KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY,

KUMASI, GHANA

Network Intrusion Detection and Countermeasure Selection in Virtual Network (NIDCS)

By

Maxwell Cobbah (B.E. Electronics and Communications Engineering)

PG7765812

A Thesis submitted to the Department of Electrical and Electronic Engineering,

College of Engineering

In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

(Telecommunication Engineering)

OCTOBER 2015

DECLARATION

I hereby declare that, except for specific references which have been duly acknowledged, this submission is my own work towards the MSc and that to the best of my knowledge, it contains no material previously published by another person or material which has been accepted for the award of another degree of the University.

KNUST

Maxwell Cobbah	……………………….	…………………………
(Student)	Signature	Date

Certified by:

Dr. J. D. Gadze	……………………….	…………………………
(Supervisor)	Signature	Date

Certified by:

Mr. E. A. Frimpong	……………………….	…………………………
(Head of Department)	Signature	Date

ACKNOWLEDGEMENT

First and foremost, I would like to thank my loving Creator for making me a curious being who loves to explore His creation and for giving me the opportunity to write this thesis. Without Him, I can do nothing.

I would like to express my sincere gratitude to my research supervisor, Dr. J. D. Gadze for his support during this thesis period. The inexhaustible supply of help, advice and most importantly patience has been invaluable in the completion of this thesis. I am deeply grateful to Dr. Griffith Selorm Klogo, KNUST, Computer
Engineering Department for his great support and encouragement. I would also like to Thank Professor Willie Ofosu, Penn State University USA, who gives his endless support and encouragement. In addition, I wish to thank Carlos Alberto Lozano Espinosa, Associate Professor, Electronic and Computer Science Department, Pontificia Universidad Javeriana, Colombia, for his guidance and technical support throughout this research.

I would like to thank my family, in particular my mother Margaret Twumasi for her effort and prayer to let my dream in education come true and not forgetting my children Benedict O. Cobbah, Bernard O. Cobbah and Bevelyn O. Cobbah for the happiness they created in my life to have sound mind to complete this thesis.

Finally, I would like to thank my course mates and all persons who in diverse ways contributed to the success of this project.
God richly bless you all.

KNUST
NYANSAPO WOSANE NO BADWENMA

DEDICATION

This thesis is dedicated to my family especially my wife Janet Amakye and my children Benedict Oduro Cobbah, Bernard Oduro Cobbah and Bevelyn Owusua Cobbah.

KNUST

NYANSAPƆ WƆSANE NO BADWENMA

ABSTRACT

Intrusion in a network or a system is a problem today as the trend of successful network attacks continue to rise. Intruders can explore vulnerabilities of a network system to gain access in order to deploy some virus or malware such as Denial of Service (DOS) attack. In this work, a frequency-based Intrusion Detection System (IDS) is proposed to detect DOS attack. The frequency data is extracted from the time-series data created by the traffic flow using Discrete Fourier Transform (DFT). An algorithm is developed for anomaly-based intrusion detection without any false alarm which further detect known and unknown attack signature in a network. The frequency of the traffic data of the virus or malware would be inconsistent with the frequency of the legitimate traffic data. A Centralized Traffic Analyzer Intrusion Detection System called CTA-IDS is introduced to further detect inside attackers in a network. The strategy is effective in detecting abnormal content in the traffic data during information passing from one node to another and also detects known attack signature and unknown attack. This approach is tested by running the artificial network intrusion data in simulated networks using the Network Simulator2 (NS2) software.

KNUST
NYANSAPO WO SANE NO BADWENMA

TABLE OF CONTENTS

KNUST

KNUST
NYANSAPO WO SANE NO BADWENMA

KNUST

KNUST

LIST OF TABLES

LIST OF FIGURES

KNUST

NYANSAPƆ WƆSANE NO BADWENMA

CHAPTER ONE

INTRODUCTION

1.1 Background of the study

Today, many organizations are relying on their virtual networks to process information at a minimal cost and at a faster rate to yield high quality in their work. As the cost of information processing falls, organizations are becoming increasingly vulnerable to potential cyber threats such as network intrusion [1], [2]. Network intrusion is growing very fast these days and it is generating a serious threat to digital and electronic information hungry society. The quality of the attack techniques is also experiencing an exponential growth. Some few years ago, only skilled professionals or system experts could break into a computer or network resources; today a neophyte or an ordinary Internet surfer can launch an attack on a network using easily downloaded tools. This necessitates the use of Network Security to defend network systems against intrusion.

Network Security is the property of computer systems and networks that prevents intruders from unauthorized access to the network [3]. It can also monitor the network activities against modification of network resources. The network access can be controlled by the network administrator. The network administrator is the only authoritative user who ensures the goals of the security of the network. The goals of a network security include confidentiality, integrity, control, authenticity, availability, and utility. The 'confidentiality' property ensures only authoritative users can read or copy a file or object. The 'Integrity' property allows only authoritative users to alter or delete a file or object. The 'control' property allows only authoritative users to decide when to allow access to information. The ‘authenticity’ property is the correctness of attribution or description. The ‘availability’ property also means no unauthorized user can deny authorized users timely access to files or other system resources and the ‘utility’ property denotes the fitness for a specified purpose [4], [5], [6].

The recent attack trend has put many security challenges to network administrators. DOS attack is increasingly becoming a major challenge for network administrators to handle, a prevalent threat and a leading Internet attack today which is easy to launch.

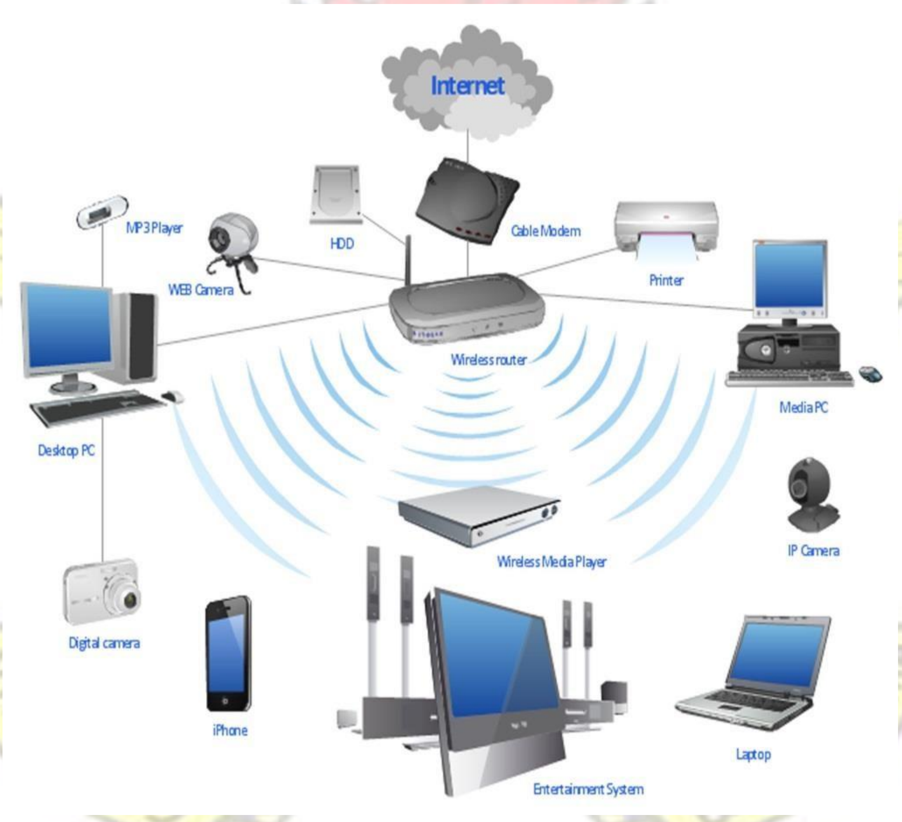


Figure 1.1: Wired and wireless Internet connection for many users [7]

Figure 1.1 shows a typical network setup having various addressable entities or network devices connected together through wired and wireless medium with internet connectivity and to deny access to any of these addressable entities, network resources would be unavailable and this can be caused by a DOS attack.

DOS can occur in a network system when victim network resources are barraged with high amount of fabricated attacking packets devised from a single or a large number of machines. The aim of the attack is to overload or send an abnormal high volume of requests to the servers that cause the server resources to collapse or to disrupt the network services and render the legitimate user incapable of performing normal transactions. This is done by exploiting essential vulnerabilities in the network and rapidly becoming the weapon of choice for hackers around the globe. DOS attack has been developed from the simplest direct flooding attacks, Distributed Denied of Service (DDOS) attack and to remote controlled Reflexive Distributed Denied of Service (RDDOS) attacks. Although the technique of DOS attacks is relatively simple, it can attack both the Internet and system resources.

In February 2014 a new DOS attack type was introduced called Network Time Protocol (NTP) Amplification attack; the biggest so far reaching 180Gbps which means volume of attack packets continue to grow. Such network based attacks are increasing frequently resulting in a huge financial loss to the organizations and causing the network to be paralyzed for several hours [8]. The cost of malicious cyber activity is mainly related to the theft of intellectual property and the loss of financial assets [9].

The global impact of malicious code attacks is estimated to be over US $113 billion annually according to Norton Cybercrime report in 2012. In 2013, the McAfee security firm also estimated that cybercrime and cyber espionage are costing the US economy US $100 billion per year, and the global impact is nearly US$300 billion annually [10], [11]. These challenges have become an urgent problem for researchers.

Since the extent of financial loses and damage by DOS attacks are increasing, many studies on Intrusion Detection (ID) mechanism have been carried out by many security experts. ID includes a range of security techniques designed to detect and report malicious system and network activity or to record evidence of intrusion. A tremendous effort has also been made by experts to design very efficient and effective Intrusion Detection System (IDS) which is the main security tool leading in the field of cyber defense currently.

An IDS is a software that automate the intrusion detection process with the primary aim of detecting unwanted and malicious activities in the network system. IDS are used to collect information from various systems and some sources of networks and then analyze the information for the attacks arriving from outside and inside of the organization. An IDS will signal the computers to prepare themselves for attacks and helps the firewalls of the network to enhance the security management like screening, attack identification, security audit and response. It can also perform the following operations:

- Supervise, display and analyze the content coming through internet.
- Take care of system configurations and any type of vulnerabilities.
- Analyze the reliability of decisive system and files which contain data. It can report if any changes are happening in the data files.
- Recognize the known and unknown attacks.
- Continuously monitor the network activities for abnormal behavior.
- Detect the nonfunctional activities of OS (Operating Systems) while attackers attack the system through the internet.

Though many researches' have been carried out extensively in this area, DOS attacks continue to harm, as the attackers adapt to the newer protection mechanisms. The existence of system vulnerabilities in the traffic data during information passing from one node to another in a network is unavoidable simply because of human nature. As long as humans make mistakes, such as inappropriate system configuration, bugs in the codes, poorly designed network architecture, security violations will continue to exist. In fact, many of these system vulnerabilities are discovered only after being attacked.

In order to maintain confidentiality, integrity, control, authenticity, availability, and utility of the network resources, there is the need to introduce an effective and efficient approach for network intrusion detection as an effective countermeasure for various virtual network attacks.

1.2 Research Problem

The trend of successful network attacks such as DOS attack continues to rise. These attackers can explore vulnerabilities of a network system to gain access in order to deploy sophisticated attacks. When the security mechanisms at the external parameters of a network are strong, the internal part of the network becomes soft and therefore the vulnerabilities in the traffic data become high. Many algorithms are developed and implemented in order to achieve intrusion free network. But most of these algorithms are using time-based approach and they are not so accurate to detect and destroy the malicious content in the network system.

The time-based method is not effective to detect inside attackers of a network though many techniques are used such as data mining approaches introduced in neural network [12], [13] and Bayesian classifiers. It is also difficult for such techniques to calculate the accurate threshold value above which an abnormality is to be considered 'intrusive' and therefore lead to high rate of false alarm. Can we sit down for network intruders to take over our networks due to system vulnerabilities? Obviously no, an appropriate system suitable in monitoring network activities with low rate of false alarm is needed to discourage cyber-attacks.

1.3 Research Objectives

1.3.1 General Objective

The General objective of this work is to develop a frequency-based Intrusion Detection System (IDS) that can detect Denied of Service (DOS) attack.

A Centralized Traffic Analyzer Intrusion Detection System called CTA-IDS is introduced. This CTA-IDS combines two important strategies: traffic visualization and connection-based activities analysis in order to detect the network intrusion. This will help to reduce false alarm rate. 1.3.2 Specific Objectives

The following are the specific objectives of this thesis:

1. Develop an intrusion algorithm for both anomaly-based and signature-based detection techniques.
2. Derive a mathematical model to reduce false alarm rate during detection process.

1.4 Motivation

There are several IDS techniques for host-based and network-based intrusion detection as well as anomaly-based and signature-based intrusion detection techniques available and they all proved to be very efficient and effective to detect intrusions in a network. However, there are some drawbacks despite the extensive research in recent years. The following are some of the drawbacks considered in the existing IDS;

1. The signature-based intrusion detection cannot detect unknown attacks.
2. The anomaly-based intrusion detection can detect both known and unknown attacks but generate high rate of false alarm.

A signature-based IDS technique determines intrusion by comparing packets on the network against a database of signatures or characteristic from known malicious threats [14]. Basically, this type of IDS looks for a specific attack that has already been documented. Like a virus detection system, detection software is only as good as the database of intrusion signatures used to compare against packets. This technique is capable of detecting only known attacks and cannot detect unknown attacks.

Anomaly-based (behavior-based) IDS technique references a baseline or learned pattern of normal system activity to identify active intrusion attempts. Deviations from this baseline or pattern cause an alarm to be triggered. In this technique the false alarm rate is high which makes it unreliable for practical applications. This encourages the use of frequency-based intrusion technique for network intrusion design.

1.5 Project Scope

This research focuses on the development of IDS that can detect DOS attack in a virtual network with no or low rate of false alarm and also select an appropriate action to isolate or disconnect malicious nodes in the network. This approach is effective in detecting only brute force attacks such as DOS and probe attacks which exhibit relatively long attack duration. In this work, Network Simulator version2 (NS2) is used to simulate the intrusion traffic for the evaluation of the intrusion detection algorithm.

1.6 Outline of the study

The remainder of this thesis is organized as follows:

Chapter 2 reviews the different work in the area of intrusion detection. The work for intrusion detection is organized into four categories: host-based, network-based, specific attack type and data mining approaches.

Chapter 3 describes the proposed system model and the methodology.

Chapter 4 presents the assessments of the methods described in chapter three. We give the experimental evaluation result and demonstrated by means of simulation.

Chapter 5 is the concluding part of the report and it gives an overview on the challenges faced by today's IDS. In conclusion, we summarize our work and point out the limitations and future directions.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter reviews the related work in the area of network intrusion detection. The chapter presents a general concept of Intrusion Detection System (IDS) and also classifies the detection techniques into three different levels. The first level talks about the existing intrusion detection techniques from a general perspective, covering host-based, network-based, and certain specific attack type detection techniques. The next level focuses on the network-based techniques. The final level, concentrate on data mining approach.

2.2 General Overview of Intrusion Detection System

An Intrusion Detection System (IDS) monitors network traffic and monitors for suspicious activity and alerts the system or network administrator. Currently, an IDS has been considered as an effective countermeasure for a variety of attacks. One major limitation of current IDS technologies is the requirement to filter false alarms lest the network administrator be overwhelmed with data. An IDS is classified in many different ways, including active and passive, network-based and host-based, and knowledge-based and behavior-based.

An active IDS is also known as an Intrusion Prevention System (IPS) and it is a system that is configured to automatically block suspected attacks in progress without any intervention required by the network administrator. IPS can provide a real-time corrective action in response to an attack but susceptible to attack itself. A passive IDS is configured only to monitor and analyze network traffic activity and alert the network administrator to potential vulnerabilities and attacks. It is not capable of performing any protective or corrective functions on its own.

A network-based IDS usually consists of a network sensor with a Network Interface Card (NIC) operating in promiscuous mode and a separate management interface. The IDS is placed along a network segment or boundary and monitors all traffic on that segment. A host-based IDS requires small programs or agents to be installed on individual systems to be monitored. The agents monitor the operating system and write data to log files and/or trigger alarms. A host-based IDS can only monitor the individual host systems on which the agents are installed; it doesn't monitor the entire network.

A knowledge-based (or signature-based) IDS references a database of previous attack profiles and known system vulnerabilities to identify active intrusion attempts. Knowledge-based IDS is currently more common than behavior-based IDS. It has lower false alarm rates than behavior-based IDS but cannot detect novel attacks. A behavior-based (or statistical anomaly–based) IDS references a baseline or learned pattern of normal system activity to identify active intrusion attempts. Deviations from this baseline or pattern cause an alarm to be triggered. This method can detect novel attacks but its false alarm rate is high.

2.3 Host-based Intrusion Detection

Data mining techniques have been widely used for both host-based and networkbased intrusion detection. Schultz et al in [15] developed a host-based approach which adapted the data mining technique to detect malicious executable. A malicious executable file needs to request system resources to finish it task. A system resource includes Dynamic Link Library (DLL) files, DLL functions, and other function calls within a DLL. The DLLs called by malicious executable show different patterns from those called by the normal executable. Beside the system resources, the strings extracted from executable code and the binary byte sequences are also used to distinguish malicious code from the normal ones.

With the exception of the host-based approach which was adapted in [15], other methods for detecting the abnormal behavior of malicious code are reported in [16], [17], [18], [19]. These methods build normal profiles using the system calls executed by programs; the differences lies in how to build the normal profile. The earliest work in program behavior based intrusion detection is from Forrest et al [16]. Their approach was to build a signature database with each signature composed of a short sequence of system calls. An alarm is triggered if the execution of a program (application) creates enough signatures that are different from the ones in the database. The success of this method brings more attention to the use of system calls as the source for host-based intrusion detection. Ghosh et al uses multiple machine learning techniques including neural network and Elman network to ”learn” the normal program behavior from the system calls and to further perform anomaly detection [17]. Ko employs another machine learning method: inductive logic programming to build the normal program profile for anomaly detection [18]. Sekar et al proposed an automatic and efficient method to build a finite-state automaton which distinguishes anomalous sequence of system calls from legitimate ones [19]. Abad et al showed that many attacks leave traces in different logs, thus correlating the logs could help to improve the precision of intrusion detection. The logs they used for experiment include NetFlow log, firewall log, system log, TCPDUMP log, DNS log, authentication log, web log, mail log and

FTP log. Once again, data mining technique is applied to these logs in generating the rules for intrusion detection [20].

In [21], Coull et al applied the technique from bioinformatics, specifically, a parewise sequence alignment for the similarity match, to the problem of matching user signatures. A user's signature is a sequence of commands produced by a user. An attacker could be identified by comparing his/her signature with legitimate user's signature. Another early work that also conducts the intrusion detection from the perspective of what an illegitimate user's action should be is [22]. In [22], Ilgun et al introduced State Transition Analysis and this is developed for an easily readable representation for computer penetrations. This approach models penetration as a series of state transitions described in terms of signature actions and state assertions which makes use of user behaviors in a host-based intrusion detection system. The differences are that a user's signature is represented by abstracted actions instead of commands, and a snapshot of the system status under the effect of user actions is defined as a state.

2.4 Specific Attack Type Intrusion Detection

A network can be subjected to attack if security measures and controls are not in place. A network attack can be considered as any method, process, or means used to maliciously attempt to compromise network security. This section reviews the techniques that are specially designed for certain type of attacks or malicious activities that network attackers perform. This specific attack types include DOS, worms, backdoor, and Probes.

2.4. 1 Denial of Service (DOS)

Denial of Service and Distributed Denial of Service (DOS/DDOS) attacks, which exploit vulnerabilities in the network resources, are chosen by the hackers as their weapon around the globe. DOS attack is easy to launch using readily available tools against network resources to paralyze a network. This is done by flooding the network systems with useless traffic, preventing legitimate transactions from completing.

The growing dependence on network systems makes the impact of these attacks increasingly painful for service providers, enterprises, hosting centers and government agencies. DOS attacks are already among the most difficult to defend against, and newer, more powerful tools promise to unleash even more destructive attacks in the months and years to come [23]. Due to its easy deployment and devastating outcome, DOS attack has attracted a lot of attentions from the defenders as well as attackers. At the attacker side, the technique of DOS is developed from a single launch to multiple launches and finally to reflexive multiple launches. At the defense side, a variety of techniques have been developed to counter it.

In [24], Feinstein et al detect the DOS by measuring the statistical properties of specific fields in the packet headers. Specifically, they take Entropy value and Chisquare value of certain fields of packet header to measure the randomness of the distribution of that field (such as source IP). Those values of DOS traffic would fall outside the range of normal value from the legitimate traffic.

Cheng et al [25] explores the fact that under the normal situation of the TCP protocol, a source will wait to send the next group of packets until it has received acknowledgement on the last group of packets (group size is based on sliding windows size). Thus, the packets are roughly transmitted every round-trip time. This is created by the fundamental behavior of TCP: to be reliable. In contrast, a DOS attacker would not care if the destination received the last packet and continued to send the packets unrelatedly. Therefore, a normal TCP flow will have stronger periodicity in the packet arrivals, while an abnormal TCP flow will not. Based on this observation, Cheng et al [25] applies power spectral density analysis on the signal of packet arrival and the result of it indicates how strong the periodicity is. The signal with less strong periodicity is marked as suspicious.

Gil and Poletto [26] proposed another DOS detection strategy that is based on the observation that during the attack, the packet rate of ongoing traffic is disproportional to that of incoming traffic. By monitoring the packet rate at the routers as near as possible to the attacker, their system, MULTOPS, is expected to stop the attack before they cause the actual harm. Gil and Poletto's approach is essentially a network-based approach that employs statistical analysis on network traffic for DOS attack.

The countermeasures to DOS attack are not restricted to the context of general networks. Morein et al [27] designed a DOS defense strategy with an overlay network as the background. The original idea is from Keromytis et al [28]. As a novel application of the overlay network in the area of network security, they propose

"SOS", which creates a new mechanism for preventing the DOS attacks by making redundancies in the overlay. Different from the traditional passive detection approach for a DOS attack, "SOS" will not do anything when detecting the malicious DOS traffic and will not attempt to disable the source of the attack. Instead, "SOS" adopts the proactive approach which opens the door for the attacks but tries to moderate the attack effect and reduces the possibility of the success attacks by making an adequate amount

of duplication of overlay nodes, namely SOAP (Simple Object Access Protocol) a Beacon and Secret Servlet nodes, use to eliminate the “pinch” points. In addition, they utilize the Chord routing mechanism to quickly recover from the nodes failure upon the attack. Another DOS detection strategy that is grounded on the overlay network is introduced by Andersen [29]. Andersen makes modifications to
“SOS” model so that the new model is more reflexive and has better performance. The main improvement is to separate the routing and the packet filtering. In the original “SOS” model, the filtering is done by the routers that are set in the deny mode which protects the target from any other traffic except for the traffic from the Secret Servlet (SS).

Other anti-DOS approaches focus on SYN (Synchronous) flooding attack [30], [31], [32]. A SYN flooding will open many partially completed connections with a victim and the victim will eventually exhaust all its system resources by trying to allocate resources to each half-open connection. Lemon’s approaches [30] for SYN attack is to let the system allocate the resource only to the completed connections (after three-way hand shaking). Moreover, the receiver would not allocate the resource but instead send a cryptographic secret “SYN cookie” to traffic initiator, where all the TCP options with initial SYN are encoded in the cookies and sent to the network. The idea is to let network “store” the state, thus the machine could save the resources for storing the connection state.

The defense mechanism in [30] is implemented at the place close to or within the victim machine. Therefore, if the defenders are interested at the attack source, it could even come from zombies, the extra IP trace back method such as in [31] will be needed. An alternative approach known as Flooding Detection System (FDS) is to defend the attack at the leaf routers that connect end host to the Internet [32]. Wang et al argues that the number of SYN packets and FIN (RST – Reset) packets in normal

TCP is, if not equal; close whereas the difference between the number of SYNs and FINS will dramatically increase in a SYN flooding attack. By counting the number of those flags at the leaf router, they could detect the SYN flood. However, in order to find the attack source and effectively detect the attack simultaneously, Wang et al FDS has to be installed at the routers close to the attacker and at the routers close to the victim. Therefore without the pre-knowledge about where the attacks might come from, the wide deployment of FDS is required.

2.4.2 Internet Worms

Internet worm is considered as one of the malicious software or an independent virtual virus that replicates itself and distributes copies to its network. It break into computers, and replicate without intervention from and unbeknownst to computer users. The propagation of various worms on the Internet has caused widespread damage. It’s completely autonomous spreading has created sensational and disruptive attacking eff ects as evident from the recently appeared worms Code Red and Nimda [33]. The eff orts of countermeasures have been made in two aspects: first, to analyze the existing worms ‘spreading; second, to defend them [34], [35], [36], and [37]. Unfortunately, the conclusions from the analysis are that future worm spreading could be worse and the current defense techniques are not effective for future worms [38], [39], [40], [41].

In [39], Staniford et al focus on the various target selection techniques that appear and can appear in the existing worms and future worms. The target selecting method directly decides the epidemic speed of worm spreading. If a worm could generate the target's IP appropriately by using the right random number generation algorithm to avoid the repetition, it could dramatically increase its spreading effects. Staniford et al, even further propose several feasible techniques that, if deployed by the attacker, could increase the scan rate. The essential of those techniques, such as permutation scanning, hit-list scanning, and topologically aware, avoid the repetitions and invalidate IPs and try to infect the nearest host first.

Simulations of worm spreading are reported in [38], [40], [42]. Moore et al [38] builds the simulation model which focuses on parameters of reaction time, containment strategy, and deployment scenario. Chen et al [40] models the worms' propagation under the parameters of the number of vulnerable machine, size of hitlist, scanning rate, death rate, and patching rate. Both of their results show that the spreading of a proper designed worm can be out of control.

On the defense side, Berk and Bakos [34] observes that during the worm spreading, a relatively high volume of Internet Control Message Protocol (ICMP) host unreachable error messages will be generated than the normal situation due to the random target IP selection. Thus, it is possible to detect a worm before it reaches epidemic by monitoring these error messages. Williamson [35] makes use of machine locality property to design a mechanism that could restrict or slow down an already infected machine from further spreading.

A network-based anomaly detection system for worms is proposed by Toth and Kruegel's [36]. Their system architecture includes a firewall on each host, a traffic monitor for each local network and a central analyzer. Each traffic monitor collects certain information from the connection history such as the number of connection attempts to non-existing host or non-existing service which is a good indication of worms' presence.

Sidiroglou and Keromytis [37]'s approach is to automatically generate the patches for the vulnerable software. Their system architecture includes a set of sensors, a detection engine, an analysis engine for patch generation and also a software update component and an actual sandbox that run the applications that are supposed to be protected. The patch is generated heuristically. That is, the code is generated iteratively until a version of the application that is resistant to the worm is found.

2.4.3 Backdoor

A backdoor is a means of access to a computer program that bypasses security mechanisms. A programmer may sometimes install a backdoor so that the program can be accessed for troubleshooting or other purposes. However, attackers often use backdoors that they detect or install themselves, as part of an exploit. In some cases, a worm is designed to take advantage of a backdoor created by an earlier attack. For instance, Nimda gained entrance through a backdoor left by Code Red [33].

In [43], Zhang and Paxson suggest a method for the backdoor detection. Zhang and Paxson argue that the attackers who use the backdoor to control the victim would create some interactive traffic, which has distinct characteristics on packet size and inter-arrival time from the normal traffic flow. Therefore, they set up the algorithms for normal traffic flow that was created by the legitimate service such as Secure Shell (SSH) and File Transfer Protocol (FTP) and look for deviations.

2.4.4 Probes

Probe is the first step that an attacker will take before breaking into a system. Port scan (service querying), Operating System (OS) fingerprint, application vulnerability scan and network mapping are all very common on the internet nowadays. Smart et al [44] designs a fingerprint scrubber which aims at operating system probe (TCP/IP stack fingerprinting). In order to collect the system information, the attacker will send carefully crafted packets to the remote target and watch their responses. These packets make use of the ambiguity of network protocol and cause the different OS to give responses differently because of lack of standard. The attacker then matches the returned responses with the signature database to identify the remote system type. In order to avoid this situation, a protocol scrubber is implemented at a gateway of a Local Area Network (LAN) to the internet. The scrubber will normalize (modify) the packet header at network and transport level to avoid the ambiguity, thereby, prevent the target LAN from receiving the ambiguous packets and system information leaking.

2.5 Network-based Intrusion Detection

Network-based anomaly detection system usually uses the information that can be obtained directly from communication infrastructure for the anomaly detection. The network traffic has been a major source for all kinds of information. The networkbased intrusion techniques can also be categorized into anomaly-based and signaturebased. The examples of signature-based intrusion techniques are [45], [46], [47]. In [45], a state transition model is used as the signature to describe each type of attack scenario. The state transition model, just like a string signature, is defined by the existing attack procedure, and also can be modified and updated as the new attacks emerge. The SNORT and BRO are two well-known signature based IDS. Both of them use a set of flexible rules to describe traffic and search for known attacks patterns.

Another apparent signature is strings. String matching is a simple yet effective method for known attack identification and it is employed by IDS ranged from the very earliest IDS to many current commercial IDS. String matching is a necessary step for the rule set matching. An efficient string matching in heavily loaded fast networks is essential for the good performance of a NIDS. Coit et al [48], Sommer and Paxson [49] are two examples that provide the different string matching enhancement. The first one use SNORT for experimental evaluation while the second one took BRO for experimental evaluation. Coit et al used an adapted Boyer-Moore algorithm to speed up the string matching while Sommer and Paxson use a context related string match method to increase the matching precision. Most of the networkbased intrusion detection techniques, however, are anomaly-based.

A quite frequently used method by network anomaly-based IDS is to set up a statistical model of normal network traffic. A deviation from the model will be marked as suspicious. Those statistical models are built from different perspectives of the traffic analysis. Current network anomaly systems such as NIDES [50], EMERALD [51], ADAM [52] and SPADE [53] belong to this category.

ADAM is a statistical anomaly-based IDS developed by George Mason University, which uses a rule-based data mining technique for network intrusion detection. ADAM builds abnormal profile by mining attack free data and ADAM also has a rule set, which

is initialized by user defined abnormal patterns and is constantly updated with the new rules. ADAM obtains the good result when applied to DARPA
evaluation data set.

Both NIDES and SPADE build the statistical model on the IP addresses and ports. For example, SPADE uses SNORT as the engine and builds the normal traffic model on the number of connections observed from certain IP and port pairs. The less frequent IP port pair is more likely to be flagged as suspicious. The drawback of SPADE is that its false alarm rate is high on the traffic from less frequent IP-port pairs.

EMERALD is capable of both anomaly and misuse detection. It uses distributed lightweight sensors to monitor and detect the anomalous and suspicious activity. The sensors collect the information from both the traffic stream and the hosts. The decision engine, which uses multiple machine learning techniques, will trigger an alarm in case of an abnormal situation.

2.6 Data Mining Approach for Network Intrusion Detection

Data mining has been used for host-based and network-based intrusion detection as well as anomaly-based and misuse-based intrusion detection. In this section, the focus will be on the data mining applications on network-based IDS.

Lee et al [54], [55] explored application of different data mining techniques in the area of intrusion detection. The main contribution of their 1999's work [54] is that they used multiple data mining techniques including classification, association rules and frequent episodes to build a framework for intrusion detection. They also introduced a practical feature construction system for the classification, which categorized the connection based features into low-cost and high-cost features in terms of their computation time.

Therefore, the classification model could choose different features according to the time requirement.

The classification methods are basically rule-based algorithm such as RIPPER. Lee and Solfo further extended their previous work in [55], where they applied association rules and frequent episodes to network connection record to obtain additional features, and then the classification algorithm, RIPPER, was applied on the labeled attack traffic to learn the intrusion pattern. They also extended their cost-sensitive model by explicitly defining three cost levels for features; the rule-sets that are formed by these features are also "cost-sensitive".

Barbara et al [56] applied association rules and classification for anomaly detection. Their system, ADAM, first built a normal profile by mining on the attack free data, then use the entire training data to look for the suspicious patterns that are not found in normal profiles. Finally, a classifier is trained to identify if the suspicious pattern belongs to known attack, unknown attack or normal event.

An alternative classification approaches that use fuzzy association rules is from [57]. Bridges and Vaughn used traditional rule-based expert system for misuse detection and made contribution to anomaly detection by using fuzzy logic and genetic algorithms. They created fuzzy association rules from the normal data set, and also built a set of fuzzy association rules from the new unknown data set, and then compared the similarity between the two groups of rules. If the similarity is low, it indicates a possible intrusion in the new data set.

The genetic algorithm (GA) is used for tuning the membership function of the fuzzy sets and to select the most relevant features. Basically, GA is used to give rewards to a high similarity of normal data and reference data and penalize a high similarity of intrusion data and reference data. By doing so, the similarity between fuzzy rules set of the normal dataset and the reference data set is going to slowly evolve to more, and the similarity between fuzzy rules set of the intrusion data set and the reference data set is going to evolve to less.

Decision tree is another classification technique that applied to intrusion detection [58], [59]. For example, Sinclair et al use genetic algorithm and decision trees for intrusion detection. The decision tree is constructed by applying the IDS [60] algorithm on the organized data information, such as the tabular form of connection features and the related label of whether they are intrusive. The constructed tree and genetic algorithms are then used to generate rules for classifying the network connections.

Other data mining approaches such as neural network [12], [13] and Bayesian classifiers are also exploited for the intrusion detection. The ultimate major challenge in the anomaly-based detection method is the high false alarm rate.

2.7 Conclusion

This chapter reviewed network intrusion detection techniques and their related works covering host-based, network-based and specific attack types as well as data mining approach. In this chapter, an analysis, evaluation and performance of different detection techniques are explored. Based on the literature review above, it is obvious there is a lot to consider when studying the concept of IDS. Though, certain detection techniques have emerged such as EMERALD, a method capable of both anomaly and misuse detection, the drawback is that its false alarm rate is still high. In this situation, more research is needed to develop an effective and efficient way of detecting network intrusions hence the need for this thesis. Information gathered from literature is considered in the design of the algorithm for the proposed IDS. The IDS is to detect known and novel DOS attacks in a network and also reduce the false alarm rate in the system. To achieve this goal, a frequency-based approach is considered using DFT to develop an algorithm for our IDS to extract the frequency pattern of traffic data for better analysis. The next chapter elaborates on the designing procedure for the proposed system. The detection system would be modeled using simulation and mathematical synthesis for the algorithm. DFT would be used to extract the frequency pattern of connection behavior of each network node at the connection history. The overall strategy for the frequency-based intrusion detection would be depicted in the next section.

CHAPTER THREE

METHODOLOGY

3.1 Introduction

This chapter outlines the methodology applied in this thesis and also presents the theoretical description of the problem being studied and considered solution approach. A mathematical synthesis supporting the background theory and the solution adopted is presented in this chapter. The primary focus is to use DFT to model an algorithm for IDS to generate frequency pattern in order to detect network intrusions and also reduce false alarm rate during detection process.

The frequency-based approach is specifically designed for intrusion pattern analysis on DOS, Probe and password guessing attacks and is evaluated by synthetic data using NS2 simulator. NS2 is simply an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks.

3.2 Time Domain vs. Frequency Domain Analysis

Signals can be transformed between time and frequency domain using DFT, a technique introduced in signal processing. DFT is considered to convert time-series data in the time domain into frequency data in the frequency domain. This technique is used because a complex data can easily be analyzed in a frequency domain and more accurate resolution for spectral analysis can be achieved when periodicity exist.

3.2.1 Frequency Based Intrusion Detection

A technique of signal processing on time-series data would be used to design the intrusion detection strategies. A time-series consists of a sequence of numbers each representing a data value at some point in time. There are many examples available for the time-series data. They are weather data, financial data, stock price, and traffic throughput of networks [61]. In this research, the focus is on three different kinds of time-series on network traffic. The first one is the time-series of packets throughput per unit time. The second is the time-series of inter-arrival time of the packets. The last one is the time-series of packet payload size. It is observed that the attacks generated by a brute force approach such as the DOS, probes, and guessing of passwords, create a large number of network packets by the coded scripts, which may cause some regular patterns in the traffic data. In addition, such attacks often use fabricated constant sized payloads. The intrusion detection algorithm would be designed based on the observations above and it is expected to capture anomalous traffic behaviors, known attack signature as well as unknown attack signature. In order to reduce the amount of suspicious traffic that need to be carefully examined by the IDS a technique introduced in [35] would be used to differentiate the "clean and dirty" traffic. Thus, a connection queue will be built where the traffic from familiar IP has higher priority than the traffic from newer IP. With the higher rate of new connections made, the lower priority will be given to these new connections. This method of frequency-based intrusion detection strategy will be used to watch those "unfamiliar" connections.

There are four steps for this approach. First, the connection history would be constructed for all those connections that are coming from new IPs and trace the traffic features including the packet size, the inter-arrival between packets and the packets throughput per unit time within or among the connections. Second, considering the possible huge amount of data, compressed time-series would be used to speed up the next step's analysis. In the third step, DFT would be applied to the time-series data generated and collect the resulting frequency information. Both the global frequency patterns for all connections in connection history and local frequency patterns for each single connection would be computed using DFT. Finally, by applying Fourier analysis to the time-series created by network traffic signal, the periodicity pattern that may exist in the network traffic would be identified. Figure 3.1 shows the Flow Chart for proposed Intrusion System. The detail and the whole strategy would be given at the following subsections.

3.3 Network Connection History

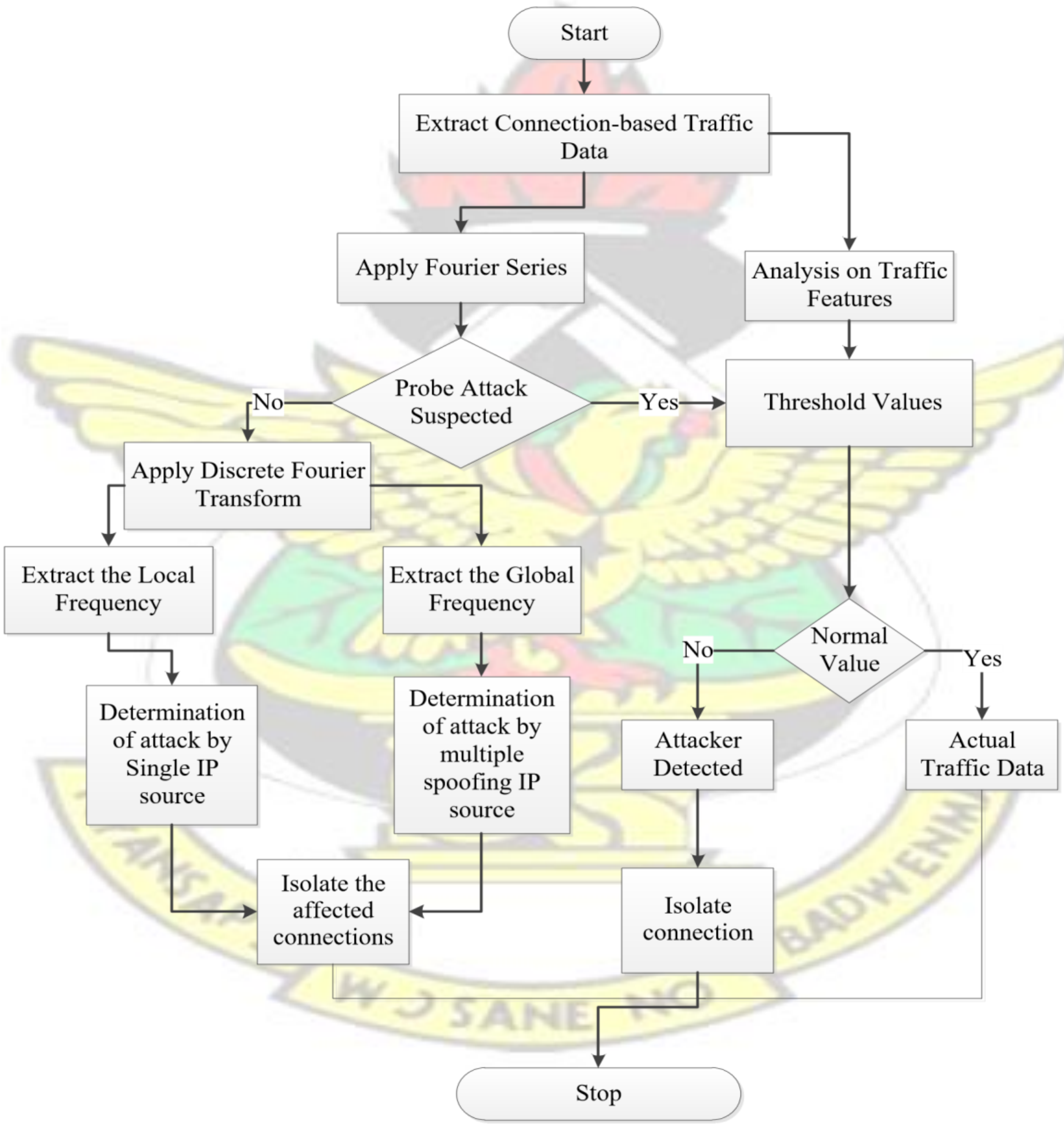
In this study, a network connection includes all the network traffic (packets) sent between two connected IP addresses in a certain time period, which has the following properties:

- Two numbers of Source and destination IP addresses.
- During the connection, only one protocol or many number of protocols used at different times.
- A set of successive packets within a connection which are not necessarily sent or received successively in terms of time by a machine.

For a protected network, a firewall is used to control the connection queue. The firewall will pass connections from familiar IP and maintain the history of recent new connections, that is, connections from new IPs will be watched and their frequency patterns will be computed and recorded. For each connection within a connection history, all the three types of time-series are generated.

3.4 General Flow Chart

The general flow chart for IDS in this work is illustrated in Figure 3.1.



Start
Extract Connection-based Traffic Data
Apply Fourier Series
Analysis on Traffic Features
Probe Attack Suspected
No
Yes
Threshold Values
Apply Discrete Fourier Transform
Extract the Local Frequency
Extract the Global Frequency
Normal Value
No
Yes
Determination of attack by Single IP source
Determination of attack by multiple spoofing IP source
Attacker Detected
Actual Traffic Data
Isolate the affected connections
Isolate connection
Stop

KNUST

Figure 3.1: Flow Chart for proposed Intrusion System

In this work, traffic can be broken down into two distinct components, local data traffic and global data traffic. The IDS program executes as follows:

In the first step, various connection-based traffic data are extracted from the firewall in the given network. Now, the traffic data is converted into time-series using Fourier series and the average variance of packet sizes and the μ values has to be computed. From the different behavior of series, probe attack can be detected during the initial stage. The Discrete Fourier Transform (DFT) is applied on the series of Local Data Traffic and Global Data Traffic. The application of DFT on the traffic extracts the frequency content available in both traffics and finally generates the Local and Global Frequency Patterns. In this work, the trace files are used to find the Suspicious IP by converting the time-based data series into frequency-base using DFT. The Local Frequency Patterns are used to find the attacks coming from a single IP source. Similarly, Global Frequency Pattern is used to find the attacks in multiple spoofing IP source.

In this system, we can find attacks coming from multiple connections. We can also prevent the attacks (from both single and multiple connections) by isolating the affected connections that were detected using Global and local frequency pattern through the connection history. This way, we can make the network free from attacks. Similarly, once the conversion takes place from the time domain into frequency patterns, the Frequency pattern (1) and Frequency pattern (2) are plotted and if the result showing is having a different frequency range from that of the normal frequency range set by the system for all nodes, then it means Suspicious attack is in either connection pattern 1 or 2 or both.

3.5 Inter-arrival Time Frequency vs. Packets Rate Frequency

The goal of an attacker is to flood or to probe the network by sending out huge amount of packets to a target automatically. The manner in which packets are sent falls into three categories. First, the packets are sent out very randomly, which means that, the inter-arrival time between any two consecutive packets would not consist any repeated periodic patterns. Second, the packets are sent out with constant speed. Third, the packets are not sent out with constant speed, but the periodic pattern exists in inter-arrival time. Figure 3.2 demonstrates the inter-arrival time for three different time patterns.

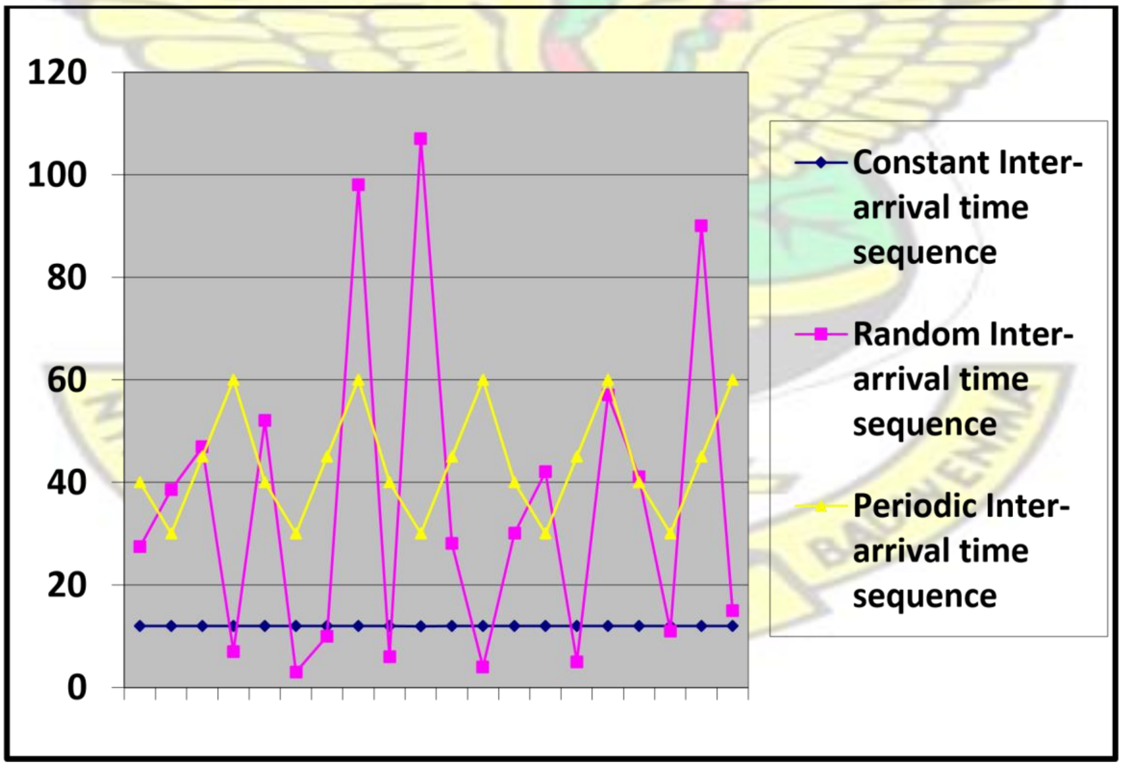


Figure 3.2: Three categories of inter-arrival time sequence of packets (X: Time in

milliseconds, Y: Inter-arrival sequence)

Each number in the above sequence represents an inter-arrival time between two consecutive arrive packets. For the cases of constant inter-arrival time sequence, instead of applying DFT on inter-arrival time, we apply DFT on data sequence of packets throughput per unit time to get the frequency pattern. The unit time is a selfadapt value depending on the duration of the connection and the sending speed of the traffic. For periodic inter-arrival time sequence, applying DFT on packets throughput per unit time would not find out the periodic pattern so DFT on periodic inter-arrival time sequence would be appropriate. For the perfectly random inter-arrival time sequence, we would not have frequency pattern. It could be a legitimate traffic or a manually operated attack or an attack that exploits a specific flaw of the system such as buffer overflow attack. The idea behind this frequency pattern analysis is that the traffic with constant Inter-arrival time is generated by the coded automated DOS, probe or dictionary attack which use the simple attack strategy, say, flooding a machine by sending a packet every millisecond or make a guess on the password for a telnet service every one second. The periodic inter-arrival time would be used by those more complicated attacks which need many packets to finish a subtask and these packets for the task are randomly generated but the subtask is repeated again and again within the whole attack process.

3.6 Application of DFT to Time-Series Sequence of Packet

Discrete value representing the time is expressed as *d(n)* for a given data sequence where $n \geq 0$. So the Discrete Fourier Transform (DFT) is chosen and applied to compute the frequency information. The DFT takes the original time series in the time domain, and transforms them into the associated frequency data in the frequency domain. Applying the DFT for the above time-series sequence, we get [62], [63]

$$F(k) = \sum_{n=0}^{N-1} d(n) W_N^{kn} \tag{3.1}$$

Where

$$W_N = Exp(-j2\pi/N) \quad \text{For} \quad 0 \le k \le \left(N-1\right) \tag{3.2}$$

Hence

$$F(k) = \sum_{n=0}^{N-1} d(n) e^{-j2\pi kn/N} \tag{3.3}$$

Expanding the right hand side of equation (3.3), we have

$$F(k) = \sum_{n=0}^{N-1} d(n)\cos(2\pi kn/N) - j\sum_{n=0}^{N-1} d(n)\sin(2\pi kn/N) \tag{3.4}$$

Where *N* is the total length of *d(n)*

However, applying equation (3.4) is time consuming for *N* samples of discrete values.

Therefore, Fast Fourier Transform (FFT) procedure would be used to compute the frequency data *F(k)* in *O(n log n)* time. Which means FFT algorithm will speed up the computation process when frequency patterns are extracted from all incoming data traffic.

3.7 Global and Local frequencies

The frequency patterns of local data traffic and global data traffic within all connections can be generated by applying equation (3.4). When an attack sends packets using multiple spoofed IPs as their source addresses, aimed at misleading the detection systems, many incoming connections to the target will be seen but it is possible that such attack packets are generated from one source computer. The global frequency analysis is effective for this type of attacks. In real-life environments when connections are established between different computers, the connection traffic from one connection is statistically unrelated to the connection traffic from others. Therefore, there should be no frequency patterns existing over the different connections. In this case, local frequency analysis is effective to detect the uncoordinated attacks occurring in different connections.

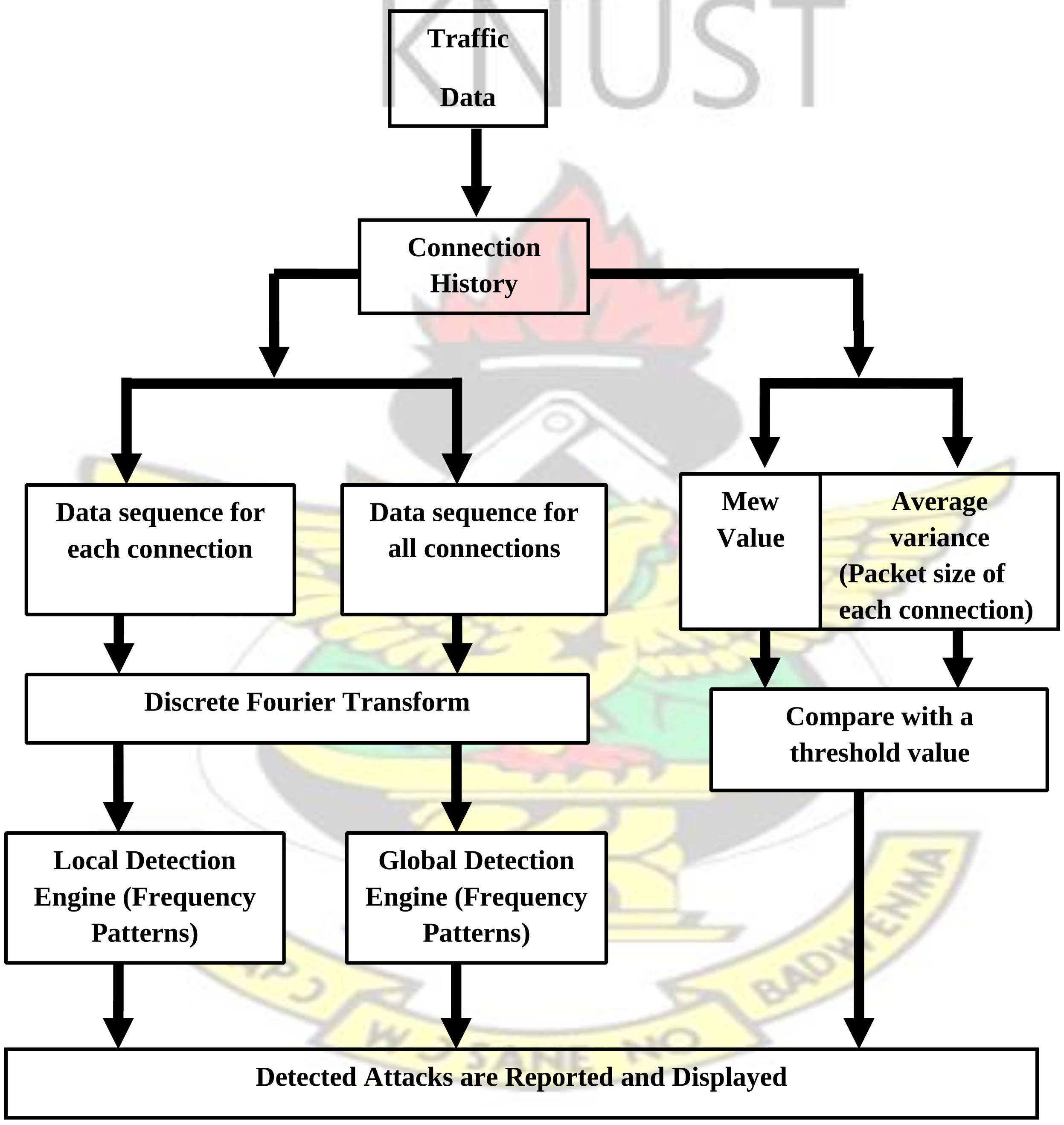

Traffic Data
Connection History
Data sequence for each connection
Data sequence for all connections
Mew Value
Average variance (Packet size of each connection)
Discrete Fourier Transform
Compare with a threshold value
Local Detection Engine (Frequency Patterns)
Global Detection Engine (Frequency Patterns)
Detected Attacks are Reported and Displayed

Figure 3.3: Strategy of frequency-based intrusion detection

Figure 3.3 shows the architecture of frequency-based intrusion detection. The Actual traffic data in the network is first segregated and it will be checking or comparing with the connection history. All the previously reported connections and its traffic are stored in the database called connection history. The algorithm written in tcl (Tool

Command Language) script which checks the data traffic is already available.

KNUST

3.8 Frequency Mining

Let the data in the time-series be represented by *d(n)*, where *n* denotes the time. Applying the DFT for the above time-series sequence, we get

From equation (3.3)

$$F(k) = \sum_{n=0}^{N-1} d(n) e^{-j2\pi kn/N} \tag{3.5}$$

From equation (3.4)

$$F(k) = \sum_{n=0}^{N-1} d(n)\cos(2\pi kn/N) - j\sum_{n=0}^{N-1} d(n)\sin(2\pi kn/N) \tag{3.6}$$

Where *N* is the total length of *d(n)*

These frequency patterns are useful to detect the intruder in the system. The timeseries data contains a number sequence. It represents the data value at some point in time. The following factors are also considered in addition to the frequency pattern to determine the anomalies in the traffic data in order to reduce or eliminate false alarm rate:

1. μ value
2. Average variance of packet size
3. Missing of flag

The μ value is the threshold value set by the proposed algorithm to determine the average number of packets in each node. The threshold algorithms activate an alarm when a value deviates from its normal (expected) threshold value set by the system by some amount. This μ value is calculated each and every time for all nodes in the system. If the μvalue of a node is found to be abnormal, comparing to the other nodes in the network then it means intruder is entered in the network.

Let the packet size of the data be represented by *pack (j)* where '*j*' varies from 1 to *n*, and '*n*' is the number of packets within a connection. Then the μ value is considered as

$$\mu = \frac{1}{n}\sum_{i=1}^{n} pack\, j() \qquad (3.7)$$

Normally, if no attacker node is found in a network packet, delivery rate is high. If any intruder exists in the network, then the network will not allow forwarding packets to it. This brings down the packet delivery ratio automatically.

The average variance, Var_{avg} of the packet size *pack* (*j*), for this work is given as follows:

KNUST

$$\mathrm{Var}_{avg} \square \frac{1}{n} \sum_{i \square 1}^{n} \sqrt{(pack\, j(\,) \square \square)^2} \qquad (3.8)$$

Where, $1 \leq j \leq n$, and 'n' is the number of packets within a connection.

The average variance of packet size $\mathrm{Var}_{avg}$of an attacker packet is very small as compared to that of the normal traffic. This is because the goal of an attacker is to consume the network resources of the victim or to probe the information, and the payload of each packet is of a similar size since it is fabricated. This factor is used to detect the anomalies in the network.

Before communication can be established between two nodes a packet called control packet would be passed between the two nodes with the help of the hub. The control packet has a field called "Flag". If the "Flag" packet is missing in the received packets of this proposed centralized network, then an intruder has entered the network.

Once the system identifies an attacker node(s) in the network through our CTA-IDS the system automatically isolates the node(s) with the attacker packet by informing the nearby nodes with the help of the hub through the control packets. The system again will not allow the intruders to come inside the network.

3.9 Testing the Intrusion Detection Strategy

Intrusion detection strategy is required for the implementation of this system. First, firewall is placed on the central device (Hub) to capture the network traffic data. The firewall has the database table of the recent connections within a specified time window. Each table entry contains an IP address that has been connected to the protected computer at least once within the time window. The traffic data of each connection is converted to a time series; the average variance of the packet sizes and the μ values are computed and recorded by the firewall. In addition, information about port connections are recorded which will be useful in detecting probe attacks. Finally, the firewall will report the suspicious IPs based on the observed frequency patterns, average variance of packet sizes and μ values. In the next section, we will describe the intrusion simulation framework using NS2, and report the simulation results evaluating the frequency-based intrusion detection strategies.

3.10 Simulation for Intrusion Detection System

Implementing and testing any new algorithm or methodology for intrusion detection in any network can be done in real environment or in a simulation environment. The simulation environment can also be a perfect environment for testing any network for the detection of anomalies. The results obtained from the simulation can be the same as if it is implemented in real time. The methodologies used in the simulation environment can also be used in real time for the intrusion detection.

The approach one takes to solving a problem determines the probability of solving it. Models have a variety of uses when implementing IDS with different approaches. Traditionally, there are two modeling approaches: analytical approach and simulation approach. Both analytical models approach and discrete event simulation models approach can help elucidate system behavior, but there can be differences in the results of these two types of models. To model a system, some simplifying assumptions are often required. It is important to note that too many assumptions would simplify the modeling but may lead to an inaccurate representation of the system.

3.10.1 Analytical Approach

The analytical approach is to describe the proposed system mathematically by using applied mathematical tools like probability and queuing theory and apply the appropriate numerical models to gain insight from an already developed mathematical model. When a system is not complex the analytical method is preferable over simulation method for implementation due to its processing time and accuracy for simple analysis.

3.10.2 Simulation Approach

Simulation of any model can be done easily using the appropriate software. For any engineering research, simulation plays a vital role for checking the correctness of model. Compared to analytical modeling, simulation is very easy, accurate and time effective. Simulation uses less abstraction in the model because almost every detail of the specifications of the system can be written in the form of a tcl code to explain the actual system. So simulation is preferred for our work. In this thesis, Network Simulator version2 (NS2) is considered for this technique.

3.11 Network Simulator2 (NS2)

Network Simulator (NS) is a software use to emulate networks or model networks. Then the model is simulated to verify the assumptions and logic that was implemented in the model. This software uses a new technique where the behavior of a network either by calculating the interaction between the different network entities by using mathematical formulas or actually by capturing and by playing back observations from a production network. There are alternate simulation tools that can be used to simulate network intrusion to achieve the same purpose. Some of these simulation tools are OPNET, NMAP and MATLAB but NS2 is considered for this work. NS2 is an open-source event driven simulation tool. NS2 was invented in the year 1989. It is more popular recently because of its flexibility and simplicity. NS2 can run perfectly in Linux operating system. Since 1995, the Defense Advanced Research Projects Agency (DARPA) has been using the NS2 tool for their work [64]. In NS2 software, the network and its traffic can be designed by writing the Tcl script which uses the C++ programming language. Tcl is a string-based command language and it is interpreted when the application runs. The interpreter makes it easy to build and refine an application in an interactive manner. Tcl is a radically simple opensource interpreted programming language that provides common facilities and some of these facilities are variables, procedures, and control structures as well as many useful features that are not found in any other major language. In general, we can say that the Network Simulator version2 (NS2) fulfills all the needs of the engineer or the designer related to any network. The network may be wired, wireless and protocols such as TCP/IP, UDP, and Routing algorithms [65]. NS2 is generally fast and cost effective as it is open source software.

3.11.1 Step by step commands for writing TCL script:

Writing tcl script has a number of commands. These commands are used to model our intrusion detection system. The detailed codes are provided at the appendix but for the purpose of this thesis, the following commands are enumerated.

1. **Define simulator.**

   Creating a simulator is essential for any NS2 simulation (NS2 is Network Simulator).

2. **Define topology.**

   This step defines the size or area under which the simulation occurs. We must define it keeping in view the possible locations of our nodes.

3. **Define output trace files.**

   Trace files record events like node creation and data transfer among others during any simulation. There are several trace formats and choosing one among them would depend on the results we are looking for and what the files log.

4. **Define the General Operations Director known as "GOD".**

   As quoted from CMU document, General Operations Director (GOD) is the object that is used to store global information such as the environment and network or nodes that an omniscient spectator would have but that should not be made known to any other participant in the simulation. That is, by definition, GOD is to manage the details of operations we supply such as the movement patterns in our simulations.

5. **Define access point/base station and configure the AP/BS.**

   To create and configure a base station or an access point in NS2, it is preferable to define the options we want to configure the AP/BS with. Then we need to disable the random motion for the AP and give it a location. After you are done with creating the AP, switch off the configuration mode.

6. **Define wireless node, configure it and attach to AP/BS.**

   Creating and configuring a wireless node is very similar to what we needed to do with the AP. Disable the random motion and set location as we did earlier.

   Finally, we "attach" the wireless node to the AP.

7. **Define the wired source, configure it and connect it to the AP/BS.** The creation and configuration of the wired source are similar to that of the wireless node except we do not "attach" it but rather "connect" it with a wired link.

8. **Define traffic generator agent and attach it to the source and destination node.**

   In order to define the traffic flow in the network, we need to assign the nodes as source or destination and decide the type of application/traffic. The first step is to define the source agent. Then, designate which node assumes the agent we defined. Define the traffic/application type and attach it to the source agent. It should be noted that each traffic type comes with default values of traffic parameters like rate and packet size which can be easily overridden with the set command. Now, define the destination agent and attach it. It must be noted that source and destination agents form a pair and though many options are available for a different type of agent, not everyone is wellmatched with each other.

Finally, we can connect the agents. It is necessary to tell the simulator when to start and stop the traffic flow. To do so, we schedule events that control the agent's activities.

9. **Run simulator.**

   To order the simulator to execute, this statement has to be provided in the script. It is necessary to clear variables and close files after the simulation ends. It is customary to create a procedure for that.

3.12 Conclusion

This chapter explains the method of using simulation to model DOS attacks, and also modeled and tested the proposed IDS algorithm with these attacks. The model is built to emulate the logical local network environment using NS2 simulator. It is very difficult to get DOS attack data set from a real environment due to security reasons. However, the simulator we used in this work is able to provide the statistical measures on different aspects of the traffic, which may provide an alternative way to simulate the countermeasure mechanism to disconnect or isolate the addressable network entity by specifying the maximum sustainable traffic load. The next chapter reports the simulation results of the proposed IDS, and illustrates how to detect DOS attacks by extracting the frequency patterns and the threshold values generated by the algorithm through our Centralized Traffic Analyzer (CTA).

CHAPTER FOUR

RESULT AND DISCUSSION

4.1 Introduction

The previous chapter discussed the system model with mathematical synthesis to demonstrate a theoretical description of Intrusion Detection System (IDS). This section renders a detailed analysis of collected results and thus identifying various observed phenomena and emphasizing the significance of made observations.

In this work, Network simulator2 (NS2) is used to detect the Intrusion in a network.

4.2 NS2 Simulation Results

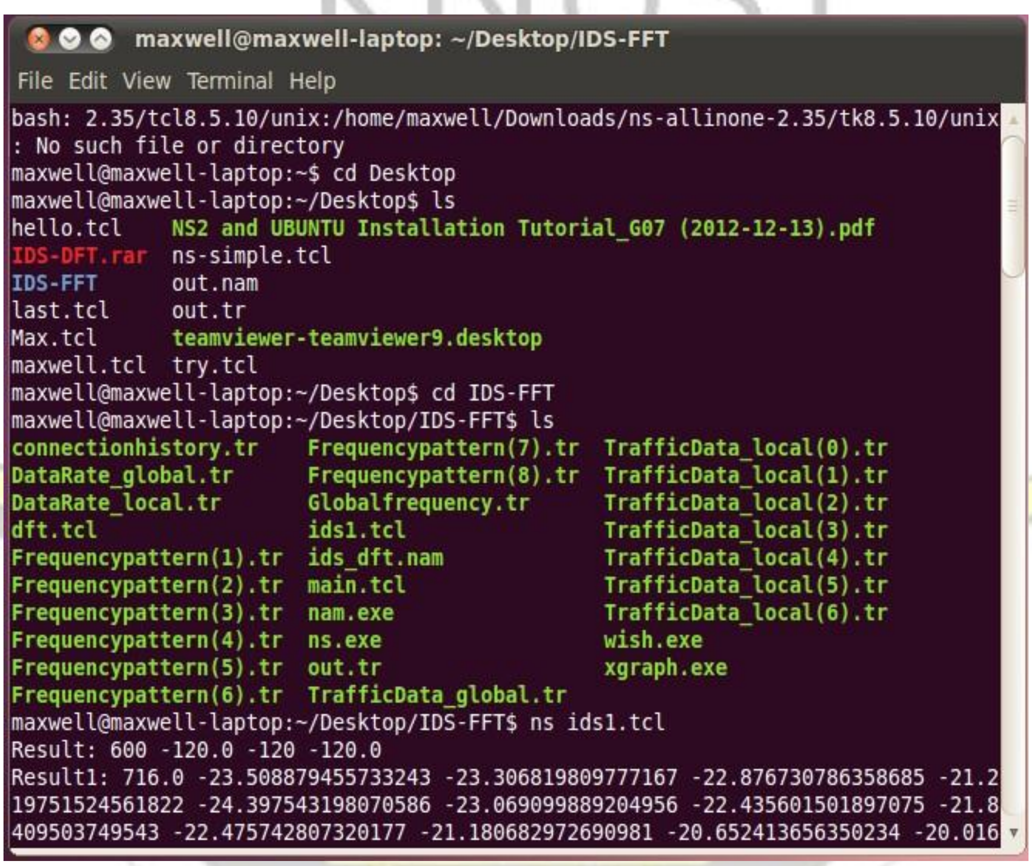
The following subsections are the simulated results for various network nodes. The results and analysis are as follows. The results indicate the traffic visualization and connection-based activities analysis of various network nodes in the form of frequency patterns. It is necessary to note that, the simulations in this work are implemented using NS2 version 2.35. DFT introduced in signal processing technique forms the basis of the algorithm for this simulation.

4.2.1 NS2 Virtual Network Simulation

During NS2 installation, a path was set in order for the application to be accessed from any subfolder on the cygwin bash shell. First we changed our cmd (command) prompt to the current working subdirectory where the script resides and then runs the script.

'Ls' is the command which displays all the files/folder in ns-allinone-2.35 package. From the list of files, the needed folder (IDS-DFT) is selected by entering "cd IDS-FFT" at the cmd prompt terminal. Again ls command displays all the files in IDS-DFT folder as shown in Figure 4.1.

KNUST



```
bash: 2.35/tcl8.5.10/unix:/home/maxwell/Downloads/ns-allinone-2.35/tk8.5.10/unix
: No such file or directory
maxwell@maxwell-laptop:~$ cd Desktop
maxwell@maxwell-laptop:~/Desktop$ ls
hello.tcl    NS2 and UBUNTU Installation Tutorial_G07 (2012-12-13).pdf
IDS-DFT.rar  ns-simple.tcl
IDS-FFT      out.nam
last.tcl     out.tr
Max.tcl      teamviewer-teamviewer9.desktop
maxwell.tcl  try.tcl
maxwell@maxwell-laptop:~/Desktop$ cd IDS-FFT
maxwell@maxwell-laptop:~/Desktop/IDS-FFT$ ls
connectionhistory.tr    Frequencypattern(7).tr  TrafficData_local(0).tr
DataRate_global.tr      Frequencypattern(8).tr  TrafficData_local(1).tr
DataRate_local.tr       Globalfrequency.tr      TrafficData_local(2).tr
dft.tcl                 ids1.tcl                TrafficData_local(3).tr
Frequencypattern(1).tr  ids_dft.nam             TrafficData_local(4).tr
Frequencypattern(2).tr  main.tcl                TrafficData_local(5).tr
Frequencypattern(3).tr  nam.exe                 TrafficData_local(6).tr
Frequencypattern(4).tr  ns.exe                  wish.exe
Frequencypattern(5).tr  out.tr                  xgraph.exe
Frequencypattern(6).tr  TrafficData_global.tr
maxwell@maxwell-laptop:~/Desktop/IDS-FFT$ ns ids1.tcl
Result: 600 -120.0 -120 -120.0
Result1: 716.0 -23.508879455733243 -23.306819809777167 -22.876730786358685 -21.2
19751524561822 -24.397543198070586 -23.069099889204956 -22.435601501897075 -21.8
409503749543 -22.475742807320177 -21.180682972690981 -20.652413656350234 -20.016
```

Figure 4.1: NS2 Command Window

From the list of files again, main.tcl is selected (this is the running main file to give output) and the results show some value which are the frequency pattern for each connection.

We set up a network model in NS2 using 13 virtual PC (Personal Computers) nodes as shown in Figure 4.2.

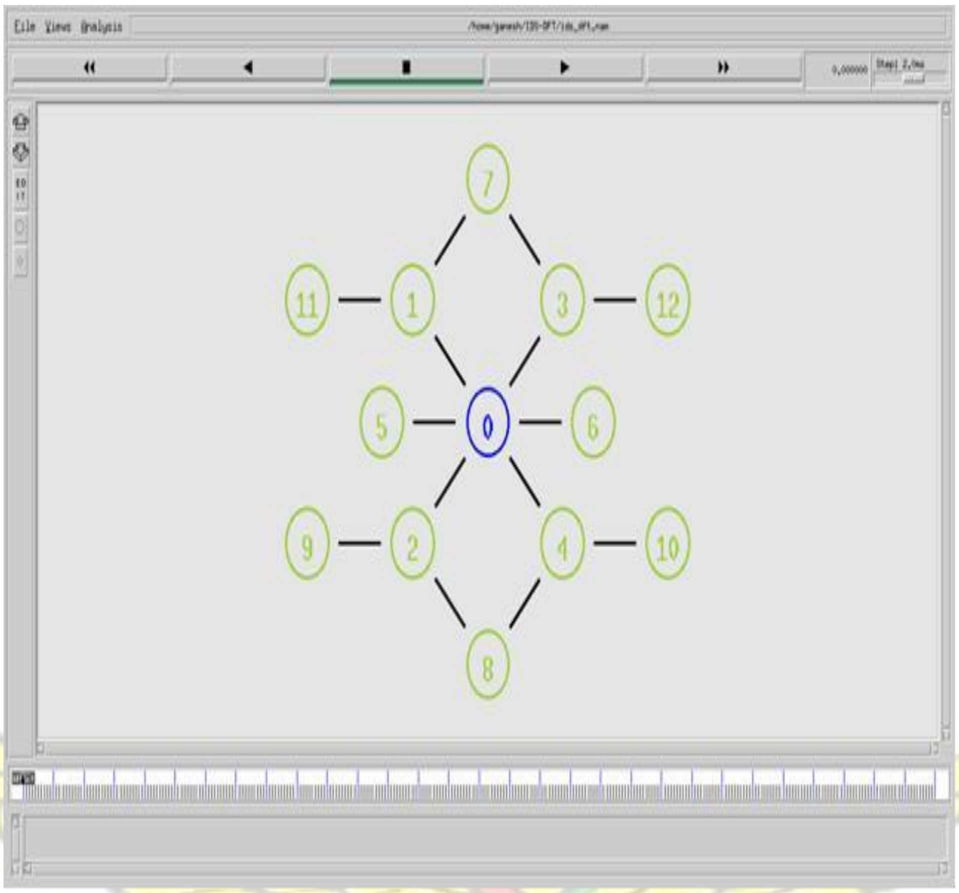
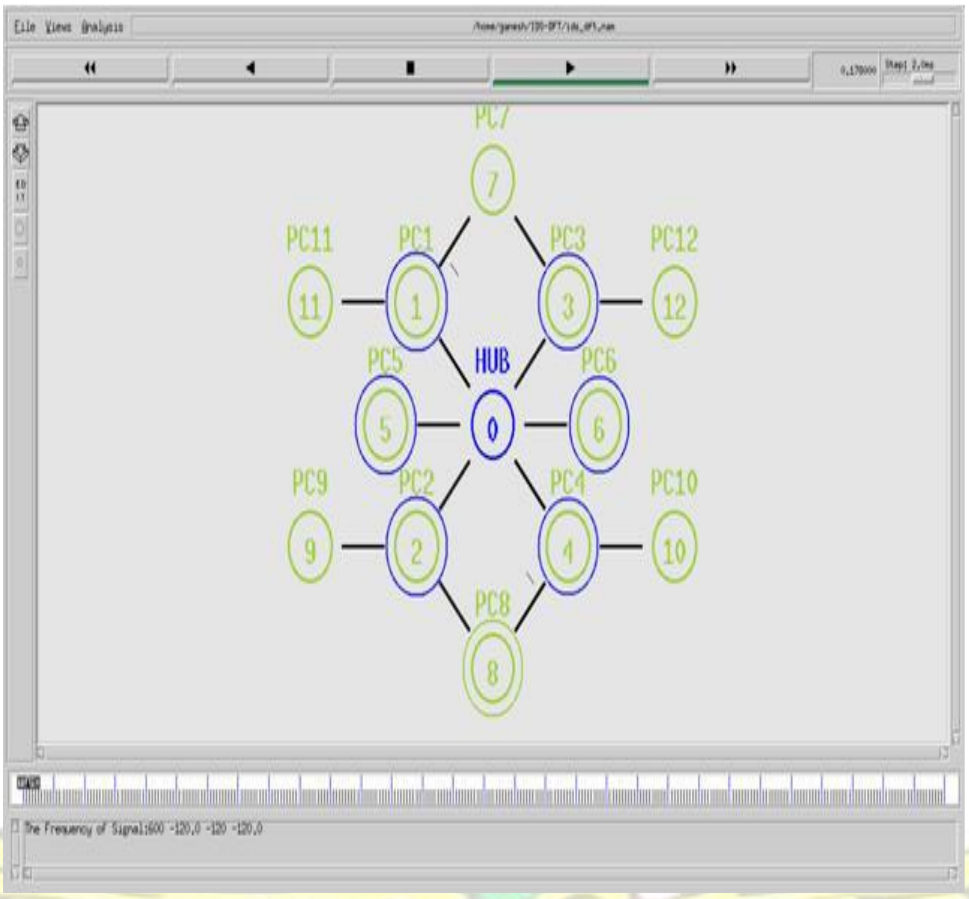


Figure 4.2: Formation of Network with Nodes

Here the 0th node is a HUB and the remaining nodes are addressable nodes that are used to assist the hub to transfer data in the network. A firewall and an intrusion detection algorithm are implemented inside the hub where network traffic would be monitored for suspicious activities.

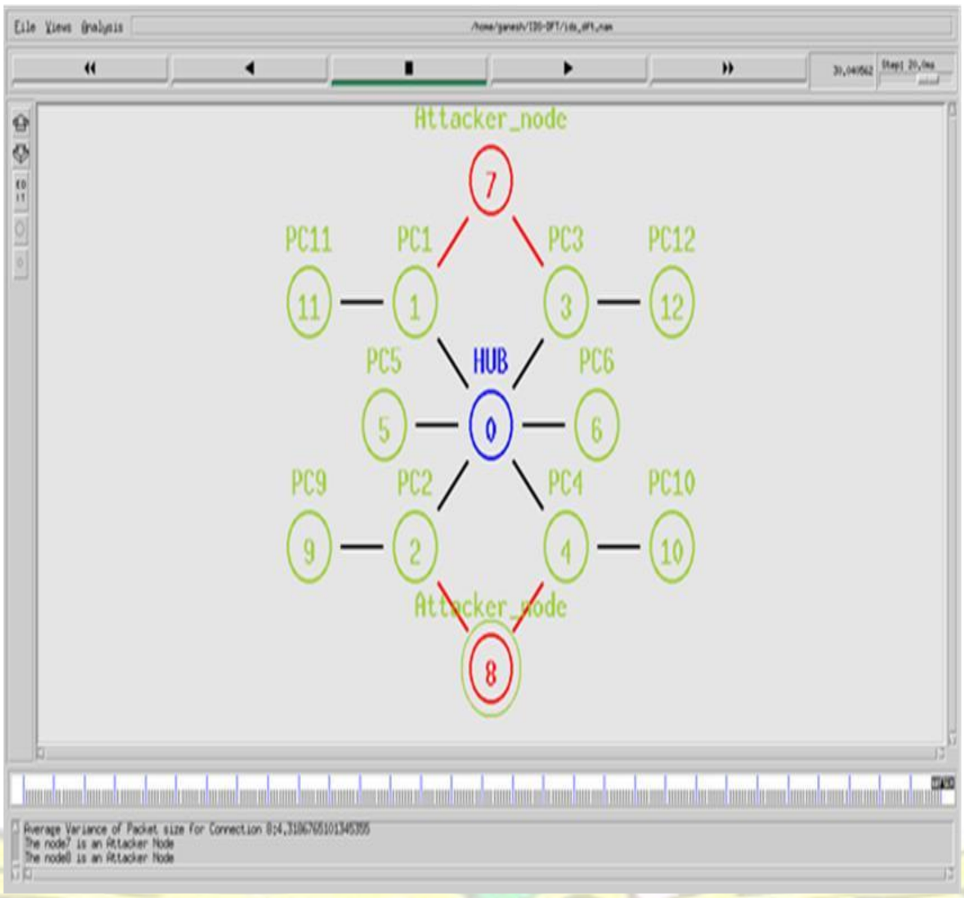


```
The Frequency of Signal:600 -120,0 -120 -120,0
```

Figure 4.3: Data Transfer from the hub

The blue circles around nodes (1, 2, 3, 4, 5, and 6) indicate that frequency signal is generated during data transfer from the hub to the nodes as shown in Figure 4.3. From this signal, frequency pattern is generated.

Applying Discrete Time Series method, traffic data is converted to time-series data using the information at the connection history. Also DFT is applied to a time series data to create frequency data at the connection history and the frequency information for each node is then used for the frequency patterns.



```
Average Variance of Packet size For Connection 8;4,310676510134535
The node7 is an Attacker Node
The node8 is an Attacker Node
```

Figure 4.4: Identification of Attackers

While passing data from node 1 or 3 to node 7 and from node 2 or 4 to 8,attacker nodes were identified, indicated as red in Figure 4.4. The attacker nodes are found by using the µ values set by this system and the average variance of packet size for each node at the connection history. The mew value is a threshold value set for each node at some point in time in the system. These values can be seen in Table 4.1 below. From the connection history, node 7 and node 8 records mew values of 4038 and 3972 respectively and these are very large and above the normal threshold values of this system as seen in Table 4.1. The average variance of packet sizes for both nodes 7 and 8 are very small, and this can be considered as automated DOS attack.

Table 4.1: Mew values and average variance of packet sizes for various connections

CONNECTION	**MEW VALUE**	**AVERAGE VARIANCE OF PACKET SIZE**
NODE 1	6	1.3656854249492381
NODE 2	4	1.3120955864630133
NODE 3	7	1.7191508225450298
NODE 4	5	1.2292528739883946
NODE 5	6	1.3656854249492381
NODE 6	7	1.7191508225450298
NODE 7	4038	0.3186765101345355
NODE 8	3972	0.4375681123928461

Table 4.1 illustrates the mew values and average variance of packet sizes for the eight network connections scanned by the NS2 simulator with the proposed algorithm introduced in this work. Nodes 7 and 8 are identified as attacker nodes and disconnected by the algorithm. From the above table, it is clear that the mew values of connection 7 and 8 are abnormal and also the average variance of the packet size for both nodes is small enough to be considered as suspicious. Thus, connections 7 and 8 are likely to be

automated attack traffic and these attack type is DOS attack. In addition to the DOS attacks other types of attack such as the Dictionary attack, which belongs to the type of remote to root attack, also falls into the scripted attack category.

4.2.2 Analysis of NS2 Simulation Results

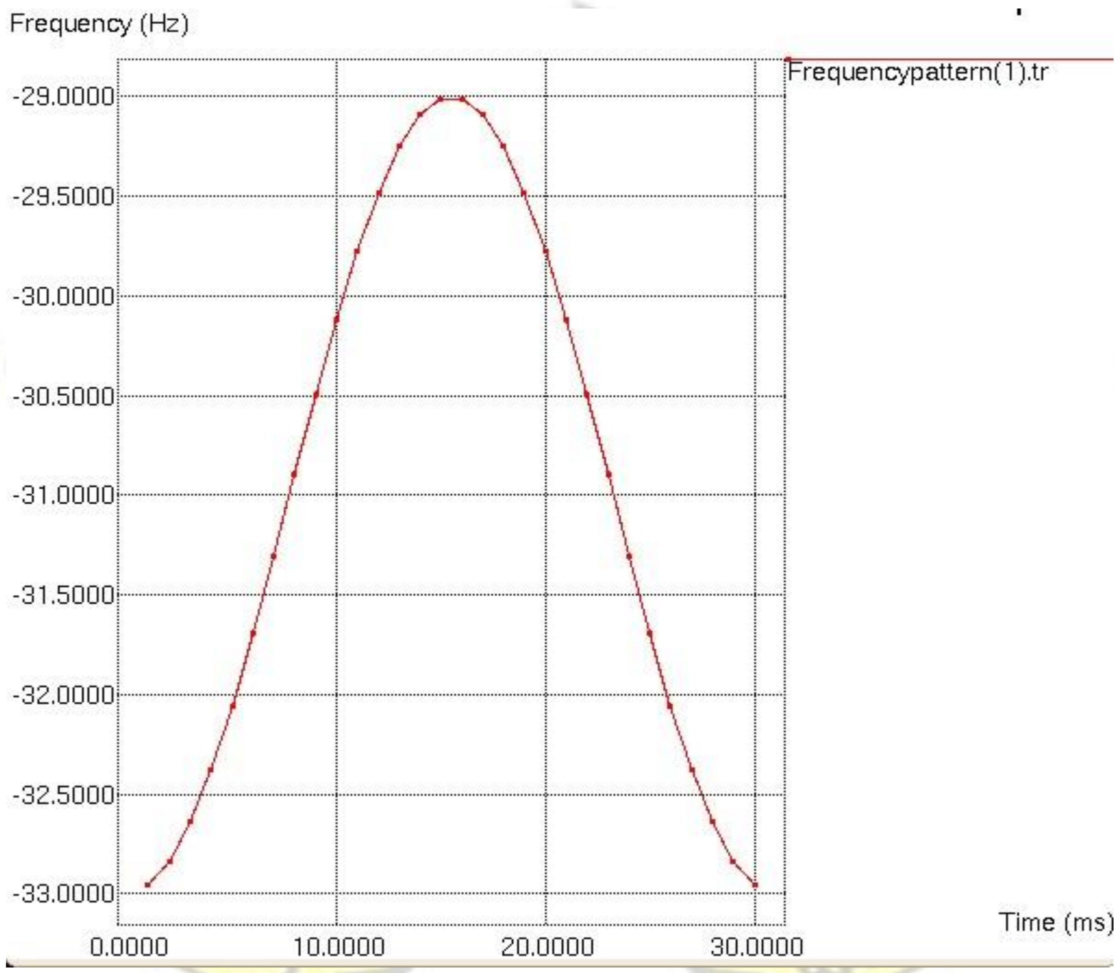


Figure 4.5: Frequency Pattern I

The frequency pattern for connection one is plotted in Figure 4.5. In this graph, there is a clear frequency pattern generated without any distortion. From the time 1.0 to

15.0, the frequency values are gradually increased from -33.0000 to -29.0000. From the time 15.0 to 16.0, we have a constant frequency value of -29.0000 and from the time 16.0 to 30.0, the frequency values are gradually decreased from -29.0000 to -33.0000. The normal frequency range set by this system as a reference for all nodes is between -45.0000 and -17.0000. Hence, since this graph is generated within the normal range as shown in the Y-axis, it means that there is no attacker packet in this node. This can be confirmed by comparing the mew value and the average variance of packet size of node 1 to that of nodes 7 and 8 which have the abnormal values as shown in Table 4.1 above. The concept of the negative frequency on this graph at the Y-axis is used for mathematical calculation only and it gives the frequency spectra in the negative domain which helps to analyze real signal using a complex number framework. A complex number can only be real if added to its conjugate [66], hence the negative frequency comes from a representation using the sum of complex exponential values for the DFT variables.

KNUST
NYANSAPƆ WƆSANE NO BADWENMA

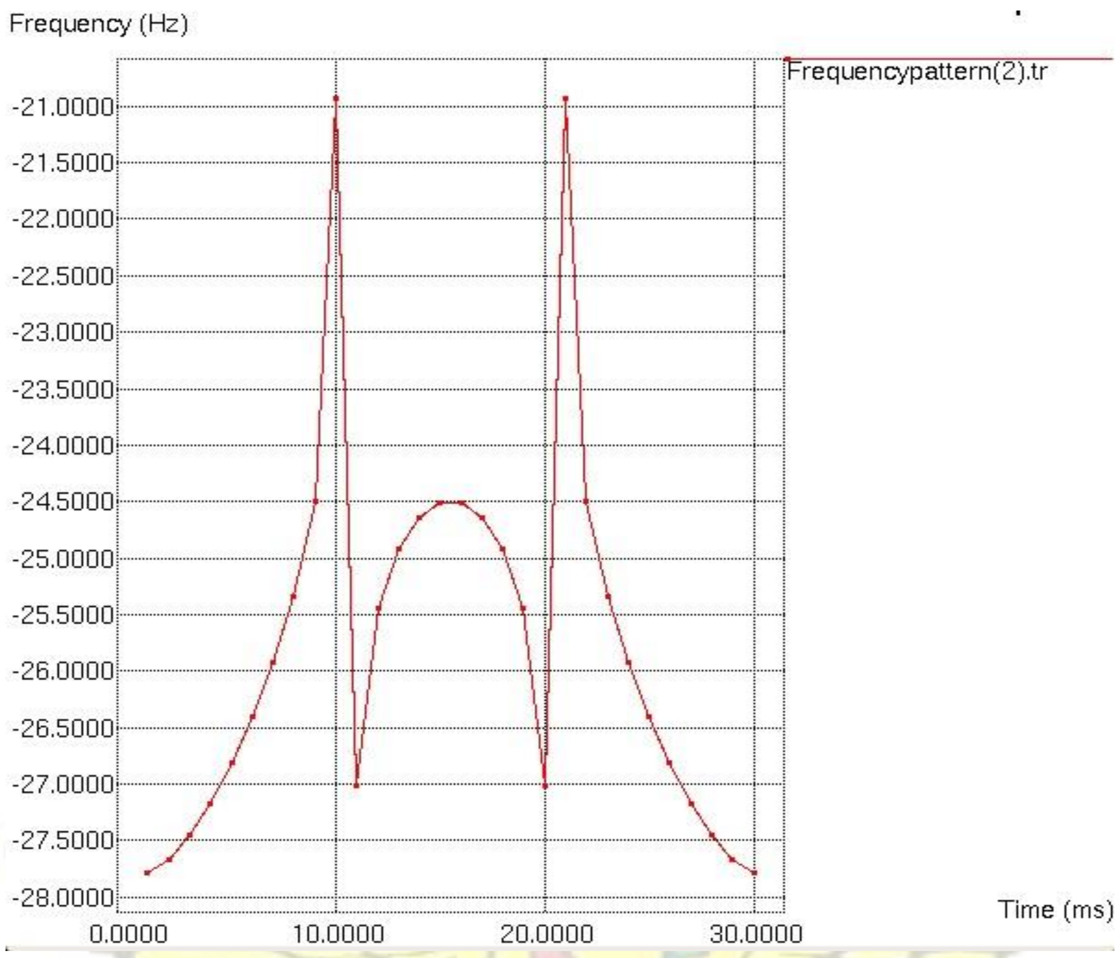


Figure 4.6: Frequency Pattern II

Figure 4.6 shows the frequency pattern for node 2 in the connection history. It can be seen that the frequency pattern of this graph looks different as compared to the graph in Figure 4.5 and the reason being that, activities at the various network nodes differ from each other and this generate different patterns. In this graph, the frequency range is set between -28.0000 and -21.0000 as illustrated in the Y-axis. This frequency range falls within the normal range and therefore concludes that node 2 does not have any malicious packets. Table 4.1 provides the mew value and the average variance of packet size for connection two as 4 and 1.3120955864630133 respectively.

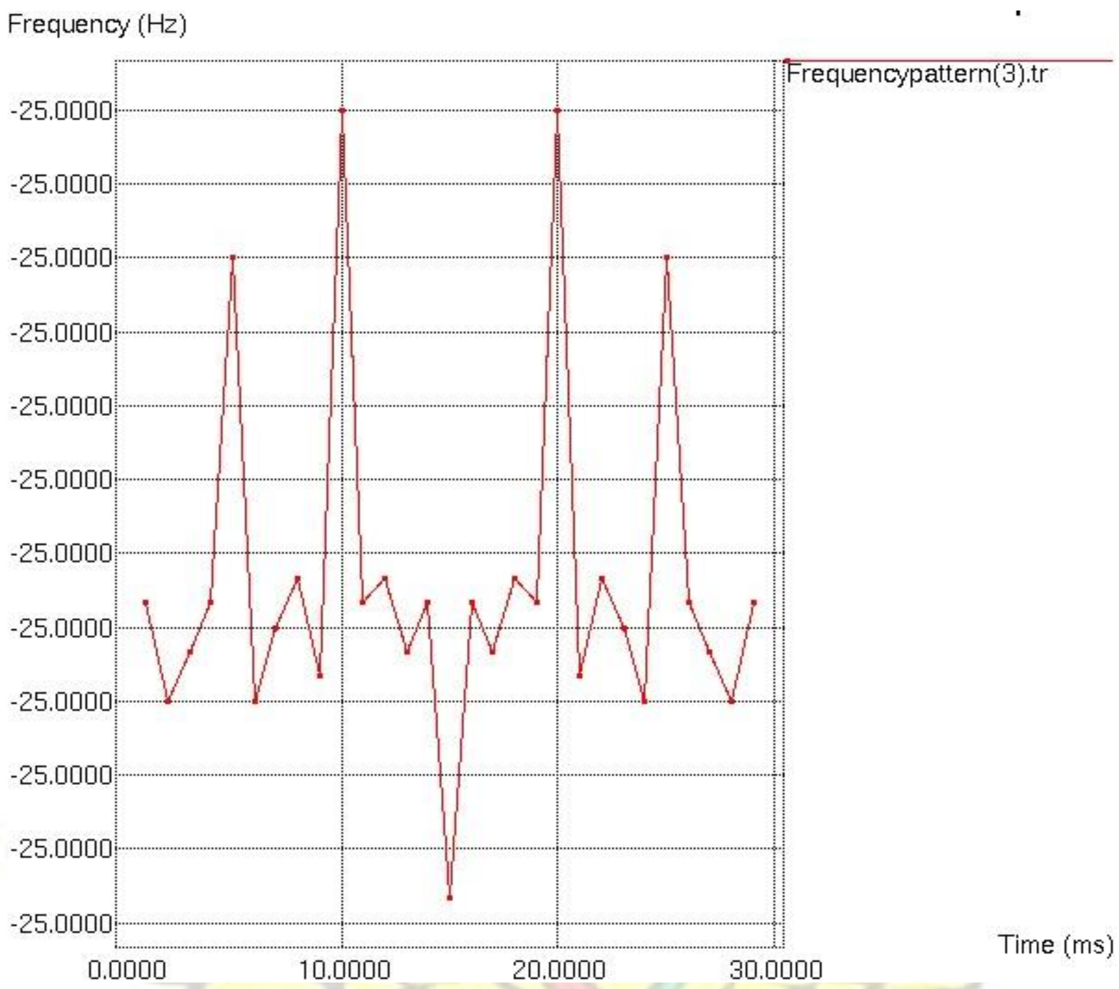


Figure 4.7: Frequency Pattern III

The third frequency pattern for node 3 is plotted in Figure 4.7. It has a distinctive frequency pattern but the frequency range is set between -25.0000 and -24.9999. The frequency range seems to be very close but in the data file (which is not showing in this report) for the graph, the frequency values are different at each point. The frequency values are in the form of fifteen decimal places which cannot be exhibited at the Y-axis of the graph hence rounded off. This format generated by the simulation helps the accuracy and precision in order to eliminate false alarm rate during detection process. The frequency range has created frequency pattern with similar frequencies in legitimate traffic due to traffic data broadcast to lists of its subnet machines periodically. The mew value recorded at the connection history is 7 and the average variance of packet size is 1.7191508225450298. This shows that node 3 is not having any malicious content.

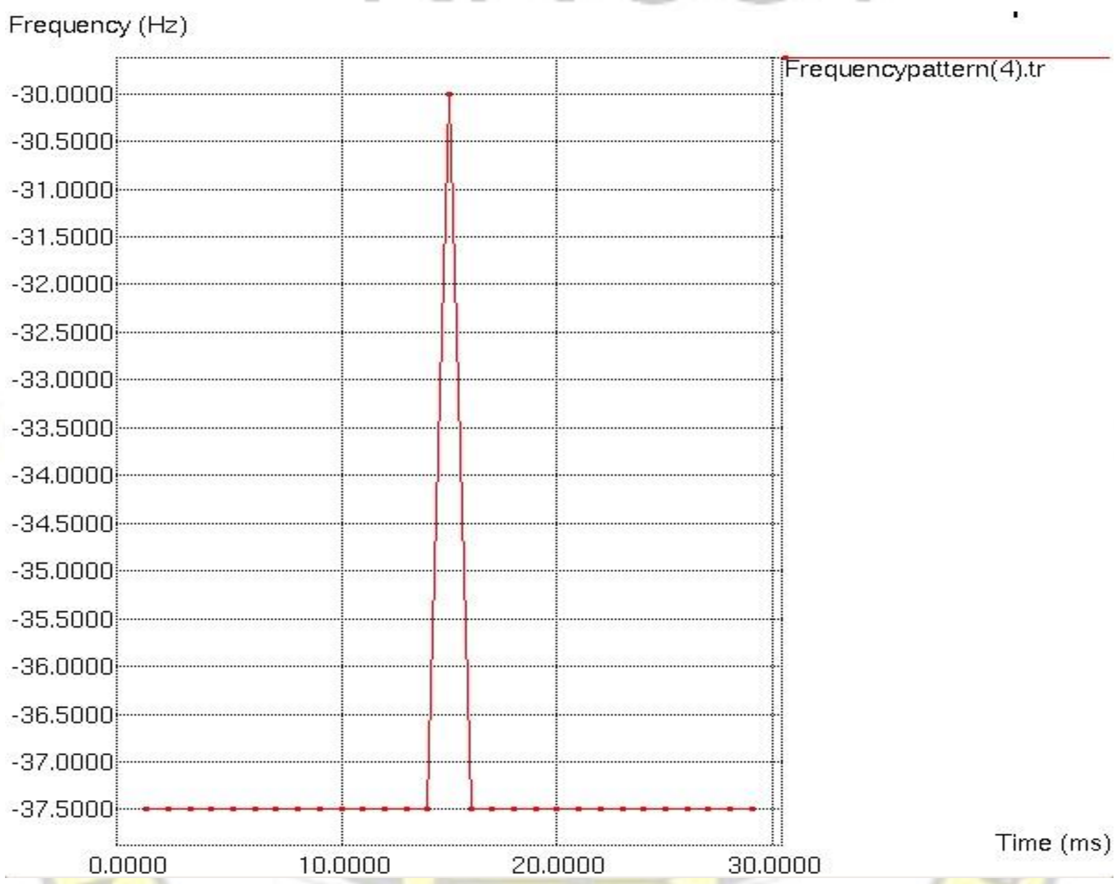


Figure 4.8: Frequency Pattern IV

The fourth frequency pattern for node 4 is plotted in Figure 4.8. From the time 1.0 to 14.0, we have a constant frequency value of -37.5000. From the time 14.0 to 16.0, the frequency values are gradually increased and decreased from -37.5000 to -30.0000 and from the time 16.0 to 30.0, we have a constant frequency value of -37.5000. The frequency range of this graph is -37.5000 and -30.0000 and this range is considered to be in a normal range. The mew value and the average variance of packet size for node 4 in the connection history are 5 and 1.2292528739883946 respectively as seen in Table 4.1.This node is having only legitimate data and does not have any attacker packets.

KNUST

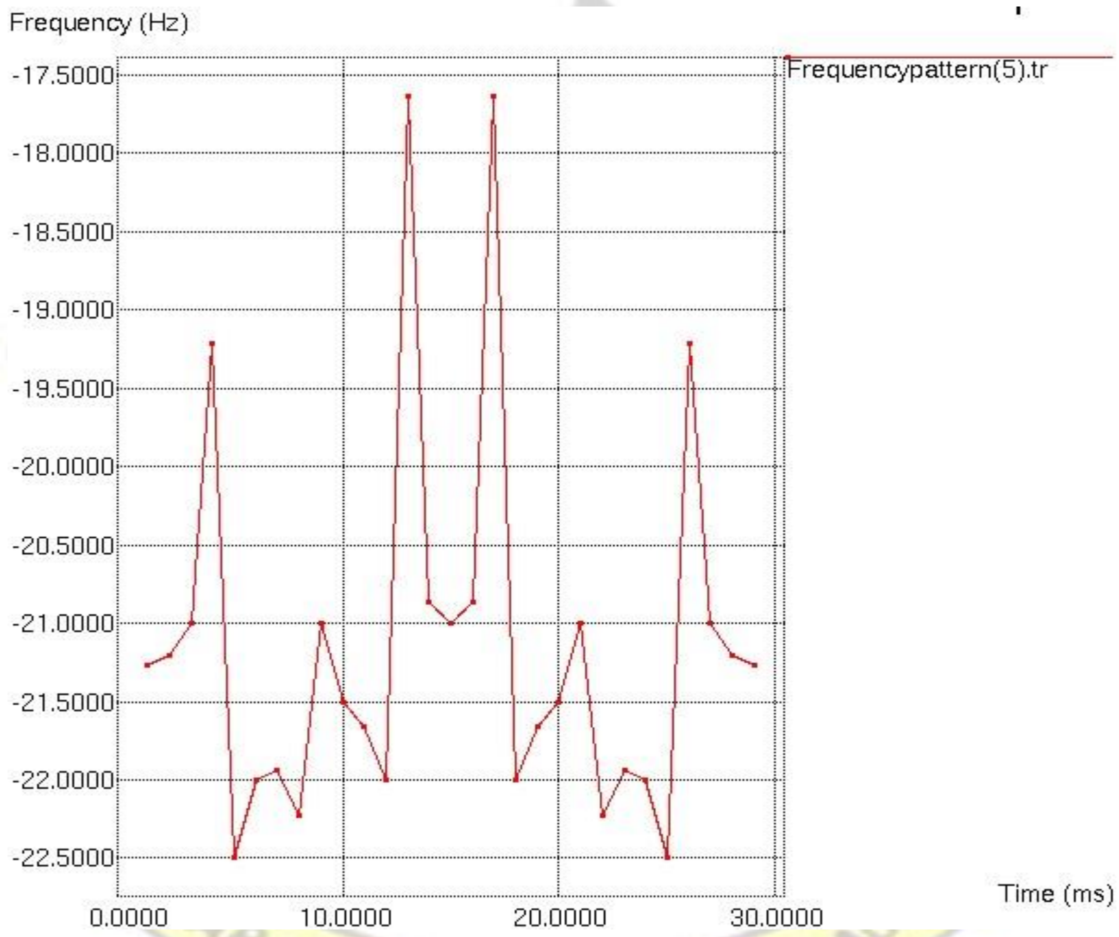


Figure 4.9: Frequency Pattern V

Figure 4.9 illustrates the frequency pattern for node 5 in the connection history. Here the frequency pattern is distinct from the one in Figure 4.8 but its frequencies are in the range between -22.5000 and -17.5000, which falls within the acceptable range. The mew value and the average variance of packet size for connection five is shown in Table 4.1 and when these values are compared to the abnormal values of nodes 7 and 8 we can conclude that node 5does not have any malicious content to collapse the network.

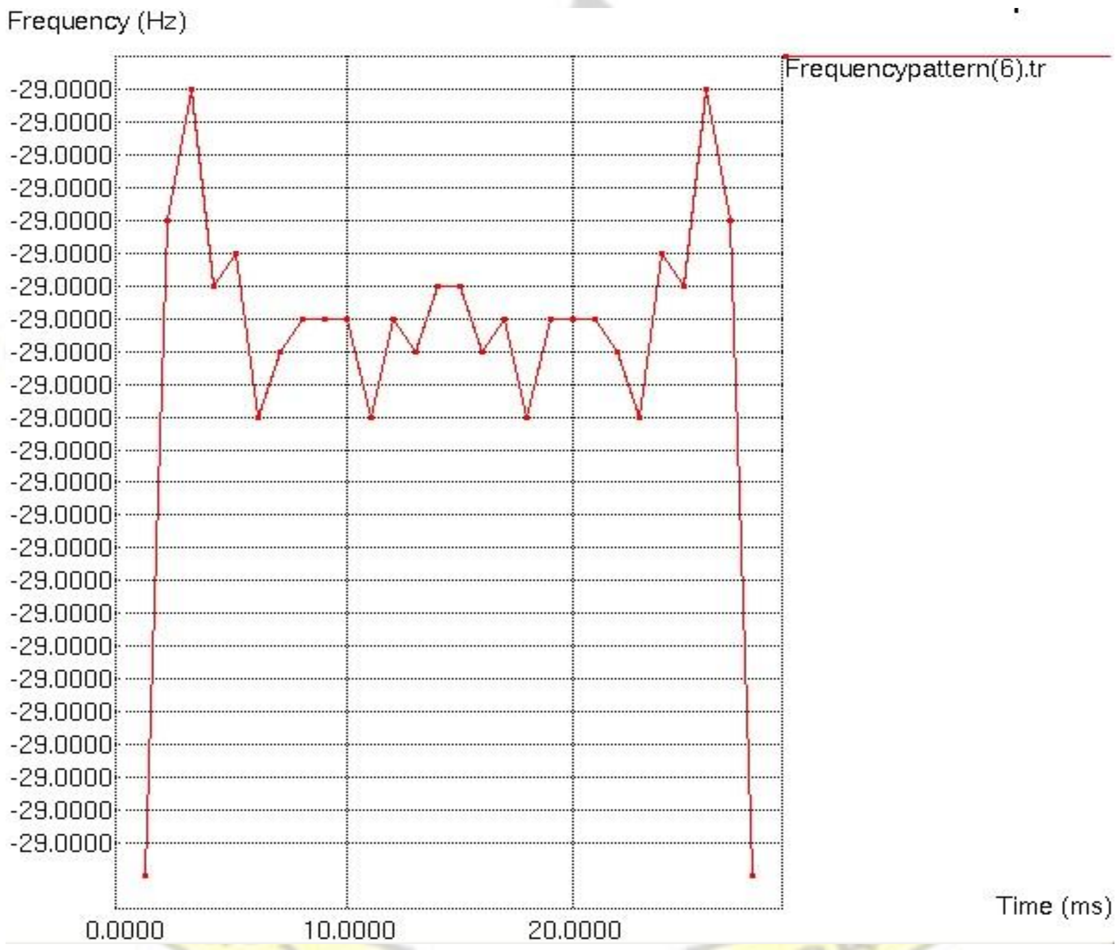


Figure 4.10: Frequency Pattern VI

The frequency pattern for node 6 in Figure 4.10 is also different from the other frequency patterns in the above Figures. In this node traffic data is broadcast to lists of its subnet machines periodically similar to the one in Figure 4.7. The frequency range of this graph is between -29.0000 and -28.9999 and it falls within the reference frequency range which declares node 6 to be free from attacker packets. This is confirmed by considering the mew value and the average variance of packet size in Table 4.1.

KNUST

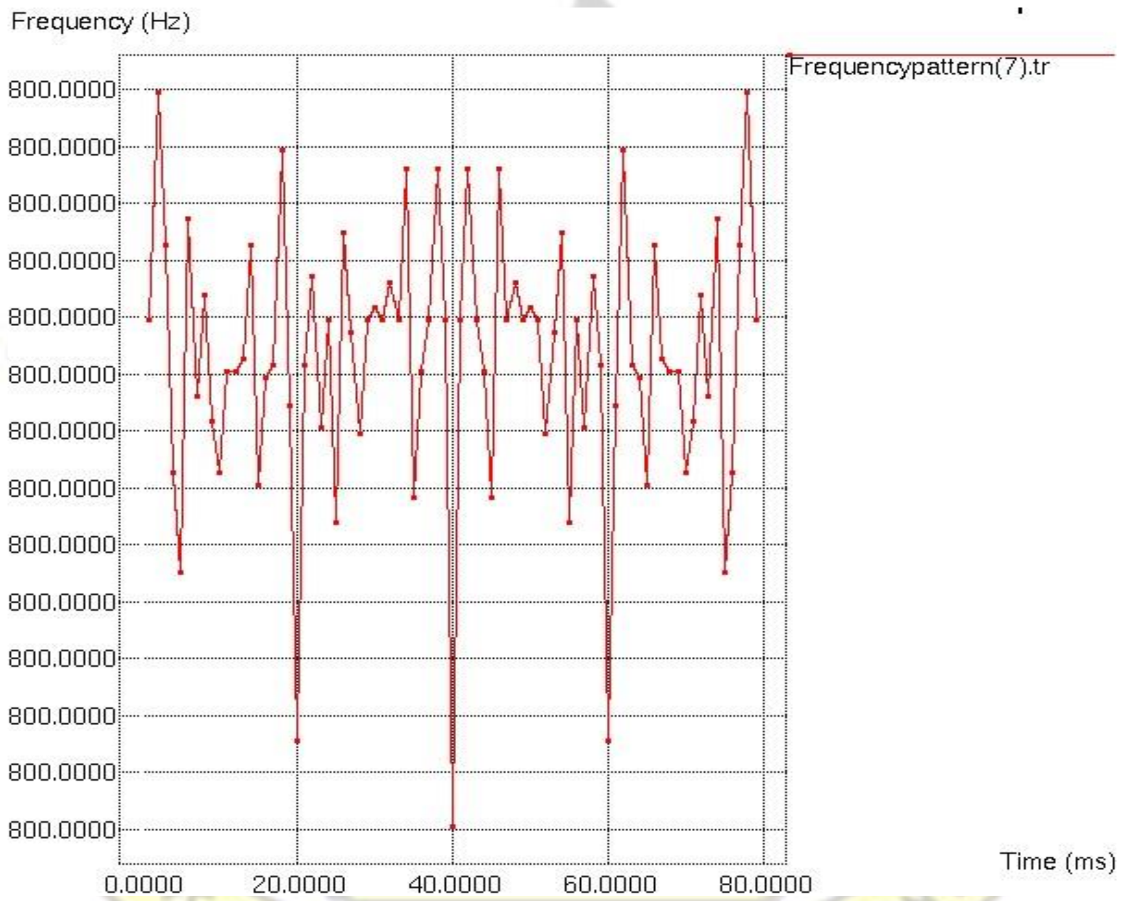


Figure 4.11: Frequency Pattern VII

W J SANE NO BADW

In this connection, it is observed that the frequency pattern and the frequency range shown in Figure 4.11 are distinct from all the above graphs. The frequency range for this node is between -800.0000and -799.0000 and it is out of the normal frequency range for this system which is between -45.0000 and -17.0000. Again, the mew value for node 7 in the connection history is 4038 and is above the normal threshold values set by the system by some amount. This mew value is abnormal as compared to the mew values of node 1 to node 6 as shown in Table 4.1. The average variance of packet size is 0.3186765101345355 which is too small and satisfies the condition of attacker packet for DOS attack. This can be considered as an automated and scripted attack demonstrates the occurrence of an intruder.

KNUST

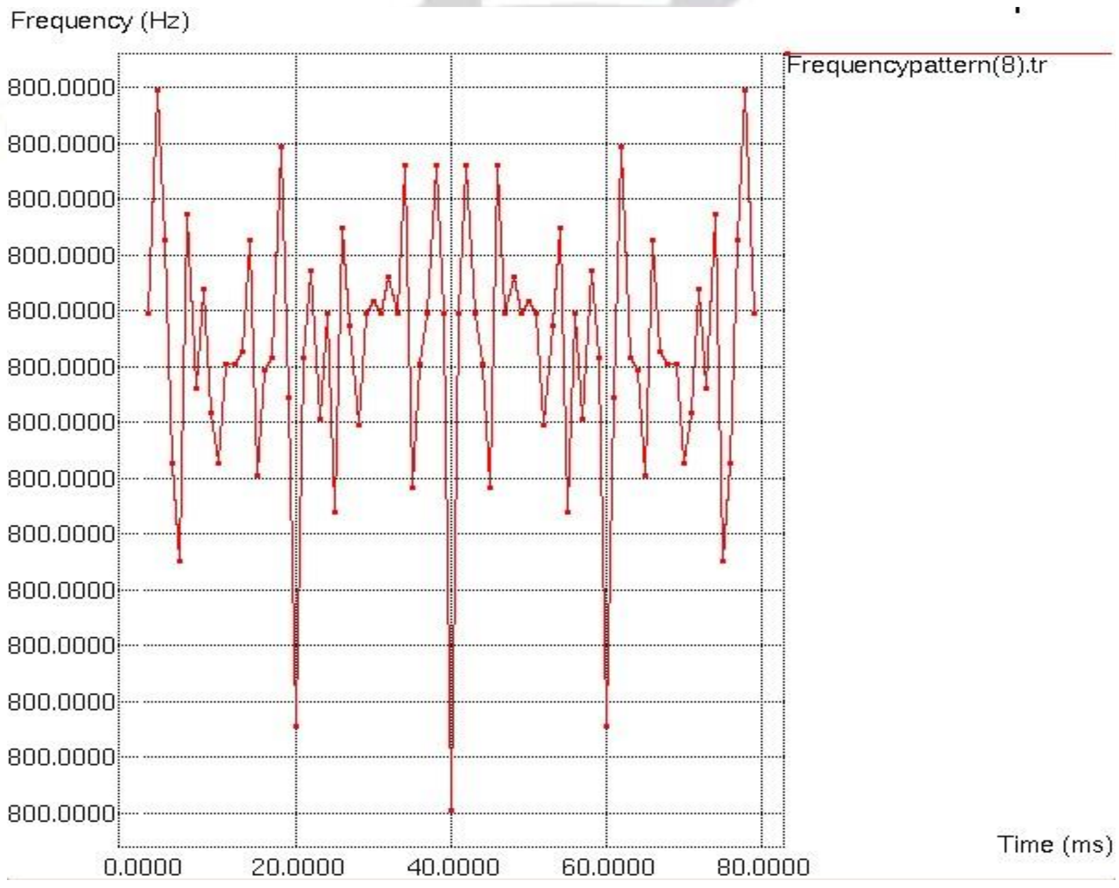


Figure 4.12: Frequency Pattern VIII

Connection eight is also having almost the same abnormalities as connection seven and it is identified as an attacker node. Figure 4.12 demonstrates the frequency pattern of node 8 and the frequency range for this node is between -800.0000 and -799.0000 as shown in the Y-axis of the above graph. It is observed that frequency range for node 8 is out of the normal frequency range for this system. The mew value and the average variance of packet size of connection eight is 3972 and 0.4375681123928461 respectively and these values are considered as anomalous.

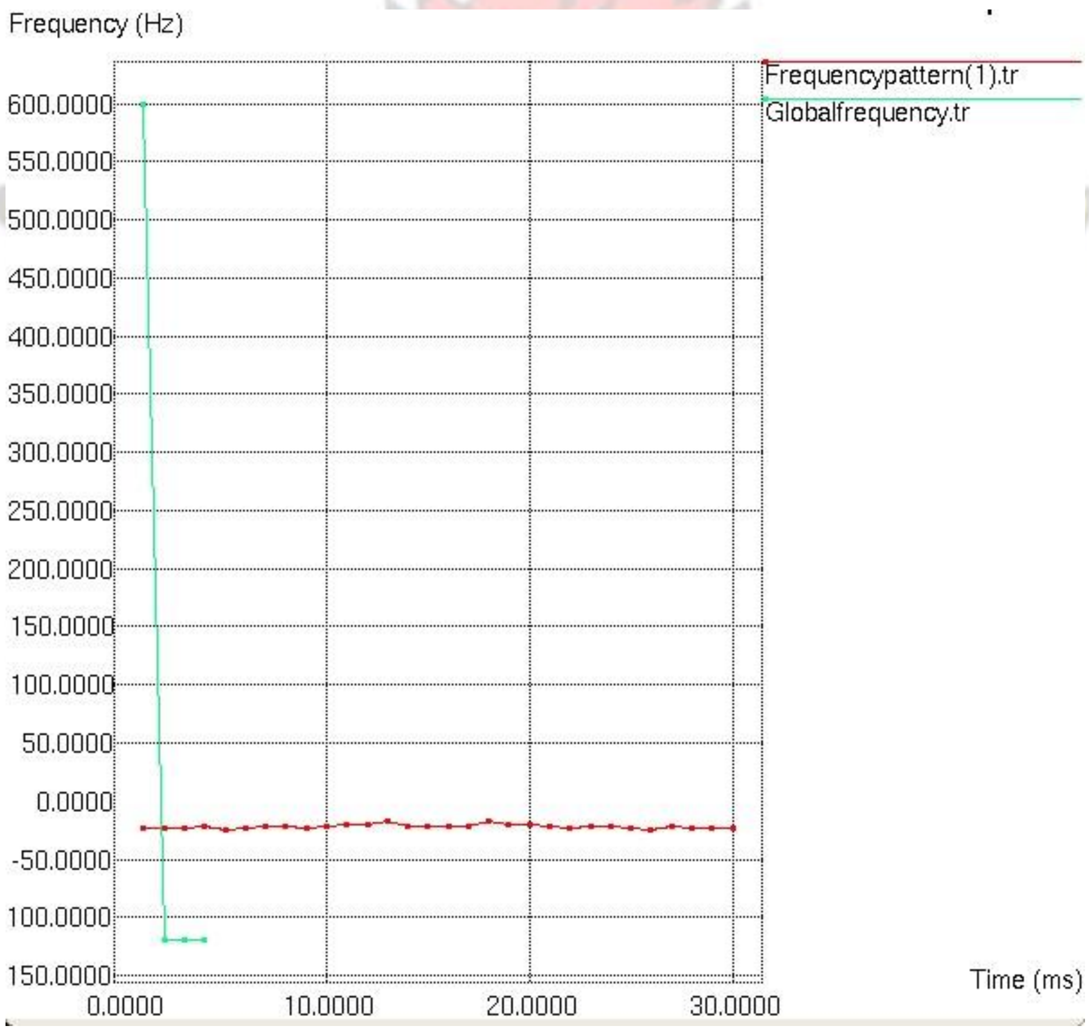


Figure 4.13: Global vs Local Frequency Patterns

Comparison of the Global and the Local frequency patterns are plotted in Figure 4.13. The Local frequency pattern is the horizontal red line in the graph and it is generated between the normal frequency range of -45.0000 and -17.0000. The green line is the Global frequency pattern and it is deviated from the Local frequency pattern. It can be seen that the frequency domain for Global frequency pattern is different from that of the Local frequency pattern which demonstrates that the Global traffic is having abnormal frequencies as compare to that of the Local traffic. The deviation of the Global frequency pattern in this situation, confirms that there is an intruder in the network.

KNUST
NYANSAPɔ Wɔ SANE NO BADWENMA

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This thesis focused on frequency-based intrusion detection approach to detect Denied of Service (DOS) attack in a virtual network system. The frequency-based approach looks for periodicity patterns in the time-series created by the traffic flow using Discrete Fourier Transform (DFT). A Centralized Traffic Analyzer Intrusion Detection System called CTA-IDS was introduced to further detect the intrusion accurately inside a network without any false alarm due to accurate threshold values set by the system. The strategy is able to detect abnormal content in the traffic data, known attack signature and unknown attack. This approach is tested by running the artificial network intrusion data in simulated networks using the Network Simulator2 (NS2) software.

From the simulation result, node 7 and node 8 did not conform to the frequency range set by the system and were detected as attacker nodes. These attacker nodes were disconnected from the network by the algorithm with the help of a control packet. Nodes 7 and 8 contained 4038 and 3972 as their average number of packets in the connection history respectively which is not normal as compared to nodes 1 to 6 in Table 4.1 above. The average variances of packet size of these two nodes were very small and are considered as automated DOS attack. The frequency pattern generated for the graphical visualization of the attacker nodes were not in range of the normal frequency range set by the system. The idea behind this frequency pattern analysis is that the traffic with constant Inter-arrival time is generated by the coded automated

DOS, probe or dictionary attacks which need many packets to finish a subtask in a repeated attack process.

5.2 Recommendations

In this thesis, network intrusion data in the simulated networks is used for frequencybased intrusion detection. Evaluating an IDS using artificial data is very effective but the overall traffic is often generated with fundamental regularity because of the absence of user interaction. For that reason, it is possible that the unnaturally attacked traffic will generate the periodic patterns while that is not often the case in the real world. Based on this observation, the future work will be tested on a real environment to confirm the effectiveness of this technique. It is highly recommended to use this approach for attack specific such as DOS, Probe and Dictionary attacks due to the fact that these types of attacks typically run from pre-written scripts and have relatively long duration which generate frequency pattern.

REFERENCES

[1] Cyber Threat Analysis, Power Point PPT Presentation, http://www. powershow.com/ view1/ 8c161-ZDc1Z/Cyber, accessed on 24 August 2014 @ 2:14pm GMT

[2] V. Marinova-Boncheva, “A Short Survey of Intrusion Detection Systems”, Institute of Information Technologies, 1113 Sofia, 2007, http://www.iit.bas. bg/PECR/58/23-30.pdf, accessed on 13November 2014 @ 10:05am GMT

[3] T. R. Peltier, “Introduction to Intrusion Detection”, http://csrc.nist.gov/ nissc/2000/proceedings/papers/300slide.pdf, accessed on 7October 2014 @ 12:48pm GMT

[4] S. Mukhekar, “Intrusion Detection Systems”, http://www.docstoc.com/ docs/ 13900942/ Intrusion-Detection-Systems-(DOC), accessed on 13June 2014 @ 11:03am GMT

[5] S. A. Khandelwal, S. A. Ade, A. A. Bhosle, R. S. Shirbhate, A Simplified Approach to Identiffy Intrusion in Network with Anti Attacking Using .net Tool, http://www.researchgate.net/publication/263550481,accessed on 21 September 2014 @ 5:37pm GMT

[6] P. Sharma, “ Artificial Neural Network”, http://www.studymode.com/ essays/Artificial-Neural-Network-1479899.html, accessed on 17 September

2014 @ 1:52pm GMT

[7] Concept draw, “Wired and wireless Internet connection for many users”, http://www.conceptdraw.com/examples/network-diagram, accessed date15th September 2014, accessed on 3February 2014 @ 7:23am GMT

[8] Incasula 2013-2014 DDOS Threat Landscape Report, http://www.imperva. com/docs/RPT_2013-2014_ddos_threat_landscape.pdf, accessed on 17 September2014 @ 3:05pm GMT

[9] Infosec Institute, “2013 The Impact of Cybercrime”, http://resources .infosecinstitute.com/ 2013-impact-cybercrime/, Posted in General Security on November 1, 2013, accessed on 16October 2014 @ 2:14pm GMT

[10] The Global Initiative Against Transnational Organized Crime, “Cybercrime and the Private Sector” http://www.globalinitiative.net/programs/cybercrime/ cybercrime-and-the-private-sector/, accessed on 10 July 2014 @ 11:34pm GMT

[11] S. Durbin, “The CIO's Secret Weapon: Stakeholder Pressure”, http://www .cioinsight.com/it-management/expert-voices/the-cios-secret-weaponstakeholder-pressure.html, accessed on 13 December 2014 @ 6:42pm GMT

[12] A. K. Ghosh, A. Schwartzbard, “A Study in Using Neural networks for

Anomaly and Misuse Detection", Ghosh, 1999.

[13] J. Ryan, M. Lin, "Intrusion Detection with Neural networks", NIPS, 1998

[14] S. Kumar, "Survey of Current Network Intrusion Detection Techniques", http://www.cse.wustl.edu/~jain/cse571-07/ftp/ids.pdf, accessed on 11thDecember 2014

[15] M. G. Schultz, E. Eskin, E. Zadok, "Data mining Methods for Detection of New Malicious Executable", Proceedings of the 2001 IEEE Symposium on Security and Privacy, IEEE 2001

[16] S. Forrest, S. A. Hofmeyr, A. Somayaji, "Computer immunology". Communications of the ACM, 40(10):88-96, October 1997.

[17] A. K. Ghosh, A. Schwartzbard, M. Schatz, "Learning Program Behavior Profiles for Intrusion Detection", 1st USENIX Workshop on Intrusion Detection and Network Monitoring, 1999

[18] C. Ko, "Logic Induction of Valid Behavior Specifications for Intrusion Detection", Proceedings of the 2000 IEEE Symposium on Security and Privacy, IEEE 2000.

[19] R. Sekar, M. Bendre, D. Dhurjati, "A Fast Automaton-Based Methods for

Detecting Anomalous Program Behavior", Proceedings of the IEEE Symposium on Security and Privacy, IEEE 200

[20] C. Abad, J. Taylor, C. Sengul, W. Yurcik, Y. Zhou, K. Rowe, "Log correlation for Intrusion Detection: A Proof of Concept", ACSAC 2003.

[21] S. Coull, J. Branch, B. Szymanski, E. Breimer, "Intrusion Detection: A Bioinfomatics Approach" Proceedings of 19th Annual Computer Security Applications Conference, Las Vegas, December, 2003.

[22] K. Ilgun, R. A. kemmerer, P. A. Porras, "State Transition Analysis: A ruleBased Intrusion Detection Approach" IEEE Transactions on software engineering, VOL21, No 3, March 1995.

[23] http://www.cisco.com/c/en/us/products/collateral/security/traffic-anomalydetector-xt-5600a/ prod_white_paper0900aecd8011e927.html, accessed on 28
September 2014 @ 3:41pm GMT

[24] L. Feinstein, D. Schnackenberg, R. Balupari, D. Kindred, "Statistical Approaches to DDoS Attack Detection and Response" Proceedings of DARPA information Survivability Conference and Exposition, IEEE, 2003.

[25] C. Cheng, H. T. Kung, K. Tan, "Use of spectral Analysis in Defense Against

DoS Attacks", IEEE GLOBECOM, 2002.

[26] T. M. Gil, M. Poletto, "MULTOPS: a data-structure for bandwidth attack detection", 10th Usenix Security Symposium, 2001.

[27] W. G. Morein, A. Stavrou, D. L. Cook, "Using Graphic Turning Test to Counter Automated DDOS Attacks Against Web Servers" CCS ACM, 2003.

[28] A. D. Keromytis, V. Misra, D. Rubenstein, "SOS: Secure overlay Services", SIGCOMM, 2002

[29] D. G. Andersen, "Mayday: Distributed Filtering for Internet Services", Proceedings of USENIX Symposium on Internet Technologies and Systems (USITS), 2003.

[30] J. Lemon, "Resisting SYN flood DOS attacks with a SYN cache", Proceedings of USENIX BSDCon', February 2002

[31] K. Park, H. Lee, "On the Effectiveness of probabilistic packet making for IP Trace back under Denial of Service Attack", Proceedings of IEEE INFOCOM, March 2001.

[32] H. Wang, D. Zhang, K. G. Shin "Detecting SYN Flooding Attacks" IEEE INFOCOM, 2002.

[33] CERT, "Code Red II: Another worm Exploiting Buffer Overflow in IIS Indexing Service DLL", Incident Note IN-2001-09, 2001.
http://www.cert.org/incident notes/IN-2001- 09.html, 2009

[34] V. Berk, G. Bakos, "Designing a Framework for Active Worm Detection on Global Networks", Proceedings of the First IEEE International Workshop on Information Assurance, 2003

[35] M. M. Williamson, "Throttling Viruses: Restricting propagation to defeat malicious mobilecode", 18th Annual Computer Security Applications Conference, 2002.

[36] T. Toth, C. Kruegel, "Connection-history Based Anomaly Detection", In Proceedings of the 2002 IEEE Workshop on Information Assurance and Security. June 2002.

[37] S. Sidiroglou, A. D. Keromytis , "A Network Worm Vaccine Architecture", 12th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, June 2003.

[38] D. Moore, C. Shannon, G. M. Voelker, S. Savage, "Internet Quarantine: Requirement for Containing Self-Propagation Code", INFOCOM IEEE, 2003.

[39] S. Staniford, V. Paxson, N. Weave, “How to own the internet in your spare time”, USENIX, 2002

[40] Z. Chen, L. Gao, K. Kwiat, “Modeling the Spread of Active Worms”, Proceedings of IEEE INFOCOM, 2003.

[41] N. C. Weaver, “A Warhol Worm: An Internet plague in 15 minutes!”, SAN technical report, 2003.

[42] D. Moore, C. Shanning, and K. Claffy. Code-Red: a case study on the spread and victims of an Internet worm. In Proceedings of the 2nd Internet Measurement Workshop (IMW), pages 273-284, November 2002.

[43] Y. Zhang, V. Paxson, “Detecting Backdoors”, Proceedings of the 9th USENIX Security Symposium, August 2000.

[44] M. Smart, G. R. Malan, F. Jahanian, “Defeating TCP/IP Stack Fingerprinting”, Proceedings of the 9th USENIX Security Symposium, August 2000.

[45] G. Vigna, R. A. Kemmerer, “NetSTAT: A Network-based Intrusion Detection Approach”,14th Annual Computer Security Applications Conference December 1998.

KNUST NYAMSAPA W J SANE NO BADWHEMA

[46] M. Roesch, “Snort: lightweight intrusion detection for networks”, USENIX 1999.

[47] V. Paxson. “Bro: A system for detecting network intruders in real-time”, Proceedings of 7thUSENIX Security Symp 1998

[48] C. J. Coit, S. Staniford, J. McAlerney, “Towards Faster String matching for Intrusion detection”, IEEE 2001.

[49] R. Sommer, V. Paxson, “Enhancing Byte-Level network Intrusion Detection Signatures with Context”, CCS’03 ACM 2003.

[50] D. Anderson, “Detecting unusual program behavior using the statistical component of the Next-generation Intrusion Detection Expert System (NIDES)”, Computer Science Laboratory SRI-CSL 95-06 May 1995.

[51] EMERALD intrusion detection system, at http://www.sdl.sri.com/projects/emerald, 2001, accessed date 14th November, 2014

[52] D. Barbara, N. Wu, S. Jajodia,”ADAM: Detecting Intrusions by Data mining”, Proceedings of the 2001 IEEE, workshop on information Assurance and Security. 2001.

[53] SPADE, Silicon Defense, http://www.silicondefense.com/software/spice

[54] W. Lee, S. J. Stolfo, K. W. Mok, "Mining in a Data-flow Environment: Experience in Network Intrusion Detection", Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining (KDD), ACM 1999.

[55] W. Lee, S. Stolfo, "Adaptive Intrusion Detection: a Data Mining Approach", Artificial Intelligence Review, 2000.

[56] D. Barbara, et al., "ADAM: A testbed for exploring the Use of Data Mining in Intrusion Detection", SIGMOD Record 2001.

[57] S. Bridges, R. Vaughn, "Fuzzy Data Mining and Genetic Algorithms Applied to Intrusion Detection", NISSC 2000.

[58] C. Sinclair, L. Pierce, S.Marzner, "An Application of Machine learning to Network Intrusion Detection", Proceedings of the 15th Annual Computer Security Applications Conference 1999

[59] X. Li, N. Ye, "Decision tree classifiers for computer intrusion detection", Real-time system security, ACM 2003.

[60] J. Ross Quinlan. C4.5 Programs for Machine Learning Morgan Kaufmann Publishers, San Mateo, CA 1993

[61] Yi-Leh Wu et.al, "A Comparison of DFT and DWT Based Similarity Search in Time-Series Databases," Proceedings of the ninth International Conference on Information and Knowledge Management, pp. 488-495, 2000

[62] K. Kawagoe and Tomohiro Ueda, "A similarity search method of Time series data with combination of Fourier and wavelet Transforms", IEEE TIME'02, 2002.

[63] S. Sharma, "Digital Signal Processing" for 6th Semester B. Tech Students of Punjab Technical University, Jalandhar, 2008

[64] http://en.wikipedia.org/wiki/Network_simulation, accessed date 28th September 2014.

[65] J. Zhang, W. Li, D. Cui, X. Zhao , Z. Yin, "The NS2-Based Simulation and Research on Wireless Sensor Network Route Protocol" Wireless Communications, Networking and Mobile Computing, WiCom '09. 5th International Conference on, 2009

[66] http://www.researchgate.net/post/Can_anyone_explain_the_concept_of_negative_frequency, accessed on 11February 2015 @ 1:21pm GMT

APPENDIX A

Code Listings

All the source codes used in the simulation are shown below.

A. 1 Tcl Coding Script for IDS Simulation Parameters

```
#===================================
#     Simulation parameters setup
#===================================
setval(stop)   30.0                    ;# time of simulation end

#===================================
#       Initialization
#===================================
#Create a ns simulator set
ns [new Simulator]

#Open the NS trace file setpacdelrat [open
TrafficData_global.tr w]
```

```
$ns trace-all $pacdelrat for {set i 0} {$i<7} {incri}
{       settdf($i) [open TrafficData_local($i).tr
w]
        $ns trace-all $tdf($i)
} setch [open connectionhistory.tr
w]
$ns trace-all $ch setgff [open
Globalfrequency.tr w]
$ns trace-all $gff
for {set i 1} {$i<=8} {incri} {      setfpf($i) [open
Frequencypattern($i).tr w]
        $ns trace-all $fpf($i)
} settracefile [open out.tr
w]
$ns trace-all $tracefile

#Open the NAM trace file setnamfile
[open ids_dft.nam w]
$ns namtrace-all $namfile

#===================================
#     Nodes Definition
#===================================
#Create 13 nodes set
n(0) [$ns node] set
n(1) [$ns node] set
n(2) [$ns node] set
```

```
n(3) [$ns node] set

n(4) [$ns node] set

n(5) [$ns node] set

n(6) [$ns node] set

n(7) [$ns node] set

n(8) [$ns node] set

n(9) [$ns node] set

n(10) [$ns node] set

n(11) [$ns node] set

n(12) [$ns node]


#Node color

$n(0) color blue

$n(0) label "HUB"

# $n(7) color red

# $n(7) label "Attacker1"

# $n(8) color red # $n(8) label

"Attacker2" set pc [list] for

{set i 1} {$i<13} {incri} {

        # if {$i!=7 && $i!=8} {

                $n($i) coloryellowgreen

$n($i) label "PC$i"                lappend

pc $i

        # }

} procrandelem {list} { lindex $list [expr

{int(rand()*[llength $list])}]

}
```

KNUST
NYANSAPƆ WƆ SANE NO BADWENMA

```
#=====================================
#       Links Definition
#=====================================
for {set i 0} {$i<13} {incri} {         for
{set j 0} {$j<13} {incr j} {
                    if {$i!=$j} {
          set link($i,$j) 0
                    }
          }
}
#Create links between nodes
$ns duplex-link $n(1) $n(0) 100.0Mb 10ms DropTail
$ns queue-limit $n(1) $n(0) 50
set link(1,0) 1
$ns duplex-link $n(5) $n(0) 100.0Mb 10ms DropTail
$ns queue-limit $n(5) $n(0) 50 set
link(5,0) 1
$ns duplex-link $n(2) $n(0) 100.0Mb 10ms DropTail
$ns queue-limit $n(2) $n(0) 50 set
link(2,0) 1
$ns duplex-link $n(4) $n(0) 100.0Mb 10ms DropTail
$ns queue-limit $n(4) $n(0) 50 set
link(4,0) 1
$ns duplex-link $n(6) $n(0) 100.0Mb 10ms DropTail
$ns queue-limit $n(6) $n(0) 50 set
link(6,0) 1
$ns duplex-link $n(3) $n(0) 100.0Mb 10ms DropTail
```

```
$ns queue-limit $n(3) $n(0) 50 set

link(3,0) 1

$ns duplex-link $n(7) $n(1) 100.0Mb 10ms DropTail

$ns queue-limit $n(7) $n(1) 50 set

link(7,1) 1

$ns duplex-link $n(7) $n(3) 100.0Mb 10ms DropTail

$ns queue-limit $n(7) $n(3) 50 set

link(7,3) 1

$ns duplex-link $n(10) $n(4) 100.0Mb 10ms DropTail

$ns queue-limit $n(10) $n(4) 50 set

link(10,4) 1

$ns duplex-link $n(9) $n(2) 100.0Mb 10ms DropTail

$ns queue-limit $n(9) $n(2) 50 set

link(9,2) 1

$ns duplex-link $n(8) $n(2) 100.0Mb 10ms DropTail

$ns queue-limit $n(8) $n(2) 50

set link(8,2) 1

$ns duplex-link $n(8) $n(4) 100.0Mb 10ms DropTail

$ns queue-limit $n(8) $n(4) 50 set

link(8,4) 1

$ns duplex-link $n(11) $n(1) 100.0Mb 10ms DropTail

$ns queue-limit $n(11) $n(1) 50 set

link(11,1) 1

$ns duplex-link $n(12) $n(3) 100.0Mb 10ms DropTail

$ns queue-limit $n(12) $n(3) 50 set

link(12,3) 1
```

```
#Give node position (for NAM)

$ns duplex-link-op $n(1) $n(0) orient right-down

$ns duplex-link-op $n(5) $n(0) orient right

$ns duplex-link-op $n(2) $n(0) orient right-up

$ns duplex-link-op $n(4) $n(0) orient left-up

$ns duplex-link-op $n(6) $n(0) orient left

$ns duplex-link-op $n(3) $n(0) orient left-down

$ns duplex-link-op $n(7) $n(1) orient left-down

$ns duplex-link-op $n(7) $n(3) orient right-down

$ns duplex-link-op $n(10) $n(4) orient left

$ns duplex-link-op $n(9) $n(2) orient right $ns

duplex-link-op $n(8) $n(2) orient left-up

$ns duplex-link-op $n(8) $n(4) orient right-up

$ns duplex-link-op $n(11) $n(1) orient right

$ns duplex-link-op $n(12) $n(3) orient left


#connection history puts $ch "Node1 Node2

IsConnection" puts $ch

"========================" for {set

i 0} {$i<13} {incri} {        for {set j 0}

{$j<13} {incr j} {

                    if {$i!=$j} {

if {$link($i,$j)==1} {

                                        puts $ch "$i          $j

yes"                          } else {                              puts

$ch "$i    $j          No"

                              }
```

KNUST

```
                    }
            }
}
#====================================
#       Agents Definition
#====================================
proc attach-cbr-traffic {node sink size interval} {
            global ns
            set source [new Agent/UDP]
$source set class_ 2          $ns attach-agent $node
$source  set traffic [new
Application/Traffic/CBR]
            $traffic set packetSize_ $size
            $traffic set interval_ $interval
            $traffic attach-agent $source
$ns connect $source $sink
return $traffic
}
#====================================
#       Applications Definition
#====================================
#Global Traffic data set null [new Agent/LossMonitor]
$ns attach-agent $n(0) $null for {set i 1} {$i<7}
{incri} {             setcbr [attach-cbr-traffic $n($i) $null
100 0.05]
            $ns at 0.0 "$n($i) add-mark m1 blue"
            $ns at 0.0 "$cbr start"
            $ns at 10.0 "$cbr stop"
} set flag 0 for {set i 1}
{$i<7}  {incri} {
            $ns at 10.0 "$n($i) delete-mark m1"
} set pd1 0 setnps 0 proc record {} {          global null ns
npspacdelratpaclossrat delay thr pd1 flag
            set now [$ns now]
set time 0.5
            set pd2 [$null set npkts_]
setpd [expr $pd2-$pd1]     puts
$pacdelrat "$now $pd2"    set pd1
$pd2      if {$flag==0} {
                        $ns at [expr $now+$time] "record"
            }
}
$ns at 0.0 "record" setdrl [list
0.04 0.05 0.06 0.07] setpsl
[list 100 150 200 250] settim
10.0


#Denial of service attack setnulla [new
Agent/LossMonitor] $ns attach-agent $n(1) $nulla
setcbra [attach-cbr-traffic $n(7) $nulla 1000 0.005]
$ns at 0.0 "$n(8) add-mark m1a yellowgreen"
$ns at 0.0 "$cbra start" $ns at 28.0 "$cbra stop"
setnullb [new Agent/LossMonitor] $ns attach-
```

```
agent $n(4) $nullb setcbrb [attach-cbr-traffic $n(8)

$nullb 1000 0.005]

$ns at 0.0 "$n(8) add-mark m1a yellowgreen"

$ns at 0.0 "$cbrb start" $ns

at 28.0 "$cbrb stop" for {set

i 0} {$i<9} {incri} {

        set f($i) 0

setlds($i) [list]

} set pd1 0 procrecorda {} {        globalnulla

ns pd1 flag array names lds         set now

[$ns now]        set time 0.1

        set pd2a [$nulla set npkts_]

setpd [expr $pd2a-$pd1]

        set pd1 $pd2a

        if {$pd2a!=0}

        {

                lappendlds(7) $pd2a

        }

        $ns at [expr $now+$time] "recorda"

}

$ns at 20.0 "recorda" set pd1 0 procrecordb {} {

globalnullb ns pd1 flag array names lds     set

now [$ns now]    set time 0.1

        set pd2b [$nullb set npkts_]

setpd [expr $pd2b-$pd1]   set pd1

$pd2b    if {$pd2b!=0} {

lappendlds(8) $pd2b
```

```
            }
            $ns at [expr $now+$time] "recordb"
}
$ns at 20.0 "recordb"

#local Traffic data for {set i
1} {$i<7}  {incri} {
            set null1($i) [new Agent/LossMonitor]
            $ns attach-agent $n(0) $null1($i)
            Setdr [randelem $drl]
            Setps($i) [randelem $psl]   set cbr1 [attach-cbr-traffic
$n($i) $null1($i) $ps($i) $dr]
            $ns at $tim "$n($i) add-mark m1 blue"
            $ns at $tim "$cbr1 start" $ns at
            [expr $tim+3.0] "$cbr1 stop"
            settim [expr $tim+3.0]

            $ns at $tim "$n($i) delete-mark m1"
} proc record1 {} {           global array names lds array names null1 ns nps  pacdelrat1 flag array names
tdf array names f
            set flag 1
set now [$ns now]
set time 0.1
            set pd21 [$null1(1) set
npkts_]  puts $tdf(1) "$now $pd21"
if {$pd21!=0} {
lappendlds(1) $pd21
            }
```

KNUST

```
          if {$f(1)==0} {
                    $ns at [expr $now+$time] "record1"
          }
}
$ns at 10.0 "record1" proc record2 {} {         global array names lds ns nps  pacdelrat1 flag array names
tdf array names f array names null1
          set f(1) 1
set now [$ns now]
set time 0.1
          set pd22 [$null1(2) set
npkts_]   puts $tdf(2) "$now $pd22"
if {$pd22!=0} {
lappendlds(2) $pd22
          }
          if {$f(2)==0} {
                    $ns at [expr $now+$time] "record2"
          }
}
$ns at 13.0 "record2" proc record3 {} {         global array names lds ns nps  pacdelrat1 flag array names
tdf array names f array names null1
          set f(2) 1
set now [$ns now]
set time 0.1
          set pd23 [$null1(3) set
npkts_]   puts $tdf(3) "$now $pd23"
if {$pd23!=0} {
lappendlds(3) $pd23
```

KNUST
NYANSAPƆ WɔSANE NO BADWENMA

```
        }
        if {$f(3)==0} {
                $ns at [expr $now+$time] "record3"
        }
}
$ns at 16.0 "record3" proc record4 {} {      global array names lds ns nps pacdelrat1 flag array names
tdf array names f array names null1
        set f(3) 1
set now [$ns now]
set time 0.1
        set pd24 [$null1(4) set
npkts_] puts $tdf(4) "$now $pd24"
if {$pd24!=0} {
lappendlds(4) $pd24
        }
        if {$f(4)==0} {
                $ns at [expr $now+$time] "record4"
        }
}
$ns at 19.0 "record4" proc record5 {} {      global array names lds ns nps pacdelrat1 flag array names
tdf array names f array names null1
        set f(4) 1
set now [$ns now]
set time 0.1
        set pd25 [$null1(5) set
npkts_] puts $tdf(5) "$now $pd25"
```

```
if {$pd25!=0} {
lappendlds(5) $pd25
            }
            if {$f(5)==0} {
                        $ns at [expr $now+$time] "record5"
            }
}
$ns at 22.0 "record5" proc record6 {} {         global array names lds ns nps pacdelrat1 flag array names
tdf array names f array names null1
            set f(5) 1
set now [$ns now]
set time 0.1
            set pd26 [$null1(6) set
npkts_]  puts $tdf(6) "$now $pd26"
if {$pd26!=0} {
lappendlds(6) $pd26
            }
            $ns at [expr $now+$time] "record6"
}
$ns at 25.0 "record6"


setgds [list 60 120 180 240]


#Fast Discrete Fourier Transform namespaceeval
::math::fourier {    #::math::constants pi
namespace export
dftinverse_dftlowpasshighpass
```

```
} proc ::math::fourier::DFT_make_roots {n sign}

{

set res [list] for {set k 0} {2*$k < $n} {incr k} { set

alpha [expr {2*3.1415926535897931*$sign*$k/$n}]

lappend res [expr {cos($alpha)}] [expr {sin($alpha)}]

} return $res } proc ::math::fourier::Fast_DFT {dataLrootL} { if {[llength

$dataL] == 8} then { foreach {Re_z0 Im_z0 Re_z1 Im_z1 Re_z2 Im_z2 Re_z3

Im_z3} $dataL {break} if {[lindex $rootL 3] > 0} then { return [list\

[expr {$Re_z0 + $Re_z1 + $Re_z2 + $Re_z3}] [expr {$Im_z0 + $Im_z1 + $Im_z2 + $Im_z3}]\

 [expr {$Re_z0 - $Im_z1 - $Re_z2 + $Im_z3}] [expr {$Im_z0 + $Re_z1 - $Im_z2 - $Re_z3}]\

  [expr {$Re_z0 - $Re_z1 + $Re_z2 - $Re_z3}] [expr {$Im_z0 - $Im_z1 + $Im_z2 - $Im_z3}]\

[expr {$Re_z0 + $Im_z1 - $Re_z2 - $Im_z3}] [expr {$Im_z0 - $Re_z1 - $Im_z2 + $Re_z3}]]

    } else { return

[list\

[expr {$Re_z0 + $Re_z1 + $Re_z2 + $Re_z3}] [expr {$Im_z0 + $Im_z1 + $Im_z2 + $Im_z3}]\

[expr {$Re_z0 + $Im_z1 - $Re_z2 - $Im_z3}] [expr {$Im_z0 - $Re_z1 - $Im_z2 + $Re_z3}]\

[expr {$Re_z0 - $Re_z1 + $Re_z2 - $Re_z3}] [expr {$Im_z0 - $Im_z1 + $Im_z2 - $Im_z3}]\ [expr
{$Re_z0 - $Im_z1 - $Re_z2 + $Im_z3}] [expr {$Im_z0 + $Re_z1 - $Im_z2 - $Re_z3}]]

       }

} elseif {[llength $dataL] > 8} then { setevenL

[list]   setoddL   [list]   foreach   {Re_z0   Im_z0

Re_z1 Im_z1} $dataL { lappendevenL $Re_z0

$Im_z0 lappendoddL $Re_z1 $Im_z1

 } setsquarerootL [list] foreach {Re_omega0 Im_omega0 Re_omega1

Im_omega1}      $rootL      {      lappendsquarerootL      $Re_omega0

$Im_omega0

 } setlowL

[list]
```

```
sethighL

[list] foreach\

        {Re_y0 Im_y0}      [Fast_DFT $evenL $squarerootL]\

        {Re_y1 Im_y1}      [Fast_DFT $oddL $squarerootL]\

        {Re_omegaIm_omega} $rootL { set Re_y1t [expr {$Re_y1 *

$Re_omega - $Im_y1 * $Im_omega}] set Im_y1t [expr {$Im_y1 *

$Re_omega + $Re_y1 * $Im_omega}] lappendlowL  [expr {$Re_y0 +

$Re_y1t}] [expr {$Im_y0 + $Im_y1t}] lappendhighL [expr {$Re_y0 -

$Re_y1t}] [expr {$Im_y0 - $Im_y1t}]

      }

return [concat $lowL $highL]

} elseif {[llength $dataL] == 4} then { foreach

{Re_z0 Im_z0 Re_z1 Im_z1} $dataL {break}

return [list\

        [expr {$Re_z0 + $Re_z1}] [expr {$Im_z0 + $Im_z1}]\

        [expr {$Re_z0 - $Re_z1}] [expr {$Im_z0 - $Im_z1}]]

} else { return

$dataL

   }

} proc ::math::fourier::Slow_DFT {dataLrootL}

{ set n [expr {[llength $dataL] / 2}]


# The missing roots are computed by complex conjugating the given

# roots. If $n is even then -1 is also needed; it is inserted explicitly.

set k [llength $rootL] if

{$n % 2 == 0} then {

lappendrootL -1.0 0.0
```

```
  } for {incr k -2} {$k > 0} {incr k -
2}  {  lappendrootL  [lindex  $rootL
$k]\
   [expr {-[lindex $rootL [expr {$k+1}]]}]
    }


# This is strictly following the naive formula.
# The product jk is kept as a separate counter variable.
set res [list] for {set k 0} {$k < $n} {incr k} {
setRe_sum 0.0 setIm_sum 0.0
setjk 0 foreach {Re_zIm_z} $dataL {
setRe_omega [lindex $rootL [expr {2*$jk}]]
setIm_omega [lindex $rootL [expr
{2*$jk+1}]] setRe_sum [expr {$Re_sum +
$Re_z * $Re_omega - $Im_z * $Im_omega}]
setIm_sum [expr {$Im_sum +
 $Im_z * $Re_omega + $Re_z * $Im_omega}]
Incrjk $k if {$jk>= $n} then {set jk [expr {$jk
- $n}]}
       }
lappend res $Re_sum $Im_sum
    }
return $res } proc
::math::fourier::dft {in_data} { #
First convert to internal format
setdataL [list] setn 0 foreach datum
```

KNUST

```
$in_data { if {[llength $datum] ==

1} then { lappenddataL $datum 0.0

  } else { lappenddataL [lindex $datum 0] [lindex

$datum 1]

   } incr

n

    }



    # Then compute a list of n'th roots of unity (explanation below) setrootL

[DFT_make_roots $n -1]



    # Check if the input length is a power of two.

setp 1 while {$p < $n} {set p [expr {$p

<< 1}]}

    # By construction, $p is a power of two. If $n==$p then $n is too.     # Finally compute the transform
using Fast_DFT or Slow_DFT,

    # and convert back to the input format.

set res [list]

foreach {Re Im} [ if

{$p == $n} then {

Fast_DFT $dataL $rootL

        } else {

Slow_DFT $dataL $rootL

        }

    ] {

lappend res $Re

    }
```

```
return $res } set result
[::math::fourier::dft $gds] puts
"Result: $result"
$ns at [$ns now] "$ns trace-annotate \"The Frequency of Signal:$result\""
set ci1 1 foreach rv1 $result {
        puts $gff "$ci1 $rv1"
        incr ci1
}


for {set i 1} {$i<=8} {incri} {
set result1($i) [list]
} procfrequencyExtraction {} {    global array names lds ns array
names result1 array names fpf     for {set i 1} {$i<=8} {incri} {
            set ci 1
            set now [$ns now]
            set result1($i) [::math::fourier::dft $lds($i)]
puts "Result$i: $result1($i)"
            #   $ns   at   $now   "$ns   trace-annotate   \"Frequency   pattern   for
connection$i:$result1($i)\""
            set index [lindex $result1($i) 0]
foreachrv $result1($i) {
                    if {$rv!=$index} {
            puts $fpf($i) "$ci $rv"
                    incr ci
                    }
            }
        }
```

KNUST

```
}
$ns at 28.0 "frequencyExtraction"


#Average variance of packet sizes #mean of the packet
sizes procAveragevariance {} { global array names lds ns
array names n array names result1 for {set i 1} {$i<=8}
{incri} {
        setic 1
        foreachelt $lds($i) {
                if {$ic<=5} {
        setps($i,$ic) $elt
                        incric
                }
        }
} setnv
5 for
{set i
1}
{$i<=
8}
{incri}
{

        set aps($i) 0
} for {set i 1} {$i<=8} {incri} {      for {set j 1}
{$j<=5} {incr j} {                     set aps($i) [expr
$aps($i)+$ps($i,$j)]
        }
```

KNUST
NYANSAPƆ WɔSANE NO BADWENMA

```
} for {set i 1} {$i<=8} {incri}

{

 setti  [$ns  now]    set  mew($i)  [expr

$aps($i)/$nv]               puts        "Mew

value($i):$mew($i)"

            $ns at $ti "$ns trace-annotate \"Mew Value for Connection $i:$mew($i)\""

}


for {set i 1} {$i<=8} {incri} {

            setavp($i) 0

} for {set i 1} {$i<=8} {incri} {      for {set j 1}

{$j<=5} {incr j} {                     setinv [expr

$ps($i,$j)-$mew($i)]

                        if {$inv<0} {

setinv [expr $inv*-1]

                        }

                        setsrv [exprsqrt($inv)]

setavp($i) [expr $avp($i)+$srv]

            }

} setavpsl [list] for {set i 1}

{$i<=8} {incri} {

setti [$ns now]    setavps($i)

[expr $avp($i)/$nv]

lappendavpsl $avps($i)

            puts "Average Variance of packet size for connection $i:$avps($i)"

             $ns at $ti "$ns trace-annotate \"Average Variance of Packet size for Connection $i:$avps($i)\""
```

```
} setsps [lsort -decreasing
$avpsl] sethpv [lindex $sps 0]
for {set i 1} {$i<=8} {incri} {
        if {$avps($i)==$hpv} {
                $ns at [$ns now] "$n($i) color red"
                $ns at [$ns now] "$n($i) label Attacker_node"
                $ns at [$ns now] "$ns trace-annotate \"The node$i is an Attacker Node\""
        }
}
$ns at [$ns now] "$ns duplex-link-op $n(7) $n(1) color red"
$ns at [$ns now] "$ns duplex-link-op $n(7) $n(3) color red"
$ns at [$ns now] "$ns duplex-link-op $n(8) $n(2) color red" $ns
at [$ns now] "$ns duplex-link-op $n(8) $n(4) color red"
}
$ns at 29.0 "Averagevariance"


#===================================
#       Termination
#===================================
#Define a 'finish' procedure
proc finish {} { global ns
tracefilenamfile
$ns flush-trace close
$tracefile close
$namfile
execnamids_dft.na
m&
```

```
exit 0

}

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "finish"

$ns at $val(stop) "puts \"done\" ; $ns halt"

$ns run
```

A. 2 Tcl Coding Script for Main Window

```
# explicitly setup our main window


wm geometry  . 850x350+300+200 wm title  .   "A Frequency-

Based Approach to Intrusion Detection"


# setup the frame stuff destroy .myArea set f [frame .myArea

-borderwidth 5 -background seagreen] pack $f -side top -

expand true -fill both


# create a menubar

destroy   .menubar

menu .menubar

. config -menu .menubar


#  create a pull down menu with a label


set File2 [menu .menubar.mFile2]

.menubar add cascade -label "IDS"  -menu  .menubar.mFile2


set File3 [menu .menubar.mFile3]
```

KNUST

```
.menubar add cascade -label "FrequencyPatterns"  -menu  .menubar.mFile3

set File4 [menu .menubar.mFile4]
.menubar add cascade -label "Comparison"  -menu  .menubar.mFile4

set close [menu .menubar.sFile]
.menubar add cascade -label Quit  -menu  .menubar.sFile

# add the menu item
$File2 add command -label Run_IDS_DFT -command {exec ns ids1.tcl &}
$File2 add command -label Run_NAM -command {exec namout.nam&}

$File3 add command -label connection1 -command {exec xgraphFrequencypattern(1).tr -x "Time
(ms)" -y "Frequency (Hz)"  -bg "white" -fg "black" -zg "white" -lw 1 -M -bb &}
$File3 add command -label connection2 -command {exec xgraphFrequencypattern(2).tr -x "Time
(ms)" -y "Frequency (Hz)"  -bg "white" -fg "black" -zg "white" -lw 1 -M -bb &}
$File3 add command -label connection3 -command {exec xgraphFrequencypattern(3).tr -x "Time
(ms)" -y "Frequency (Hz)"  -bg "white" -fg "black" -zg "white" -lw 1 -M -bb &}
$File3 add command -label connection4 -command {exec xgraphFrequencypattern(4).tr -x "Time
(ms)" -y "Frequency (Hz)"  -bg "white" -fg "black" -zg "white" -lw 1 -M -bb &}
$File3 add command -label connection5 -command {exec xgraphFrequencypattern(5).tr -x "Time
(ms)" -y "Frequency (Hz)"  -bg "white" -fg "black" -zg "white" -lw 1 -M -bb &}
$File3 add command -label connection6 -command {exec xgraphFrequencypattern(6).tr -x "Time
(ms)" -y "Frequency (Hz)"  -bg "white" -fg "black" -zg "white" -lw 1 -M -bb &}
$File3 add command -label connection7 -command {exec xgraphFrequencypattern(7).tr -x "Time
(ms)" -y "Frequency (Hz)"  -bg "white" -fg "black" -zg "white" -lw 1 -M -bb &}
```

```
$File3 add command -label connection8 -command {exec xgraphFrequencypattern(8).tr -x "Time (ms)"
-y "Frequency (Hz)"  -bg "white" -fg "black" -zg "white" -lw 1 -M -bb &}
$File4 add command -label GlobalVsLocal -command {exec xgraphFrequencypattern(1).tr
Globalfrequency.tr -x "Time (ms)" -y "Frequency (Hz)"  -bg "white" -fg "black" -zg "white" -lw 1 -M bb
&}
$close add command -label Quit -command exit
```

KNUST

A. 3 Tcl Coding Script for DFT

```
namespaceeval ::math::fourier {
  #::math::constants pi

namespace export dftinverse_dftlowpasshighpass
} proc ::math::fourier::DFT_make_roots {n sign}
{
set res [list] for {set k 0} {2*$k < $n} {incr k} { set
alpha [expr {2*3.1415926535897931*$sign*$k/$n}]
lappend res [expr {cos($alpha)}] [expr {sin($alpha)}]
   }
return $res } proc ::math::fourier::Fast_DFT {dataLrootL} { if {[llength $dataL]
== 8} then { foreach {Re_z0 Im_z0 Re_z1 Im_z1 Re_z2 Im_z2 Re_z3 Im_z3}
$dataL {break} if {[lindex $rootL 3] > 0} then { return [list\
            [expr {$Re_z0 + $Re_z1 + $Re_z2 + $Re_z3}] [expr {$Im_z0 + $Im_z1 + $Im_z2 +
$Im_z3}]\
            [expr {$Re_z0 - $Im_z1 - $Re_z2 + $Im_z3}] [expr {$Im_z0 + $Re_z1 - $Im_z2 - $Re_z3}]\
            [expr {$Re_z0 - $Re_z1 + $Re_z2 - $Re_z3}] [expr {$Im_z0 - $Im_z1 + $Im_z2 - $Im_z3}]\
            [expr {$Re_z0 + $Im_z1 - $Re_z2 - $Im_z3}] [expr {$Im_z0 - $Re_z1 - $Im_z2 + $Re_z3}]]
```

```
        } else {
return [list\
            [expr {$Re_z0 + $Re_z1 + $Re_z2 + $Re_z3}] [expr {$Im_z0 + $Im_z1 + $Im_z2 +
$Im_z3}]\
            [expr {$Re_z0 + $Im_z1 - $Re_z2 - $Im_z3}] [expr {$Im_z0 - $Re_z1 - $Im_z2 + $Re_z3}]\
            [expr {$Re_z0 - $Re_z1 + $Re_z2 - $Re_z3}] [expr {$Im_z0 - $Im_z1 + $Im_z2 - $Im_z3}]\
            [expr {$Re_z0 - $Im_z1 - $Re_z2 + $Im_z3}] [expr {$Im_z0 + $Re_z1 - $Im_z2 - $Re_z3}]]
        }
 } elseif {[llength $dataL] > 8} then { setevenL
[list]  setoddL  [list]  foreach  {Re_z0  Im_z0
Re_z1 Im_z1} $dataL { lappendevenL $Re_z0
$Im_z0 lappendoddL $Re_z1 $Im_z1
        }
setsquarerootL  [list]  foreach  {Re_omega0  Im_omega0  Re_omega1
Im_omega1}     $rootL     {     lappendsquarerootL     $Re_omega0
$Im_omega0
        }
setlowL [list] sethighL
[list] foreach\
        {Re_y0 Im_y0}       [Fast_DFT $evenL $squarerootL]\
        {Re_y1 Im_y1}       [Fast_DFT $oddL $squarerootL]\
        {Re_omegaIm_omega} $rootL { set Re_y1t [expr {$Re_y1 *
$Re_omega - $Im_y1 * $Im_omega}] set Im_y1t [expr {$Im_y1 *
$Re_omega + $Re_y1 * $Im_omega}] lappendlowL  [expr {$Re_y0 +
$Re_y1t}] [expr {$Im_y0 + $Im_y1t}] lappendhighL [expr {$Re_y0 -
$Re_y1t}] [expr {$Im_y0 - $Im_y1t}]
        }
return [concat $lowL $highL]
```

```
   } elseif {[llength $dataL] == 4} then { foreach
{Re_z0 Im_z0 Re_z1 Im_z1} $dataL {break}
return [list\
 [expr {$Re_z0 + $Re_z1}] [expr {$Im_z0 + $Im_z1}]\
[expr {$Re_z0 - $Re_z1}] [expr {$Im_z0 - $Im_z1}]]
   } else { return
$dataL
   }
} proc ::math::fourier::Slow_DFT {dataLrootL}
{ set n [expr {[llength $dataL] / 2}]

 # The missing roots are computed by complex conjugating the given
# roots. If $n is even then -1 is also needed; it is inserted
explicitly. set k [llength $rootL] if {$n % 2 == 0} then {
lappendrootL -1.0 0.0
   }
for {incr k -2} {$k > 0} {incr k -2} { lappendrootL
[lindex $rootL $k]\
        [expr {-[lindex $rootL [expr {$k+1}]]}]
   }

   # This is strictly following the naive formula.
   # The product jk is kept as a separate counter variable.
set res [list] for {set k 0} {$k <
$n} {incr k} { setRe_sum 0.0
setIm_sum 0.0
setjk 0
```

KNUST

```
foreach {Re_zIm_z} $dataL { setRe_omega

[lindex $rootL [expr {2*$jk}]] setIm_omega

[lindex $rootL [expr {2*$jk+1}]] setRe_sum

[expr {$Re_sum +

            $Re_z * $Re_omega - $Im_z * $Im_omega}] setIm_sum

[expr {$Im_sum +

            $Im_z * $Re_omega + $Re_z * $Im_omega}]

incrjk $k if {$jk>= $n} then {set jk [expr {$jk - $n}]}

       }

lappend res $Re_sum $Im_sum

   }

return          $res          }          proc

::math::fourier::dft {in_data} {     #

First   convert   to   internal   format

setdataL [list] setn 0 foreach datum

$in_data { if {[llength $datum] ==

1} then { lappenddataL $datum 0.0

       } else {

lappenddataL [lindex $datum 0] [lindex $datum 1]

       }

incr n

   }

   # Then compute a list of n'th roots of unity (explanation below) setrootL

[DFT_make_roots $n -1]


   # Check if the input length is a power of two.
```

KNUST

```
setp 1 while {$p < $n} {set p [expr {$p
<< 1}]}
   # By construction, $p is a power of two. If $n==$p then $n is too.


   # Finally compute the transform using Fast_DFT or Slow_DFT,
   # and convert back to the input format. set
res [list]
foreach {Re Im} [ if
{$p == $n} then {
Fast_DFT $dataL $rootL
       } else {
Slow_DFT $dataL $rootL
       }
   ] {
lappend res [list $Re $Im]
   }
return $res } set data [list 2 3 -1 1]
set result [::math::fourier::dft $data]
puts "Result: $result"
```