshark. (2017). *Wireshark.* Retrieved 2017-09-04TZ, from https://www.wireshark.org/

Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, *5*(2), 241–259.

Wood, J. M. (2007). Understanding and Computing CohenâĂŹs Kappa: A Tutorial. *WebPsychEmpiricist. Web Journal at http://wpe. info/*.

Xiang, Y., Li, K., & Zhou, W. (2011). Low-rate DDoS attacks detection and traceback by using new information metrics. *IEEE Transactions on Information Forensics and Security*, *6*(2), 426–437.

Yan, X., & Zhang, J. Y. (2013). *Early Detection of Cyber Security Threats Using Structured Behavior Modeling - Semantic Scholar.*

You, Y. (2007). *A defense framework for flooding-based ddos attacks* (Unpublished doctoral dissertation).

Yu, L., & Liu, H. (2004). Efficient feature selection via analysis of relevance and redundancy. *Journal of machine learning research*, *5*(Oct), 1205–1224.

Zhang, C., Cai, Z., Chen, W., Luo, X., & Yin, J. (2012). Flow level detection and filtering of low-rate DDoS. *Computer Networks*, *56*(15), 3417–3431.