DEVELOPMENT OF A MODEL FOR INTERNAL NETWORK INTRUSION DETECTION SYSTEM FOR MULTI DISCIPLINARY CAMPUS NETWORK: A CASE STUDY OF THE AHMADU BELLO UNIVERSITY ZARIA

BY

ADEDOKUN EMMANUEL ADEWALE, B.ENG (ABU) 1999, MSc (ABU) 2009
Ph.D/ENG/ 7242/2009-2010

**A DISSERTATION SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES,
AHMADU BELLO UNIVERSITY ZARIA
IN PARTIAL FUFILLMENT OF THE REQUIREMENTS FOR THE AWARD
OF A
DOCTOR OF PHILOSOPY (Ph.D) DEGREE IN ELECTRICAL ENGINEERING**

**DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
FACULTY OF ENGINEERING
AHMADU BELLO UNIVERSITY, ZARIA
NIGERIA**

MAY, 2014

DECLEARATION

I, ADEDOKUN Emmanuel Adewale hereby decleare that this dissertation titiled **"Development of a Model for Internal Network Intrusion Detection System for Multi Disciplinary Campus Network: A Case Study of the Ahmadu Bello University Zaria"** presented to the Department of Electrical and computer Engineering, Ahmadu Bello University Zaria is the result of my own research and it has not been presented in any form anywhere for the award of a degree.

All literature herein referenced and cited have been duly acknowledged. All shortcomings in this are entirely my responsibility.

ADEDOKUN Emmanuel Adewale	Signature	Date

CERTIFICATION

The project dissertation titled **Development of a Model for Internal Network Intrusion Detection System for Multi Disciplinary Campus Network: A Case Study of the Ahmadu Bello University Zaria** meet the regulations governing the award of the degree of Doctor of Philisophy (Ph.D) in Electrical Engieering of Ahmadu Bello University, and is approved for its contribution to literary presentation.

Dr. M.B Mu'azu. Chairman Supervisory Committee	_____ Signature	_____ Date
Prof. B.G Bajoga Member Supervisory Committee	_____ Signature	_____ Date
Dr.D.D Dajab Member Supervisiory Committee	_____ Signature	_____ Date
Dr. M.B.Mu'azu Head of Department	_____ Signature	_____ Date
Prof A.A. Joshua Dean School of Post-graduate Studies	_____ Signature	_____ Date



DEDICATION

My Wife, 'Ronke …. the most peaceful person I ever knew, am grateful to God for what we share!



ACKNOWLEDGEMENTS

To God the Father, Son and the sweet Holy spirit for grace and favour.

I am very grateful for the guidance and support of the supervisory committee chaired by Dr M.B Muazu with the support of Prof B.G Bajoga and Dr. D.D. Dajab, may God bless you abundantly. I am most grateful to Dr Bashir Muazu for beign a big brother and friend. I would have given up this work long time ago but for his persistence and encouragement, God bless you sir.

Sincere appreciation to the Vice chancellor, Prof. Abdullahi Mustapha for the opportunity and support for this study. I also appreciate the Mac Arthur PIMC for the dissertation completion grant. I appreciate the encouragement of Prof S. O Abah, Prof (Mrs) Adeyanju, Mal Nasir Bello and Late Prof Jenks Okwori.

Many thanks to Dr. Akinsanmi and Dr. S. G. Garba for their encouragement.I most sincerely thank Dr. S.M. Sani, for his invaluable contributions to make this work worth the while.

Big thanks to my colleagues: Akin, Yusuf, Chris, Keji, Ovie, John, Al-Hassan, Emma Oloye for their help thoughout the period of this study. My siblings from other mothers, Maryam, Philip and Tamon for your contributions and support.

My big brothers and their families: Barr. & Mrs Niran Adedokun, Dr & Dr (Mrs) Keshinro, Pastor & Mrs Adeshina, Dr & Dr(Mrs) Ogunremi for their encouragments and prayers.

I appreciate the support and prayers of my lovely parents, Mrs C.O. Adedokun, Prof and Dr. (Mrs) Onaolapo, my siblings and their spouses: Scott, Nike, Deola, Bola, Tosin, Seun, Bolu, and Yemi. Thank you all for your support.

And finally a big thanks to my wife 'Ronke and our precious kids Dara and Joba and Seyi, for their patience and support thought the period of this study.Thanks and God bless.

I acknowledge the contributions of authors, so referred in the field of this research work.

Once again this work is a clear testimony of God's Faithfullness.

Adedokun Adewale

Electrical Engineering Dept.ABU Zaria



ABSTRACT

The aim of the research is the modeling and implementation of an intrusion detection system (IDS) with focus on internally generated intrusions on a multi-Disciplinary campus network using Ahmadu Bello University (ABU) Zaria as a case study. A.B.U Zaria campus network is a complex network covering three campuses, 12 faculties and over a hundred Departments and as such there are varied types of users with diverse application requirements. Traffic data for a period of 24 months (January 2011 to December 2012) comprising of local to local and local to remote were used in developing the knowledge base of the IDS. The overhead of each packet derived from properties and composition of each packet is determined and used to form the behavior base of the IDS. The overhead determined is compared to a threshold called a safe score. This is the maximum allowable overhead for an Ethernet frame and has a value of 0.45. The developed model hybridizes the knowledge –based and behavior-based detection techniques and is implemented as an agent based application (using autonomous and adaptic agents or workers) in a program written in C-sharp(C#) and works in such a way that if any packet has an overhead greater than the safe score, it is logged for further analysis. In the validation stage, the model processed and logged 3,422,000 packets and 2328 packets respectively. Using Wireshark for the analysis of the logged packets, 298 packets (13% of the logged packets) were determined not to be malicious but of applications and or protocols not in the original knowledge-base and as such the knowledge base was updated. The IDS model developed is capable of updating its knowledge base, can sniff traffic directly on the network and utilizes less than 17% of ROM and CPU capacity at peak traffic period. It also showed about 50% improvement in the number of unknown applications and protocols identified by Netflow anlayzer upon implementation.

TABLE OF CONTENTS

Title Page i

Declearation ii

Certification iii

Dedication iv

Acknowledgement v

Abstract vi

Table of Contents vii

List of Figures xi

List of Tables xiii

List of Appendices xiv

List of Abbreviations xv

CHAPTER ONE

1.0 INTRODUCTION 1

1.1 Background Study. 1

1.1.1 The Ahmadu Bello University Network 2

1.1.2 The CISCO Three-Layer Hierarchical Model 4

1.1.3 The Core Layer 5

1.1.4 The Distribution Layer 6

1.1.5 The Access Layer 6

1.1.6 Agents and Their Roles in IDS Model 10

1.1.7 C# Programming Language. 11

1.2 Aim and Objectives 12

1.3 Statement of Problem 13

1.4 Methodology 13

1.5 Dissertation 14

CHAPTER TWO

2.0 LITERATURE REVIEW 15

2.1 Review of Fundamental Concepts 15

2.1.1 ISO/OSI Reference Model for Networks 12

2.1.2 Local Area Network 18

2.1.2.1 Local Area Network Technologies 19

2.1.2.2 Ethernet 19

2.1.2.3 Frame Formats 20

2.1.3 Security in ISO/OSI Reference Model 25

2.1.3.1 Security Services and Security Mechnaisms 26

2.1.4 Network Security Attacks 33

2.1.5 Concept of Intrusion Detection 35

2.1.5.1 Types of Intrusion Detection Systems 37

2.1.5.2 Knowledge-Based Intrusion Detection 38

2.1.5.3 Behaviour-Based Intrusion Detection. 38

2.1.5.4 Network Vulnerabilities to Internal Intrusion. 39

2.1.6 Acceptable Use Policy (AUP) 41

2.2 Review of Similar Works 43

CHAPTER THREE

3.0 DATA COLLECTION AND MODEL DEVELOPMENT . . . 50

3.1 Introduction 50

3.2 Network Setup for Data Collection 50

3.2 CISCO IOS Netflow Probes 52

3.3 Data Collection Methods 52

3.3.1 Netflow Analyser 52

3.3.2 Nfsen 54

3.3.3 Paessler Router Traffic Grapher (PRTG) 55

3.3.4 Ntop 56

3.4 Data Partitioning 58

3.4.1 Sample Data Collected 58

3.4.2 Identification and Extraction of Traffic Information 62

3.5 Intrusion Detection Model Development 63

3.6 Metrics for the Model 63

3.7 Initial Testing of the Model 72

3.7.1 Model Modifications 74

CHAPTER FOUR

4.0 MODEL VALIDATION 77

4.1 Model Validation on Live Network 77

4.2 Analysis of Logged Packets 77

4.2.1 Mechnaisms for Analysing Logged Packets 77

4.2.2 Packet Analyser 77

4.2.3 Wireshark 79

4.2.4 Analysis of Logged Packets 80

4.2.5 Results of Packet Analysis 81

4.2.6 Sample of Intrusive Packets 82

4.3 IDS Knowledge Base Update 83

4.4 Updating the Knowledge Base 84

4.5 Comparison of Initial Netflow Analyser Data before and after Knowledge Base Update 85

4.6 Observation of Model Performance during Validation Period . . 88

CHAPTER FIVE

5.0 CONCLUSION AND SUGGESTION FOR FURTHER WORK . . 90

5.1 Introduction 90

5.2 Significance of this Research 90

5.3 Limitations 91

5.4 Conclusion 91

5.5 Suggestion for Further Research 91

REFERENCES 93

Appendix I: List of Defined (Known) Applications and Protocols. . . 98

Appendix II: Some Traffic with Unknown Applications and Protocols . 113

Appendix III: Programme Listing 114

Appendix IV: Captured Packets on a Small LAN. 145

Appendix V: Log Files 169

LIST OF FIGURES

Figure 1.1: Ahamdu Bello Unviersity Network 3

Figure1. 2: CISCO Hierarchical Model 5

Figure1. 3: A3 Layer Network Showing Core, Distribution and Access Layers . 7

Figure1. 4: Logical Diagram of the Core and Distribution Layer of the ABU Network 8

Figure 2.1: OSI and TCP/IP Models and Protocol Distribution . . . 16

Figure 2.2: Ethernet and IEEE 802.3 Frame Formats. 20

Figure 2.3: Ethernet Frame Format Showing Minmum and Maximum Payload. 22

Figure 2.4: Profiles of Behaviour of Intruders and Authorized Users . . 36

Figure 3.1: Modified Layout with Monitoring Servers Installed . . . 51

Figure 3.2: Screenshot of Netflow Analyser 53

Figure 3.3: Screenshot of Nfsen 54

Figure 3.4: Screenshot of PRTG 55

Figure 3.5: Screenshot of Ntop 57

Figure 3.6: Bar Graph showing Traffic Summary from Netflow Analyser . . 60

Figure 3.7: Flowchart for the Intrusion Detection Hybrid Model . . . 66

Figure 3.8: Representation of Local LAN used for Initial Testing of Developed Model. 71

Figure 3.9: Ping Output During Intial Testing of Model . . . 73

Figure 3.10: Ping output During Testing Period after Modification of Model. . 75

Figure 3.11: Screenshot of the IDS Model Data Capturing Interface. . . 76

Figure 4.1: Screenshot of Wireshark Capture Interface 80

Figure 4.2: Screeshot of Packet showing an LLMNR Protocol . . . 81

Figure 4.3: Screenshot of Packet showing IMAP Protocol 82

Figure 4.4: Screenshot a Packet showing IMAP Protocol with Spamming Tendency 83

Figure 4.5: Improved IDS Interface for Update of Knowledge Base . . . 85

Figure 4.6: Screenshot of Netflow Analyser during Model Validation Period . 86

Figure 4.7: Bar Graph showing Traffic Summary from Netflow Analyser during Model Validation Period 87

Figure 4.8: Screenshot of System Resource Usage during Validation . . 89

LIST OF TABLES

Table 1.1: List of Major Network Nodes and Service Areas 9

Table 2.1: Total Minimum Frame Physical Size 23

Table 2.2: Total Maximum Frame Physical Size 24

Table 2.3: Relationship of Security Services and Layers 1 – 7. . . . 28

Table 2.4: Relationship between Security Services and Mechanisms. . . 32

Table 3.1: Traffic Summary from Ntop. 59

Table 3.2: Traffic Summary from Netflow Analyser 59

Table 3.3: Traffic Summary from PRTG 61

Table 3.4: Traffic Summary from Nfsen 62

Table 3.5: Comparism of Object Oriented Programming Language Feautres. . . 63

Table 4.1: Comparism of technical and general Properties of Packet Analysers 78

Table 4.2: List of Additional Protocols/Application Signatures added to the Knowledge Base 84

Table 4.3: Traffic Summary from Netflow Analyser druing Model Validation Period 87



LIST OF APPENDIXES

Appendix I: List of Defined (Known) Applications and Protocols. . . 98

Appendix II: Some Traffic with Unknown Applications and Protocols . 113

Appendix III: Programme Listing 114

Appendix IV: Captured Packets on a Small LAN. 145

Appendix V: Log Files 169

LIST OF ABBREVIATIONS

AES	Advanced Encryption Standard
ASA	Adaptive Security Appliance
ASCII	American Standard Code for Information Interchange
AUP	Acceptable Usage Policy
BPNN	Back Propagation Neural Network
BPNN	Back Propagation Neural Network
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DNS	Domain Name System
DMZ	Demilitarized Zone
DSAP	Destination Service Access Point
ElGamal	Taher El-Gamal (Egyptian Crytographer)
FCS	Frame Check Sequence
FDDI	Fibre Distributed Data Interface
FTP	File Transfer Protocol
HTML	Hyper Text Mark-up Language
HTTP	Hyper Text Transfer Protocol
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronic Engineers
IFG	Inter-Frame Gap
IP	Internet Protocol
IPG	Inter-packet Gap
IPX	Internetwork packet Exchange
ISO	International Organisation for Standardization

JPEG	Joint Photographic Expect Group
LAN	Local Area Network
MAC	Media Access Control
MIDI	Musical Instrument Digital Interface
OSAS	Open Source application Analyzer for Snort
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
PDU	Protocol Data Unit
PRTG	Paessler Router Traffic Grapher
QoS	Quality of Service
RC5	Rivest Cipher 5
RPC	Remote Procedure Call
RRD	Round Robin Database
RSA	Rivest, Shamir Adjeman
RTP	Real-time Transfer Protocol
SCTP	Stream Control Transmission Protocol
SIFT	Snort Intrusion Filter for TCP
SIP	Session Initiation Protocol
SMTP	Simple Mail Transfer Protocol ASCII
SNMP	Simple Network Management Protocol
SSAP	Source Service Access Point
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TIFF	Tagged Image File Format
TTL	Time-to-Live

UDP	User Datagram Protocol

CHAPTER ONE

INTRODUCTION

1.1 Background Study

With the increasing growth in the use of computers for teaching, learning, research and even administrative activities on campuses, network function is increasingly powerful. As campus networks keep growing in size and functionality, network security issues have also become major challenges on campus networks for a long time (Xiong C.H and Peng Z 2012). Like any other network, campus networks have suffered various attacks ranging from unauthorised access from outside the network to attacks by users within the network and also attack by users within the network to remote sites.

A lot of work has been focused on attacks from remotes sites into the network or malware attacks in form of viruses, spy ware etc (Ismail et.al (2009), Yi et al (2010)). Not much work has been done to address internal attacks carried out by legitimate users within the same network. Such attacks prove more difficult to detect and arrest without resulting in disruption of some network services that may be required by legitimate users especially on a multi-disciplinary campus network like that of the Ahmadu Bello University Zaria. This is because it may be very difficult to categorise traffic as illegal due to the diverse nature of research, learning, instructional and administrative activities going on within the campus network.

An intrusion detection system (IDS) inspects all inbound and outbound network activity and identifies suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromise a system. IDS are mostly designed to identify what might have gotten past the firewall. Network security issues have been a major challenge on campus networks for a long time. There have been various attacks ranging from unauthorised

access from outside the network to attacks by users within the network and also attack by users within the network to remote sites.

1.1.1 The Ahmadu Bello University Network

The Ahmadu Bello network is designed around an optical fibre based backbone comprising of three rings on the 3 campuses: Samaru, Kongo and Shika as shown in Figure 1.1 Each ring is a 4Gbps network transmission while intercampus links are 10Gbps transmission.

Currently the University runs one of the largest campus networks by any tertiary institution within the country serving twelve academic Faculties with 85 academic departments running different academic programmes leading to Bachelors, Masters and Doctoral degrees, a School of Postgraduate Studies, five Institutes, six Specialized Centers, a Division of Agricultural Colleges, a Teaching Hospital, Demonstration Secondary School, Primary School and a Consultancy Outfit.

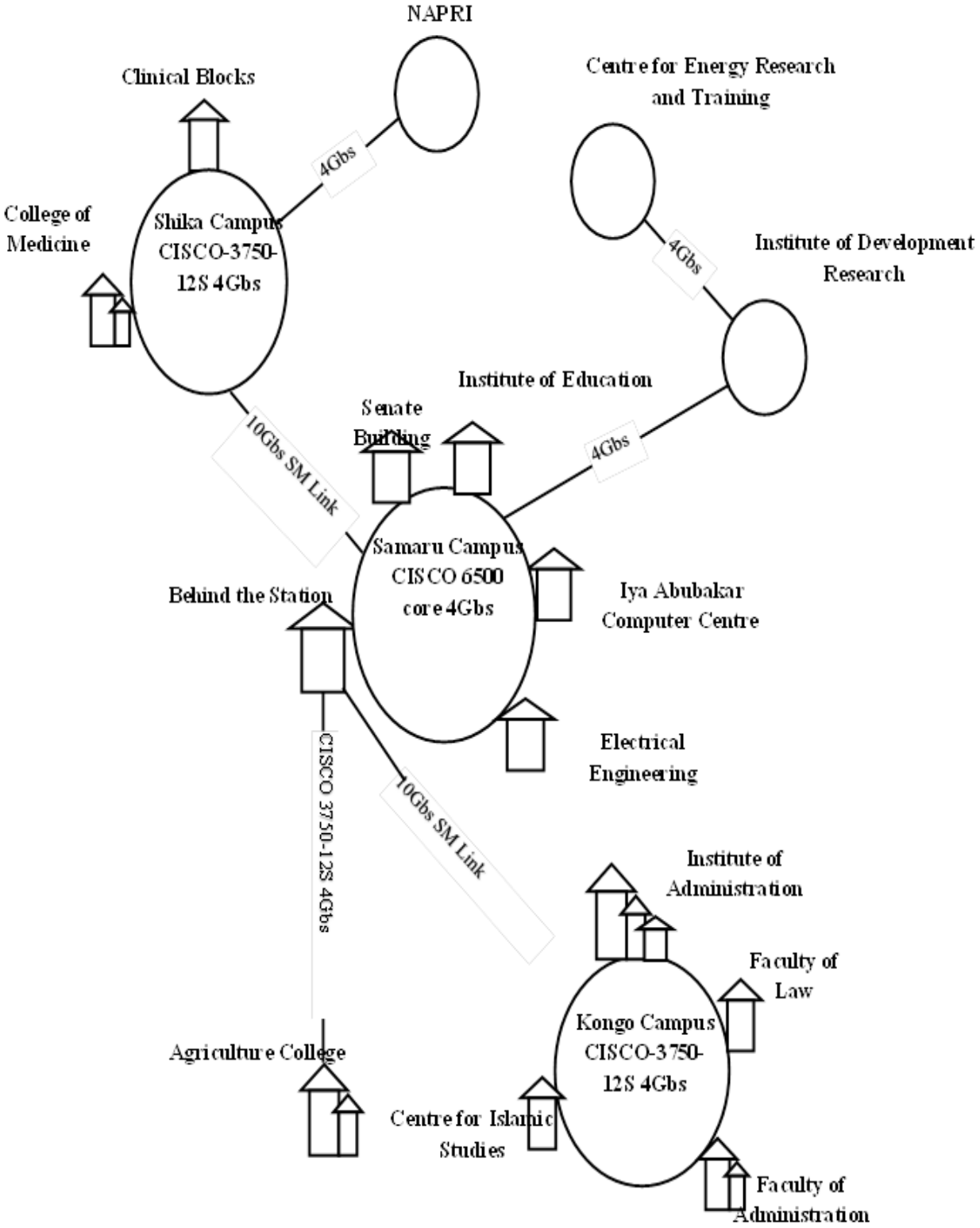


Figure 1.1: **Ahmadu Bello University Network**

The network runs almost entirely on Cisco active devices and hence the Cisco three layer hierarchy; Core, Distribution and Access model is employed.

1.1.2 **The Cisco Three-Layer Hierarchical Model**

In network design, hierarchical Model has several benefits just as it does in other areas of life. When a network is properly designed and implementated, a hierarchical model network is more predictable. It helps to define which areas should perform certain functions.

Large networks can be extremely complicated, with multiple protocols, detailed configurations, and diverse technologies. Hierarchy helps to summarize a complex collection of details into an understandable model. Then, as specific configurations are needed, the model dictates the appropriate manner to apply them. With the Cisco hierarchical model it is easier to design, implement, and maintain a scalable, reliable, cost-effective hierarchical internetwork. (Lamme 2005)

The three layers of hierarchy as defined by Cisco is as shown in Figure1.2, each with specific functions:

i. The core layer: Backbone
ii. The distribution layer: Routing
iii. The access layer: Switching

Even though each layer has specific responsibilities, it is important to note that the definition of the layers are basically logical and not nessesarily physical , this implies that in the physical implementations of hierarchical networks,there may be several devices in a single layer, or there might be a single device performing functions at two layers. (Lamme 2005)

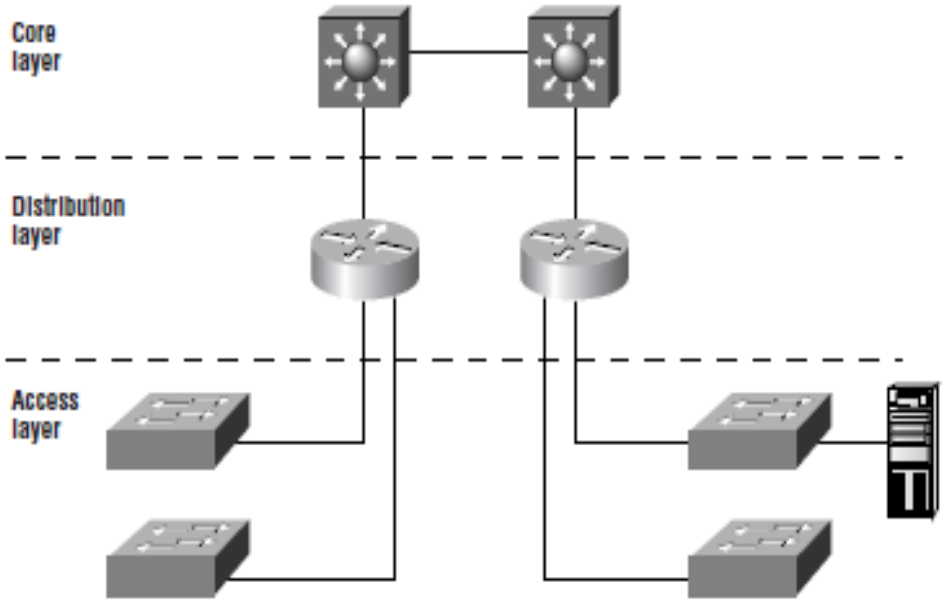


Figure1. 2: **Cisco Hierarchical Model**

1.1.3 The Core Layer

The core layer is the backbone of the network because all other layers rely on it. Sitiing at the top of the hierarchy, the core layer is responsible for transporting large amounts of traffic both reliably and quickly. The only purpose of the network's core layer is to switch traffic as fast as possible with the least possible manipulations. The traffic transported across the core is common to a majority of users. However, user data is processed at the distribution layer, which forwards the requests to the core if needed.

If there is a failure in the core, every single user can be affected. Therefore, fault tolerance at this layer is an important issue. Because the core layer is likely to deal with very large volume of traffic, high speed and very low latency are driving concerns at this layer (Lamme 2005).

1.1.4 **The Distribution Layer**

The distribution layer is also referred to as the workgroup layer. It is the communication point between the access layer and the core. The distribution layer primarily provides routing, filtering, and WAN access and determines how packets can access the core, if needed. The distribution layer is also expected to determine the fastest way to handle network service requests, for example, how a file request is forwarded to a server. After the distribution layer determines the best path, it forwards the request to the core layer if needed. The core layer then quickly transports the request to the correct service (Lamme, 2005).

1.1.5 **The Access Layer**

The access layer which can also be referred to as the desktop layer, contains devices that allow workgroup and users to use the services provided by the distribution and core layers.This layer controls user and workgroup access to internetwork resources.The network resources most users need will be available locally. The distribution layer handles any traffic for remote services.

It is important to emphasis that though there are three separate levels, three separate levels do not imply three separate devices. There could be fewer, or there could be more (Lamme,2005).

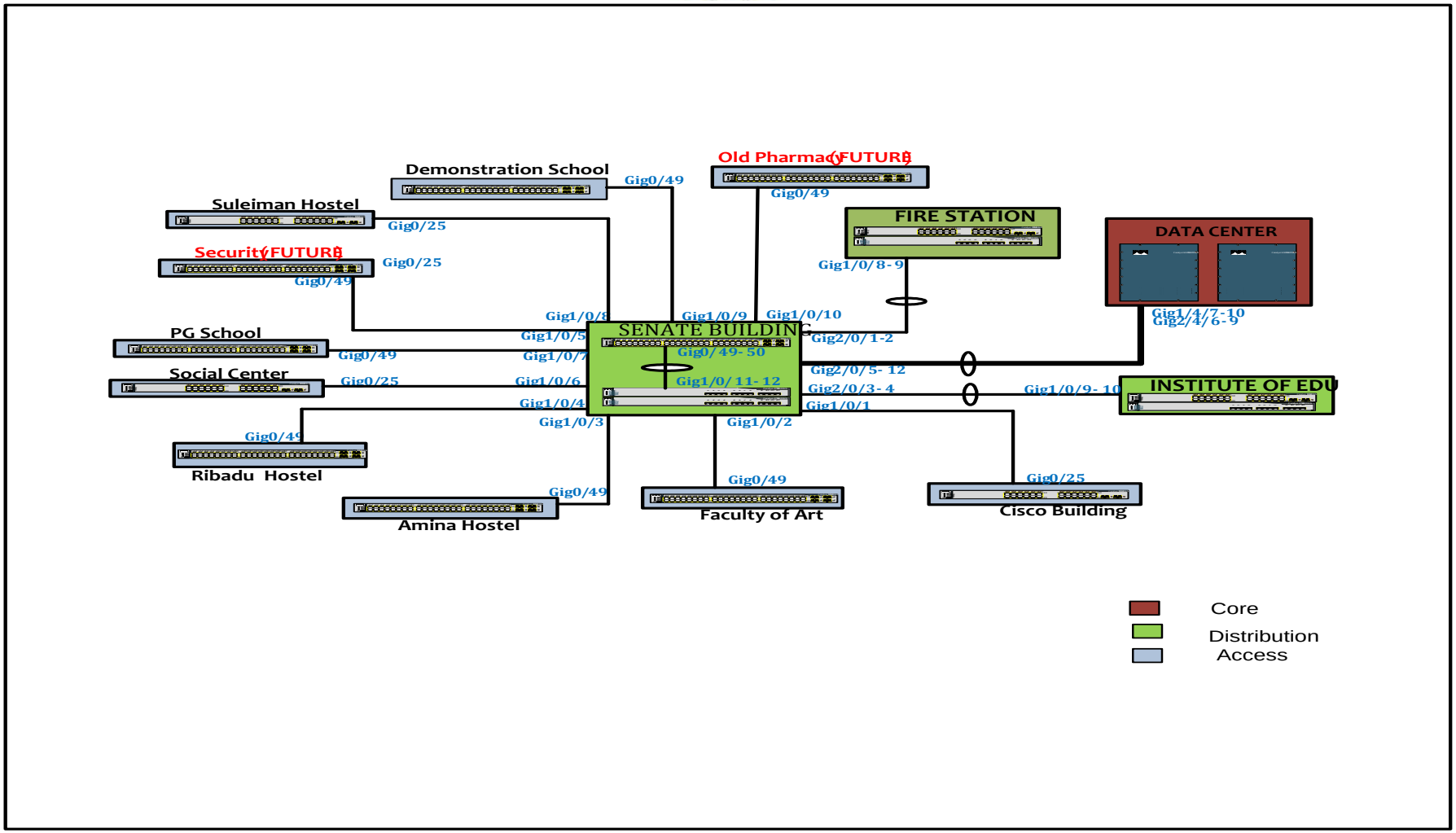


Figure 1.3: **A 3 Layer Network showing Core, Distribution and Access Layers**

[*Source: Ahmadu Bello University Fibre Network as-Built Documentation 2010]*

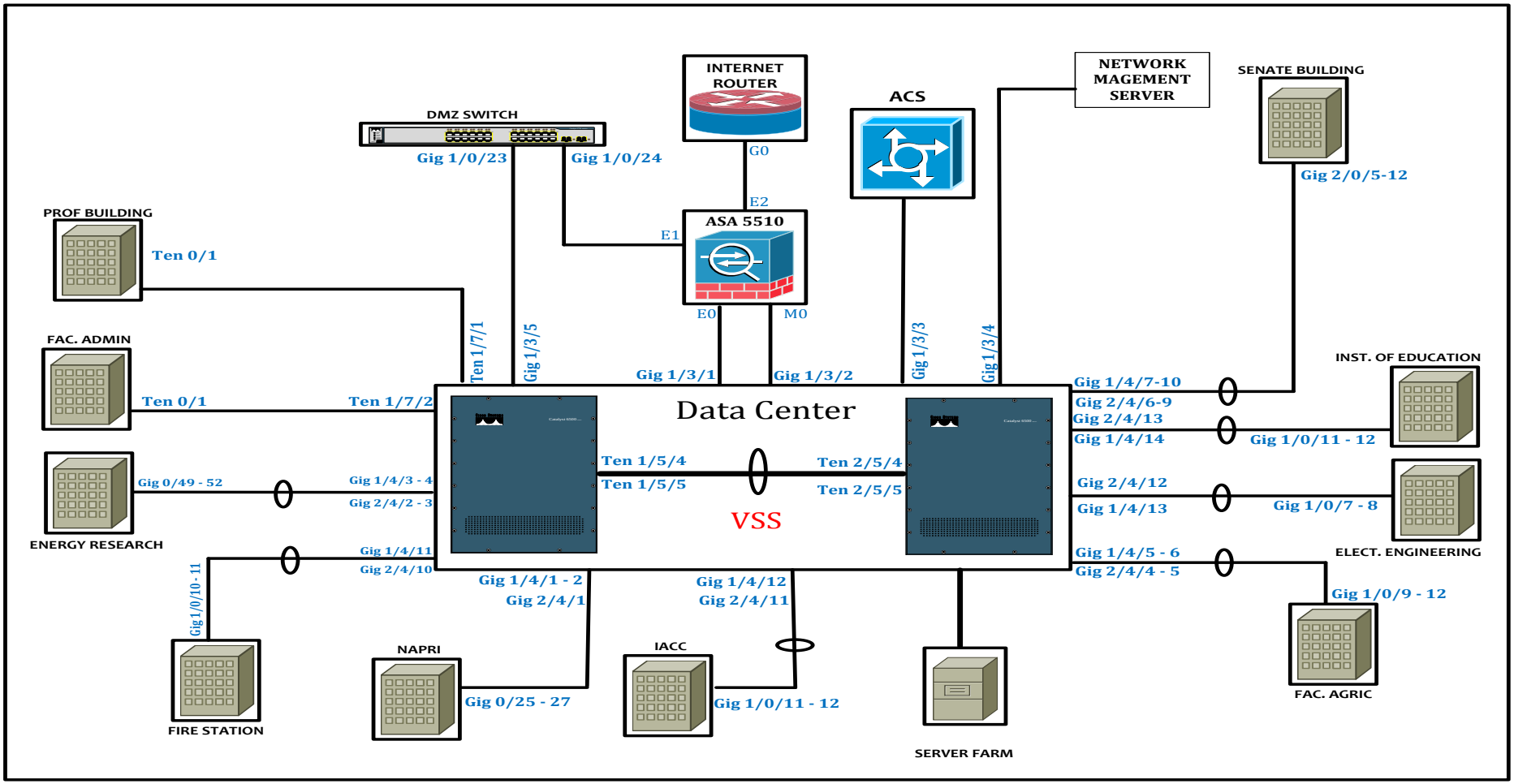


Figure 1.4: **Logical Diagram of the Core and Distribution Layer of the ABU Network**

[*Source: Ahmadu Bello University Fibre Network as-Built Documentation 2010]*

All traffic from outside the network pass through the Adaptive Security Appliance (ASA) that provides both firewall services and intrusion detection system to prevent any unwanted traffic entering the network. A second security layer is the Demilitarized Zone (DMZ) which further filters inbound traffic before sending any traffic to the core router. The ASA and DMZ can be seen on Figures1.3 and 1. 4

Table 1.1 presents the list of network nodes, their locations and service areas on the Ahmadu Bello University network as defined on the core; it further explains the complexity in terms of network size and diverse activities as it relates to different field of research, learning and instructions carried out on the University network.

Table 1.1: **List of Major Network Nodes and Service Areas**

S/N	Node ID	Location	Service Areas
1	Senate Building	Senate	Senate Building, Cisco Academy, Amina Hostel, Ribadu Hall etc
2	Fire Station	Fire Station	Faculties of Pharmacy, Vet Medicine, Medicine, Danfodio Hostel, BBC Studio
3	Iya Abubakar Computer Centre	IACC	Faculty of Social Sciences, IACC,KIL,
4	Electrical Engineering Dept	Elect Dept	Bio Tech Centre, Faculty of Engineering, New Engineering Complex, Environmental Design
5	Institute of Education	IOE	Institute of Education, ICT Park, Chemical Engineering, IDR, Faculty of Education
6	Faculty of Sciences	Bio Sciences	Faculty of Science
7	Estate	Estate	Faculty of Agric, Agric Engineering, IAR, Agric Ext, Estate, NAERLS,DAC
8	ABUTH	Prof Building	Male and Female hostel, Prof Building and Doctors Common Room
9	Kongo	Fac. of Admin	Department of Public Admin, business admin, Local Govt, Staff School, Sick Bay, Law, Centre For Islamic Legal Studies, Libraries, Hostels
10	Energy Research	CERT	Centre for Energy research and training
11	NAPRI	NAPRI	National Animal production research institute

1.1.6 Agents and their Role in the IDS Model

The intrusion detection model developed is an agent based model, this section will briefly describe the types of agents, their properties and role in the model.

1.1.6.1 Agents

At a minimum, an IT agent is generally regarded to be an autonomous entity that can interact with its environment. In other words, it must be able to perceive its environment through sensors and act upon its environment with effectors.

An agent can be a person, a machine, a piece of software, or a variety of other things. Some of the properties that agents may possess in various combinations include (but not limited to):

i. Autonomous: capable of acting without direct external intervention. It has some degree of control over its internal state and actions.
ii. Interactive: communicates with the environment and other agents.
iii. Adaptive: capable of responding to other agents and/or its environment to some degree.
iv. Mobile: able to transport itself from one environment to another.
v. Proxy: acting in the interest of, as a representative of, or for the benefit of some entity.

1.1.6.2 Software Agents

Agent developers have identified several forms of agents that are important for IT system development. These agents must, however, execute as software, hardware, robotics or a combination of these. In this work, the focus is on software agents.

Software agents are a more specific kind of agent. At a minimum, a software agent is defined as an autonomous software entity that can interact with its environment. In other words, they are agents that are implemented using software. They are normally autonomous

and can react with other entities, including humans, machines, and other software agents in various environments and across various platforms. Basically, software agents are design patterns for software. Tools, languages, and environments can be specifically developed to support the agent-based pattern. However, the agent design pattern can also be implemented using object-oriented tools, languages, and environments—or any other tool, language, and environment that is capable of supporting software entities that are autonomous, interactive, and adaptive. Agent-based tools are preferable primarily because the agent design patterns are inherent in the software—rather than explicitly programmed. In other words, object technology can be used to enable agent-based technology.

This means that object oriented languages are quite suited for developing agent-based applications and in this work, C# was the language of choice.

1.1.7 C# Programming Language

C# is a modern, general-purpose object-oriented and structured programming language developed by Microsoft as part of the .Net framework and approved by ISO. It produces efficient programs which can be compiled on a variety of computer platforms and architectures. Although C# constructs closely follow traditional high level languages C and C++, it however has the following specific feautures (Sempf et al., 2010):

i. **Flexible:** C# programs can execute on the current machine, or can be transmitted over the Web and executed on a remote computer.

ii. **Powerful:** C# has basicly the same commands as C++, however with the rough edges filed smooth.

iii. **Easier to use:** C# error-proofs the commands responsible for most C++ errors, so you spend far less time chasing down those errors.

iv. **Visually oriented:** The .NET code library that C# uses for many of its capabilities provides the help needed to readily create complicated display frames with drop-down

lists, tabbed windows, grouped buttons, scroll bars, and background images, to name just a few.

v. **Internet-friendly:** C# plays a pivotal role in the .NET Framework, Microsoft's current approach to programming for Windows, the Internet, and beyond. .NET is pronounced *dot net*.

vi. **Secure:** Any language intended for use on the Internet must include serious security to protect against malevolent hackers.

1.2 Aim and Objectives

The aim of this research is to develop an agent- based intrusion detection model that can detect internally generated intrusion on a multidisciplinary campus network using the Ahmadu Bello University Zaria as a case study. It is also expected that the resulting model can be implemented with little or no disruption of network services to authorized legitimate users.

The objectives of this research, therefore, are as follows:

i) Deployment of traffic sniffers like Ntop, Cisco Netflow, PRTG and Nfsen to obtain and classify live traffic data according to IP and application details.Two of these sniffers work on free and open source operating system (Ntop and Nfsen) while the other two (Netflow Analyser and PRTG) are proprietary and work on Windows environment. This is to give comparable results across different operating system environments. The data collected will serve as a knowledge base for the model;

ii) Development of an agent-based hybrid intrusion detection system that will have universal application to most campus networks based on defined criteria and constraints. This is developed using C# (C-sharp) and its platform independent;

iii) Validation of the developed model in real time on a live network.

1.3 Statement of Problem

Campus networks have proved to be one of the most challenging networks to manage due to diversity of network services required by the various members of campus community based on their diverse needs and activities. Most times when these unwanted network traffics are noticed on a campus network, it is always a challenge finding simple and quick means to arrest them in such a manner that would not disrupt authorized legitimate network activities and services.

This research is thus aimed at developing an intrusion detection model that will detect unwanted network traffic generated within a network that can lead to degradation of quality of service(QoS) or complete denial of sevice on a typical multi-disciplinary tertiary institution network like that of the Ahmadu Bello University (ABU) Network in real time.

1.10

1.4 Methodology

The methodology adopted in this research is as follows:

i) Obtaining campus data traffic with IP and port details: To obtain this data, Netflow is enabled on the core router and Cisco Netflow installed to collect the data. Other network monitoring software like NTOP, NFSEN and PRTG were also installed on the network.

ii) Partitioning the data into:

   a) Local to remote traffic,

   b) Remote to Local Traffic and

   c) Remote to Remote traffic

   This stage is accomplished using the source and destination IP of traffic. The NTOP traffic partitioning module is a good tool at this stage of the work, as it partitions traffics into local and remote traffic using the properties and directions of each traffic.

iii) Identifying unusual traffic pattern and extracting peculiarities of such traffic especially the headers and frames including their behavioural patterns

iv) Modelling of Intrusion detection system that utilizes:

   a. the list of applications and protocols recorded from the data collected as its knowledge base;and

   b. the behavioural pattern of network traffic (overhead) to set the safe score metrics for the model.

iv. Trial test of model on a small campus network.

v. Evaluation of model performance to ascertain that model does not create bottleneck on the network during trial period.

vi. Validation of the model on a multi-disciplinary campus network using the ABU Zaria as a case study.

1.5 DISSERTATION OUTLINE.

The Dissertation is divided into five chapters. Chapter One is the Backgound Introduction of the research. It explains the objective and methodology used for the research. Chapter Two describes the review of Fundermental concepts and review of similar works done. The Setup for data collection, collected data and model development including initial test and modification of the developed model was discussed in Chapter Three. Validation of the developmed model on a live network and analysis of logged packets is presented in Chapter Four. The Conclusion is in Chapter Five.

CHAPTER TWO

LITERATURE REVIEW

2.1 Review of Fundamental Concepts

The following section presents a review of fundamental concepts which are key to this study.

2.1.1 ISO/OSI REFERENCE MODEL FOR NETWORKS

In order to have a good understanding of the way that networking is performed, network reference models have been developed that group similar functions into abstractions known as layers. Each layer's functions can communicate with the same layer's functions of another network host. On the same host, the functions of a particular layer have interfaces to communicate with the layers below and above it. This abstraction simplifies and properly defines the necessary actions for networking.

The ISO Open Systems Interconnection (OSI) reference model, defines seven network layers as well as their interfaces. Each layer depends on the services provided by its intermediate lower layer all the way down to the physical network interface card and the wiring. Then, it provides its services to its immediate upper layer, all the way up to the running application. It needs to be noted that not all protocol stacks include all seven layers. (ISO 1984)

Fig 2.1 presents a comparison of the seven -layered OSI reference model to the four-layered TCP/IP (Internet) model which is an earlier model as it relates to the different applications and protocols in each layer. The two models are similar in concept, however, the OSI model further simplifies the TCP/IP model to achieve easier troubleshooting and better interoperability.

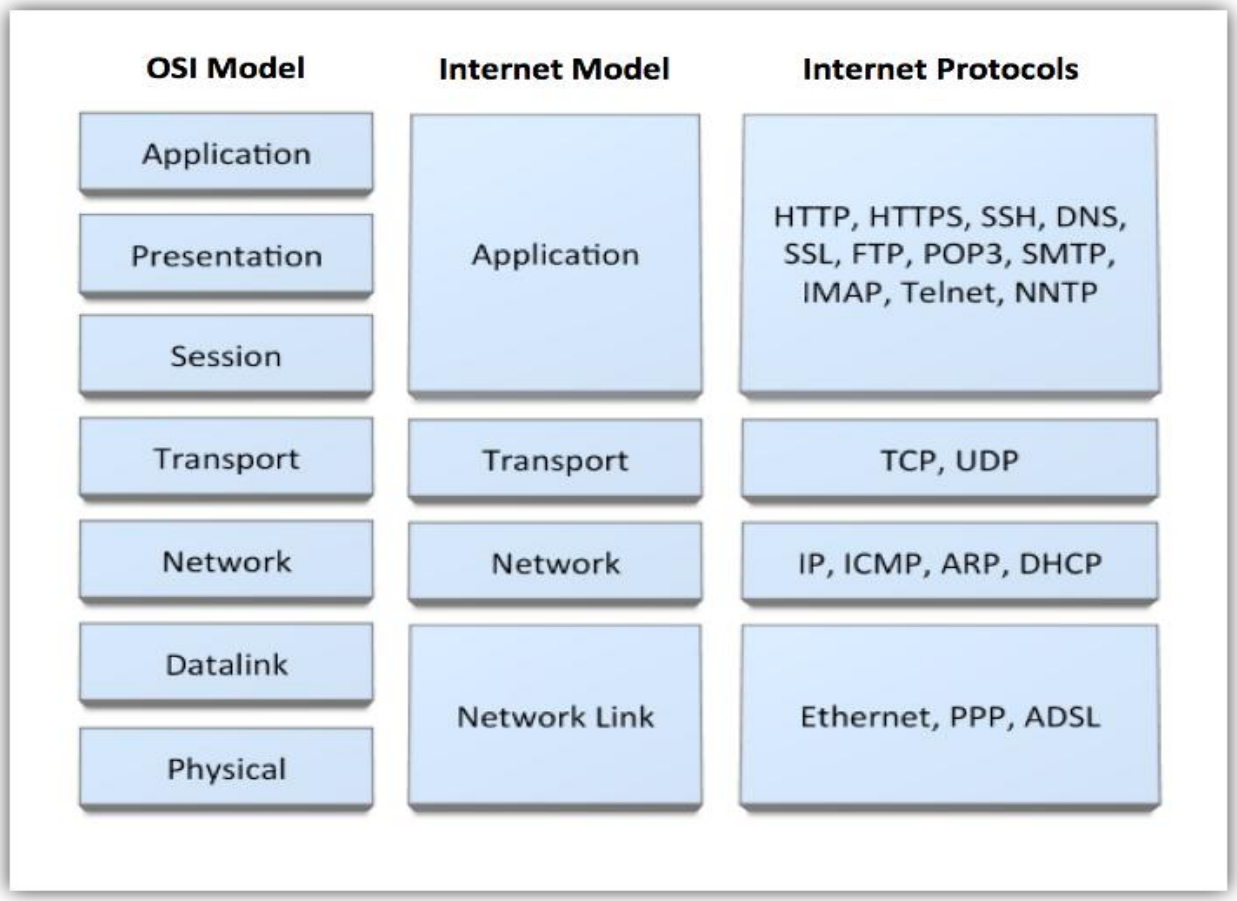


Figure 2.1: **OSI and TCP/IP Models and Protocol Distribution**

(Source: http://bit.kuas.edu.tw/~csshieh/teach/np/tcpip/)

The OSI reference model, from the highest to the lowest, is described as follows (ISO/OSI 1994):

i) **Layer 7 - Application Layer:** The application layer is the seventh and topmost layer in the OSI Architecture and this layer deals with the communication issues of an application. It identifies and establishes the availability of the communicating principals and is also responsible for interfacing with the user. Protocols of this layer directly serve the end user by providing the distributed information service appropriate to an application. Examples of application layer protocols include the Session Initiation Protocol (SIP), the HyperText Transfer Protocol (HTTP), the File Transfer Protocol (FTP), the Simple Mail Transfer Protocol (SMTP), Telnet etc.

Applications or application processes may be of any kind (manual, computerized, industrial or physical)

ii) **Layer 6 - Presentation Layer:** The Presentation layer provides the set of services which may be selected by the Application Layer to enable it interpret the meaning of the data exchanged. Essentially, it translates the data and it performs tasks like data compression and decompression and data encryption and decryption. Applications in an OSI environment can communicate without unacceptable costs in interference variability, application modification or transformations through the use of the services provided by the presentation layer. Some of the well-known standards and protocols of this layer include ASCII, ZIP, JPEG, TIFF, RTP, and the MIDI format.

iii) **Layer 5 - Session Layer:** This layer is responsible for initiating the contact between two computers and setting up the communication lines.The Session layer assist in the support of the interaction between presentation entities. This is achieved by the provision of session administration services and session dialogue service by the Session Layer. It formats the data for transfer and it maintains the end-to-end connection. Two examples of session layer protocols are the remote procedure call (RPC) and the secure sockets layer (SSL) protocols.

iv) **Layer 4 - Transport Layer:** This layer defines how to address the physical locations of the network, establishes connections between hosts, and handles network messaging. It also maintains the end-to-end integrity of the session and provides mechanisms to support session establishment for the upper layers. The Transport Layer provides a universal transport service in association with underlying services provided by lower layers.The TCP and the User Datagram Protocol (UDP) are the most widely known protocols of this layer, with the Stream Control Transmission Protocol (SCTP) gaining in usage.

v) **Layer 3 - Network Layer:** The Network layer provides functional and procedural means to exchange network service data units between two transport entities over a network connection.The Network layer facilitates routing and relaying of data between the network hosts, it sends fragments of data called packets from a source to a destination hosts as it also includes the management of error detection, message routing, and traffic control. The Internet Protocol (IP) belongs to this layer.

vi) **Layer 2 - Data Link Layer:** The Data Link layer provides functional and procedural means to establish, maintain and realse data links between network entities.The layer defines the conditions that must be followed by a host in order to access the network. It establishes the link between the hosts over a physical channel. It ensures message delivery to the proper device and translates the transmitted bits for the lowest physical layer. Ethernet and Token Ring are typical examples of protocols that operate at this layer.

vii) **Layer 1 - Physical Layer:** The Physical layer provides mechanical, electrical functional and procedural characteristics to establish maintain and release physical connections. It defines the physical connection between a host and a network. It mainly converts the bits into physical signalling suitable for transmission, such as voltages or light impulse. The device drivers that handle the communications hardware (network cards, wireless cards etc) operate at this layer.

2.1.2 Local Area Network

Local Area Network (LAN) is a connection of group of computers and associated devices within a buildings or group of buildings sharing a common communications line or wireless link. These connected devices share the common resources within a small geographic area. Usually, the server has applications and data storage that are shared in

common by multiple computer users. LANs can be characterized by their topology, protocols and media.

Topology is the geometric arrangement of devices on the network. For example, devices can be arranged in a ring or in a straight line. Protocols are the rules and encoding specifications for sending data. They also determine whether the network uses a peer-to-peer or client/server architecture. Media is what is used to connect the devices, i.e., twisted-pair copper wire, coaxial cables, fiber optic cables or radio waves.

LANs are capable of transmitting data at very fast rates, much faster than data can be transmitted over a telephone line because of their short distances and the fact that they do not have to rely on copper wire media (with its limited capacity). A local area network may serve as few as two or three users or as many as thousands of users.

2.1.2.1 **Local Area Network Technologies**

Major local area network technologies are:

i. Ethernet;

ii. Token Ring; and

iii. Fibre Distributed Data Interface (FDDI).

Ethernet is by far the most commonly used LAN technology. A number of corporations use the Token Ring technology. FDDI is sometimes used as a backbone LAN interconnecting Ethernet or Token Ring LANs. Since this research is focused on a campus network, more emphasis will be placed on the Ethernet technology since vitually all campus network utilises the ethernet technology especially at the access layer.

2.1.2.2 **Ethernet**

In 1985, the Institute of Electrical and Electronic Engineers (IEEE) in the United States of America produced a series of standards for Local Area Networks (LANs) called the IEEE 802 standards. These have found widespread acceptability and now form the core of

most LANs. One of the IEEE 802 standards, IEEE 802.3, is a standard known as "Ethernet". This is the most widely used LAN technology in the world today.

2.1.2.3 **Frame Formats**

Ethernet and IEEE 802.3 frame formats are shown in Figure 2.2.

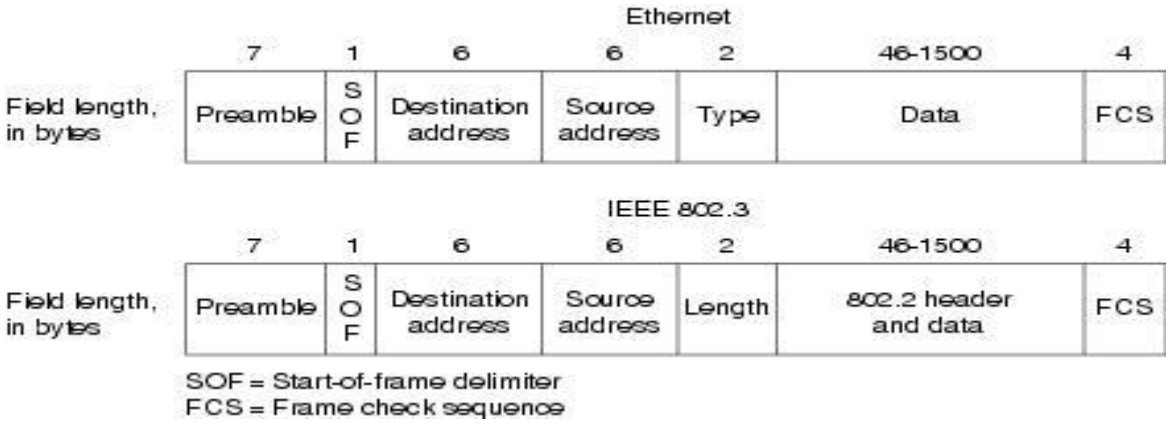


Figure 2.2**: Ethernet and IEEE 802.3 Frame Formats**

(Source: Inter network Technology Overview Second Edition June 1999)

Both Ethernet and IEEE 802.3 frames begin with an alternating pattern of ones and zeros called a *preamble*. The preamble tells receiving stations that a frame is coming. The byte before the destination address in both an Ethernet and an IEEE 802.3 frame is a start-of-frame (SOF) delimiter. This byte ends with 2 consecutive 1 bits, which serve to synchronize the frame reception portions of all stations on the LAN. (IEEE 802.3/ITO 1999).

Immediately following the preamble in both Ethernet and IEEE 802.3 LANs are the destination and source address fields. Both Ethernet and IEEE 802.3 addresses are 6 bytes long. Addresses are contained in hardware on the Ethernet and IEEE 802.3 interface cards. The first 3 bytes of the addresses are specified by the IEEE on a vendor-dependent basis, and the last 3 bytes are specified by the Ethernet or IEEE 802.3 vendor. The source address is

always a unicast (single node) address, whereas the destination address may be unicast, multicast (group), or broadcast (all nodes).

In Ethernet frames, the 2-byte field following the source address is a type field. This field specifies the upper-layer protocol to receive the data after Ethernet processing is complete. In IEEE 802.3 frames, the 2-byte field following the source address is a length field, which indicates the number of bytes of data that follow this field and precede the frame check sequence (FCS) field.

Following the type/length field is the actual data contained in the frame. After physical layer and link layer processing is complete, this data will eventually be sent to an upper-layer protocol. In the case of Ethernet, the upper-layer protocol is identified in the type field. In the case of IEEE 802.3, the upper-layer protocol must be defined within the data portion of the frame, if at all. If data in the frame is insufficient to fill the frame to its minimum 64-byte size, padding bytes are inserted to ensure at least a 64-byte frame.

In 802.3 the data field carries a payload header in addition to the payload itself. This header serves the logical link control sublayer of the OSI model and is completely independent of the MAC sublayer and physical layer below it. This header, functionally known as 802.2 encapsulation contains destination service access point (DSAP) and source service access point (SSAP) information. This will notify higher protocols what type of payload is actually riding in the frame. It functions like the "type" field in traditional Ethernet and is used by upper-layer network protocols such as IPX. Network software developed to support the TCP/IP networking suite uses the type field to determine protocol type in an Ethernet frame. The type field and the LLC header are not replacements for each other, but they serve to offer backward compatibility between network protocol implementations without rewriting the entire Ethernet frame.

After the data field is a 4-byte frame check sequence (FCS) field containing a cyclic redundancy check (CRC) value as in Figure 2.3. The CRC is created by the sending device and is recalculated by the receiving device to check for damage that might have occurred to the frame in transit.

The ISO Ethernet standard dictates a minimum size of frame, which requires at least 46 bytes of data (payload) to be present in every MAC frame. If the network layer wishes to send less than 46 bytes of data the MAC protocol adds sufficient number of zero bytes (0x00, is also known as null padding characters) to satisfy this requirement. The maximum size of data (payload) which may be carried in a MAC frame using Ethernet is 1500 bytes (this is known as the MTU in IP).

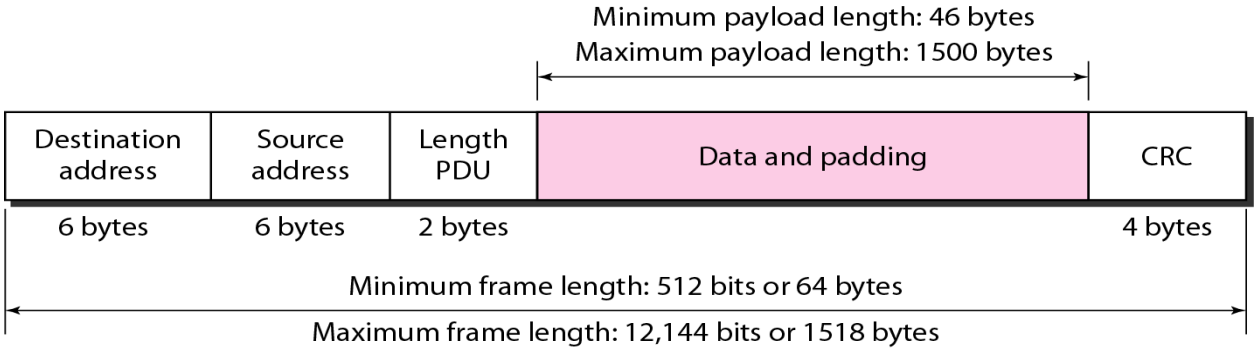


Fig 2.3: Ethernet Frame Format Showing Minmum and Maximum Payload.

The minimum frame payload is 46 Bytes (dictated by the slot time of the Ethernet LAN architecture). The maximum frame rate is achieved by a single transmitting node. This implies a frame consisting of 72 Bytes with a 9.6 μs inter-frame gap (IFG). The total utilised period (measured in bits) corresponds to 84 Bytes (Fairhurst, 2002) as shown in Table 2.1

Table 2.1: **Total Minimum Frame Physical Size** *(Source:* IEEE 802.3)

Frame Part	**Minimum Size Frame**
Inter Frame Gap (9.6μs)	12 Bytes
MAC Preamble (+ SFD)	8 Bytes
MAC Destination Address	6 Bytes
MAC Source Address	6 Bytes
MAC Type (or Length)	2 Bytes
Payload (Network PDU)	46 Bytes
Check Sequence (CRC)	4 Bytes
Total Frame Physical Size	*84 Bytes*

Calculation of number of bit periods occupied by smallest size of Ethernet frame is done as follows:

The maximum number of frames per second is:

$$= \frac{\text{Ethernet Data Rate (bits per second)}}{\text{TotalFrame PhysicalSi ze(bits)}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.1)$$

$$= \frac{10000000}{84 \text{x} 8}$$

= 14880 Frames per second

In practice, this exceeds the forwarding capacity of many routers and bridges (typically 1000's of frames per second). This is however not usually a concern, since most Ethernet networks carry packets with a range of lengths and usually transmit a significant proportion of maximum sized frames (the maximum rate of transmission of maximum sized frames is only 812 frames per second). If the forwarding capacity is momentarily exceeded due to a large number of small frames, the bridge/router will simply discard the excess frames which it cannot process (this is allowed, since Ethernet provides only a best effort link layer service).

The maximum frame payload is 1500 Bytes; this will offer the highest throughput, when the frames are transmitted by only one node on the network. To calculate the

throughput provided by the link layer, the maximum frame rate for this size of frame must be calculated first (Fairhurst, 2002)

Table 2.2: **Total Maximum Frame Physical Size** *(Source:* IEEE 802.3)

Frame Part	Maximum Size Frame
Inter Frame Gap (9.6µs)	12 Bytes
MAC Preamble (+ SFD)	8 Bytes
MAC Destination Address	6 Bytes
MAC Source Address	6 Bytes
MAC Type (or Length)	2 Bytes
Payload (Network PDU)	1500 Bytes
Check Sequence (CRC)	4 Bytes
Total Frame Physical Size	*1538 Bytes*

Calculation of number of bit periods occupied by largest size of Ethernet frame is carried out as follows:

The largest frame consists of 1526 Bytes (as Table 2.2) with a 9.6 µs inter-frame gap (corresponding to 12 Bytes at 10 Mbps). The total utilised period (measured in bits) therefore corresponds to1538 Bytes.

The maximum frame rate is:

$$= \frac{\text{Ethernet Data Rate (bits per second)}}{\text{TotalFrame PhysicalSize (bits)}} \qquad \text{..... (2.2)}$$

$$= \frac{10000000}{1538 \text{x} 8}$$

= 812.74 frames per second

The link layer throughput (i.e. number of payload bits transferred per second) is:

$$= \text{Frame Rate x Size of Frame Payload (bits)}$$
$$= 812.74 \text{ x } (1500 \text{ x } 8)$$
$$= 9\,752\,880 \text{ bps.}$$

This represents a throughput efficiency of 97.5 %.

2.1.3 **Security in ISO/OSI Reference Model**

There are three functional planes in each protocol layer:

i. Users (also called bearers);

ii. Signalling and control; and

iii. Management.

To achieve secure network communications, the security objectives should be accomplished in each appropriate protocol layer and in each suitable functional plane. ISO/IEC 7498-1 standard (ISO Part 1, 1984),

The ISO/IEC 7498-2 standard (ISO Geneva,1989), and the ITU-T X.800 Security Architecture for Open Systems Interconnection recommendation (ITU, Geneva, 1991), extended the ISO/OSI 7498-1 reference model (also described in the ITU-T recommendation X.200) to cover security aspects which are general architectural elements of communications protocols. The X.800 recommendation provides a general description of security services and related mechanisms, which may be provided by the reference model. It also defines the positions within the reference model where the services and mechanisms may be provided.

Following the ISO and ITU Standards, the security objectives are accomplished through security policies and security services. (ISO Geneva 1989; ITU Geneva 1991).

A security policy is the set of criteria that define the provision of security services, where a security service is a service which is provided by a layer of communicating open systems, in order to ensure adequate security of the systems or of data transfers. The security services are implemented by security mechanisms which are in general mechanisms that can be used to technically enforce and implement a security service. (Douligeris and Serpanos 2007).

2.1.3.1 **Security Services and Security Mechanisms**

The basic security services in OSI communications include the following (ISO Geneva 1989; ITU Geneva 1991):

i) **Authentication:** Authentication service is employed to validate the claimed identity of a communicating principal (peer entity authentication) or that the claimed source of a data unit is valid (data origin authentication).

ii) **Access Control:** Acess Control service is used to protect the information assets and resources available via OSI from unauthorized access. This service may be applied to various types of access, such as read, write, or execute or combinations of the above. Access to resources may be controlled through various types of access policies, such as rule-based or identity-based security policies. The access control services is expected to work with the authentication services, since granting access rights to a principal requires prior authentication of the principal requesting a particular access.

iii) **Data Confidentiality:** The Data Confidentiality service is employed to protect the data from access by unauthorized principals. According to the X.800 recommendation, variants of this service include connection confidentiality (when it involves all the layers of communication), connectionless confidentiality (when it provides confidentiality in a connectionless service data unit), selective field confidentiality (when it protects selective fields of the data), and traffic flow confidentiality (when it protects information that could be potentially derived from observation of traffic flows).

iv) **Data Integrity:** This service prevents unauthorised principals from altering the data during their transmission. There are several forms of Data Intgrity service. Connection integrity with recovery provides integrity of the data and also detects modification, insertion, deletion, and replay of da ta. In contrast, connection integrity

with recovery does not attempt recovery. Selective field connection integrity provides integrity for selective data fields within a connection. Connectionless versions of the above services also exist for connectionless data units.

v) **Non-Repudiation:** Non-Repudiation identifies the signature of all principals and ensures that a principal cannot deny the transmission or the receipt of a message. This service may take one or both of two forms. With non-repudiation with proof of origin, the recipient of data is provided with proof of the origin of data, so that the sender cannot later deny that he or she sent the particular data. With non-repudiation with proof of delivery the sender of data is provided with proof of the delivery of data, so that the receiver cannot later deny having received the particular data.

Table 2.3 describes the relationship of security services and layers. It should be noted that in the application layer 7, it is possible that the application process itself provides security services (ITU-T X.800, ITU, Geneva, 1991).

Table 2.3: **Relationship of Security Services and Layers 1 – 7** *(Source: ITU, Geneva, 1991)*

Service	1	2	3	4	5	6	7
Peer Entity Authentication			X	X			X
Data Origin Authentication			X	X			X
Access Control Service			X	X			X
Connection Confidentiality	X	X	X	X		X	X
Connectionless Confidentiality		X	X	X		X	X
Selective Field Confidentiality						X	X
Traffic Flow Confidentiality	X		X				X
Connection Integrity with Recovery				X			X
Connection Integrity without Recovery				X			X
Selective Field Connection Integrity							X
Connectionless Integrity			X	X			X
Selective Field Connectionless Integrity							X
Nonrepudiation of Origin							X
Nonrepudiation of Deliver							X

The implementation of the security services is provided through security mechanisms. These can also be divided into several categories (Douligeris and Serpanos 2007):

i) **Encipherment Mechanisms:** These mechanisms provide data confidentiality services by transforming the data to forms not readable by unauthorized principals. The encipherment mechanisms can also complement a number of other security mechanisms. The encipherment algorithms are generally divided into symmetric (or secret key), where the same secret key is used for both encipherment and decipherment, and asymmetric (or public key), where two mathematically bounded keys are used, the public key for encipherment and the private, or secret, key for decipherment. Knowledge of the public key does not imply knowledge of the secret key. Issues related with the management of the keys are raised both in symmetric and asymmetric encipherment mechanisms. Examples of symmetric encipherment algorithms are AES, Twofish, and RC5, while examples of asymmetric encipherment algorithms are RSA and ElGamal.

ii) **Digital Signatures:** Digital signatures are the electronic equivalent of ordinary signatures in electronic data. Such mechanisms are constructed by properly applying asymmetric encipherment. The decipherment of a data unit with the private key of an entity corresponds to the signature procedure of the data unit. The result is the digital signature of the particular data unit produced by the holder of the private key. The encipherment of the generated digital signature with the corresponding public key of the particular entity corresponds to the verification procedure. Digital signatures can be used to provide peer entity authentication and data origin authentication, data integrity, and non-repudiation services.

iii) **Access Control Mechanisms:** The access control mechanisms are used to provide access control services. They may use the authenticated identity of an entity or other

information related with an entity (e.g., membership, permissions, or capabilities of the entity) in order to determine and enforce the access rights of the entity. They may also report unauthorized access attempts as part of a security audit trail. Examples of access control mechanisms are firewalls and operating system user access privileges.

iv) **Data Integrity Mechanisms:** These mechanisms provide data integrity services by appending some kind of checksums to the data which may prove alteration of the data. Data integrity may involve a single data unit or field or a stream of data units or fields. In general, provision of the second without the first is not practical. The message authentication codes (MACs) and the digital signatures can be used as data integrity mechanisms.

v) **Authentication Mechanisms:** These mechanisms provide authentication services by assuring the identity of a principal. Examples of such mechanisms are passwords, cryptographic techniques, and biometrics. Authentication mechanisms may also be based on cryptographic techniques and trust infrastructures such as public key infrastructure (PKI).

vi) **Traffic-Padding Mechanisms:** These mechanisms provide protection from traffic analysis attacks. Several network protocols and security mechanisms include padding mechanisms to protect the exchanged communication. These can be effective only if the traffic padding is protected by a confidentiality service.

vii) **Routing Control Mechanisms:** These mechanisms allow the selection of a specific route for the communicating data, either dynamically or statically through prearranged routes. Moreover, by applying security policies, data carrying certain security labels may be routed through certain sub-networks, relays, or links. Hackers, viruses, and malicious programs frequently exploit the security vulnerabilities of routing protocols in order to launch network security attacks.

viii) **Notarization Mechanisms:** Finally, notarization mechanisms are used to assure the integrity, the source or destination, and the time of sending or delivering of transmitted data. Such assurance mechanisms may be part of the networking protocols in use and/or of a trusted third party which may be used to assure the communication consistency and non-repudiation. A notarization mechanism may be supported by other mechanisms such as digital signatures, encipherment, or integrity mechanisms.

Table 2.4 describes the relationship between security services and security mechanisms. If a mechanism is indicated as appropriate for a given service, this may be either on its own or in combination with other mechanisms.

Table 2.4: Relationship between Security Services and Mechanisms (*Source: Douligeris and Serpanos 2007*):

Service	Encipherment	Digital Signaure	Access Control	Data Integrity	Authentication Exchange	Traffic Padding Routing Control	Notarization
Peer Entity Authentication	X		X	X			X
Data Origin Authentication	X		X	X			X
Access Control Service			X	X			X
Connection Confidentiality	X	X	X	X		X	X
Connectionless Confidentiality	X	X	X	X		X	X
Selective Field Confidentiality	X					X	X
Traffic Flow Confidentiality	X		X				X
Connection Integrity with Recovery	X			X			X
Connection Integrity without Recovery	X			X			X
Selective Field Connection Integrity	X						X
Connectionless Integrity	X		X	X			X
Selective Field Connectionless Integrity	X						X
Nonrepudiation of Origin	X						X
Nonrepudiation of Deliver	X						X

2.1.4 Network Security Attacks

The description above clearly shows that security threats and attacks are not particular to specific layers. Security threats and attacks may be from the physical to the application layer. It is possible that a successful attack in one layer may render useless the security measures taken in the other layers. Some basic network security attacks are described as follows (Douligeris and Serpanos 2007):

i. **Eavesdropping Attacks:** Eavesdropping is the process of gathering information from a network by snooping on transmitted data. And to eavesdrop is to secretly overhear a private conversation over a confidential communication in a not legally authorized way. The information remains intact, but its privacy is compromised.This can be performed in several different layers—for example, in the network layer by sniffing into the exchanged packets or in the physical layer by physically wire tapping the access medium.

ii. **Logon Abuse Attacks:** A logon abuse attact circumvents all access control mechanisms put in place to allow a user more privileges than authorized.

iii. **Spoofing Attacks:** Spoofing is a situation in which one person or program successfully masquerades as another by falsifying information/data and thereby gaining an illegitimate advantage. A simple instance of this type of attacks is IP spoofing, through which a system is convinced that it is communicating with a known principal and thus provides access to the attacker. The attacker sends a packet with an IP source address of a known trusted host by altering the packet at the transport layer. The target host may be deceived and accept the modified packet as valid.

iv. **Intrusion Attacks:** These are attacks where unauthorized users gain access to a system through the network. These attacks target specific vulnerabilities in assets. For example, a typical web server intrusion attack is a buffer overflow attack, which

occurs when a web service receives more data than it has been programmed to handle and thus reacts in unexpected and unpredicted ways.

v. **Hijacking Attacks:** These attacks are essentially attempts to gain unauthorized access to a system by using a legitimate entity's existing connection. For example, at the session layer, if a user leaves an open session, this can be subject to session hijacking by an attacker. An example of session hijacking is the TCP sequence number attack: This attack exploits the communication session which was established between the target host and a legitimate host that initiated the session. The attacker hijacks the session of the legitimate host by predicting a sequence number selected by the target host, which is used by the TCP.

vi. **Denial-of-Service (DoS) Attacks:** These attacks attempt to exhaust the network or server resources in order to render it useless for legitimate hosts and users. A more advanced type is the distributed denial-of-service (DDoS) attack, where the attacker uses resources from a distributed environment against a target host. Some well-known DoS attacks are as follows (Douligeris and Serpanos 2007)::

   a. **SYN Attack.** In a SYN attack, the attacker exploits the inability of a server process to handle unfinished connection requests. The attacker floods a server process with connection requests, but it does not respond when the server answers those requests. This causes the attacked system to crash, while waiting for the proper acknowledgments of the initial requests.

   b. **Ping of Death.** This is an obsolete DoS attack in which an attacker sends a ping request that is larger than 65,536 bytes, which is the maximum allowed size for the IP, causing the system to crash or restart. Such attacks are not in use today, since most operating systems have implemented measures against it.

c. **Application-Level Attacks.** These attacks are concerned with the exploitation of weaknesses in the application layer and really focus on intrusion attacks in most cases—for example, security weaknesses in the Web server, in the specific technology used in the website, or in faulty controls in the filtering of an input on the server side. Examples of these attacks include malicious software attacks (viruses, Trojans,etc.), Web server attacks, remote command execution, Structured Query Language(SQL) injection, and cross-site scripting (XSS).

2.1.5 Concept of Intrusion Detection

Intrusion threat is defined as the possibility of deliberate unauthorized attempt to access information, manipulate information or render a system unreliable or unusable. Hence an intrusion detection system (IDS) can be defined as a program or hardware/software combination whose primary purpose is to detect unauthorized attempts to access information, manipulate information or render a system unreliable or unusable (Rehman,2003).

A system's next line of defence after the firewall is intrusion detection, and this has been the focus of much research in recent years. This interest is motivated by a number of considerations, including the following:

i. If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised. Even if the detection is not sufficiently timely to pre-empt the intruder, the sooner that the intrusion is detected, the less the amount of damage and the more quickly that recovery can be achieved.
ii. An effective intrusion detection system can serve as a deterrent, acting to prevent intrusions.
iii. Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility. Intrusion detection is

based on the assumption that the behaviour of the intruder differs from that of a legitimate user in ways that can be quantified. Of course, it cannot be expected that there will be a crisp, exact distinction between an attack by an intruder and the normal use of resources by an authorized user rather we must expect that there will be some overlap (Stallings 2011).

Figure 2.4 suggests, the nature of the task confronting the designer of an intrusion detection system.

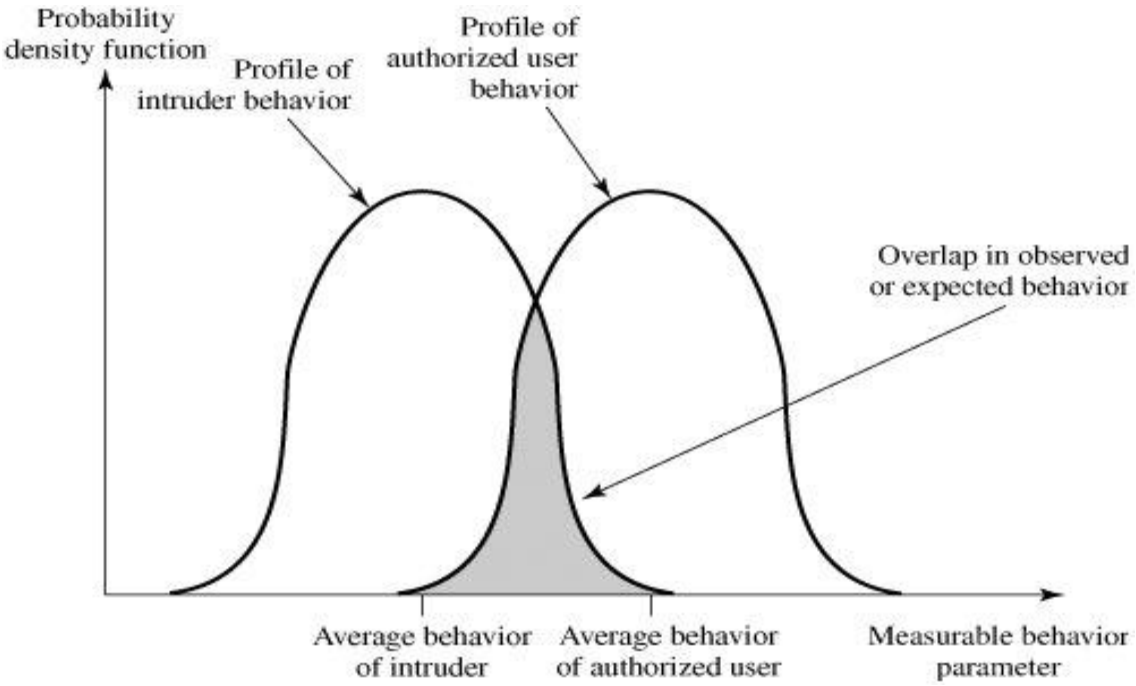


Figure 2.4: **Profiles of Behaviour of Intruders and Authorized Users** (Source: Stallings 2011b)

Although the typical behaviour of an intruder differs from the typical behaviour of an authorized user, there is an overlap in these behaviours. Thus, a loose interpretation of intruder behaviour, which will catch more intruders, but can also lead to a number of "false positives," or authorized users identified as intruders. On the other hand, an attempt to limit false positives by a tight interpretation of intruder behaviour can lead to an increase in false negatives, or intruders not identified as intruders.

Thus, there is an element of compromise in the practice of intrusion detection. Anderson (1980) postulated that one could, with reasonable confidence, distinguish between a masquerader and a legitimate user. Patterns of legitimate user behaviour can be established by observing past history, and significant deviation from such patterns can be detected. Anderson (1980) also suggested that the task of detecting a misfeasor (legitimate user performing in an unauthorized fashion) is more difficult, in that the distinction between abnormal and normal behaviour may be small. Anderson (1980) then concluded that such violations would be undetectable solely through the search for anomalous behaviour. However, misfeasor behaviour might nevertheless be detectable by intelligent definition of the class of conditions that suggest unauthorized use.

2.1.5.1 **Types of Intrusion Detection Systems**

Intrusion detection systems come in two basic forms namely:

a. Host based Intrusion detection system; and

b. Network based Intrusion detection system.

However, whether host based or network based, an intrusion detection system is based on two fundamental paradigms: Behaviour based (Anomaly detection) and Knowledge based (Signature based).

Each of these has their own merits and demerits. The two complement each other and are often deployed together to give more complete coverage of a given system (Debar, 2002) There are two complementary trends in intrusion detection: To use the knowledge accumulated about attacks and look for evidence of the exploitation of these attacks (Almgren et. al, 2000), and to build a reference model of the usual behaviour of the information system being monitored and look for deviations from the observed usage (Anderson et. al, 1992). The first trend is often referred to as misuse detection (Jagannathan et.al, 1993; Kumar, 1994), but also as detection by appearance (Sobirey, 1998). The second

trend is referred to as anomaly detection (Jagannathan et al, 1993) or detection by behaviour (Sobirey, 1998). Here, the term knowledge-based intrusion detection is used for the first trend, as it describes the technique being used more accurately. The term behaviour-based intrusion detection is used for the second trend.

2.1.5.2 **Knowledge-Based Intrusion Detection**

Knowledge-based intrusion-detection techniques is the application of the knowledge accumulated about specific attacks and system vulnerabilities over a period of time. Such knowledge base of vulnerabilities is contained in the intrusion-detection system and looks for attempts to exploit them. When such an attempt is detected, an alarm is raised. Meanwhile, any action that is not explicitly recognized as an attack based on the systems knowledge base is considered acceptable. The accuracy of knowledge-based intrusion-detection systems is considered good. However, their completeness depends on the regular update of knowledge about attacks (Jagannathan et. al, 1993).

Advantages of the knowledge-based approaches are that they have, in theory, very low false-alarm rates, and that the contextual analysis proposed by the intrusion-detection system is detailed, making it easier for the security officer using this intrusion-detection system to understand the problem and to take preventive or corrective action (Jagannathan et.al, 1993).

Drawbacks include the difficulty of gathering the required information on the known attacks and keeping it up to date with new vulnerabilities and environments. Maintenance of the knowledge base of the intrusion detection system requires careful analysis of each vulnerability and is therefore a time-consuming task.

2.1.5.3 **Behaviour-Based Intrusion Detection**

Behaviour-based intrusion detection techniques assume that an intrusion can be detected by observing a deviation from normal or expected behaviour of the system or the

users. The model of normal or valid behaviour is extracted from reference information collected by various means. The intrusion detection system later compares this model with the current activity. When a deviation is observed, an alarm is generated. In other words, anything that does not correspond to a previously learned behaviour is considered intrusive. Therefore, the intrusion detection system might be complete (i.e. all attacks should be caught), but its accuracy is a difficult issue (i.e. a lot of false alarms can be obtained). Advantages of behaviour-based approaches are that they can detect attempts to exploit new and unforeseen vulnerabilities. They can even contribute to the (partially) automatic discovery of these new attacks. They also help detect 'abuse of privileges' types of attacks that do not actually involve exploiting any security vulnerability. This is basically a paranoid approach: Everything which has not been seen previously is dangerous (Jagannathan et al, 1993).

The high false alarm rate is generally cited as the main drawback of behaviour-based techniques because the entire scope of the behaviour of an information system may not be covered during the learning phase. Also, behaviour can change over time, introducing the need for periodic online retraining of the behaviour profile resulting either in unavailability of the intrusion detection system or in additional false alarms. The information system can undergo attacks at the same time the intrusion detection system is learning the behaviour. As a result, the behaviour profile contains intrusive behaviour, which is not detected as an anomaly (Debar, 2002).

2.1.5.4 **Network Vulnerabilities to Internal Intrusion**

Ensuring adequate security on inbound traffic seem much easier to accomplish as all form of rules can be put in place at the border to ensure unwanted traffic is not allowed into the system. However, the same principle is not necessarily applicable to traffic generated within the same network. It is even more difficult on a campus network where diverse

activities result in different forms of traffic which makes defining particular set of traffic as illegitimate very difficult. Firewalls have been used to take care of these security holes but they have been found not sufficient to address intrusions that present themselves in several forms.

The following examples illustrate several but not exhaustive forms of intrusion:

i) **Attempted Break-In**: Someone attempting to break into a system might generate an abnormally high rate of password failures with respect to a single account or the system as a whole.

ii) **Masquerading or Successful Break-In**: Someone logging into a system through an unauthorized account and password might have a different login time, location, or connection type from that of the account's legitimate user. In addition, the penetrator's behaviour may differ considerably from that of the legitimate, user, in particular, the masquerader might spend most of the time browsing through directories and executing system status commands, whereas the legitimate user might concentrate on editing or compiling and linking programs. Many break-ins have been discovered by security officers or other users on the system who have noticed the alleged user behaving strangely.

iii) **Penetration by Legitimate User**: A user attempting to penetrate the security mechanisms in the operating system might execute different programs or trigger more protection violations from attempts to access unauthorized files or programs. If the attempt succeeds, that user will have access to commands and files not normally permitted to such user.

iv) **Leakage by Legitimate User**: A user trying to leak sensitive documents might log into the system at unusual times or route data to remote printers not normally used.

v) **Inference by Legitimate User**: A user attempting to obtain unauthorized data from a database through aggregation and inference might retrieve more records than usual.

vi) **Trojan Horse**: The behaviour of a Trojan horse planted in or substituted for a program may differ from the legitimate program in terms of its CPU time or I/O activity.

vii) **Virus**: A virus planted in a system might cause an increase in the frequency of executable files rewritten, storage used by executable files, or a particular program being executed as the virus spreads.

viii) **Denial-of-Service**: An intruder able to monopolize a resource (e.g., network) might have abnormally high activity with respect to the resource, while activity for all other users is abnormally low. (Denning 1987)

2.1.6 Acceptable Use Policy (AUP)

The ICT Directorate of the Ahmadu Bello University has developed an AUP; some of its highlights include the following:

Unacceptable use of University computers and network resources may be summarised as (this is not an exhaustive list):

a) The retention or propagation of material that is offensive, obscene or indecent, except in the course of recognised research or teaching that is permitted under ABU Zaria and international laws, propagation will normally be considered to be a much more serious offence.

b) Intellectual property rights infringement, including copyright, trademark, patent, design and moral rights;

c) Causing annoyance, inconvenience or needless anxiety to others, for example by propagating rumours that have the potential to disrupt peaceful activities on campus.

d) Unsolicited advertising, often referred to as "spamming";

e) Sending e-mails that purport to come from an individual other than the person actually sending the message using, e.g., a forged address or false identity.

f) Actions or inactions which intentionally, or unintentionally, aid the distribution of computer viruses or other malicious software;

g) Attempts to access or actions intended to facilitate access to computers for which the individual is not authorised;

h) Unauthorised resale of ABU networking services or information. These restrictions should be taken to mean, for example, that the following activities will normally be considered to be a breach of this policy (potential exceptions should be discussed with ICT Directorate):

i) The downloading, distribution, or storage of music, video, film, or other material, for which you do not hold a valid licence, or other valid permission from the copyright holder;

j) The distribution or storage by any means of pirated software;

k) Connecting an unauthorised device to the University network, i.e. one that has not been configured to comply with this policy and any other relevant regulations and guidelines relating to security, IT purchasing policy, and acceptable use;

l) Circumvention of Network Access Control;

m) Monitoring or interception of network traffic, without permission;

n) Probing for the security weaknesses of systems by methods such as port-scanning, without permission;

o) Associating any device to network Access Points, including wireless, to which you are not authorised;

p) Non-academic activities which generate heavy network traffic, especially those which interfere with others' legitimate use of IT services or which incur financial costs;

q) Excessive use of resources such as bandwidth and file shares, leading to a denial of service to others, especially when compounded by not responding to requests for action;

r) The deliberate viewing and/or printing of pornographic images;

s) The passing on of electronic chain mail;

t) The posting of defamatory comments about staff or fellow students on virtual learning environment or social networking sites;

u) The copying of other people's web site material without the express permission of the copyright holder;

v) The use of peer-to-peer and related applications within the University. These include, but are not limited to, Ares, BitTorrent, Direct Connect, Morpheus, KaZaA, Skype.

2.2 Review of Similar Works

With the widespread dependence by individuals and enterprises on the Intranet and Internet, the issue of security of information, abuse of privileges and misuse amongst others has come to the forefront and necessitated the development of such security systems as the IDS. Several researchers have looked at different approaches and models to address the various types of security breaches. Only of recent has searchlight been on internal attacks which is claimed to be much more difficult to address due to network and security policies for a campus network (Wang et. al, 2009). Some previous works reviewed are presented as follows:

Denning (1987) developed a model of a real-time intrusion-detection expert system capable of detecting break-ins, penetrations, and other forms of computer abuse. The model was based on the hypothesis that security violations can be detected by monitoring a system's audit records for abnormal patterns of system usage. The model

included profiles for representing the behaviour of subjects with respect to objects in terms of metrics and statistical models. The model is independent of any particular system, application environment, system vulnerability, or type of intrusion, thereby providing a framework for a general-purpose intrusion-detection expert system. However, due to its non-independence, the amount of false negatives will be high in certain environments like campus network environment.

Maselli *et al* (2001) designed and implemented an open source application for the detection of known anomalies on a network using defined generic network rules. The work was however restricted to detections at the IP level alone and as a generic approach, any intrusion that can be introduced beyond this protocol level will swiftly pass beyond the detection of the system.

Attig *et al* (2005) developed a Snort Intrusion Filter for TCP(SIFT) to reduce the load in processing IP packets by an intrusion detection system running Snort in order to ensure that all packets are processed to provide a complete protection. This work is a very interesting and useful work, however it only reduced IP packets processing load for Snort.

Dreger *et al* (2008) developed a system to predict the CPU and memory consumption of Intrusion Detection Systems. This was to ensure that a realistic configuration is built for IDSs especially for operators where large parameters and analysis option are needed instead of considering trade-offs between detection accuracy and resource requirements. This work did not develop a detection system but a system that helps to optimise the functionality of IDSs.

Ismail *et al* (2009) developed a framework of Intrusion Detection System (IDS) implementation using Snort application in campus network environment. The work was basically targeted at detecting malware attack with emphasis on external attacks.

This was achieved by developing an Open Source application Analyzer for Snort (OSAS) which was used to achieve alerts generated by Snort and is able to analyze and detect malware activity in client server environment over LAN. Malware is just one of the several intrusions that could happen on a network and as such detection of malware alone on a network does not address security on such network. The work was limited to issues of remote attacks only.

Wang *et al* (2009) analyzed security problem of campus network, and designed an intrusion detection structure that can detect attacks using a data mining algorithm. Associated rules were created in intrusion detection to find association relation in network data stream in the algorithm. The system could detect the intrusion attack behaviour in campus network, especially the internal attacks on a campus network. It was, however a host based intrusions detection system and limited its data set to only applications on the network without considering other features and elements on the network.

Yi *et al* (2010) designed and implemented a campus network security system based on distributed network intrusion detection technology. The system integrated a variety of attack detection technologies which could effectively detect complex and steeped coordinated or combination attacks to good effect. However, the implementation of the IDS showed clearly that it was focused on remote attacks and not attacks generated from within the network.

Gandhi *et al* (2010) designed a system to detect and combat some common attacks on network systems, and was basically a signature based intrusion detection methodology implemented in VC++. A signature based IDS will monitor packets on the network and compare them against a database of signatures or attributes from known malicious threats. The system worked as an alert device in the event of attacks

directed towards an entire network. Like any other signature based IDS, it only detects known attacks by comparing traffic with signatures in its database. Attacks with variant signatures and new attacks escape detection.

Bartos *et al* (2010) proposed a flow based network intrusion detection system that addressed hardware-accelerated probes used to collect unsampled Net Flow data from gigabit-speed network links. This was a multi-agent data fusion method that combined several anomaly detection algorithms. The data acquired on the network is pre-processed and passed to anomaly detection models to gather independent anomaly opinions for each flow. The anomaly data is passed to several trust models to aggregate the anomalies with past experience, and the flows are re-evaluated to obtain their trustfulness, which is further aggregated to detect malicious traffic. Experiments performed on-line on real campus network illustrated system suitability for real-time networks. However, the system is signature based and does not make provision for regular update of the signatures which makes easy passage for new signatures or variant of old ones, this may render the system almost inefficient within a very short time.

Hari *et al* (2010) designed an intrusion detection system for web documents using neural network. The Back Propagation Neural Network (BPNN) architecture was applied to supervise the learning of the neural network. The log of Internet Security Acceleration server 2000 was analyzed to determine the web documents that should not be accessed by unauthorized persons in an organisation. This work focused mainly on detecting intrusion on web pages only, and as such it is not a complete intrusion detection system.

Carliss *et al* (2010) developed an intelligent system to analyse network traffic flow and detect potential exfiltration of internal data, targeting packet level to high threat

level traffic flows. An algorithm that used currently available network monitoring tools to transmit data to an intelligent analyzer was used. The analysis then reduced both the amount of traffic analysed and the amount of analysis performed per network flow thus enabling the system to detect potentially malicious traffic very accurately. The accuracy of this is however in doubt because of its dependence soley on a network monitoring tool that will definitely not be able to report all elements contained on a network adequately, but only with strong reporting tendencies for some features.

Lu Yao and Chen (2010) Proposed a network Intrusion detection system with Snooping Agents (NIDS-SA) based on network-based intrusion detection techniques, the NIDS-SA proposed is based on three components: Intrusion detection Node (IDN), Intrusion detection Coordinator (IDC) and the Snooper Agent (SA). The IDN is responsible for packet capturing and local intrusion detection while the IDC takes IDN's intrusion information as input and decides whether the suspicious activity is intrusive by comparing the information to the IDC intrusion signature, the Snooper agent lunches and gather network information. The proposed model has three components and eleven (11) sub-components,the functionality of the model is too distributed and will affect performance and throughput on the network. The model did not only consider protocols but also did not make provision for signature update which gives room for high rate of false alarm

KHAN Ali M.Sadiq (2011) applied Genetic Algorithm to devise rules that can be used efficiently in the classification of various types of attacks. With emphasis on DoS or probing attack which are relatively more common and can be detected more accurately if the contributing parameters are formulated in terms of rules. He discovered that the accuracy of rule based learning increases with the number of

iteration. However, any system that uses such algorithm will record high accuracy only on old and relatively known attacks.

Kim and Kim (2012) proposed the development of an Agent-Based Honeypot framework for protecting servers in campus network from attacks. The proposed framework is aimed at solving the problem of honeypot being unable to lure attackers for it to gather information on new form of attackers on the network for the purpose of updating knowledge-based attack. The proposed framework's success however depends on successful installation of agents on all clients/host on the network without wihich the Honeypot may not be able to gather signatures of new attacks, which will inturn impact on the update of knowledge base of any form of IDS on the network since it depends on the information gathered by the Honeypot for its signature update.

Xiong and Peng (2012) developed a distributed Snort Intrusion detection system model applying protocol analysis and pattern matching detecting method to Snort IDS to improve the speed and accuracy of Snort intrusion detection system. The developed distributed systems consisted of these parts based on the function: the sensors, data management centre and the management decision centre. The model used the sensor to collect data from the network, while the data management centre collects the alarm information for storage and classification of the alarm. The data management centre is to collect and analyze alarm information. However since two of the three parts of the model depends solely on the sensors and the sensors only collect and reacts to protocols on the network, any other network element that can be intrusive will pass undetected.

Considering the reviewed similar works done in the area, this work is focused on the development of an intrusion detection system for a multidisciplinary campus with a view to addressing the issue of internally generated intrusions (local to local and local

to remote). It also considers not only applications and protocols but also IP details of the traffic which is a major departure from the study of Wang et al (2009). The model also hybridizes the behaviour and knowledge bases of the IDS and is implemented as an agent-based application written in C# unlike wang et al (2009) that employed data mining technique.

CHAPTER THREE

DATA COLLECTION AND MODEL DEVELOPMENT

3.1 **Introduction**

This chapter discusses the network setup that was put in place for data collection and the development of the model.

3.2 **Network Setup for Data Collection**

Two Network monitoring systems were installed in the Data Centre of the Ahmadu Bello University Zaria network. The two systems were placed on the live network to collect live network data in real time. Two different servers were used each with a different operating system to ensure that the data collected for use as the knowledge base of the model will have universal application across major operating systems. Figure 3.1 is a modified version of Figure 1.4 with the inclusion of the monitoring servers.

One of the servers runs Windows Server 2008 and also has installed and configured on it, PRTG and Netflow Analyser. While the second monitoring system runs a free and open source operating system (Ubuntu Server Version 11) and has installed and configured on it two other collectors, NFSEN and NTOP.

The core router is a Cisco router and has a Cisco switch connected to it as the public switch. On the switch, the netflow feature is enabled such that it will allow the netflow collector to analyse probed aggregate flows sent to it.

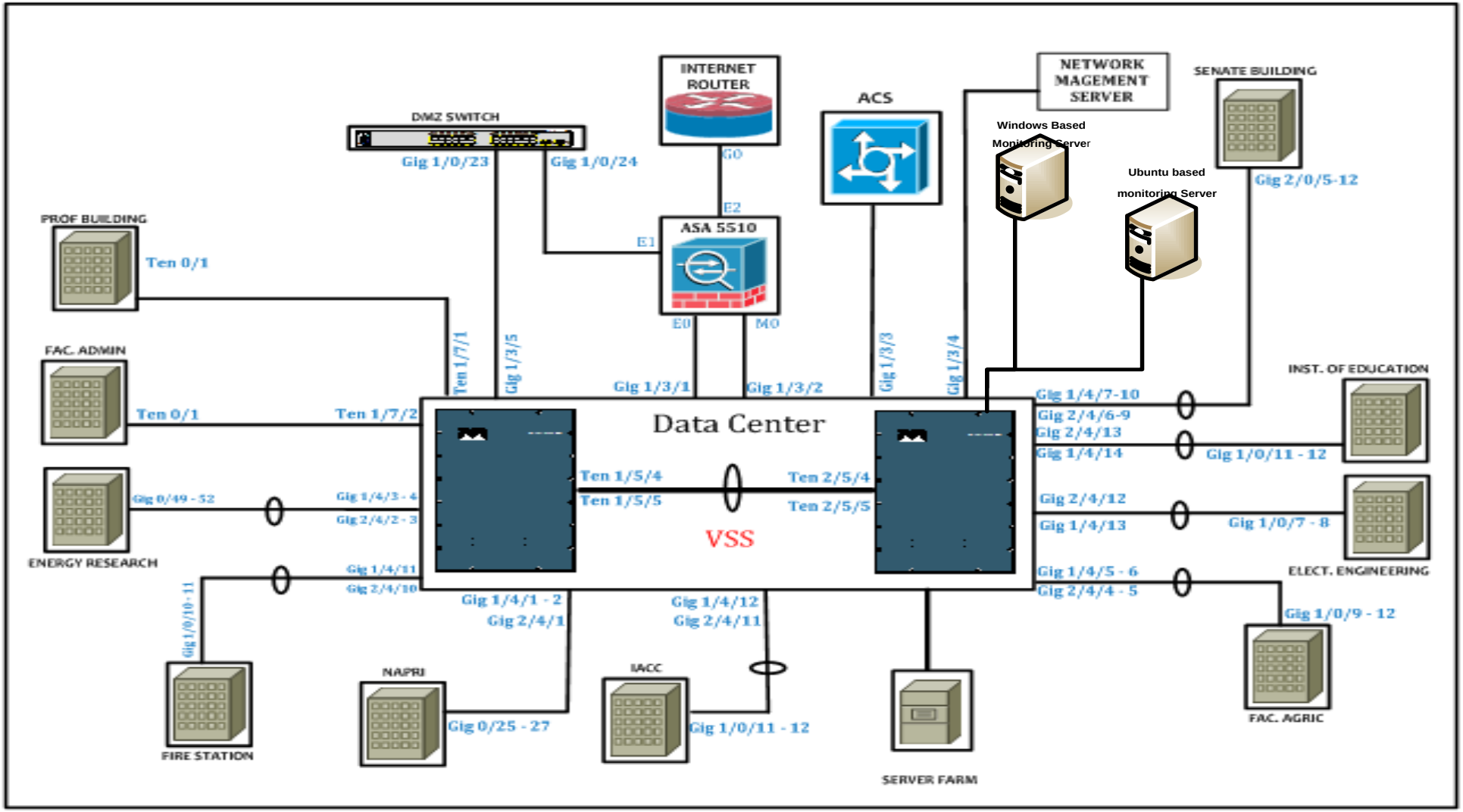


Figure 3.1: **Modified Layout with Monitoring Servers Installed**

3.2 Cisco IOS Netflow Probes

Cisco IOS Netflow efficiently provides a key set of services for IP applications, including network traffic accounting, usage-based network billing, network planning, security, Denial of Service monitoring capabilities, and network monitoring. Netflow provides valuable information about network users and applications, peak usage times, and traffic routing. Netflow is a Cisco proprietary product and is the leader in IP traffic flow technology. It probes aggregates flows and can send them to a flow collector for analysis.

Cisco Netflow is the collector (sniffer) of choice here because the University runs a Cisco based hierarchical network from the core to access layers. However, three other network monitoring application (NTOP, NFSEN and PRTG) were also installed to provide for comparison of data collected for informed decision at the data partitioning stage and also ensure that the developed model is not hardware or software limited.

3.3 Data Collection Methods

Live data of network traffic on the Ahmadu Bello University campus network comprising of three campuses was obtained from the ICT Data Centre. These data were collected over a period of 24 months (January 2011 through December 2012) with the help of software tools installed for the purpose of collecting and analysing of probed aggregate traffic on the network. These are, Netflow Analyser, Nfsen, Ntop and PRTG.

3.3.1 Netflow Analyser

NetFlow Analyzer is a net-flow collector with complete traffic analytical tools and leverages flow technologies to provide real time visibility into the network bandwidth performance. Netflow analyser collects, analyzes and reports about what the network bandwidth is being used for and by whom. A typical screenshot of the Netflow Analyser interface presents a list of applications and protocols on the network with percentage of traffic transmitted by each protocol and/or application as shown in Figure 3.2. The summary

of applications and protocols used on the network as detected by Netflow analyser are arranged in descending order.

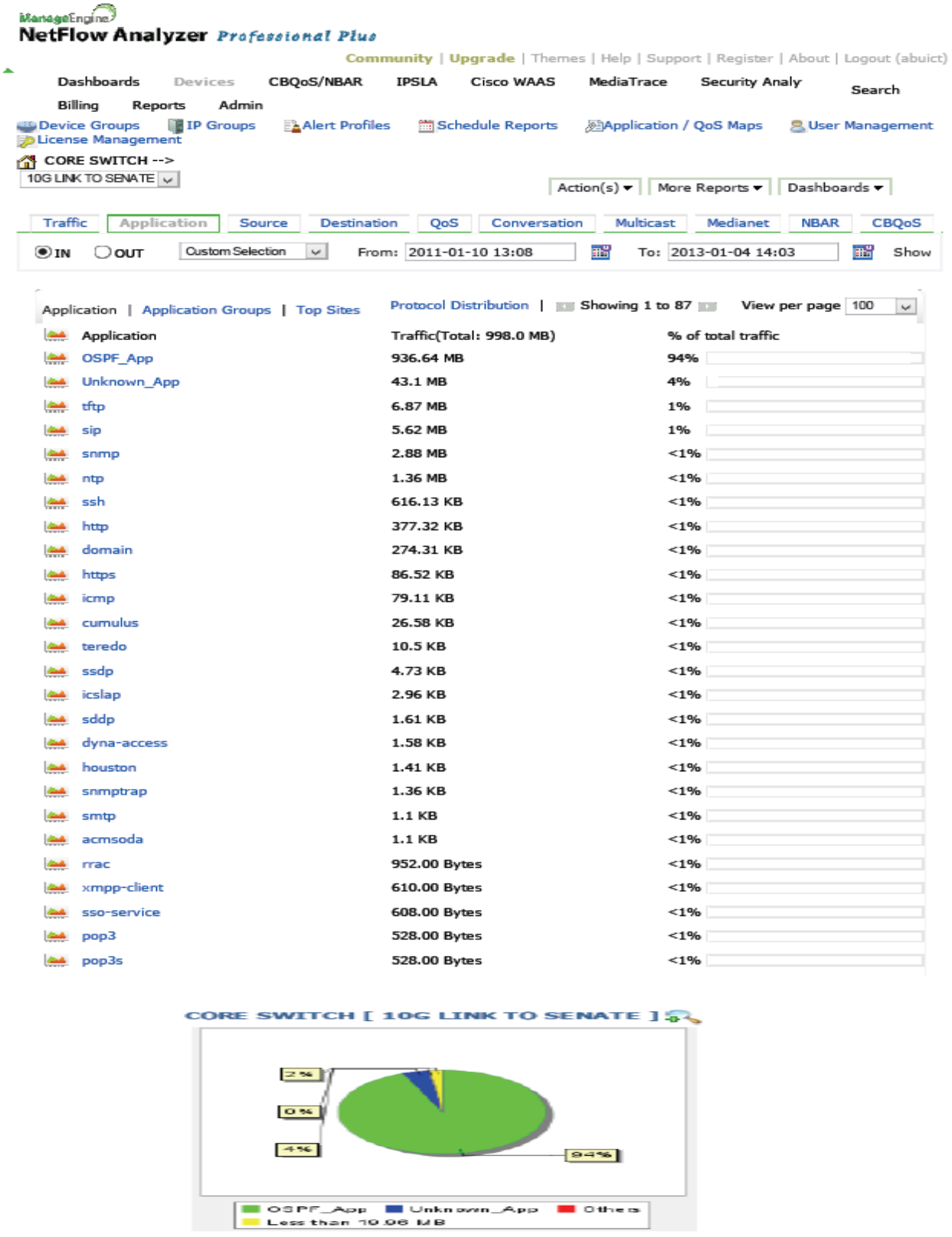
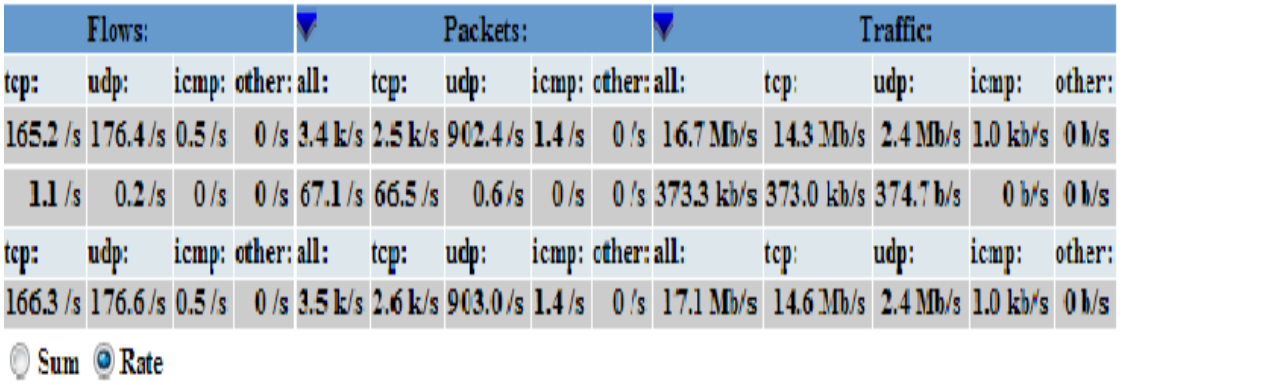


Figure 3.2: **Screenshot of Netflow Analyser**

3.3.2 **NFSEN**

NFSEN is a graphical web-based front end for the nfdump netflow tools. With Nfsen, it is possible to display the netflow data from many sources (Flows, Packets and Bytes), easily navigate through the netflow data, process the netflow data within the specified time span, create history as well as continuous profiles and develop plug-ins to process and display netflow data on a regular interval. A typical screenshot of the Nfsen interface presents source and destination IP of host with packet size and time stamp as shown in Figure 3.3.



Flows:				Packets:					Traffic:				
tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:
165.2 /s	176.4 /s	0.5 /s	0 /s	3.4 k/s	2.5 k/s	902.4 /s	1.4 /s	0 /s	16.7 Mb/s	14.3 Mb/s	2.4 Mb/s	1.0 kb/s	0 b/s
1.1 /s	0.2 /s	0 /s	0 /s	67.1 /s	66.5 /s	0.6 /s	0 /s	0 /s	373.3 kb/s	373.0 kb/s	374.7 b/s	0 b/s	0 b/s
tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:
166.3 /s	176.6 /s	0.5 /s	0 /s	3.5 k/s	2.6 k/s	903.0 /s	1.4 /s	0 /s	17.1 Mb/s	14.6 Mb/s	2.4 Mb/s	1.0 kb/s	0 b/s

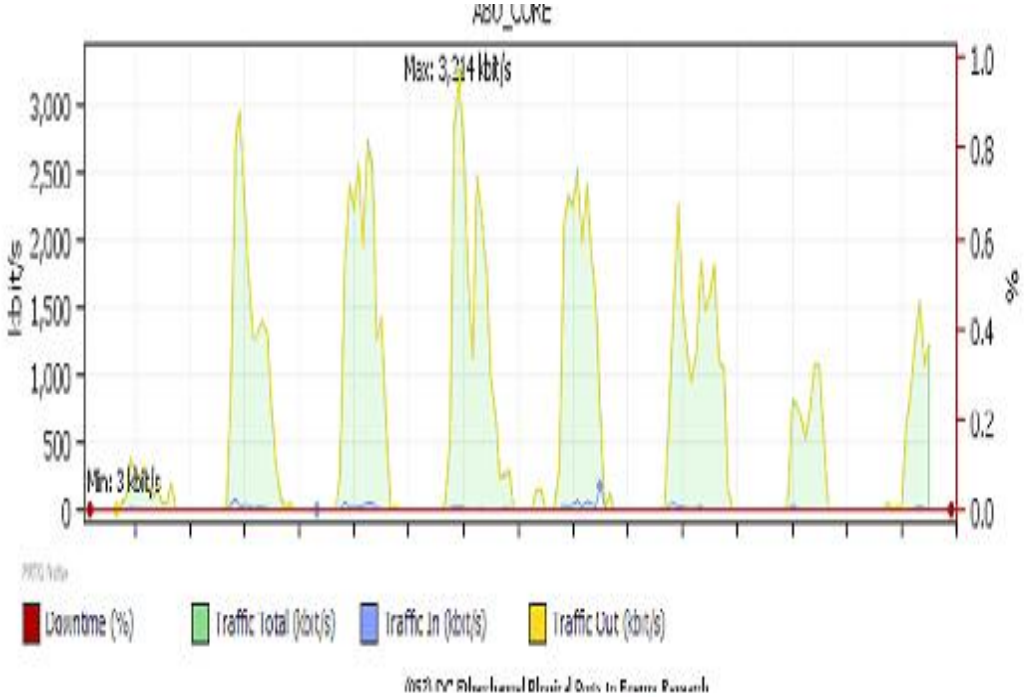
Sum Rate

Duration	Proto	Src IP Addr	Dst IP Addr	Packets	Bytes	bps	Bpp	Flows
1795.040	UDP	10.255.12.114	98.116.224.140	1050	80335	358	76	1
1799.010	UDP	27.106.65.20	10.1.48.131	11462	14.2 M	62977	1235	1
1681.460	TCP	178.79.206.141	10.1.51.41	10057	15.0 M	71580	1495	6
1261.990	TCP	139.216.201.123	10.1.50.148	782	191823	1216	245	3
1087.970	UDP	122.108.76.91	10.1.34.203	653	840096	6177	1286	1
1305.700	TCP	95.211.214.230	10.1.51.41	9872	14.7 M	90259	1492	12
1068.750	UDP	10.1.34.203	98.154.218.0	5220	5.7 M	42739	1093	1
1068.260	UDP	98.154.218.0	10.1.34.203	5329	289675	2169	54	1
1094.360	UDP	10.1.29.38	192.168.0.27	275	22000	160	80	1
1094.360	UDP	10.1.29.38	187.64.74.189	275	22000	160	80	1
1098.350	UDP	187.64.74.189	41.203.92.114	276	22080	160	80	1
1098.350	UDP	41.203.92.114	187.64.74.189	276	29808	217	108	1
1290.460	UDP	94.212.142.82	10.1.34.203	4283	6.0 M	36888	1389	1
1297.780	TCP	178.79.194.168	10.1.51.41	4502	6.7 M	41511	1495	3
1275.240	UDP	10.1.34.220	88.112.90.207	1269	65677	412	51	4

Figure 3.3**: Screenshot of Nfsen**

3.3.3 **Paessler Router Traffic Grapher (PRTG)**

PRTG is a network and bandwidth usage monitoring software for Microsoft Windows. It can monitor and classify bandwidth usage in a network A typical screenshot of the PRTG interface presents total traffic size and time stamp as shown in Figure 3.4



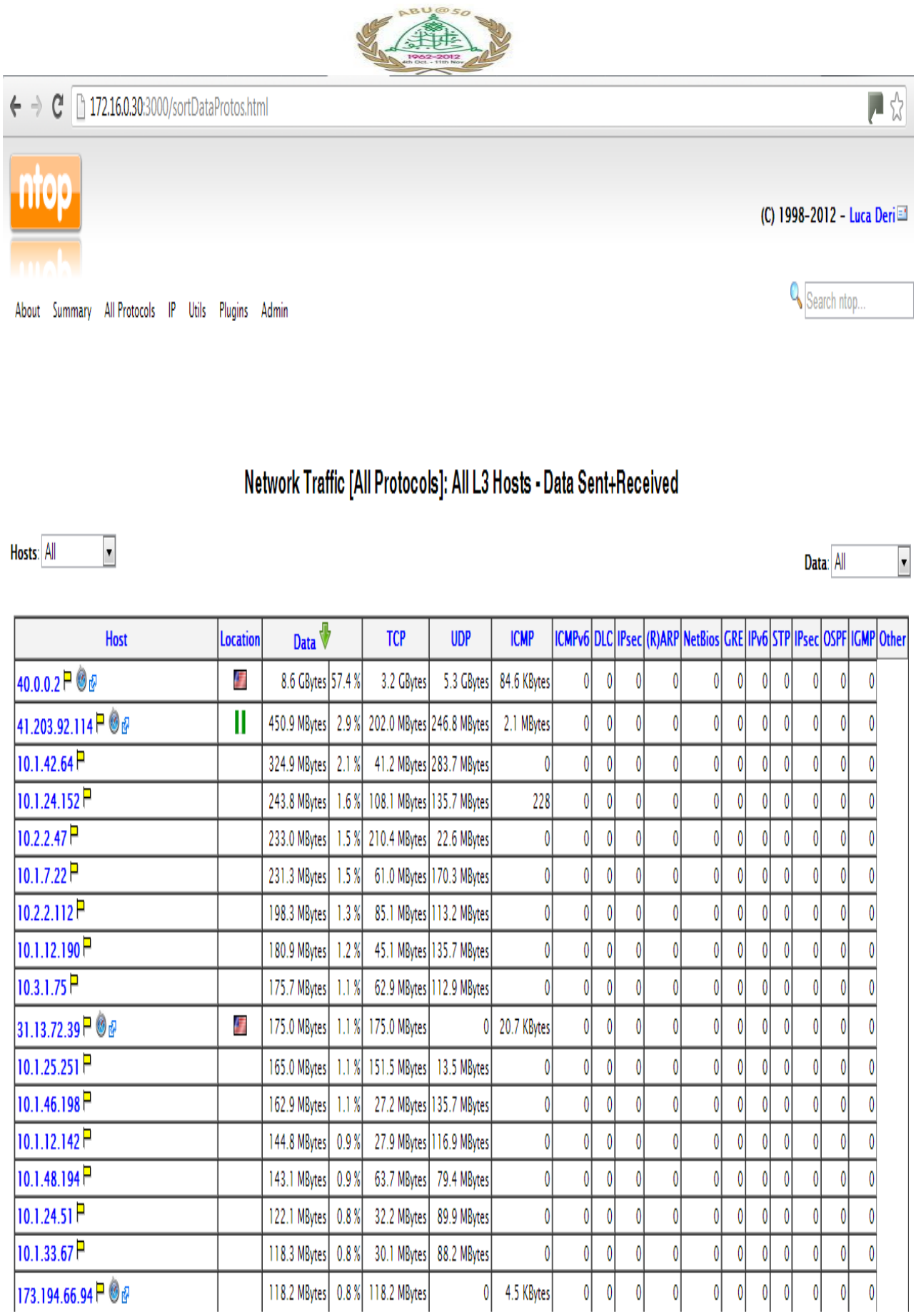
Channel	Average	Total
Traffic Total	595 kbit/s	48,634,078 KByte
Traffic In	8 kbit/s	624,527 KByte
Traffic Out	588 kbit/s	48,009,550 KByte

Traffic Total (volume)	Traffic Total (speed)	Traffic In (volume)	Traffic In (speed)	Traffic Out (volume)	Traffic Out (speed)	Downtime	Coverage
453,834 KByte	1,206 kbit/s	24 KByte	0.06 kbit/s	453,826 KByte	1,206 kbit/s	0 %	86 %
475,043 KByte	1,081 kbit/s	5,035 KByte	11 kbit/s	470,009 KByte	1,070 kbit/s	0 %	100 %
682,882 KByte	1,554 kbit/s	9,301 KByte	21 kbit/s	673,581 KByte	1,533 kbit/s	0 %	100 %
568,446 KByte	1,294 kbit/s	5,249 KByte	12 kbit/s	563,200 KByte	1,282 kbit/s	0 %	100 %
393,400 KByte	895 kbit/s	27 KByte	0.06 kbit/s	393,373 KByte	895 kbit/s	0 %	100 %
261,477 KByte	595 kbit/s	28 KByte	0.06 kbit/s	261,449 KByte	595 kbit/s	0 %	100 %

Figure 3.4: **Screenshot of PRTG**

3.3.4 **NTOP**

Ntop is a network probe that shows network usage. In the interactive mode, it displays the network status on the user's terminal and in the Web mode, it acts as a web server, creating a HTML dump of the network status. It supports a NetFlow/sFlow emitter/collector, a HTTP-based client interface for creating Ntop-centric monitoring applications, and Round Robin Database (RRD) for persistently storing traffic statistics. A typical screenshot of the Ntop interface is as shown in Figure 3.5. The Ntop shows the IP address of host, different protocol in use by each host and the data size transmitted by each protocol.



Host	Location	Data		TCP	UDP	ICMP	ICMPv6	DLC	IPsec	(R)ARP	NetBios	GRE	IPv6	STP	IPsec	OSPF	IGMP	Other
40.0.0.2		8.6 GBytes	57.4 %	3.2 GBytes	5.3 GBytes	84.6 KBytes	0	0	0	0	0	0	0	0	0	0	0	
41.203.92.114		450.9 MBytes	2.9 %	202.0 MBytes	246.8 MBytes	2.1 MBytes	0	0	0	0	0	0	0	0	0	0	0	
10.1.42.64		324.9 MBytes	2.1 %	41.2 MBytes	283.7 MBytes	0	0	0	0	0	0	0	0	0	0	0	0	
10.1.24.152		243.8 MBytes	1.6 %	108.1 MBytes	135.7 MBytes	228	0	0	0	0	0	0	0	0	0	0	0	
10.2.2.47		233.0 MBytes	1.5 %	210.4 MBytes	22.6 MBytes	0	0	0	0	0	0	0	0	0	0	0	0	
10.1.7.22		231.3 MBytes	1.5 %	61.0 MBytes	170.3 MBytes	0	0	0	0	0	0	0	0	0	0	0	0	
10.2.2.112		198.3 MBytes	1.3 %	85.1 MBytes	113.2 MBytes	0	0	0	0	0	0	0	0	0	0	0	0	
10.1.12.190		180.9 MBytes	1.2 %	45.1 MBytes	135.7 MBytes	0	0	0	0	0	0	0	0	0	0	0	0	
10.3.1.75		175.7 MBytes	1.1 %	62.9 MBytes	112.9 MBytes	0	0	0	0	0	0	0	0	0	0	0	0	
31.13.72.39		175.0 MBytes	1.1 %	175.0 MBytes	0	20.7 KBytes	0	0	0	0	0	0	0	0	0	0	0	
10.1.25.251		165.0 MBytes	1.1 %	151.5 MBytes	13.5 MBytes	0	0	0	0	0	0	0	0	0	0	0	0	
10.1.46.198		162.9 MBytes	1.1 %	27.2 MBytes	135.7 MBytes	0	0	0	0	0	0	0	0	0	0	0	0	
10.1.12.142		144.8 MBytes	0.9 %	27.9 MBytes	116.9 MBytes	0	0	0	0	0	0	0	0	0	0	0	0	
10.1.48.194		143.1 MBytes	0.9 %	63.7 MBytes	79.4 MBytes	0	0	0	0	0	0	0	0	0	0	0	0	
10.1.24.51		122.1 MBytes	0.8 %	32.2 MBytes	89.9 MBytes	0	0	0	0	0	0	0	0	0	0	0	0	
10.1.33.67		118.3 MBytes	0.8 %	30.1 MBytes	88.2 MBytes	0	0	0	0	0	0	0	0	0	0	0	0	
173.194.66.94		118.2 MBytes	0.8 %	118.2 MBytes	0	4.5 KBytes	0	0	0	0	0	0	0	0	0	0	0	

Figure 3.5: **Screenshot of Ntop**

3.4 Data Partitioning

This research work is focused mainly on internal network traffic; hence two data categories are needed:

i. **Local-to-Local Traffic**: Most campus network have servers and services hosted on the same network, therefore any traffic from host to host on the same network or from host to any of the servers on the network are regarded as local-to-local traffic. The largest percentage of this traffic is expected to be from host to servers/services on the network.

ii. **Local-to-Remote Traffic**: Host within the campus network will also generate a lot of data to remote sites outside the network. The traffic going outside the network is equally important to ensure that hosts on the University network are not using the network to bridge security of other networks.

3.4.1 Sample Data Collected

Data was collected from the four different monitoring systems for the period of twenty four (24) months (January 2011 to December 2012). The data collected ran into several hundreds of pages, however Tables 3.1 and 3.4 show samples of such data collected from the monitoring systems.

Based on the data collected, some wre classified as ‘Known’ i.e the monitoring system could identify the application or protocol type. These are as shown in Appendix I. Appendix II contains those that the monitoring systems could not identify either the application or protocol type and these were classified as unknown. Details of some of the unknown traffic are shown in Appendix III.

Table 3.1 presents an extract of host traffic on the network as seen by NTOP. NTOP presents each host with their data size and further breaks down the data size into various components by protocol and application usage

Table 3.1: Traffic Summary from Ntop

S/N	Host	Data Size	%	Protocol			
				TCP	UDP	ICMP	Others
1	**10.1.14.40**	272.8 MBytes	2.4 %	272.7 MBytes	131.5 Kbytes	0	
2	**10.1.48.193**	238.1 MBytes	2.1 %	238.1 MBytes	77.9 Kbytes	0	
3	**10.1.14.83**	233.4 MBytes	2.0 %	233.3 MBytes	131.5 Kbytes	0	
4	**10.1.40.38**	231.3 MBytes	2.0 %	231.1 MBytes	250.7 Kbytes	0	
5	**10.1.17.104**	202.6 MBytes	1.8 %	202.2 MBytes	399.5 Kbytes	0	
6	**172.16.0.30**	202.1 MBytes	1.8 %	0	202.1 Mbytes	0	
7	**172.16.0.10**	199.1 MBytes	1.7 %	0	199.1 Mbytes	10.7 KBytes	
8	**10.1.38.24**	194.3 MBytes	1.7 %	179.8 MBytes	14.5 Mbytes	0	
9	**10.4.7.50**	182.3 MBytes	1.6 %	182.2 MBytes	144.5 Kbytes	0	
10	**10.1.48.148**	179.5 MBytes	1.6 %	179.4 MBytes	154.3 Kbytes	360	
11	**10.3.2.103**	176.0 MBytes	1.5 %	175.7 MBytes	349.1 Kbytes	0	
12	**10.1.14.42**	163.6 MBytes	1.4 %	163.5 MBytes	64.2 Kbytes	0	
13	**10.1.37.52**	161.4 MBytes	1.4 %	160.0 MBytes	1.5 Mbytes	0	

It is important to note that only hosts 7 and 10 on table 6 stream icmp packets on the network during this period.

Table 3.2 presents traffic as captured by Netflow which reports traffic size according to application or protocol. It can be noted that the second largest traffic type reported is unknown. Figure 3.3 is a bar chart representation of Table 3.2

Table 3.2: **Traffic Summary from Netflow Analyser**

Application	Traffic	% of total traffic
OSPF	936.64 MB	94%
Unknown	43.10MB	4%
Tftp	6.87MB	1%
Sip	5.62MB	1%
Ntp	1.36MB	<1%
Dzoglserver	2.24 KB	<1%

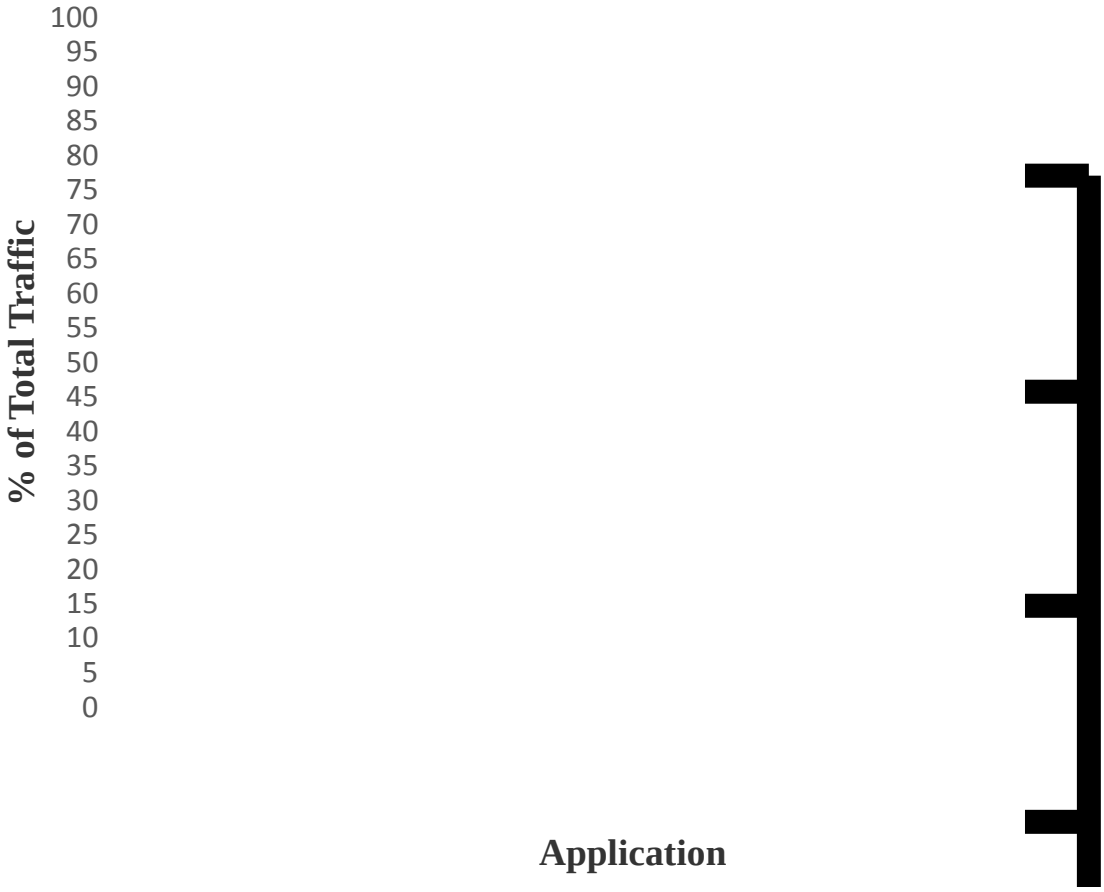


Figure 3.6: **Bar Graph showing Traffic Summary from Netflow**

Table 3.3 presents traffic captured and reported by the PRTG which record a cumulative inflow and outflow size and speed of packets over a given period of time.

Table 3.3**: Traffic Summary from PRTG**

Traffic Total (volume)	Traffic Total (speed)	Traffic In (volume)	Traffic In (speed)	Traffic Out (volume)	Traffic Out (speed)	Downtime	Coverage
701 Kbyte	5,683 kbit/s	57 KByte	462 kbit/s	644 KByte	5,221 kbit/s	0 %	0 %
1,952,110 KByte	4,443 kbit/s	198,017 KByte	451 kbit/s	1,754,093 KByte	3,992 kbit/s	0 %	100 %
2,592,428 KByte	5,900 kbit/s	456,678 KByte	1,039 kbit/s	2,135,750 KByte	4,860 kbit/s	0 %	100 %
1,887,093 KByte	4,295 kbit/s	472,044 KByte	1,074 kbit/s	1,415,049 KByte	3,220 kbit/s	0 %	100 %
1,376,593 KByte	3,133 kbit/s	382,593 KByte	871 kbit/s	994,001 KByte	2,262 kbit/s	0 %	100 %
1,217,842 KByte	2,771 kbit/s	346,422 KByte	788 kbit/s	871,420 KByte	1,983 kbit/s	0 %	100 %
647,496 KByte	1,474 kbit/s	102,196 KByte	233 kbit/s	545,299 KByte	1,241 kbit/s	0 %	100 %
690,225 KByte	1,571 kbit/s	122,912 KByte	280 kbit/s	567,314 KByte	1,291 kbit/s	0 %	100 %
930,287 KByte	2,117 kbit/s	102,765 KByte	234 kbit/s	827,522 KByte	1,883 kbit/s	0 %	100 %
843,002 KByte	1,918 kbit/s	148,670 KByte	338 kbit/s	694,331 KByte	1,580 kbit/s	0 %	100 %
1,007,717 KByte	2,293 kbit/s	153,216 KByte	349 kbit/s	854,501 KByte	1,945 kbit/s	0 %	100 %
1,903,169 KByte	4,331 kbit/s	114,332 KByte	260 kbit/s	1,788,837 KByte	4,071 kbit/s	0 %	100 %
2,960,547 KByte	6,737 kbit/s	243,497 KByte	554 kbit/s	2,717,049 KByte	6,183 kbit/s	0 %	100 %
3,440,111 KByte	7,829 kbit/s	327,959 KByte	746 kbit/s	3,112,153 KByte	7,082 kbit/s	0 %	100 %
3,044,554 KByte	6,929 kbit/s	288,956 KByte	658 kbit/s	2,755,597 KByte	6,271 kbit/s	0 %	100 %
2,092,686 KByte	4,762 kbit/s	259,970 KByte	592 kbit/s	1,832,717 KByte	4,171 kbit/s	0 %	100 %
1,455,965 KByte	3,313 kbit/s	152,078 KByte	346 kbit/s	1,303,887 KByte	2,967 kbit/s	0 %	100 %
1,250,629 KByte	2,846 kbit/s	104,154 KByte	237 kbit/s	1,146,474 KByte	2,609 kbit/s	0 %	100 %
1,931,161 KByte	4,395 kbit/s	219,590 KByte	500 kbit/s	1,711,570 KByte	3,895 kbit/s	0 %	100 %
2,106,442 KByte	4,794 kbit/s	219,659 KByte	500 kbit/s	1,886,783 KByte	4,294 kbit/s	0 %	100 %
2,924,822 KByte	6,656 kbit/s	289,966 KByte	660 kbit/s	2,634,857 KByte	5,996 kbit/s	0 %	100 %

Table 3.4 presents traffic as captured and reported by Nfsen, this includes the duration of each packet on the network, the protocol in use including the source and destination IP of each packet.

Table 3.4: **Traffic Summary from Nfsen**

Durationn	Proto	Src IP Addr	Src Pt	Dst IP Addr	Dst Pt	Packets	Bytes	Flows
0.000	UDP	10.1.49.31	22320	187.10.130.228	54274	2	252	1
0.000	TCP	10.1.28.176	4416	4.26.242.125	80	2	96	1
0.000	TCP	10.1.28.219	56819	115.132.88.174	52268	6	332	1
9.000	TCP	10.2.0.95	58193	118.69.169.125	80	4	204	1
5.790	TCP	10.1.28.233	3060	31.186.225.23	80	9	2649	1
0.000	TCP	41.107.253.216	10777	41.203.92.114	49480	2	80	1
1.630	TCP	41.203.92.114	58759	105.236.224.213	62067	4	160	1
37.570	TCP	74.125.163.46	80	10.1.10.248	53064	26	29220	1
16.700	TCP	10.1.10.234	34524	173.204.162.2	80	4	224	1
0.000	UDP	10.1.34.104	58585	41.133.20.15	57059	2	190	1
0.000	UDP	71.167.144.183	11783	10.1.49.31	22320	2	638	1
16.000	TCP	69.55.61.166	80	10.1.16.187	53699	6	708	2
2.950	UDP	99.225.94.254	60186	41.203.92.114	24150	3	174	1
0.000	UDP	10.255.16.50	6882	41.222.183.40	51164	2	190	1
2.010	UDP	10.1.10.207	59802	40.0.0.2	64872	3	183	1
135.980	TCP	10.1.25.161	54910	95.101.238.131	80	16	3395	3

3.4.2 **Identification and Extraction of Traffic Information**

Upon comparing the data collected by NTOP, NFSEN, PRTG and Netflow, it is concluded that Netflow and NTOP (Tables 3.1. and 3.2) presented the data best suited for the work and indeed have similar features. It can be seen that Ntop identified some traffic as unknown, yet it could not assign values to them. Netflow analyser also identified same set of traffic as unknown traffic and clearly showed them to be the second largest traffic on the network after the default Open Shortest Path First (OSPF) protocol application used on the network for routing and communications.

The Ntop collector also clearly differentiated the various categories of data for ease of use. The source and destination IP addresses captured by the Netflow analyser can also be

used to separate the data into different categories. Based on the clarity of these data sets from Ntop and Netflow analyser, the two data were used as training data in the development of intrusion detection model

3.5 **Intrusion Detection Model Development**

The hybrid model developed using the behavioural pattern of the network traffic and the knowledge base built up based on the 24 months of data collected is based on an agent-based program written in C# (C-Sharp) programming language. Its strong programming features as indicated in table 3.5 such as automatic garbage collection, delegates and events management (necessary for development of agent-based applications) make it suitable for this work.

Table 3.5: **Comparison of Object Oriented Programming Language Feautres**

Programming Languages	Auto Garbage collection	Supports Delegates	Event management	Operator overloading	Integers
C#	Yes	Yes	Yes	Not Restricted	Support both Signed and Unsigned
C++	No	No	yes	Restricted	Support Signed
Java	yes	No	Yes	Not Included	Support Only Signed

A worker (or agent) in the context of this work is essentially an autonomous, adaptive and mobile software agent. A software agent is an entity (or program) implemented in software capable of acting to attain a certain objective and it is autonomous if it is capable of such actions without external intervention. It is adaptive if it has the ability to adapt to changes in its environment while mobility ensures that it can source its information from different sources operating on different platforms (OMG, 2000)

3.6 Metrics for the Model

Since the research is essentially concerned about internally generated traffic, the protocol usage on the network which has a direct relation to the applications running on the network was examined. Some traffic properties such as the source IP, destination IP, packet sizes and percentages were considered in the development of the model and determination of the metrics (or constraints in this case). However, port usage was not considered in this work for two major reasons:

i. Ports can be manually changed and masqueraded ;
ii. Default ports are normally under consideration with applications and protocols.

Two separate constraints are loaded by the model:

i. Constraints for defined (known) traffic;
ii. Constraints for undefined (unknown) traffic

The constraints are calculated based on overhead generated by network frames. It is noted that most parameters that form the properties of a frame (preamble, destination address, source address cyclic redundancy check etc) have standard values, however, the value of the payload of each frame which is also the message carried by each frame differs depending on the size and composition of the message. The minimum and maximum frame sizes were used to calculate the default minimum and maximum overheads as shown in Section 2.1.2.3 and Tables 2.1 and 2.2.

The largest frame consists of 1526 Bytes, as in Table 3.1 with a 9.6μs inter-frame gap; therefore

$$\text{Overhead} = \frac{\text{Frame Size} - \text{Payload Size}}{\text{Frame Size}} \quad \text{.....(3.1)}$$

Thus, for the largest size of Ethernet frame :

$$\text{Overhead} = \frac{1538 - 1500}{1538}$$

$$= 0.0247$$

Unknown packets are initially treated as suspicious at this stage. Therefore the constraints for unknown traffic will take into consideration the minimum frame size allowable on Ethernet. The minimum frame payload is 46 Bytes (dictated by the slot time of the Ethernet LAN architecture). The maximum frame rate is achieved by a single transmitting node. This implies a frame consisting of 72 Bytes (as in Table 2.1) with a 9.6μs inter-frame gap (corresponding to 12 Bytes at 10Mbps). The total utilised period (measured in bits) corresponds to 84 Bytes.

$$\text{Overhead} = \frac{\text{Frame Size} - \text{Payload Size}}{\text{Frame Size}} \quad \text{.....(3.2)}$$

Thus, for the smallest size of Ethernet frame

$$\text{Overhead} = \frac{84 - 46}{84}$$

$$= 0.4523$$

From the analysis carried out in determining the default minimum and maximum frame sizes, the maximum overhead allowed on the Ethernet frame is 0.45. As such any frame with score above the maximum overhead will be logged for further analysis.

The hybrid model developed in this research is described with the aid of the flowchart shown in Figure 3.7.

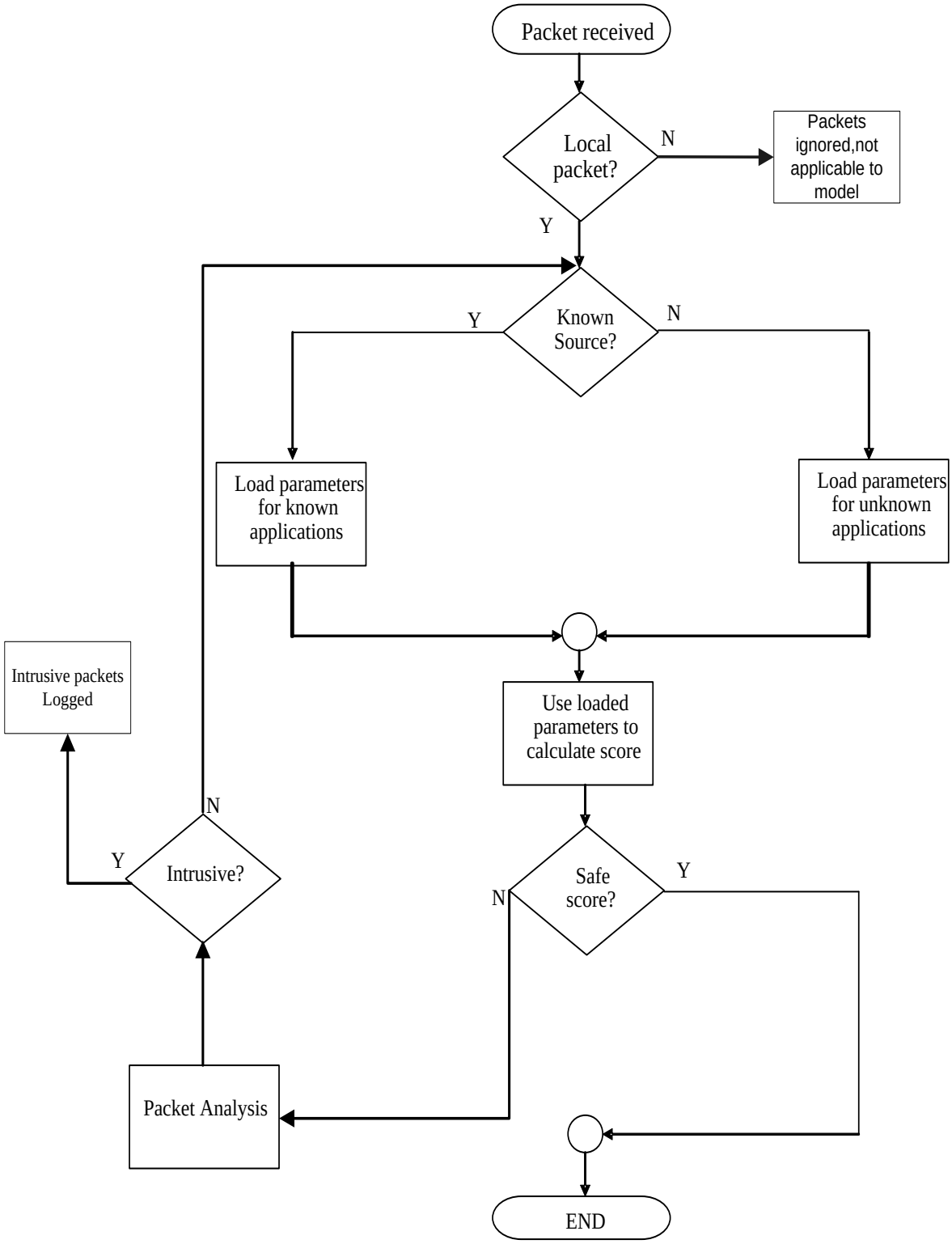


Figure 3.7: **Flowchart for the Intrusion Detection Hybrid Model**

The program functions as follows:

i. When a data packet is received by the model from a Netflow enabled port on the network, the agent (also called worker) first checks the packet's header information for the source and destination IP to determine the origin and destination of the packet and whether the packet is a local packet i.e. whether it is internally-generated and internally-bound traffic or not. Confirmed local packets are sent to the next stage of the programme while confirmed non-local packets are ignored (the model treats only internally generated traffic alone) and allowed to follow the normal route without being processed by the model. The following programme listing shows snipets of the portion that executes the Source and destination address information checks.

```
private void Packet_Processor(Packet _inputPacket)
{
if (!this.Running) return;
var ipv4Dgram = _inputPacket.Ethernet.IpV4;
var transportProps = _inputPacket.Ethernet.IpV4.Transport;
if (transportProps == null) return;
var sourceIp = ipv4Dgram.Source;
var destinationIp = ipv4Dgram.Destination;
var sourcePort = transportProps.SourcePort;
var destinationPort = transportProps.DestinationPort;
var size = ipv4Dgram.Length;
string transport = "";
if (this.Running != true) return;
```

ii. Then the program checks the protocol information of verified local traffic to determine whether the packet is carrying data of a defined (known) protocol and/or application (UDP, TCP etc). At this stage, packets are divided into two depending on whether their properties are on the knowledge base or not. The following programme listing shows snipets of the portion that executes the protocol information checks.

```
        if ((transportProps as PcapDotNet.Packets.Transport.TcpDatagram) != null)
        {
           transport = "TCP";
        }
        else if ((transportProps as PcapDotNet.Packets.Transport.UdpDatagram) !=
null)
        {
           transport = "UDP";
        }
        else
        {
           Log.Success("Something else");
           while (true) ;
        }

        Log.Info(transport + " " + sourceIp.ToString() + ":" + sourcePort.ToString() + "
--> " + destinationIp.ToString()+ ":" + destinationPort.ToString());
        if (transport == "TCP")
        {
           try
           {
              _knownPacketsQueue.Send(_inputPacket.Buffer);

           }
           catch
           {
              return;
           }
        }
        else if (transport == "UDP")
        {
           try
           {
              _unknownPacketsQueue.Send(_inputPacket.Buffer);
           }
           catch
           {
              return;
           }
        }
     }
```

iii. The program has two defined constraint sets; the packets are treated by either of the constraint sets based on result of activity described in (ii). The constraint sets are developed based on the rationale that traffic that utilizes an undefined (unknown)

protocol is more likely to be malicious than defined (known) traffic. Each constraint set consists of a group of mathematical weights which will be applied to each packet. The maximum allowed payload is used as constraint for known traffic while the minimum allowed payload is used as constraint for unknown traffic to determin the risk score. The following programme listing shows snipets of the portion that executes the two sets of constraints on packets:

```
    private const int KNOWN_PACKET_SIZE = 1500;
private const int UNKNOWN_PACKET_SIZE = 84;
...
...
    private void LoadFunctionObjects()
     {
        _UnknownPacketClassifier = new Func<Packet, bool>((p) =>
        {
        double packetScore = (UNKNOWN_PACKET_SIZE -
        p.Ethernet.IpV4.Payload.Length) / (double)UNKNOWN_PACKET_SIZE;
        if (packetScore > this.getSafeValue()) return true;
           return false;
        });

        _KnownPacketClassifier = new Func<Packet, bool>((p) =>
        {
         double packetScore = (KNOWN_PACKET_SIZE - p.Ethernet.IpV4.Payload.Length) /
        (double)KNOWN_PACKET_SIZE;
           if (packetScore > this.getSafeValue()) return true;
           return false;
        });
     }
```

iv. After the constraints are loaded by the program, the relevant properties of a packet are extracted and used to calculate its risk score by the worker. The risk score used by the program is the maximum overhead calculated with default ISO Ethernet standard values of maximum and minimum allowable frames sizes. A unified scale is used for packets containing ‘known’ data and ‘unknown’ data (though from (iii), it was stated clearly that different constraint sets were used for each type). This is to reduce the possibility of false positive bias in the program. The

following programme listing shows snipets of the portion that compares the result of activity (iii) with the safe score

```
protected override void WorkerMethod()
    {

        {
            result = ListenSocket.Receive(buffer, SocketFlags.DontWait);
        }
        catch (Exception e)
        {
            continue;
        }

        if (result > 0)
        {
            Packet _thisPacket = new Packet(buffer, DateTime.Now,
DataLinkKind.Ethernet);
            if (!_thisPacket.IsValid) continue;

            if (QualifierFunction(_thisPacket))
            {
                 try
                {
                    AffirmativeSocket.Send(_thisPacket.Buffer);
                }
                catch (Exception positiveSocketException)
                {
                    continue;
                }
            }
            else
            {
                if (NegativeSocket != null)
                    try
                    {
                        NegativeSocket.Send(_thisPacket.Buffer);
                    }
                    catch (Exception negativeSocketException)
                    {
                        continue;
                    }
            }
        for (int i = 0; i < buffer.Length; i++)
                {
          buffer[i] = 0;
                }
        }
```

```
else
{
                _UnknownPacketClassifier = new Func<Packet, bool>((p) =>
{
double packetScore = (UNKNOWN_PACKET_SIZE -
p.Ethernet.IpV4.Payload.Length) / (double)UNKNOWN_PACKET_SIZE;
if (packetScore > this.getSafeValue()) return true;
   return false;
});

_KnownPacketClassifier = new Func<Packet, bool>((p) =>
{
 double packetScore = (KNOWN_PACKET_SIZE - p.Ethernet.IpV4.Payload.Length) /
(double)KNOWN_PACKET_SIZE;
   if (packetScore > this.getSafeValue()) return true;
   return false;
});
}
```

v. The calculated over head is compared against the risk score which is a given threshold defined on the unified scale. If the overhead exceeds the threshold (risk score), the packet is suspected to be malicious and its content and details are subjected to further analysis. However, if the packet's overhead is lower than the threshold, then the packet details and content are allowed into the network without further analysis.The failed packets are further examined with a view of achieving the following:

a. Identify malicious or intrusive traffic and logging them for determination of the best preventive actions. The following programme listing shows snipets of the portion that logs failed packets.

```
_LoggerTask = new Action<Packet>((p) =>
{
  _logInstance.Error("Suspicious packet detected");
PacketDumpFile.Dump("./Logs/" + p.Timestamp.Ticks.ToString(),
DataLinkKind.Ethernet, 3000, new Packet[] { p });
});
```

b. Provide a feedback to the database of known traffic with the aim of updating the database with traffic that was termed unknown but found not to be intrusive.

3.7 Initial Testing of the Model

The model was tested on a small LAN as shown in figure 3.8 below. The Small LAN has over 30 desktop systems connected to it and it also has a small outdoor wireless Acess point which allows ourdoor browsing within a radius of 150meters.

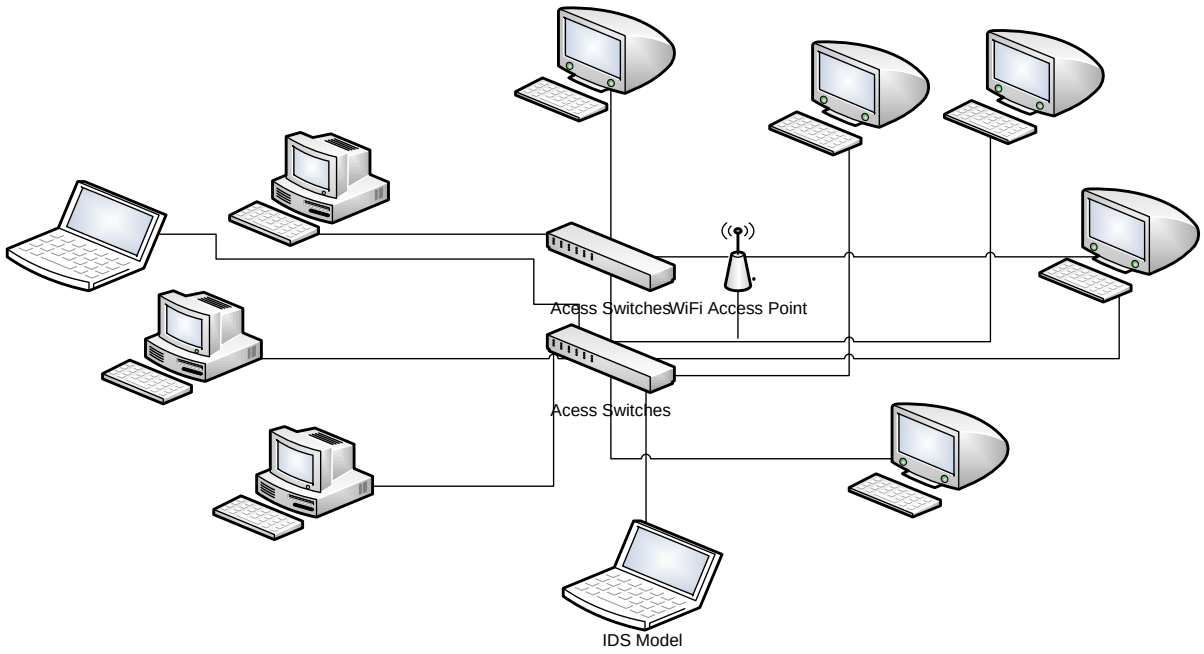


Figure 3.8: Representation of Local LAN Used for Initial Testing of Developed Model

During the testing period, the model captured over 600 packets over a period of 60 minutes. During the testing period, it was discovered that traffic on the small LAN was dragging. A ping test to another host on the same LAN from the output as shown in Figure 3.9 discovered that there is irregularity in packet transmission time on the network resulting in some latency on the network. The output also suggests that the model reacts to the volume of traffic it has to process per time. So when it has more packets coming in, it takes a longer time to transmit depending on the number of packets on qeue for the model to process.

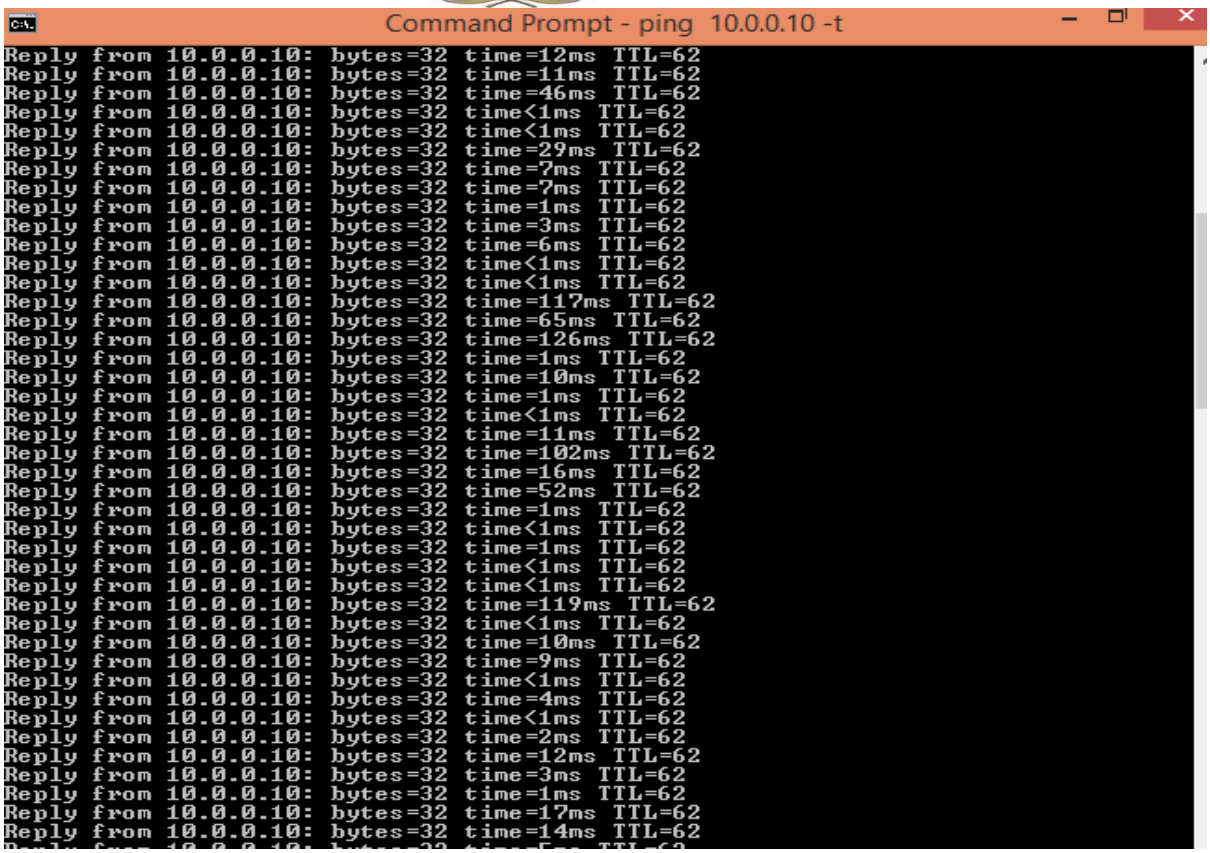


Figure 3.9: Ping Output During Initial Test of Model

From the ping output, it was concluded that the inconsistency in transmission time on the network was as a result of the IDS on the network since it has to capture and treat all packets before sending them over the network this resulted into latency on the network whenever there are several packets to process by the IDS. This was discovered to be as a result of the fact that the model was initially developed to have only one worker (or agent) each for known and unknown packets. This affected the system as there were several other packets that needed to be captured and treated by the model so packets are qeued for processing by the IDS. This suggested that the model will need more workers for processing of packets by both constratints to treat packets simultaneously hence, the need to modify the model.

3.7.1 Model Modifications.

The developed model was modified to initialize with 5 workers and dynamically add more workers as soon as more packets arrived for processing, which added scalability feature to the model. However, to reduce CPU usage and overhead on the model, the workers were also programmed to go to sleep mode when they are not needed and automatically resume when packets arrive for processing (adaptability). The snipets of the program that modified for number of workers is as shown below.

```
{
   public partial class MainPage : Form
   {
      private PcapUtility _utility;
      private bool hasDevices = false;
      private Logger Log;
      private LivePacketDevice _activeInterface;
      private PacketCommunicator _commObject = null;
      private const int N_WORKERS = 5;
      private const string QUEUE_NAME_KNOWN =
@"inproc://valkyrie/known";
      private const string QUEUE_NAME_UNKNOWN =
@"inproc://valkyrie/unknown";
      private const string QUEUE_NAME_LOGGER =
@"inproc://valkyrie/logger";

      private double thisSafeValue = 0.44; //initial safe value.

      private bool Running { get; set; }

      private ZmqContext _baseContext;
      private ZmqSocket _knownPacketsQueue;
      private ZmqSocket _unknownPacketsQueue;

      private List<string> KnownPacketQueues = new List<string>();
      private List<string> UnknownPacketQueues = new List<string>();
      private List<string> LoggingQueues = new List<string>();

      private List<QueueProcessor> WorkerQueue;
      private ValkyrieWorkFactory factory;


      public MainPage()
      {
         InitializeComponent();
         _utility = new PcapUtility();
```

Log = new Logger(_rtbLogWindow);

LoadDeviceNames();
if (!hasDevices)

This tremendously improved the functionality and efficiency of the model. These features were possible due to those inherent characteristics of agents of being autonomous, interactive and adaptive. After the modification, the model was put on the same network and a similar ping was initiated to the same host on the network, the output of the ping test is as shown in Figure 3.10 below. This time the output showed a very consistent transmission time with less than 1ms compared to between 1ms -126ms before the model was modified.

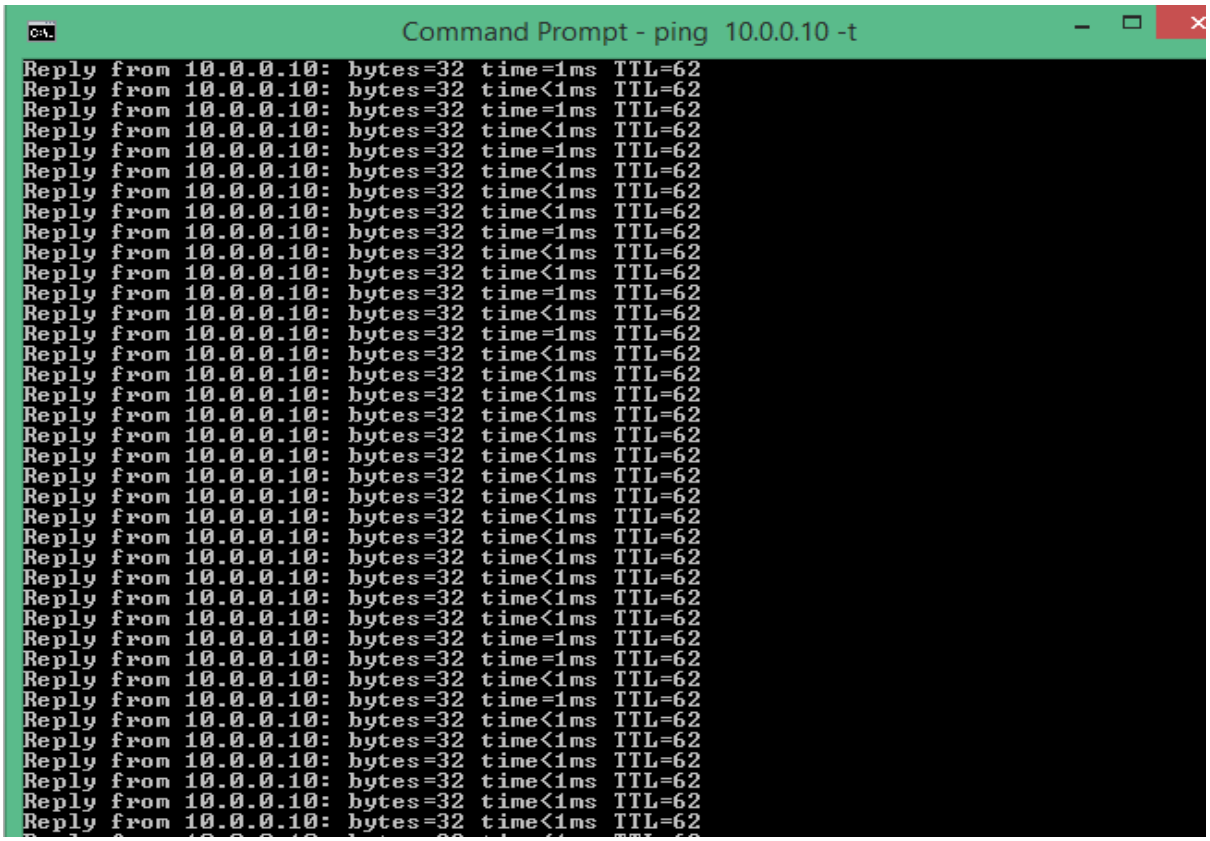


Figure 3.10: Ping Output during test period after Modification of Model

During the testing after the modification, of the 20,495 packets captured over a period of 60 minutes, a total of 269 packets were seen and logged as suspicious packets. A screen

shot is as shown in Figure 3.11. Though this was less than 2% of the total packets captured yet it is of great significance to a network.

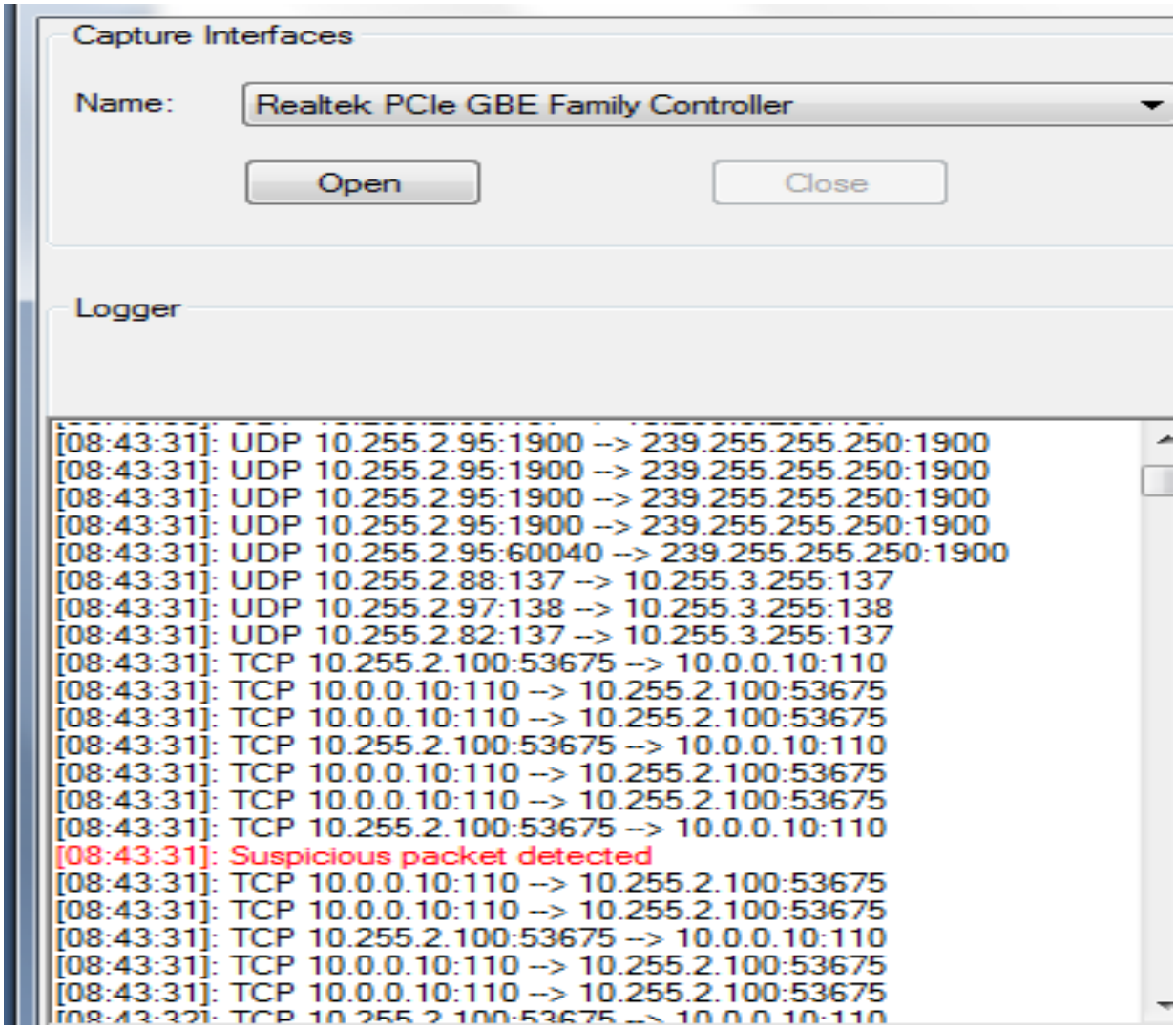


Figure 3.11: **Screenshot of the IDS Model Data Capturing Interface**

CHAPTER FOUR

MODEL VALIDATION

4.1 Model Validation on Live Network

The model was validated on the Ahmadu Bello University network, severally with a two hour test period each time and eventually for a period of sixty days (two months).The model was installed on a high end laptop and connected to the core switch at the University Data centre. It was installed on a Netflow enabled port to capture and analyse network traffic on the live network.

4.2 Analysis of Logged Packets

This section will discuss the packets logged by the model during the validation period, mechanism used in analysing the packet, the choice of packet analyzer and the results of the analysis.

4.2.1 Mechanisms for Analysing Logged Packets

An automated mechanism for analysing logged packets based solely on their content seemed to be a major issue. Hence, using the expert knowledge contained in packet analysis is considered in checking the contents of the logged packets and making appropriate inference. This informed the decision to employ the use of a packet analyzer to verify the properties of all logged packets for intrusive and/or malicious packets at the same time to isolate properties of other packets that are not intrusive for possible update of the knowledge-base of the IDS.

4.2.2 Packet Analyzer

A packet analyzer displays the contents of all fields within a protocol message, it understands the structure of all messages exchanged by protocols. For example, in displaying the various fields in messages exchanged by a HTTP protocol, the packet analyzer understands the format of Ethernet frames, and so can identify the IP datagram within an

Ethernet frame. It also understands the IP datagram format, so that it can extract the TCP segment within the IP datagram. It also understands the TCP segment structure, so it can extract the HTTP message contained in the TCP segment. And finally, it understands the HTTP protocol and so, for example, knows that the first bytes of an HTTP message will contain the string "GET," "POST," or "HEAD". There are serveral types of Packet analysers; the Table below compares both technical and general properties of some packet analysers.

Table 4.1: Comparism of Technical and General Properties of Packet Analysers

S/N	Name of Network Analyzer	Management Console	Operating system
1	Dsniff	Console	Linux, Mac, SPAC
2	Just Sniff	Console	Linux ,Unix
3	kismet	Console	Windows, Linux
4	ngrep	Console	Windows, Linux, Mac, SPAC
5	Observer	GUI	Windows
6	Ominpeek	GUI	Windows
7	Snoop	Console	SPAC
8	tcpdump	Console	Windows, SPAC, Linux, Unix
9	WireShark	Console and GUI	Windows, SPAC, Linux, Unix
10	Xplico	Console and GUI	Linux

From the comparism in Table 4.1, the packet analyzer of choice for this research is "Wirehsark" due to its provision to be managed both via console and graphic user interface (GUI) and also its availability in most operating systems.

4.2.3 **Wireshark**

Wireshark is a free network protocol analyzer that runs on Windows, Linux/Unix, and Mac computers. It is an ideal packet analyzer for this research because it is stable, and has rich functionalities that include the capability to analyze hundreds of protocols, and a well-designed user interface. It operates in computers using Ethernet, Token-Ring, FDDI, serial (PPP and SLIP), 802.11 wireless LANs, and ATM connections (if the Operating System on which it is running allows Wireshark to do so).

It also has a large user base and well-documented support that includes a user-guide (http://www.wireshark.org/docs/wsug_html_chunked)

Wireshark is, however, not an intrusion detection system and as such will not trigger an alarm when a user does unauthorized things on the network but it can help in figuring out what happened. Wireshark will also not manipulate things (such as sending packets) on the network but will only "measure" things from it. A typical Wireshark capture interface is shown in Figure.4.1

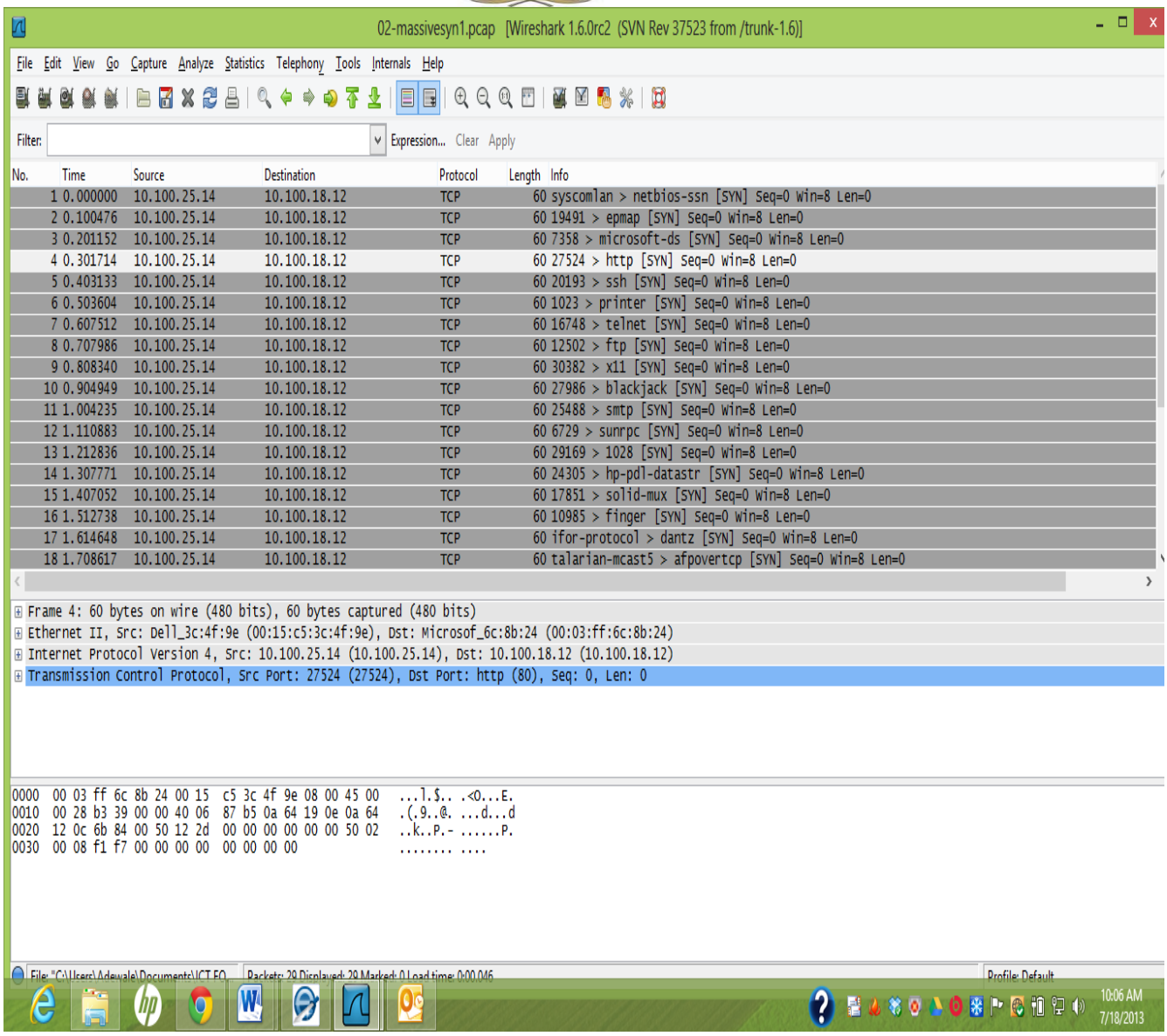


Figure 4.1: **Screenshot of Wireshark Capture Interface**

4.2.4 **Analysis of Logged Packets**

Using Wireshark, the entire properties of a network frame can be seen and thoroughly looked into for malicious traces in packets. This analysis will look closely into the following properties of all logged frames:

i. Protocol field;

ii. Protocol stream;

iii. Frame Size; and

iv. Frame content.

4.2.5 **Results of Packet Analysis**

Over 2000 packets were analysed using Wireshark and most of the packets were eventually resolved to be malicious even though they have known streams of protocols in the database. However, several of these frames were seen to have very large PDUs and at the same time are broadcast traffic with very high time stamps. Such packets are intrusive packets capable of affecting the QoS on the network.

Several other packets were also seen to have useful protocols on the network but such protocols were not defined in data collected which was used as the knowledge base of the model as shown in Figures 4.2 and 4.3

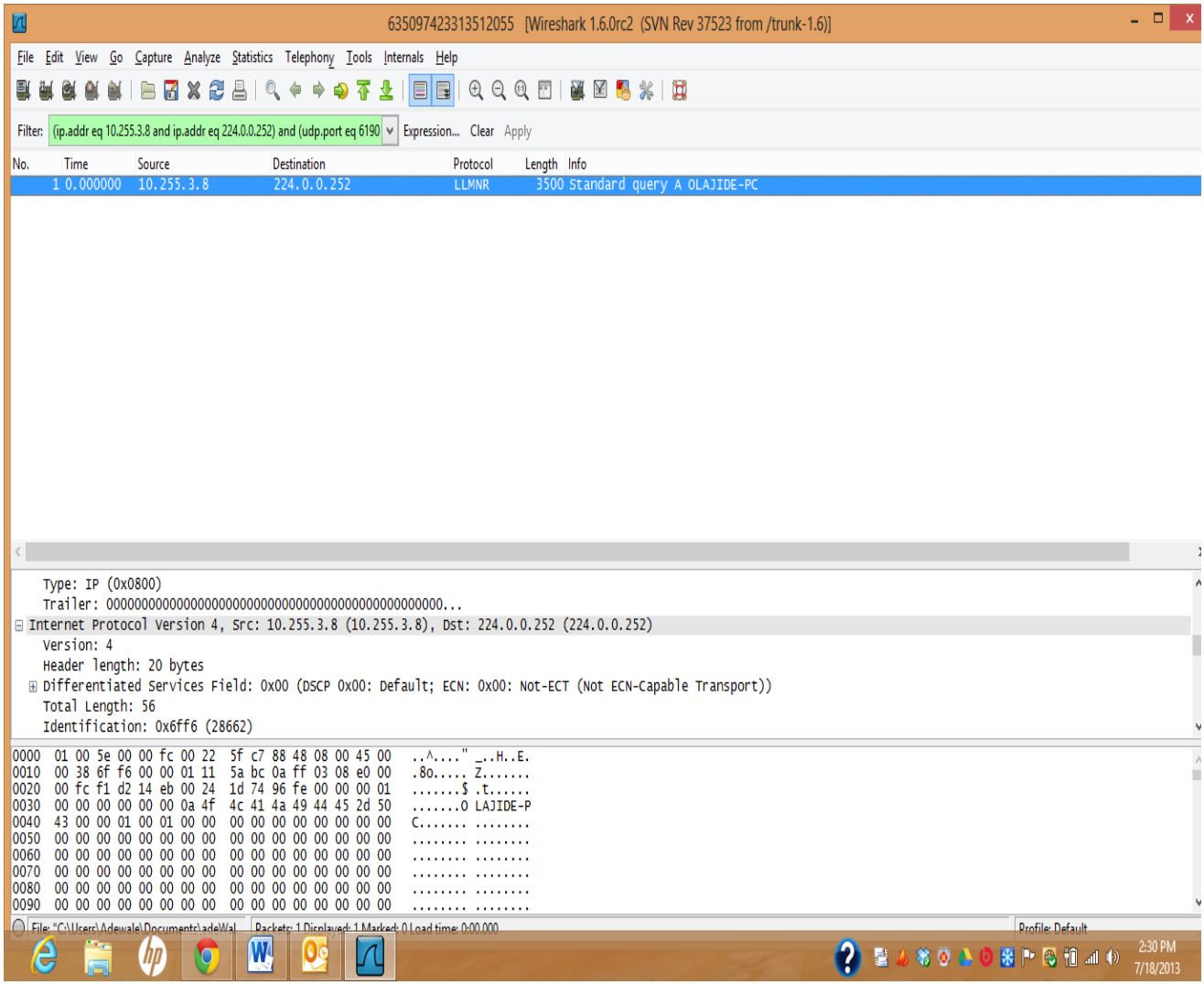


Figure 4.2: **Screenshot of Packet showing an LLMNR Protocol**

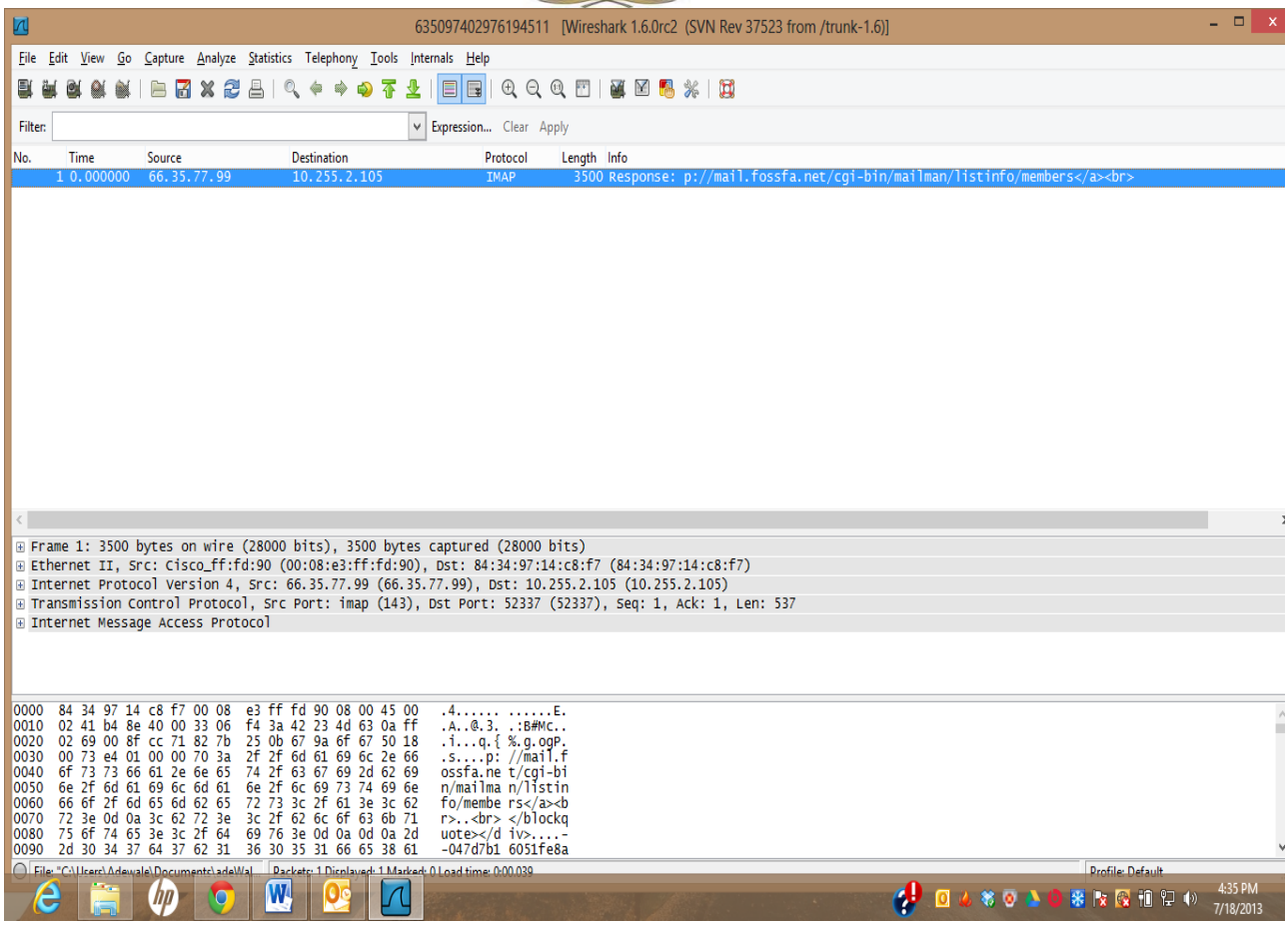


Figure 4.3: **Screenshot a Packet showing IMAP Protocol**

Example of such protocol/application is the Link Local Multicast Name Resolution (LLMNR) which is a protocol that allows computers to perform name resolution for addresses on the same local network without the need for a centrally coordinating DNS server. Table 4.1 presents the list of protocols/applications that were not in the knowledge base of the model but discovered during packet analysis.

4.2.6 **Sample of Intrusive Packets**

Several of the packets that are obviously intrusive were packets that have their application and/or protocols defined in knowledge base but some were found to either be streaming traffic to another host or server on the network over a long period of time while some have information that are more of spams. An IMAP packet with spamming tendency is shown in Figure 4.4

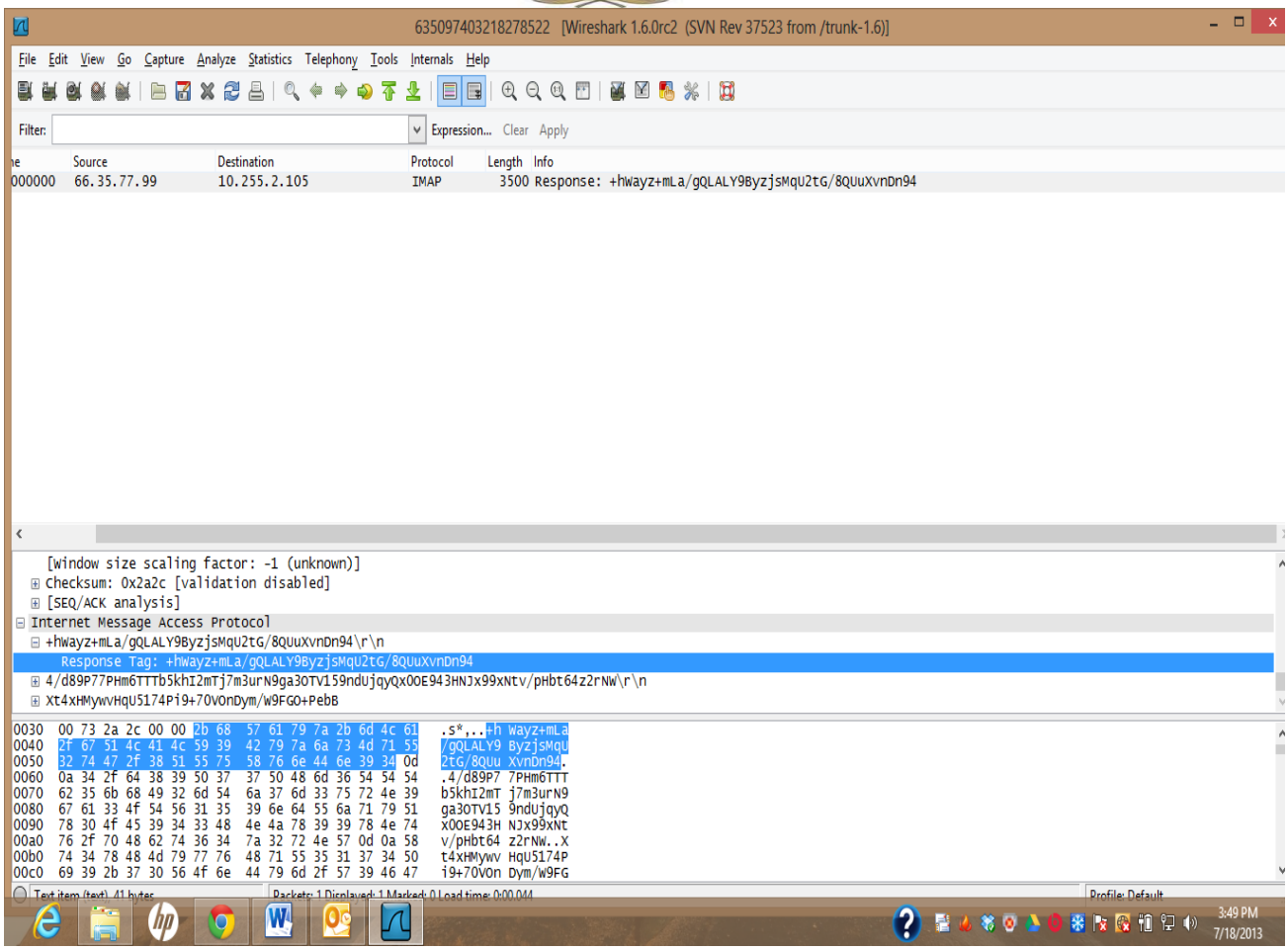


Figure 4.4: **Screenshot a Packet showing IMAP Protocol with Spamming Tendency**

4.3 IDS Knowledge Base Update

After several test run of the model on a live network and analysis of logged packets, the knowledge base of the model was updated to include signatures of applications and protocols detected on the network but not included in the knowledge base used for the model training. These signatures were added to the knowledge base and are expected to reduce false alarms in the number and details of packets that will be logged by the IDS. Table 4.1 presents the details of newly discovered signatures for protocols and applications that were added to the knowledge base of the IDS.

Table 4.2: **List of Additional Protocols/Application Signatures Added to the Knowledge Base**

S/No	Protocol/Application	Justification/Remarks
1	Llmnr	Link Local Multicast Name Resolution Protocol
2	Imap	Internet Message Access Protocol
3	Fmtp	Flight Message Transfer Protocol
4	rfid-rp1	RFID Reader Protocol 1.0
5	ds-slp	Televaulting DS-Sleeper Service
6	Vytalvaultvsmp	Network Services
7	oracle-oms	Oracle Management Server
8	bsquare-voip	Network Services
9	Ideesrv	Network Services
10	bacula-dir	Controller of the Network Backup System
11	Dbabble	Provides Secure Instant Messaging (Network Services)
12	Omscontact	Network Services
13	konshus-lm	License Manager
14	office-tools	Network services
15	msfw-storage	MS Firewall Storage
16	Unot	Network services
17	cloanto-lm	port testing your firewall security
18	new-rwho	Network services
19	Drip	Network services
20	Sbl	Network services
21	Rnmap	Network services
22	mit-ml-dev	Network services
24	fmpro-internal	Network services
25	Ggz	Network services
26	Stun	Network services
27	Bess	Network services
28	Proaxess	Network services

4.4 Updating IDS Interface.

An interface as shown in Figure 4.5 was developed for the IDS model's knowledge base to be updated whenever a new application or protocol is detected and confirmed not to be malicious or intrusive. The new interface has a window where the newly detected application or protocol can be added to the knowledge-base. This additional feature is to ease the task of knowledge base update.

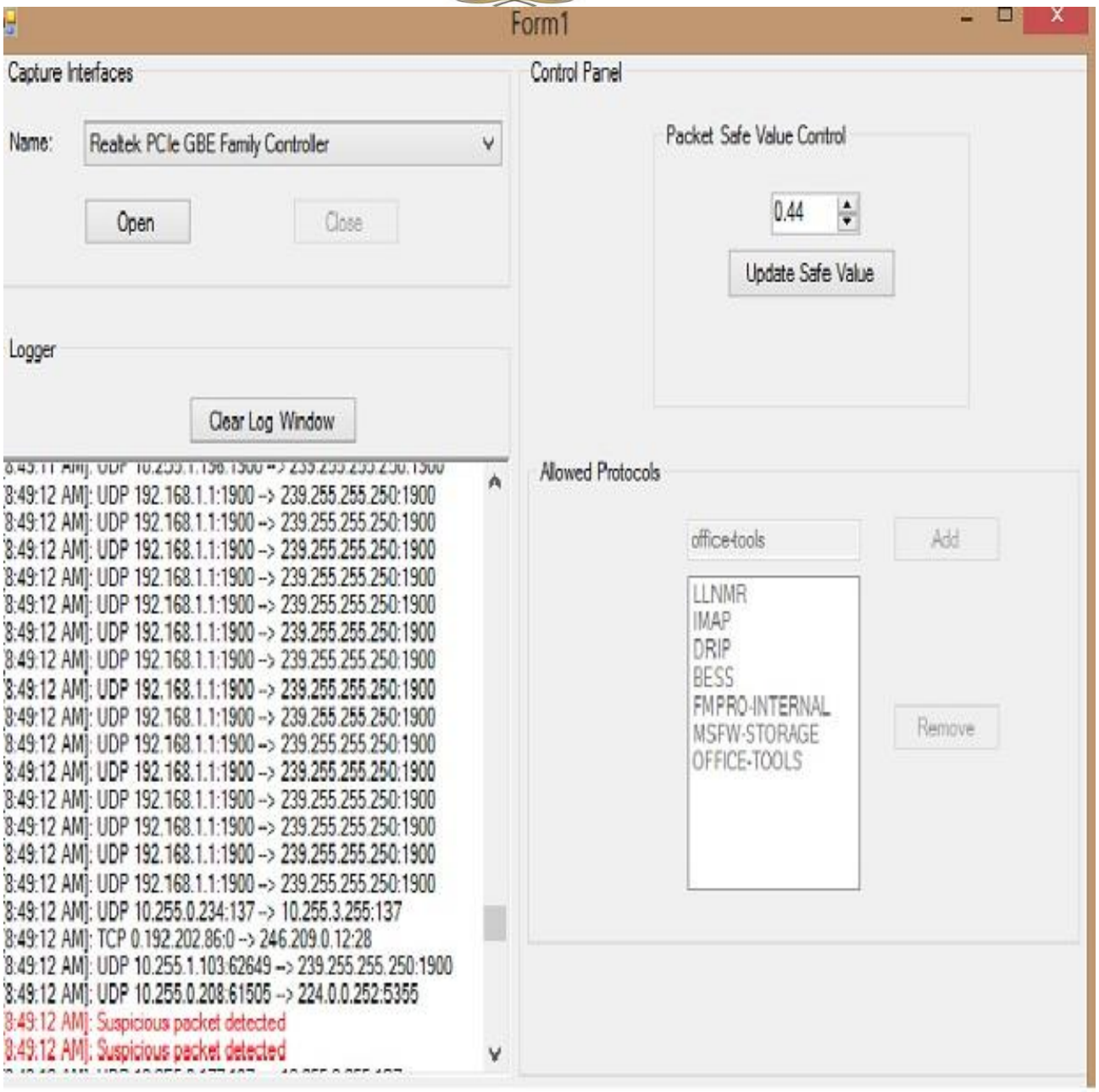


Figure 4.5: Improved IDS Interface for Update of Knowledge Base

4.5 Comparison of Initial Netflow Analyser Data before and after Knowledge Base Update

During the period of testing the model, and knowledge base update, the output of sniffers on the network was compared to the initial output that was used as training data for the model. It was discovered that the percentage of unknown traffic detected was consistently reducing as the knowledge base of model was updated and consequently the core router. The screenshot of Netflow during the validation period is as shown in Figure 4.6

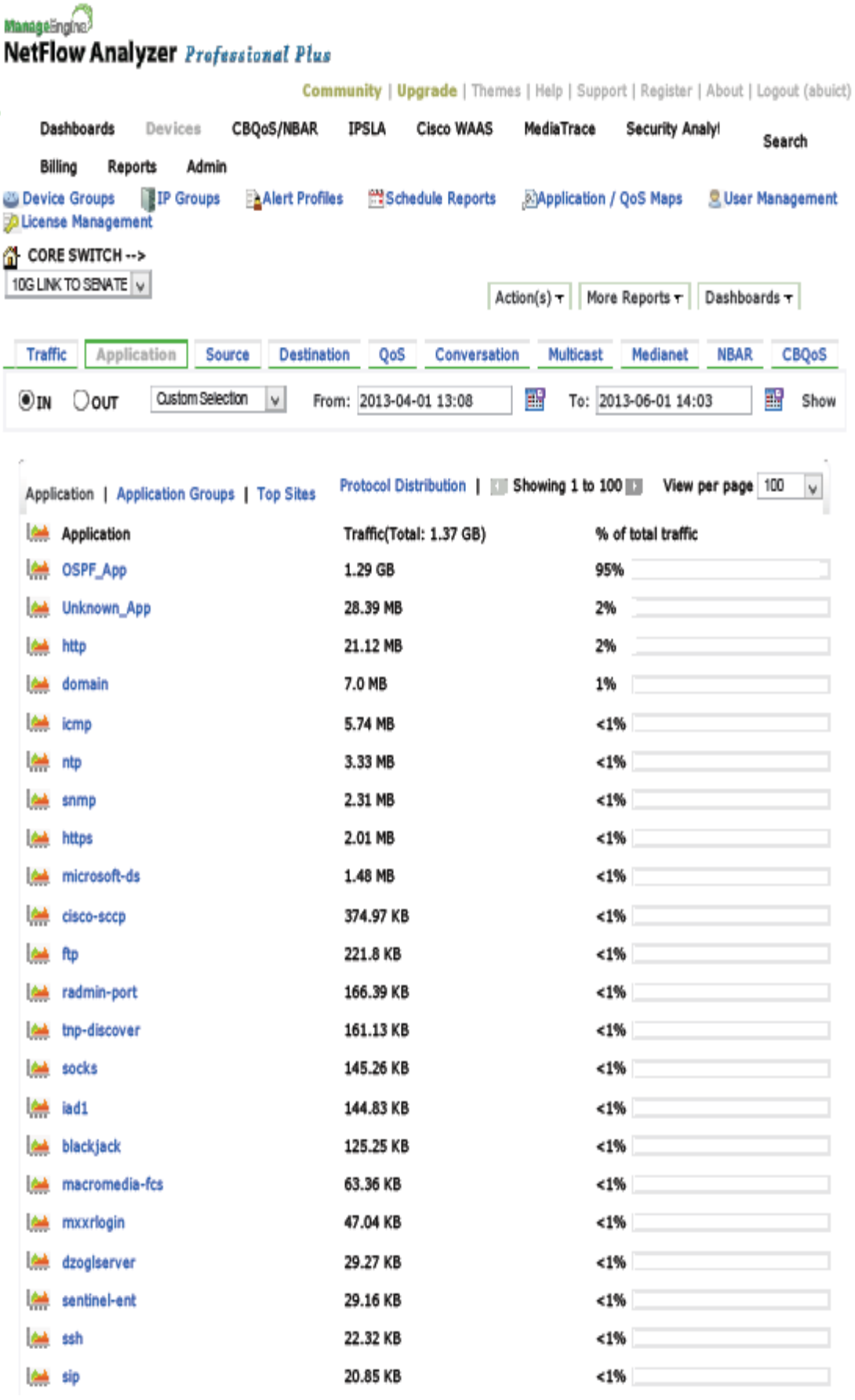

ManageEngine
NetFlow Analyzer Professional Plus
Community | Upgrade | Themes | Help | Support | Register | About | Logout (abuict)
Dashboards Devices CBQoS/NBAR IPSLA Cisco WAAS MediaTrace Security Analyl Search
Billing Reports Admin
Device Groups IP Groups Alert Profiles Schedule Reports Application / QoS Maps User Management
License Management
CORE SWITCH -->
10G LINK TO SENATE
Action(s) More Reports Dashboards
Traffic Application Source Destination QoS Conversation Multicast Medianet NBAR CBQoS
IN OUT Custom Selection From: 2013-04-01 13:08 To: 2013-06-01 14:03 Show
Application | Application Groups | Top Sites Protocol Distribution | Showing 1 to 100 View per page 100
Application Traffic(Total: 1.37 GB) % of total traffic
OSPF_App 1.29 GB 95%
Unknown_App 28.39 MB 2%
http 21.12 MB 2%
domain 7.0 MB 1%
icmp 5.74 MB <1%
ntp 3.33 MB <1%
snmp 2.31 MB <1%
https 2.01 MB <1%
microsoft-ds 1.48 MB <1%
cisco-sccp 374.97 KB <1%
ftp 221.8 KB <1%
radmin-port 166.39 KB <1%
tnp-discover 161.13 KB <1%
socks 145.26 KB <1%
iad1 144.83 KB <1%
blackjack 125.25 KB <1%
macromedia-fcs 63.36 KB <1%
mxxrlogin 47.04 KB <1%
dzogiserver 29.27 KB <1%
sentinel-ent 29.16 KB <1%
ssh 22.32 KB <1%
sip 20.85 KB <1%

Figure 4.6: **Screenshot of Netflow Analyser during Model Validation Period**

The traffic summary from Netflow and the graphical representation are as shown in tables 4.2 and Figure 4.7 respectively

Table 4.3: **Traffic Summary from Netflow Analyser during Model Validation Period**

Application	Traffic	% of Total Traffic
OSPF	1.29 GB	95%
Unknown	28.39 MB	2%
http	21.12MB	2%
ICMP	5.74MB	1%
Ntp	3.33MB	<1%
dzoglserver	29.27 KB	<1%
Others		<1%

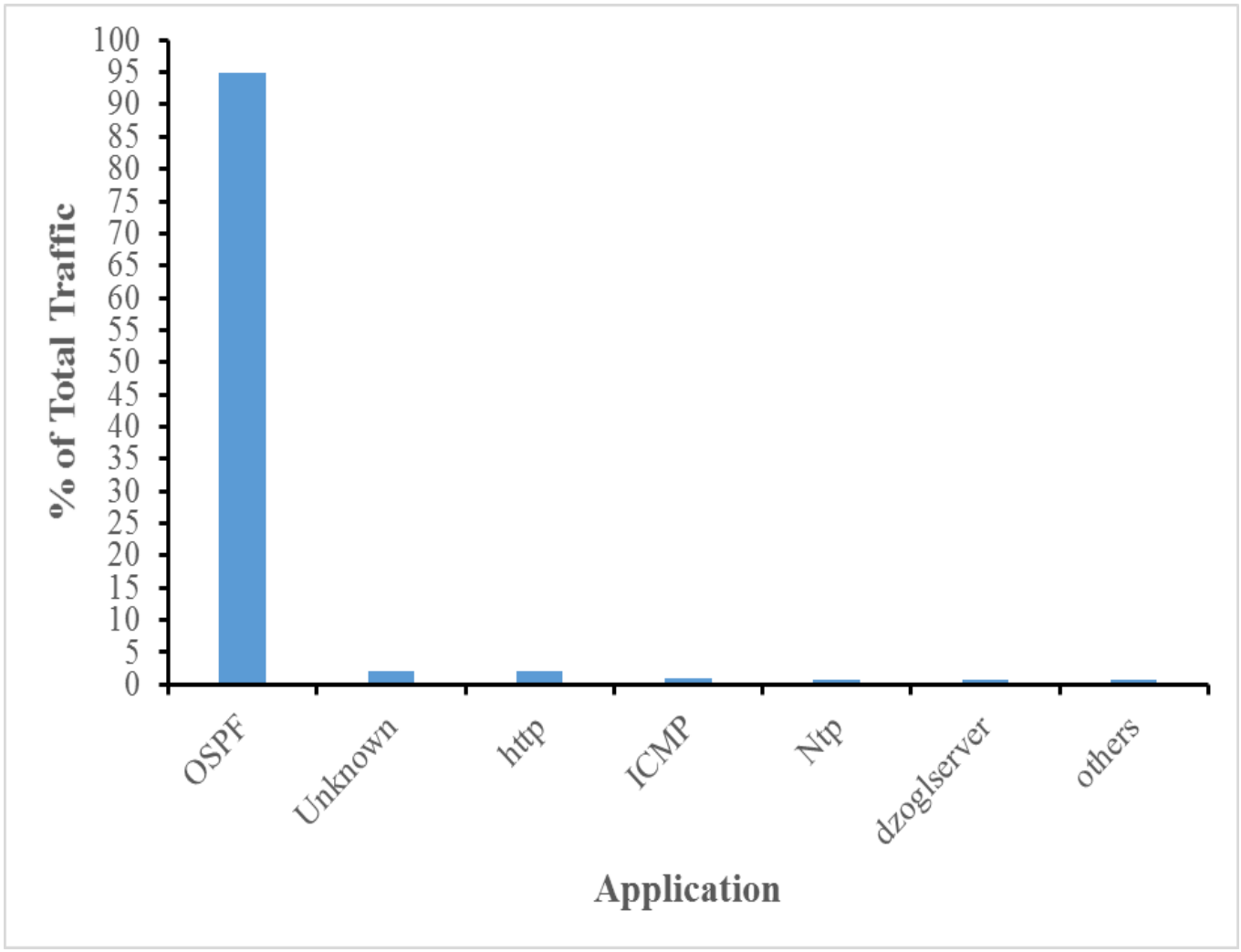


Figure 4.7: **Traffic Summary from Netflow Analyser during Model Validation Period**

4.6 **Observation of Model Performance during Validation Period.**

During the validation period, the model was monitored during peak usage period to ensure that the maximum number of packets possible on the network were captured and processed by the model. Hence the important features of the system in-use were captured severally to arrive at the minimum and maximum possible resources the model will need for optimal performance.

The following important features and resources were observed.

i. CPU usage;
ii. Memory Usage; and
iii. Disk Usage.

It was discovered that the model usage of CPU at peak period on the network was varying between 7 and 15.5 % as shown in Figure 4.8. But it never went beyond 15.5% CPU usage.

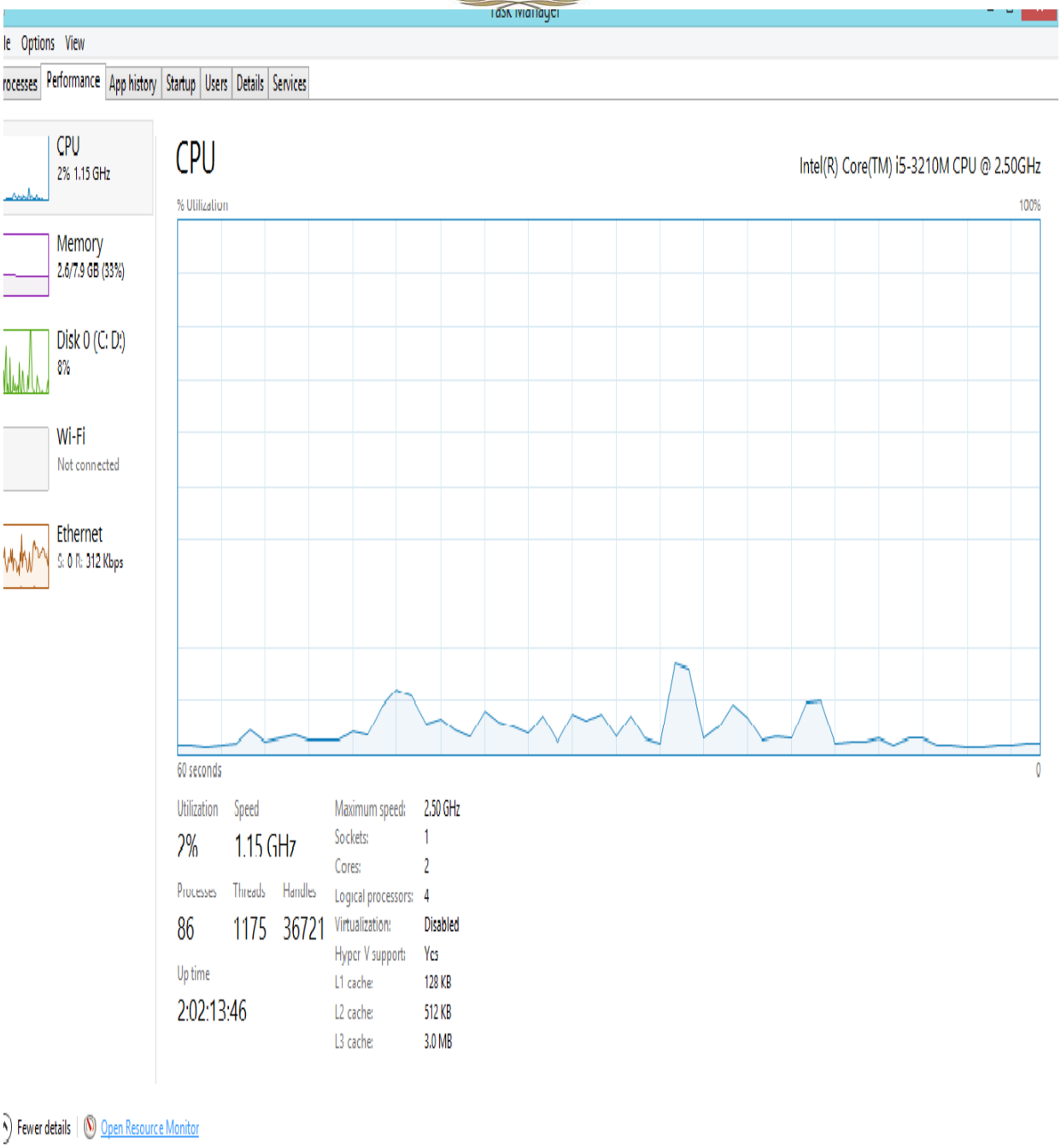


Figure 4.8: **Screenshot of System Resource Usage During Validation**

CHAPTER FIVE

CONCLUSION AND SUGGESTION FOR FURTHER WORK

This section presents the conclusion of this research work and suggestions for further work.

5.1 Introduction

Agent technology was applied in this research work to develop an intrusion detection model for internally generated inturusion in a multi-disciplinary campus network. The IDS model used data collected over a 24 month period as a knowledge base for applications and protocols. The model also used the behavioural pattern of packets by calculating the overhead of each packet on the network and comparing to the maximum allowable overhead of an Ethernet frame which was set as threshold and refered to as the Safe Score.

5.2 Significance of this Research

The major contributions to knowledge of this research include the following:

i) Development of an agent-based IDS using hybrid of knowledge-based and behavioural based detection techniques implemented as a program developed using c# programming language.

ii) Development of an IDS that is simple and adaptable to any multi-disciplinary University campus and capable of detecting unwanted or malicious traffic even from legitimate users.

iii) The IDS has the ability of sniffing data on its own in real time without depending on a third party, it is also capable of updating its knowledge base.

iv) The IDS developed has a very low system memory and CPU utilization at peak period was less than 17%.

5.3 Limitations

i. The training data collected from sniffers over a period of two years was running into several hundred of pages. This made the partitioning stage of the work very cumbersome.

ii. The IDS developed did not incorporate an automated analytical tool for processing the logged packets.

5.4 Conclusion

The development of the model, which is an agent-based hybrid model utilizing the behavioural pattern of network traffic to set the safe score metrics for the model and the knowledge base determined by defined applications and protocols, is presented in this work. The model was developed using a 24 month data collected using various network sniffers like Ntop, Netflow etc. However, the eventual IDS is capable of listening and capturing its own data without depending on any sniffer. It is also capable of reducing overhead on the system by ensuring that a worker (or agent) is assigned to only one packet and more workers are automatically introduced by the system as new packets are discovered. For the IDS to be deployed elsewhere few parameters are to be changed on the network and these are basically the trusted IP range from where data should be captured.

5.5 Suggestion for Further Research

i. The developed IDS model can further be enhanced by exploring the possibility of automating the analysis of logged packets for the detection of intrusive or malicious packets.

ii. The method of developing an IDS model using an agent based technology can also be explored in developing metrics for analysis of intrusive packets.

iii. An Intrusion prevention system may be developed using the properties of intrusive packets as its knowledge base, update of such system should be possible and easy to achieve.

iv. Profilling the behaviour of users on a multi-disciplinary campus network may be explored with the aim of developing a user behavioural predictive system.

REFERENCES

Abdelaziz Mounji (1997) "Languages and Tools for Rule-Based Distributed Intrusion Detection", Doctor of Science, Faculties Universitaires Notre-Dame de la Paix, Namur (Belgium). http://asax.info.ucl.ac.be/papers/mounji_phd.pdf.Retrived on July 11th 2010

Anderson, T. A. Avizienis, W.C. Carter, A. Costes, F. Cristian, Y. Koga, H. Kopetz, J.H. Lala, J.C.Laprie, J.F. Meyer, B. Randell, A.S. Robinson, L. Simonici, and U. Voges (1992), "Dependability: Basic Concepts and Terminology", Dependable Computing and Fault Tolerance. Springer Verlag,

Anderson, J.P. (1980). "Computer Security Threat Monitoring and Surveillance", http://seclab.cs.ucdavis.edu/projects/history/papers/ande80.pdf retrived July 12 2010

Attig, M., Lockwood, J (2005), "SIFT: snort intrusion filter for TCP," High Performance Interconnects, 2005.Proceedings. 13th Symposium on, vol., no., pp. 121- 127, 17-19 Aug.2005
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1544587&isnumber=32970 Retrived June 26th 2010

Corliss, J and El- Sheikh, E.M (2010), "An intelligent System for Network Traffic Flow Analysis", International Conference of Artificial Intelligence Pp 716-722.

Denning, D. E. (1987), "An Intrusion Detection Model", IEEE Transaction on Software Engineering, Vol. SE13, No.2, pp. 222-232,

Denning, D. E. and Neumann, P. G. (1985), "Requirements and model for IDES-A real-time intrusion detection system," Comput. Sci. Lab, SRI International, Menlo Park, CA, Tech. Rep.

Douligeris,C and Serpanos, D. N. (2007) Network Security: Current Status and Future Directions, Institute of Electrical and Electronics Engineers, Inc. Pp 5-9

Dreger, H. and Feldmann, A. (2008) "Predicting the Rosource Consumption of network intrusion Detection Systems", SIGMERTRICS Annapolis Maryland USA.June 2-6 2008

Fairhurst, G. (2008) Local Area Networkshttp://www.erg.abdn.ac.uk/~gorry/eg3567 retrived April 23rd 2011

Gandhi, M. and Srivatsa, S.K (2010), "Detecting and Preventing Attacks Using Network Intrusion Detection Systems" International Journal of Computer Science and Security, Vol. (2) Issue (1) Pp 49-60.

Hari, O.M. and Tapas, K. S. (2010), "Designing Intrusion detection System for Web Documents Using Neural Network", IEE Communication and Network, Pp 54-61.

He, Y. and Zhang, Y. (2010), "Research of campus network security system based on Intrusion Detection" International Conference on Computer Design and Applications (ICCDA) ppV4-618-V4-621

Herve, D. (2000), "An introduction to Intrusion Detection System", IBM Research Zurich Research Laboratory. **http://bit.kuas.edu.tw/~csshieh/teach/np/tcpip/ Retrived April 16th 2011**

http://infohost.nmt.edu/~sfs/Students/HarleyKozushko/Papers/IntrusionDetectionPaper.pdf Retrived: 20th August 2011

http://www.cs.ubc.ca/~krasic/cse585/ilgun92ustat.pdf Retrived: 30th June 2012

International Journal of Computer and Telecommunications Networking (1999) Volume 31 Issue 23- 24, Dec.1999 Pp 2435-2463

International Organization for Standardization (ISO), Information Processing Systems—Open Systems Interconnection (1994) "Part 1: Basic Reference Model", ISO/IEC 7498-1, ISO, 1984, also ISO/OSI 7498-1, Geneva, 1994.

International Organization for Standardization (ISO), Information Processing Systems—Open Systems Interconnection (1989) "Part 2: Security Architecture", ISO/IEC 7498-2, ISO, Geneva, 1989.

International Organization for Standardization (ISO), Information Processing Systems—Open Systems Interconnection(1989), "Part 1: Security Frameworks for Open Systems: Overview", ISO/IEC 10181-1, ISO, Geneva, 1996.

International Organization for Standardization (ISO), Information Processing Systems—Open Systems Interconnection(1996), "Part 2: Security Frameworks for Open Systems: Authentication Framework", ISO/IEC 10181-2, ISO, Geneva, 1996.

International Organization for Standardization (ISO), Information Processing Systems—Open Systems Interconnection (1996), "Part 3: Security Frameworks for Open Systems: Access Control Framework", ISO/IEC 10181-3, ISO, Geneva, 1996.

International Organization for Standardization (ISO), Information Processing Systems—Open Systems Interconnection(1996), "Part 4: Security Frameworks for Open Systems: Non-Repudiation Framework", ISO/IEC 10181-4, ISO, Geneva, 1996.

International Organization for Standardization (ISO), Information Processing Systems—Open Systems Interconnection (1996), **"**Part 5: Security Frameworks for Open Systems: Confidentiality Framework", ISO/IEC 10181-5, ISO, Geneva, 1996.

International Organization for Standardization (ISO), Information Processing Systems—Open Systems Interconnection (1996), **"**Part 6: Security Frameworks for Open Systems: Integrity Framework", ISO/IEC 10181-6, ISO, Geneva, 1996.

International Organization for Standardization (ISO), Information Processing Systems—Open Systems Interconnection (1996), "Part 7: Security Frameworks for Open Systems: Security Audit and Alarms Framework", ISO/IEC 10181-7, ISO, Geneva, 1996.

International Organization for Standardization (ISO), Information Technology—Open Systems Interconnection(1996), **"**Generic Upper Layers Security: Overview, Models and Notation", ISO/IEC 11586-1, ISO, Geneva, 1996.

International Organization for Standardization (ISO), Information Technology—Open Systems Interconnection(1996), **"**Generic Upper Layers Security: Exchange Service Element (SESE) Service Definition", ISO/IEC 11586-2, ISO, Geneva, 1996

International Telecommunication Union (ITU) (1991), "Security Architecture for Open Systems Interconnection for CCIT Applications", Recommendation ITU-T X.800, ITU, Geneva, 1991.

International Telecommunication Union (ITU) , (Information Technology—Open Systems Interconnection (1994), "Basic Reference Model: The Basic Model", Recommendation ITU-T X.200, ITU, Geneva, 1994.

International Telecommunication Union (ITU), Information Technology—Open Systems Interconnection(1995), **, "**Lower Layers Security Model", Recommendation ITU-T X.802, ITU, Geneva, 1995.

International Telecommunication Union (ITU), Information Technology—Open Systems Interconnection (1994), "Upper Layers Security Model", Recommendation ITU-T X.802, ITU, Geneva, 1994.

International Telecommunication Union (ITU), Information Technology—Open Systems Interconnection(1995), **"**Security Frameworks for Open Systems Overview", Recommendation ITU-T X.810, ITU, Geneva, 1995.

International Telecommunication Union (ITU), Information Technology—Open Systems Interconnection(1995), "Security Frameworks for Open Systems Authentication Framework", Recommendation ITU-T X.811, ITU, Geneva, 1995.

International Telecommunication Union (ITU), Information Technology—Open Systems Interconnection (1995), "Security Frameworks for Open Systems: Access Control Framework", Recommendation ITU-T X.812, ITU, Geneva, 1995.

International Telecommunication Union (ITU), Information Technology—Open Systems Interconnection (1996), "Security Frameworks for Open Systems: Non-Repudiation Framework", Recommendation ITU-T X.813, ITU, Geneva, 1996.

International Telecommunication Union (ITU), Information Technology—Open Systems Interconnection (1995), "Security Frameworks for Open Systems: Confidentiality Framework", Recommendation ITU-T X.814, ITU, Geneva, 1995.

International Telecommunication Union (ITU), Information Technology—Open Systems Interconnection (1995), "Security Frameworks for Open Systems: Integrity Framework", Recommendation ITU-T X.815, ITU, Geneva, 1995.

International Telecommunication Union (ITU), Information Technology—Open Systems Interconnection (1995), "Security Frameworks for Open Systems: Security Audit and Alarms Framework", Recommendation ITU-T X.815, ITU, Geneva, 1995.

Internetwork Technology Overview (1999) Second Edition. Cisco Press/Techmedia, New Delhi, 1999

Ismail, M.N. and Ismail, M.T (2009), "Framework of Intrusion Detection System via Snort Application on Campus Network Environment", International Conference on Future Computer and Communication, ICFCC pp 455-459.

Jagannathan, R.; Lunt, T.; Anderson, D.; Dodd, C.;Gilham,F,; Caveh,J.; Hal Javitz, Neumann,F.;Tamaru, A.; and Valdes, A.(1993). "System Design Document: Next-Generation Intrusion Detection Expert System (NIDES)", Technical Report A007/A008/A009/A011/A012/A014, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025.

Khan, A. and Sadiq, M. (2011) "Rule based Network Intrusion Detection Using Generic Algorithm" International Journal of Computer Applications Vol. 18 Pp 26-29

Kim I.S and Kim M.H (2012), "Agent-based Honeynet Framework for Protecting Servers in Campus Network" Institute of Engineering Technology Information Security 2012, Vol.6 Issue 3 pp 202-211

Kim, G.H and Spafford. E (1994), "Experiences with Tripwire: Using Integrity Checkers for Intrusion Detection", Purdue Technical Report CSD-TR-94-012.

Kozushko, H. (2003), "Intrusion Detection: Host Based and Network Intrusion Systems", an independent Study.

Lammle T. (2005), Cisco Certified Network Associate Study Guide 5th Edition.Sybex inc USA Pp 42-45

Lammle T. H., Gihan V., Dias, K.N., Levitt, Biswanath Mukherjee, Jef Wood, and David, W. (1990). "A Network Security Monitor". Proceedings of the IEEE Symposium on Research in Security and Privacy, pp 296-304, Oakland, CA, May 1990. IEEE Computer Society Press, Los Alamitos, CA.

Leon-Garcia A, Widjaja I (2000), "Communication Networks: Fundamental Concepts and Key Architechures", Mcgraw-Hill Inc (2000)

Lu Yao, B. and Chen Z. (2010), "A Network Intrusion Detection System with the Snooping Agents" 2010 IEEE International Conference on Computer Application and System Modeling (ICCASM 2010)

Magnus, A., Herve D., and Marc, D.(2000), A lightweight tool for detecting web server attacks Proceedings of Network and Distributed System Security Symposium, Pp 157-170, San Diego, CA.USA

Maselli, G. L. and Stefano, S. (2001) "Design and Implementation of an Anomaly Detection System: An Empirical Approach" University of Pisa, Italy, http://luca.ntop.org/ADS.pdf. Retrived 13th June 2011

Object Management Group (OMG) (2000)' "Agent Technology: Green Paper, OMG Document ec/2000-08-01"

Porras, P. (1992), "A State Transition Analysis Tool for Intrusion Detection". Master's Thesis, University of California at Santa Barbara. URL:www.cs.ubc.**ca**/~krasic/cse585/ilgun92ustat.pdf Retrived 14th July 2012

Rehman, R. (2003), "Intrusion Detection Systems with Snort Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID" Pearson Education,

Sandeep, K. and Eugene, S. (1994), "A Pattern Matching Model for Misuse Intrusion Detection", In Proceedings of the 17th National Computer Security Conference, pp.11-21.

Sempf, B., Sphar C. and Davis S.R (2010). "C# 2010 all-in-one for dummies. Wiley Publishing, Inc., Indianapolis, Indiana USA

Sobirey, M. (1998), "Intrusion detection system bibliography", www.rnks.informatik.tucottbus.de/sobirey/ids.html. Retrived: 14th July 2012

Stallings W. (2011a) "Cryptography and Network Security: Principles and Practice", 5th Edition, pp 20-8

Stallings W. (2011b) "Network Security Essentials Applications and Standards", 4th Edition, pp 312-313

Stephane, G. (1998), "Design and Implementation of a Snifer Detector", In Proceedings of RAID 98,Workshop on Recent Advances in Intrusion Detection, Louvain-la-Neuve, Belgium.

Vern, P. (1999), "BRO: A System for Detecting Network Intruders in Real-Time",

Wang. B., Feng, L. and Shaomin, Z. (2009), "Research on Intrusion Detection Based on Campus Network", Intelligent Information Technology Application, IITA 2009 pp 468-471

Xiong C.H and Peng Z (2012), "Applied Research on Snort Intrusion Detection Model in Campus Network" IEE 2012 Symosium on Robotics and Application pp 596-599.

APPENDIX I

List of Defined (Known) Applications and Protocols

S/No	Applications	Traffic (Total: 4.41 GB)	% of Total Traffic
1	OSPF_App	4.15 GB	94%
2	domain	36.89 MB	1%
3	http	28.71 MB	1%
4	tftp	13.3 MB	<1%
5	snmp	12.98 MB	<1%
6	sip	8.4 MB	<1%
7	ntp	8.31 MB	<1%
8	https	5.1 MB	<1%
9	icmp	5.02 MB	<1%
10	ssh	4.38 MB	<1%
11	cisco-sccp	2.74 MB	<1%
12	http-alt	1.5 MB	<1%
13	elpro_tunnel	1.25 MB	<1%
14	microsoft-ds	1.19 MB	<1%
15	macromedia-fcs	522.15 KB	<1%
16	icslap	481.35 KB	<1%
17	dzoglserver	163.32 KB	<1%
18	teredo	160.74 KB	<1%
19	telnet	138.28 KB	<1%
20	bmcpatrolrnvu	130.61 KB	<1%
21	ardus-trns	112.54 KB	<1%
22	iad1	76.8 KB	<1%
23	ddi-tcp-1	73.95 KB	<1%
24	ssdp	43.51 KB	<1%
25	blackjack	34.85 KB	<1%
26	wfremotertm	31.53 KB	<1%
27	cumulus	26.58 KB	<1%
28	netbios-ns	20.33 KB	<1%

29	epmd	16.32 KB	<1%
30	wxbrief	16.32 KB	<1%
31	ddi-tcp-2	13.97 KB	<1%
32	xmpp-client	12.31 KB	<1%
33	nacnl	11.82 KB	<1%
34	asprovatalk	7.92 KB	<1%
35	igi-lm	7.71 KB	<1%
36	ivcollector	4.81 KB	<1%
37	smtp	4.51 KB	<1%
38	houston	4.27 KB	<1%
39	echo	3.42 KB	<1%
40	uacs	3.42 KB	<1%
41	socks	3.0 KB	<1%
42	guibase	2.89 KB	<1%
43	snmptrap	2.72 KB	<1%
44	Epp	2.63 KB	<1%
45	availant-mgr	2.6 KB	<1%
46	cichild-lm	2.6 KB	<1%
47	es-elmd	2.55 KB	<1%
48	cddbp-alt	2.38 KB	<1%
49	beacon-port	2.24 KB	<1%
50	qadmifevent	2.18 KB	<1%
51	signet-ctf	2.01 KB	<1%
52	editbench	1.89 KB	<1%
53	gmrupdateserv	1.86 KB	<1%
54	vrts-auth-port	1.86 KB	<1%
55	fj-hdnet	1.82 KB	<1%
56	ariliamulti	1.79 KB	<1%
57	vpa-disc	1.68 KB	<1%
58	canon-bjnp1	1.64 KB	<1%
59	fhsp	1.64 KB	<1%
60	tbrpf	1.62 KB	<1%
61	sddp	1.61 KB	<1%
62	dyna-access	1.58 KB	<1%
63	at-zis	1.29 KB	<1%
64	netbios-dgm	1.28 KB	<1%
65	eyetv	1.22 KB	<1%
66	acmsoda	1.1 KB	<1%
67	amiganetfs	1.08 KB	<1%
68	bspne-pcc	1.04 KB	<1%
69	sctp-tunneling	1.0 KB	<1%
70	sso-service	992.00 Bytes	<1%
71	rrac	952.00 Bytes	<1%
72	nsstp	940.00 Bytes	<1%
73	dwf	929.00 Bytes	<1%
74	3par-rcopy	925.00 Bytes	<1%
75	payrouter	867.00 Bytes	<1%
76	dpserveadmin	834.00 Bytes	<1%

77	ndmp	816.00 Bytes	<1%
78	passgo	811.00 Bytes	<1%
78	sntp-heartbeat	811.00 Bytes	<1%
90	psrserver	811.00 Bytes	<1%
91	genisar-port	768.00 Bytes	<1%
92	theta-lm	768.00 Bytes	<1%
92	iad2	768.00 Bytes	<1%
94	abcsoftware	768.00 Bytes	<1%
93	nettest	768.00 Bytes	<1%
96	optika-emedia	768.00 Bytes	<1%
97	infiniswitchcl	768.00 Bytes	<1%
98	abr-secure	768.00 Bytes	<1%
99	elan	768.00 Bytes	<1%
100	veritas-vis2	768.00 Bytes	<1%
101	nessus	736.00 Bytes	<1%
102	ams	712.00 Bytes	<1%
103	topflow	704.00 Bytes	<1%
104	rrilwm	704.00 Bytes	<1%
105	futrix	704.00 Bytes	<1%
106	ip-qsig	704.00 Bytes	<1%
107	hri-port	704.00 Bytes	<1%
108	csdm	704.00 Bytes	<1%
109	bpcp-trap	704.00 Bytes	<1%
110	dict	704.00 Bytes	<1%
111	advant-lm	704.00 Bytes	<1%
112	novell-ipx-cmd	704.00 Bytes	<1%
113	funk-logger	674.00 Bytes	<1%
114	cplscrambler-lg	671.00 Bytes	<1%
115	unet	644.00 Bytes	<1%
116	blaze	644.00 Bytes	<1%
117	brvread	637.00 Bytes	<1%
118	bmap	624.00 Bytes	<1%
119	sbook	613.00 Bytes	<1%
120	odn-castraq	610.00 Bytes	<1%
121	x11	607.00 Bytes	<1%
122	dfn	592.00 Bytes	<1%
123	kazaa	592.00 Bytes	<1%
124	bnetgame	574.00 Bytes	<1%
125	metricadbc	545.00 Bytes	<1%
126	ds-clnt	545.00 Bytes	<1%
127	vrtstrapserver	545.00 Bytes	<1%
128	instantia	540.00 Bytes	<1%
129	trim	540.00 Bytes	<1%
130	seclayer-tcp	536.00 Bytes	<1%
131	xdsxdm	532.00 Bytes	<1%
132	pop3	528.00 Bytes	<1%
133	ibm-ssd	528.00 Bytes	<1%
134	pop3s	528.00 Bytes	<1%

135	vrtp	515.00 Bytes	<1%
136	adobeserver-1	515.00 Bytes	<1%
137	audit-transfer	476.00 Bytes	<1%
138	isoft-p2p	472.00 Bytes	<1%
139	nmsigport	472.00 Bytes	<1%
140	rtmp-port	472.00 Bytes	<1%
141	amt-cnf-prot	472.00 Bytes	<1%
142	distinct	470.00 Bytes	<1%
143	avinstalldisc	456.00 Bytes	<1%
144	vlsi-lm	452.00 Bytes	<1%
145	chip-lm	446.00 Bytes	<1%
146	acr-nema	427.00 Bytes	<1%
147	nat-pmp	426.00 Bytes	<1%
148	fpitp	416.00 Bytes	<1%
149	veronica	413.00 Bytes	<1%
150	netarx	412.00 Bytes	<1%
151	mtqp	412.00 Bytes	<1%
152	mxxrlogin	412.00 Bytes	<1%
153	pacerforum	402.00 Bytes	<1%
154	danf-ak2	401.00 Bytes	<1%
155	dynamid	393.00 Bytes	<1%
156	opennl-voice	391.00 Bytes	<1%
157	raventbs	384.00 Bytes	<1%
158	sun-as-iiops-ca	384.00 Bytes	<1%
159	acptsys	384.00 Bytes	<1%
160	avenyo	384.00 Bytes	<1%
161	concomp1	384.00 Bytes	<1%
162	umm-port	384.00 Bytes	<1%
163	rdlap	384.00 Bytes	<1%
164	naap	384.00 Bytes	<1%
165	wfm	384.00 Bytes	<1%
166	bv-smcsrv	384.00 Bytes	<1%
167	estamp	384.00 Bytes	<1%
168	nds_sso	384.00 Bytes	<1%
169	mmcals	384.00 Bytes	<1%
170	kfxaclicensing	384.00 Bytes	<1%
171	equationbuilder	384.00 Bytes	<1%
172	sftsrv	384.00 Bytes	<1%
173	fryeserv	384.00 Bytes	<1%
174	ept-machine	384.00 Bytes	<1%
175	start-network	384.00 Bytes	<1%
176	vytalvaultpipe	384.00 Bytes	<1%
177	listmgr-port	384.00 Bytes	<1%
178	mt-scaleserver	384.00 Bytes	<1%
179	serverview-rm	384.00 Bytes	<1%
180	abr-api	384.00 Bytes	<1%
181	iph-policy-adm	384.00 Bytes	<1%
182	rib-slm	384.00 Bytes	<1%

183	asoki-sma	384.00 Bytes	<1%
184	ice-location	384.00 Bytes	<1%
185	glogger	384.00 Bytes	<1%
186	nati-svrloc	384.00 Bytes	<1%
187	apwi-rxserver	384.00 Bytes	<1%
188	ltctcp	384.00 Bytes	<1%
189	tcpdataserver	384.00 Bytes	<1%
190	oi-2000	384.00 Bytes	<1%
191	skytelnet	384.00 Bytes	<1%
192	interhdl_elmd	384.00 Bytes	<1%
193	dvcprov-port	384.00 Bytes	<1%
194	nuts_bootp	384.00 Bytes	<1%
195	mcs-mailsvr	384.00 Bytes	<1%
196	edix	384.00 Bytes	<1%
197	eq-office-4942	384.00 Bytes	<1%
198	caller9	384.00 Bytes	<1%
199	msfw-s-storage	384.00 Bytes	<1%
200	netbill-prod	384.00 Bytes	<1%
201	jpegmpeg	384.00 Bytes	<1%
202	sysopt	384.00 Bytes	<1%
203	ncdmirroring	384.00 Bytes	<1%
204	udrive	384.00 Bytes	<1%
205	ttc-etap-ns	384.00 Bytes	<1%
206	sixxsconfig	384.00 Bytes	<1%
207	data-port	384.00 Bytes	<1%
208	femis	384.00 Bytes	<1%
209	d2k-datamover1	384.00 Bytes	<1%
210	commonspace	384.00 Bytes	<1%
211	bv-agent	384.00 Bytes	<1%
212	nm-assessor	384.00 Bytes	<1%
213	drp	384.00 Bytes	<1%
214	orasrv	384.00 Bytes	<1%
215	udt_os	384.00 Bytes	<1%
216	gw-call-port	384.00 Bytes	<1%
217	fjicl-tep-c	384.00 Bytes	<1%
218	myrtle	384.00 Bytes	<1%
219	idmaps	384.00 Bytes	<1%
220	c-h-it-port	384.00 Bytes	<1%
221	videobeans	384.00 Bytes	<1%
222	gdbremote	384.00 Bytes	<1%
223	nbx-ser	384.00 Bytes	<1%
224	simple-push	384.00 Bytes	<1%
225	udpradio	384.00 Bytes	<1%
226	topx	384.00 Bytes	<1%
227	ace-server	384.00 Bytes	<1%
228	boinc-client	384.00 Bytes	<1%
229	rfio	384.00 Bytes	<1%
230	resource_mgr	384.00 Bytes	<1%

231	surveyinst	384.00 Bytes	<1%
232	rrirtr	384.00 Bytes	<1%
233	f5-iquery	384.00 Bytes	<1%
234	hillrserv	384.00 Bytes	<1%
235	whosockami	384.00 Bytes	<1%
236	pctrader	384.00 Bytes	<1%
237	dssiapi	384.00 Bytes	<1%
238	cimtrak	384.00 Bytes	<1%
239	a14	384.00 Bytes	<1%
240	bruce	384.00 Bytes	<1%
241	remote-collab	384.00 Bytes	<1%
242	vipremoteagent	384.00 Bytes	<1%
243	net8-cman	384.00 Bytes	<1%
244	cdbroker	384.00 Bytes	<1%
245	vytalvaultbrtp	384.00 Bytes	<1%
246	donnyworld	384.00 Bytes	<1%
247	sm-pas-5	384.00 Bytes	<1%
248	srvc_registry	384.00 Bytes	<1%
249	netobjects2	384.00 Bytes	<1%
250	faxportwinport	384.00 Bytes	<1%
251	netview-aix-4	384.00 Bytes	<1%
252	msft-gc-ssl	384.00 Bytes	<1%
253	cops	384.00 Bytes	<1%
254	slingshot	384.00 Bytes	<1%
255	facelink	384.00 Bytes	<1%
256	sds-admin	384.00 Bytes	<1%
257	bxp	384.00 Bytes	<1%
258	bmc-ea	384.00 Bytes	<1%
259	sdproxy	384.00 Bytes	<1%
260	wininstall-ipc	384.00 Bytes	<1%
261	valisys-lm	384.00 Bytes	<1%
262	t128-gateway	384.00 Bytes	<1%
263	univ-appserver	384.00 Bytes	<1%
264	netchat	384.00 Bytes	<1%
265	ds-mail	384.00 Bytes	<1%
266	galileolog	384.00 Bytes	<1%
267	cecsvc	384.00 Bytes	<1%
268	tsilb	384.00 Bytes	<1%
269	netadmin	384.00 Bytes	<1%
270	ncu-1	384.00 Bytes	<1%
271	telefinder	384.00 Bytes	<1%
272	timelot	384.00 Bytes	<1%
273	lnvpoller	384.00 Bytes	<1%
274	ctp	384.00 Bytes	<1%
275	arepa-raft	384.00 Bytes	<1%
276	itv-control	384.00 Bytes	<1%
277	novell-zen	384.00 Bytes	<1%
278	pmas	384.00 Bytes	<1%

279	isis-bcast	384.00 Bytes	<1%
280	LiebDevMgmt_C	384.00 Bytes	<1%
281	vocaltec-admin	384.00 Bytes	<1%
282	iascontrol-oms	384.00 Bytes	<1%
283	krb524	384.00 Bytes	<1%
284	timestenbroker	384.00 Bytes	<1%
285	wlanauth	384.00 Bytes	<1%
286	event_listener	384.00 Bytes	<1%
287	adaptecmgr	384.00 Bytes	<1%
288	wkstn-mon	384.00 Bytes	<1%
289	rtsserv	384.00 Bytes	<1%
290	itm-mcell-u	384.00 Bytes	<1%
291	brlp-3	384.00 Bytes	<1%
292	wimd	384.00 Bytes	<1%
293	as-debug	384.00 Bytes	<1%
294	ufastro-instr	384.00 Bytes	<1%
295	apocd	384.00 Bytes	<1%
296	mediaspace	384.00 Bytes	<1%
297	nucleus	384.00 Bytes	<1%
298	imyx	384.00 Bytes	<1%
299	csnotify	384.00 Bytes	<1%
300	cs-remote-db	384.00 Bytes	<1%
301	indura	384.00 Bytes	<1%
302	pn-requester	384.00 Bytes	<1%
303	pcptcpservice	384.00 Bytes	<1%
304	spw-dnspreload	384.00 Bytes	<1%
305	qsm-gui	384.00 Bytes	<1%
306	iclpv-nls	384.00 Bytes	<1%
307	bip	384.00 Bytes	<1%
308	appliance-cfg	384.00 Bytes	<1%
309	csbphonemaster	384.00 Bytes	<1%
310	fagordnc	384.00 Bytes	<1%
311	t1-e1-over-ip	384.00 Bytes	<1%
312	vpnz	384.00 Bytes	<1%
313	interact	384.00 Bytes	<1%
314	nei-management	384.00 Bytes	<1%
315	item	384.00 Bytes	<1%
316	pcihreq	384.00 Bytes	<1%
317	shockwave2	384.00 Bytes	<1%
318	amx-axbnet	384.00 Bytes	<1%
319	camac	382.00 Bytes	<1%
320	sns-quote	380.00 Bytes	<1%
321	fc-ser	380.00 Bytes	<1%
322	creativepartnr	380.00 Bytes	<1%
323	saphostctrl	364.00 Bytes	<1%
324	caspssl	364.00 Bytes	<1%
325	ansyslmd	364.00 Bytes	<1%
326	CBT_App	349.00 Bytes	<1%

327	pcsync-https	336.00 Bytes	<1%
328	emprise-lls	336.00 Bytes	<1%
329	mapper-nodemgr	320.00 Bytes	<1%
330	flamenco-proxy	320.00 Bytes	<1%
331	udp-sr-port	320.00 Bytes	<1%
332	orbplus-iiop	320.00 Bytes	<1%
333	3comnetman	320.00 Bytes	<1%
334	ncl	320.00 Bytes	<1%
335	cl-db-attach	320.00 Bytes	<1%
336	EtherNet/IP-1	320.00 Bytes	<1%
337	ni-visa-remote	320.00 Bytes	<1%
338	ns-cfg-server	320.00 Bytes	<1%
339	xnds	320.00 Bytes	<1%
340	dkmessenger	320.00 Bytes	<1%
341	fs-rh-srv	320.00 Bytes	<1%
342	service-ctrl	320.00 Bytes	<1%
343	finle-lm	320.00 Bytes	<1%
344	wiegand	320.00 Bytes	<1%
345	noaaport	320.00 Bytes	<1%
346	fdt-rcatp	320.00 Bytes	<1%
347	seagull-ais	320.00 Bytes	<1%
348	arkivio	320.00 Bytes	<1%
349	unisql	320.00 Bytes	<1%
350	jlicelmd	320.00 Bytes	<1%
351	sunclustermgr	320.00 Bytes	<1%
352	fujitsu-dtcns	320.00 Bytes	<1%
353	lipsinc1	320.00 Bytes	<1%
354	cadencecontrol	320.00 Bytes	<1%
355	zephyr-hm	320.00 Bytes	<1%
356	alias	320.00 Bytes	<1%
357	tns-adv	320.00 Bytes	<1%
358	slscc	320.00 Bytes	<1%
359	apollo-gms	320.00 Bytes	<1%
360	tsp	320.00 Bytes	<1%
361	ordinox-dbase	320.00 Bytes	<1%
362	seclayer-tls	320.00 Bytes	<1%
363	raw-serial	320.00 Bytes	<1%
364	iscsi-target	320.00 Bytes	<1%
365	gandalf-lm	320.00 Bytes	<1%
366	abbaccuray	320.00 Bytes	<1%
367	tolteces	320.00 Bytes	<1%
368	cisco-net-mgmt	320.00 Bytes	<1%
369	jaxer-manager	320.00 Bytes	<1%
370	prnrequest	320.00 Bytes	<1%
371	redwood-chat	320.00 Bytes	<1%
372	3Com-nsd	320.00 Bytes	<1%
373	g-talk	320.00 Bytes	<1%
374	cardbox	320.00 Bytes	<1%

375	tvdumtray-port	320.00 Bytes	<1%
376	twcss	320.00 Bytes	<1%
377	attachmate-s2s	320.00 Bytes	<1%
378	de-cache-query	320.00 Bytes	<1%
379	osdcp	320.00 Bytes	<1%
380	miroconnect	320.00 Bytes	<1%
381	dvt-data	320.00 Bytes	<1%
382	ms-sql-m	320.00 Bytes	<1%
383	troff	320.00 Bytes	<1%
384	ehs-ssl	320.00 Bytes	<1%
385	avocent-adsap	320.00 Bytes	<1%
386	dicom-tls	320.00 Bytes	<1%
387	pdp	320.00 Bytes	<1%
388	rcst	320.00 Bytes	<1%
389	optilogic	320.00 Bytes	<1%
390	anoto-rendezv	320.00 Bytes	<1%
391	ewinstaller	320.00 Bytes	<1%
392	discp-client	320.00 Bytes	<1%
393	polestar	320.00 Bytes	<1%
394	snifferclient	320.00 Bytes	<1%
395	global-dtserv	320.00 Bytes	<1%
396	ace-proxy	320.00 Bytes	<1%
397	tripe	320.00 Bytes	<1%
398	proshare-mc-1	320.00 Bytes	<1%
399	entp	320.00 Bytes	<1%
400	ft-role	320.00 Bytes	<1%
401	pluribus	320.00 Bytes	<1%
402	kjtsiteserver	320.00 Bytes	<1%
403	redstorm_find	320.00 Bytes	<1%
404	dbref	320.00 Bytes	<1%
405	epl-slp	320.00 Bytes	<1%
406	camp	320.00 Bytes	<1%
407	owserver	320.00 Bytes	<1%
408	rgtp	320.00 Bytes	<1%
409	cinegrfx-lm	320.00 Bytes	<1%
410	apple-sasl	320.00 Bytes	<1%
411	sophia-lm	320.00 Bytes	<1%
412	msfw-replica	320.00 Bytes	<1%
413	munin	320.00 Bytes	<1%
414	ccu-comm-3	320.00 Bytes	<1%
415	nhserver	320.00 Bytes	<1%
416	dic-aida	320.00 Bytes	<1%
417	mylxamport	320.00 Bytes	<1%
418	nmmp	320.00 Bytes	<1%
419	ttc	320.00 Bytes	<1%
420	hicp	320.00 Bytes	<1%
421	npsp	320.00 Bytes	<1%
422	ibm-pps	320.00 Bytes	<1%

423	csoftragent	320.00 Bytes	<1%
424	close-combat	320.00 Bytes	<1%
425	hydra	320.00 Bytes	<1%
426	chromagrafx	320.00 Bytes	<1%
427	menandmice-dns	320.00 Bytes	<1%
428	cajo-discovery	320.00 Bytes	<1%
429	ccu-comm-2	320.00 Bytes	<1%
430	murx	320.00 Bytes	<1%
431	stat-cc	320.00 Bytes	<1%
432	acp-discovery	320.00 Bytes	<1%
433	odi-port	320.00 Bytes	<1%
434	avenue	320.00 Bytes	<1%
435	qtp	320.00 Bytes	<1%
436	hp-webadmin	320.00 Bytes	<1%
437	vieo-fe	320.00 Bytes	<1%
438	remographlm	320.00 Bytes	<1%
439	pserver	320.00 Bytes	<1%
440	fc-cli	320.00 Bytes	<1%
441	orbix-locator	320.00 Bytes	<1%
442	lnvmailmon	320.00 Bytes	<1%
443	m3ua	320.00 Bytes	<1%
444	isoipsigport-2	320.00 Bytes	<1%
445	starfish	320.00 Bytes	<1%
446	tabula	320.00 Bytes	<1%
447	jetformpreview	320.00 Bytes	<1%
448	travsoft-ipx-t	320.00 Bytes	<1%
449	cadabra-lm	320.00 Bytes	<1%
450	nnsp	318.00 Bytes	<1%
451	documentum	316.00 Bytes	<1%
452	veracity	312.00 Bytes	<1%
453	spss-lm	312.00 Bytes	<1%
454	aplx	312.00 Bytes	<1%
455	ftp	308.00 Bytes	<1%
456	netviewdm3	305.00 Bytes	<1%
457	raven-rmp	304.00 Bytes	<1%
458	mynahautostart	304.00 Bytes	<1%
459	pm-cmdsvr	304.00 Bytes	<1%
460	netattachsdmp	304.00 Bytes	<1%
461	arcpd	304.00 Bytes	<1%
462	identify	304.00 Bytes	<1%
463	sip-tls	304.00 Bytes	<1%
464	ccss-qsm	304.00 Bytes	<1%
465	orbix-cfg-ssl	304.00 Bytes	<1%
466	dtp-dia	304.00 Bytes	<1%
467	onscreen	304.00 Bytes	<1%
468	raven-rdp	304.00 Bytes	<1%
469	tekpls	304.00 Bytes	<1%
470	cft-1	301.00 Bytes	<1%

471	netinfo-local	300.00 Bytes	<1%
472	ayiya	290.00 Bytes	<1%
473	at-nbp	290.00 Bytes	<1%
474	ns	290.00 Bytes	<1%
475	cbos-ip-port	290.00 Bytes	<1%
476	cmip-agent	290.00 Bytes	<1%
477	h323hostcallsc	288.00 Bytes	<1%
478	hpvmmdata	282.00 Bytes	<1%
479	s8-client-port	280.00 Bytes	<1%
480	spramsd	268.00 Bytes	<1%
481	s-net	268.00 Bytes	<1%
482	bl-idm	268.00 Bytes	<1%
483	mpfwsas	268.00 Bytes	<1%
484	wsdapi-s	268.00 Bytes	<1%
485	ovtopmd	268.00 Bytes	<1%
486	can-ferret	268.00 Bytes	<1%
487	ms-wbt-server	268.00 Bytes	<1%
488	iniserve-port	268.00 Bytes	<1%
489	jetform	268.00 Bytes	<1%
490	isi-irp	268.00 Bytes	<1%
491	ms-theater	268.00 Bytes	<1%
492	dwr	268.00 Bytes	<1%
493	apc-6548	268.00 Bytes	<1%
494	canon-capt	268.00 Bytes	<1%
495	synchronet-upd	264.00 Bytes	<1%
496	ddi-udp-6	262.00 Bytes	<1%
497	unihub-server	262.00 Bytes	<1%
498	saiscm	262.00 Bytes	<1%
499	priority-e-com	262.00 Bytes	<1%
500	MUX_App	261.00 Bytes	<1%
501	nmsd	260.00 Bytes	<1%
502	ivmanager	260.00 Bytes	<1%
503	commlinx-avl	260.00 Bytes	<1%
504	tgp	260.00 Bytes	<1%
505	omnivision	260.00 Bytes	<1%
506	sweetware-apps	260.00 Bytes	<1%
507	q55-pcc	260.00 Bytes	<1%
508	miva-mqs	260.00 Bytes	<1%
509	openvpn	260.00 Bytes	<1%
510	cadkey-licman	260.00 Bytes	<1%
511	cft-0	260.00 Bytes	<1%
512	nmasoverip	260.00 Bytes	<1%
513	dpi-proxy	260.00 Bytes	<1%
514	dbsyncarbiter	256.00 Bytes	<1%
515	nms	256.00 Bytes	<1%
516	mc-client	256.00 Bytes	<1%
517	netlabs-lm	256.00 Bytes	<1%
518	jamserverport	256.00 Bytes	<1%

519	wag-service	256.00 Bytes	<1%
520	radio	256.00 Bytes	<1%
521	activesync	256.00 Bytes	<1%
522	wrs_registry	256.00 Bytes	<1%
523	izm	256.00 Bytes	<1%
524	vat-control	256.00 Bytes	<1%
525	paragent	252.00 Bytes	<1%
526	scp-config	252.00 Bytes	<1%
527	vcrp	252.00 Bytes	<1%
528	prodigy-intrnet	252.00 Bytes	<1%
529	cpqrpm-agent	252.00 Bytes	<1%
530	ls3	252.00 Bytes	<1%
531	orbix-loc-ssl	252.00 Bytes	<1%
532	propel-msgsys	240.00 Bytes	<1%
533	excw	208.00 Bytes	<1%
534	casp	208.00 Bytes	<1%
535	x9-icue	208.00 Bytes	<1%
536	lupa	208.00 Bytes	<1%
537	rmpp	208.00 Bytes	<1%
538	visionpyramid	208.00 Bytes	<1%
539	neod2	208.00 Bytes	<1%
540	netbios-ssn	208.00 Bytes	<1%
541	isbconference2	208.00 Bytes	<1%
542	apdap	208.00 Bytes	<1%
543	groove-dpp	208.00 Bytes	<1%
544	tripwire	208.00 Bytes	<1%
545	cspmlockmgr	208.00 Bytes	<1%
546	emperion	208.00 Bytes	<1%
547	murray	208.00 Bytes	<1%
548	search-agent	208.00 Bytes	<1%
549	jt400-ssl	208.00 Bytes	<1%
550	hpss-ndapi	208.00 Bytes	<1%
551	servergraph	208.00 Bytes	<1%
552	IPV6_App	204.00 Bytes	<1%
553	xgrid	192.00 Bytes	<1%
554	cpqrpm-server	192.00 Bytes	<1%
555	navbuddy	192.00 Bytes	<1%
556	fiveacross	192.00 Bytes	<1%
557	dvl-activemail	192.00 Bytes	<1%
558	mmaeds	192.00 Bytes	<1%
559	agslb	192.00 Bytes	<1%
560	exlm-agent	190.00 Bytes	<1%
561	razor	190.00 Bytes	<1%
562	di-drm	190.00 Bytes	<1%
563	trap-daemon	190.00 Bytes	<1%
564	ars-master	190.00 Bytes	<1%
565	hello-port	190.00 Bytes	<1%
566	versa-tek	190.00 Bytes	<1%

567	multip-msg	190.00 Bytes	<1%
568	ctdp	190.00 Bytes	<1%
569	taskman-port	190.00 Bytes	<1%
570	iad3	182.00 Bytes	<1%
571	solid-mux	179.00 Bytes	<1%
572	cap	174.00 Bytes	<1%
573	dectalk	156.00 Bytes	<1%
574	encrypted_admin	156.00 Bytes	<1%
575	egs	156.00 Bytes	<1%
576	hhb-gateway	156.00 Bytes	<1%
577	mpshrsv	156.00 Bytes	<1%
578	vmrdp	156.00 Bytes	<1%
579	kwdb-commn	156.00 Bytes	<1%
580	neod1	156.00 Bytes	<1%
581	slinterbase	156.00 Bytes	<1%
582	dcutility	156.00 Bytes	<1%
583	atc-lm	156.00 Bytes	<1%
584	hpvmmagent	156.00 Bytes	<1%
585	afrog	156.00 Bytes	<1%
586	iden-ralp	156.00 Bytes	<1%
587	hpvmmcontrol	156.00 Bytes	<1%
588	kvm-via-ip	156.00 Bytes	<1%
589	scanstat-1	156.00 Bytes	<1%
590	mpc-lifenet	156.00 Bytes	<1%
591	autonoc	156.00 Bytes	<1%
592	dka	156.00 Bytes	<1%
593	amqps	152.00 Bytes	<1%
594	jvserver	152.00 Bytes	<1%
595	sabams	152.00 Bytes	<1%
596	jwclient	152.00 Bytes	<1%
597	fcp-addr-srvr1	152.00 Bytes	<1%
598	ibm3494	152.00 Bytes	<1%
599	jwserver	152.00 Bytes	<1%
600	svs-omagent	152.00 Bytes	<1%
601	dlms-cosem	152.00 Bytes	<1%
602	starbot	152.00 Bytes	<1%
603	oracle-vp2	152.00 Bytes	<1%
604	ftps	152.00 Bytes	<1%
605	repliweb	152.00 Bytes	<1%
606	sccip-media	152.00 Bytes	<1%
607	iclpv-nlc	152.00 Bytes	<1%
608	netbill-keyrep	152.00 Bytes	<1%
609	dashpas-port	152.00 Bytes	<1%
610	vnc-server	152.00 Bytes	<1%
611	jetcmeserver	152.00 Bytes	<1%
612	otlp	152.00 Bytes	<1%
613	jms	152.00 Bytes	<1%
614	idcp	152.00 Bytes	<1%

615	xaapi	152.00 Bytes	<1%
616	ipether232port	152.00 Bytes	<1%
617	xmapi	152.00 Bytes	<1%
618	tpcsrvr	152.00 Bytes	<1%
619	amqp	152.00 Bytes	<1%
620	ctt-broker	152.00 Bytes	<1%
621	softaudit	152.00 Bytes	<1%
622	centra	144.00 Bytes	<1%
623	ms-streaming	144.00 Bytes	<1%
624	swdtp-sv	134.00 Bytes	<1%
625	rvs-isdn-dcp	134.00 Bytes	<1%
626	esbroker	134.00 Bytes	<1%
627	comscm	134.00 Bytes	<1%
628	opsmgr	134.00 Bytes	<1%
629	aises	134.00 Bytes	<1%
630	essbase	134.00 Bytes	<1%
631	a4-sdunode	134.00 Bytes	<1%
632	ups-onlinet	134.00 Bytes	<1%
633	ischat	134.00 Bytes	<1%
634	veritas-udp1	134.00 Bytes	<1%
635	fiorano-msgsvc	134.00 Bytes	<1%
636	routematch	134.00 Bytes	<1%
637	prat	134.00 Bytes	<1%
638	tmi	134.00 Bytes	<1%
639	gv-us	134.00 Bytes	<1%
640	ibm-res	134.00 Bytes	<1%
641	extensis	134.00 Bytes	<1%
642	ansoft-lm-2	134.00 Bytes	<1%
643	f5-globalsite	134.00 Bytes	<1%
644	videte-cipc	131.00 Bytes	<1%
645	1ci-smcs	131.00 Bytes	<1%
646	lpsrecommender	131.00 Bytes	<1%
647	ebinsite	131.00 Bytes	<1%
648	git	131.00 Bytes	<1%
649	productinfo	131.00 Bytes	<1%
650	saratoga	131.00 Bytes	<1%
651	capfast-lmd	131.00 Bytes	<1%
652	imgames	131.00 Bytes	<1%
653	imqtunnel	131.00 Bytes	<1%
654	netview-aix-10	131.00 Bytes	<1%
655	ehtp	131.00 Bytes	<1%
656	vqp	131.00 Bytes	<1%
657	winddlb	131.00 Bytes	<1%
658	acp-conduit	131.00 Bytes	<1%
659	ardus-mtrns	131.00 Bytes	<1%
660	x500ms	131.00 Bytes	<1%
661	nvcnet	131.00 Bytes	<1%
662	umsp	131.00 Bytes	<1%

663	quosa	131.00 Bytes	<1%
664	descent3	129.00 Bytes	<1%
665	netview-aix-5	129.00 Bytes	<1%
666	nimreg	129.00 Bytes	<1%
667	vfo	129.00 Bytes	<1%
668	vantronix-mgmt	129.00 Bytes	<1%
669	osu-nms	128.00 Bytes	<1%
670	pxc-roid	126.00 Bytes	<1%
671	mpnjsomg	126.00 Bytes	<1%
672	funkproxy	126.00 Bytes	<1%
673	sntlkeyssrvr	126.00 Bytes	<1%
674	digital-notary	126.00 Bytes	<1%
675	sscan	126.00 Bytes	<1%
676	boomerang	126.00 Bytes	<1%
677	3com-njack-2	126.00 Bytes	<1%
678	documentum_s	126.00 Bytes	<1%
679	caicci	126.00 Bytes	<1%
680	ff-fms	126.00 Bytes	<1%
681	ccs-software	126.00 Bytes	<1%
682	talon-disc	126.00 Bytes	<1%
683	terminaldb	126.00 Bytes	<1%
684	terabase	126.00 Bytes	<1%
685	synapse	126.00 Bytes	<1%
686	jt400	126.00 Bytes	<1%

APPENDIX II

Some Traffic with Unknown Applications and Protocols

Src IP	Dst IP	Application/ Protocol	Traffic	Percent (%)
10.1.12.88	192.168.10.105	Unknown	314.86 KB	1%
10.1.8.57	192.168.10.17	Unknown	323.28 KB	1%
10.1.12.230	192.168.10.102	Unknown	340.48 KB	1%
10.1.12.141	192.168.10.102	Unknown	364.06 KB	1%
10.1.12.152	192.168.10.15	Unknown	375.84 KB	1%
10.1.12.55	172.16.0.2	Unknown	380.26 KB	1%
10.1.12.37	192.168.10.103	Unknown	396.12 KB	1%
10.1.10.91	192.168.10.22	Unknown	398.7 KB	1%
10.1.5.36	192.168.20.104	Unknown	426.77 KB	1%
10.1.12.88	192.168.10.13	Unknown	434.1 KB	1%
10.1.5.244	192.168.20.5	Unknown	480.55 KB	1%
10.1.12.41	172.16.0.38	Unknown	564.95 KB	1%
10.1.13.165	192.168.10.235	Unknown	594.98 KB	1%
10.1.12.35	192.168.10.6	Unknown	685.99 KB	1%
10.1.12.79	192.168.10.104	Unknown	731.67 KB	1%
10.1.12.41	192.168.10.194	Unknown	923.38 KB	1%
10.1.12.33	192.168.10.40	Unknown	1.05 MB	2%
10.1.13.44	192.168.10.4	Unknown	1.44 MB	2%
10.1.12.37	192.168.10.1	Unknown	2.05 MB	3%
10.1.103.166	192.168.10.15	Unknown	2.11 MB	3%
10.1.13.44	192.168.10.6	Unknown	3.12 MB	5%
10.1.12.37	192.168.10.6	Unknown	11.96 MB	19%
10.1.12.87	192.168.10.1	Unknown	1.04 KB	<1%
10.1.10.25	192.168.60.137	Unknown	1.04 KB	<1%
10.1.0.95	192.168.10.180	Unknown	1.04 KB	<1%
10.1.0.162	192.168.10.1	Unknown	1.04 KB	<1%
10.1.13.41	192.168.10.192	Unknown	1.28 KB	<1%
10.1.13.41	192.168.10.3	Unknown	1.28 KB	<1%
10.1.12.95	192.168.10.101	Unknown	1.28 KB	<1%
10.1.0.171	192.168.10.101	Unknown	1.28 KB	<1%
10.1.5.179	192.168.10.1	Unknown	1.36 KB	<1%
10.1.12.209	192.168.10.3	Unknown	1.36 KB	<1%
10.1.0.87	192.168.10.104	Unknown	1.36 KB	<1%
10.1.0.55	192.168.10.111	Unknown	1.36 KB	<1%
10.1.0.171	192.168.10.180	Unknown	1.36 KB	<1%
10.1.5.244	192.168.10.9	Unknown	1.42 KB	<1%
192.168.100.21	192.168.10.252	Unknown	1.44 KB	<1%
10.1.12.160	192.168.10.100	Unknown	1.44 KB	<1%



APPENDIX III

Programme Listing

```
using System;
using System.Windows.Forms;
using PcapDotNet.Core;
using PcapDotNet.Packets;
using Valkyrie_IDS.Helpers;
using ZeroMQ;
using System.Collections.Generic;
using Valkyrie_IDS.Core;
namespace Valkyrie_IDS
{
   public partial class MainPage : Form
   {
      private PcapUtility _utility;
      private bool hasDevices = false;
      private Logger Log;
      private LivePacketDevice _activeInterface;
      private PacketCommunicator _commObject = null;
      private const int N_WORKERS = 5;
      private const string QUEUE_NAME_KNOWN = @"inproc://valkyrie/known";
      private const string QUEUE_NAME_UNKNOWN = @"inproc://valkyrie/unknown";
      private const string QUEUE_NAME_LOGGER = @"inproc://valkyrie/logger";
      private double thisSafeValue = 0.44; //initial safe value.
      private bool Running { get; set; }
      private ZmqContext _baseContext;
      private ZmqSocket _knownPacketsQueue;
      private ZmqSocket _unknownPacketsQueue;

      private List<string> KnownPacketQueues = new List<string>();
      private List<string> UnknownPacketQueues = new List<string>();
      private List<string> LoggingQueues = new List<string>();
      private List<QueueProcessor> WorkerQueue;
      private ValkyrieWorkFactory factory;
      public MainPage()
      {
         InitializeComponent();
```



```
        _utility = new PcapUtility();
        Log = new Logger(_rtbLogWindow);
        LoadDeviceNames();
        if (!hasDevices)
        {
            DisableInterfaceSelectors();
            Log.Error("No compatible interfaces found.");
        }
        _baseContext = ZmqContext.Create();
        WorkerQueue = new List<QueueProcessor>();
        this.Running = false;
        InitPublishers();
        factory = new ValkyrieWorkFactory(Log);
    }
    private void LoadDeviceNames()
    {
        _cmbInterfaces.Items.Clear();
        string[] deviceNames = _utility.GetDeviceNames(false);
        hasDevices = deviceNames.Length > 0;
        _cmbInterfaces.DataSource = deviceNames;
     }
    private void _btnOpenIFace_Click(object sender, EventArgs e)
    {
        PacketDeviceOpenAttributes _attribs = PacketDeviceOpenAttributes.NoCaptureLocal |
PacketDeviceOpenAttributes.MaximumResponsiveness | PacketDeviceOpenAttributes.Promiscuous;
        try
        {
            InitializeProcessors(); //sets up a queue, so now objects can be created.
            StartProcessors(); //let the dragon ride...no need yet.

            _activeInterface = _utility.GetDeviceByIndex(_cmbInterfaces.SelectedIndex);

#warning Check filter below
#warning Snapshot length used here
            _commObject = _activeInterface.Open(3000, _attribs, 1000);
           // _commObject.SetFilter("tcp or udp");
            BeginPacketCapture();
        }
```



```
        catch (Exception ex)
        {
            Log.Error("An error occurred while opening the interface. Please restart.");
            Log.Error("Debug: " + ex.Message);
            DisableInterfaceSelectors();
            return;
        }
        Log.Success("Opened interface " + _cmbInterfaces.SelectedValue + " successfully.");
        Open_Interface_GUIRoutine();
        Running = true;
}
    private void _btnCloseIFace_Click(object sender, EventArgs e)
    {
        this.Running = false;
        _commObject.Break(); //break the event handler: the working thread will dispose of it.
        Close_Interface_GUIRoutine();
        StopProcessors(); //stop the queue listeners and likely the logger itself.
        Log.Success("Closed Interface " + _cmbInterfaces.SelectedValue + " successfully.");

    }

    private void Packet_Processor(Packet _inputPacket)
    {
    //event handler per packet
        //
        if (!this.Running) return;
        var ipv4Dgram = _inputPacket.Ethernet.IpV4;
        var transportProps = _inputPacket.Ethernet.IpV4.Transport;
        if (transportProps == null) return;
        var sourceIp = ipv4Dgram.Source;
        var destinationIp = ipv4Dgram.Destination;
        var sourcePort = transportProps.SourcePort;
        var destinationPort = transportProps.DestinationPort;
        var size = ipv4Dgram.Length;
        string transport = "";
        if (this.Running != true) return;

        //if (transportProps is PcapDotNet.Packets.Transport.TcpDatagram)
```

```
        if ((transportProps as PcapDotNet.Packets.Transport.TcpDatagram) != null)
        {
           transport = "TCP";
        }
        //else if (transportProps is PcapDotNet.Packets.Transport.UdpDatagram)
        else if ((transportProps as PcapDotNet.Packets.Transport.UdpDatagram) != null)
        {
           transport = "UDP";
        }
  else
        {
           Log.Success("Something else");
           while (true) ;
        }

        Log.Info(transport + " " + sourceIp.ToString() + ":" + sourcePort.ToString() + " --> " +
destinationIp.ToString()+ ":" + destinationPort.ToString());
        //next, implement IP address filtering and protocol-based classification into known and unknown traffic.
        if (transport == "TCP")
        {
           try
           {
              _knownPacketsQueue.Send(_inputPacket.Buffer);

           }
           catch
           {
              return;
           }
        }
        else if (transport == "UDP")
        {
           try
           {
              _unknownPacketsQueue.Send(_inputPacket.Buffer);
           }
           catch
           {
```



```
                return;
            }

        }
        //if ultimately, we must reach a situation whereby processing after filtering and logging are carried out
in
        //this callback, then here is the space to do it.
        //On pain of forking my own code, I will implement the message-passing portions in here and in their
respective
        //classes so as to move functionality in there.
    }
    #region Misc
    private void BeginPacketCapture()
    {
        new System.Threading.Thread(new System.Threading.ThreadStart(StartCapture)).Start();
    }
    private void StartCapture()
    {
      //possible exception point
        _commObject.ReceivePackets(-1, Packet_Processor);
        _commObject.Dispose(); //destroy this object because its operation has been broken.
    }
    #endregion

    private void Form_Closing(object sender, FormClosingEventArgs e)
    {
        this.Running = false;
        DoCleanup();
        Environment.Exit(Environment.ExitCode);
    }
    private void _btnClearLog_Click(object sender, EventArgs e)
    {
        _rtbLogWindow.Clear();
    }

    private void _btnUpdateSValue_Click(object sender, EventArgs e)
    {
        if ((double)_nudSafeValue.Value != thisSafeValue)
```

```
        {
            factory.setSafeValue((double)_nudSafeValue.Value);
            Log.Info("Packet Safe Value set to " + _nudSafeValue.Value.ToString());
            thisSafeValue = (double)_nudSafeValue.Value;
        }
    }
    private void _btnAddProto_Click(object sender, EventArgs e)
    {
        //string protocolName = _cmbProtocols.SelectedItem.ToString();
        string protocolName = _txtAllowedProtos.Text.Trim().ToUpper();
        var response = MessageBox.Show("Add Protocol \"" + protocolName + "\" to allowed protocols?",
"Confirm Add", MessageBoxButtons.YesNo);
        if (response == System.Windows.Forms.DialogResult.No) return;
       foreach (var item in _lstAllowedProtos.Items)
        {
            if (item.ToString() == protocolName)
            {
                MessageBox.Show("Protocol already in list.");
                return;
            }
        }

        _lstAllowedProtos.Items.Add(protocolName);
        Log.Info("Protocol \"" + protocolName + "\" added to allowed protocols.");
    }
    private void _btnRemoveProto_Click(object sender, EventArgs e)
    {
        if (_lstAllowedProtos.SelectedItem == null)
        {
            MessageBox.Show("Please select a protocol to disable.");
            return;
        }
        string protocolName = _lstAllowedProtos.SelectedItem.ToString();
        var result = MessageBox.Show("Remove Protocol \"" + protocolName + "\" from allowed protocols?",
"Confirm Remove", MessageBoxButtons.YesNo);
        if (result == DialogResult.No) return;
        _lstAllowedProtos.Items.Remove(_lstAllowedProtos.SelectedItem);
        Log.Info("Protocol \"" + protocolName + "\" removed from allowed protocols.");
```

```
    }
  }
}
namespace Valkyrie_IDS
{
  partial class MainPage
  {
    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.IContainer components = null;

    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
    protected override void Dispose(bool disposing)
    {
      if (disposing && (components != null))
      {
        components.Dispose();
      }
      base.Dispose(disposing);
    }
    #region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
      System.Windows.Forms.GroupBox _grpInterfaces;
      System.Windows.Forms.Label _lblIFaceName;
      System.Windows.Forms.GroupBox _grpLogger;
      System.Windows.Forms.GroupBox _grpCtrlPanel;
      System.Windows.Forms.GroupBox _grpProtocols;
      System.Windows.Forms.GroupBox _grpSafeValue;
      this._btnCloseIFace = new System.Windows.Forms.Button();
```



```
this._btnOpenIFace = new System.Windows.Forms.Button();
this._cmbInterfaces = new System.Windows.Forms.ComboBox();
this._btnClearLog = new System.Windows.Forms.Button();
this._rtbLogWindow = new System.Windows.Forms.RichTextBox();
this._txtAllowedProtos = new System.Windows.Forms.TextBox();
this._btnRemoveProto = new System.Windows.Forms.Button();
this._lstAllowedProtos = new System.Windows.Forms.ListBox();
this._btnAddProto = new System.Windows.Forms.Button();
this._btnUpdateSValue = new System.Windows.Forms.Button();
this._nudSafeValue = new System.Windows.Forms.NumericUpDown();
_grpInterfaces = new System.Windows.Forms.GroupBox();
_lblIFaceName = new System.Windows.Forms.Label();
_grpLogger = new System.Windows.Forms.GroupBox();
_grpCtrlPanel = new System.Windows.Forms.GroupBox();
_grpProtocols = new System.Windows.Forms.GroupBox();
_grpSafeValue = new System.Windows.Forms.GroupBox();
_grpInterfaces.SuspendLayout();
_grpLogger.SuspendLayout();
_grpCtrlPanel.SuspendLayout();
_grpProtocols.SuspendLayout();
_grpSafeValue.SuspendLayout();
((System.ComponentModel.ISupportInitialize)(this._nudSafeValue)).BeginInit();
this.SuspendLayout();
//
// _grpInterfaces
//
_grpInterfaces.Controls.Add(_lblIFaceName);
_grpInterfaces.Controls.Add(this._btnCloseIFace);
_grpInterfaces.Controls.Add(this._btnOpenIFace);
_grpInterfaces.Controls.Add(this._cmbInterfaces);
_grpInterfaces.Location = new System.Drawing.Point(2, 1);
_grpInterfaces.Name = "_grpInterfaces";
_grpInterfaces.Size = new System.Drawing.Size(358, 108);
_grpInterfaces.TabIndex = 0;
_grpInterfaces.TabStop = false;
_grpInterfaces.Text = "Capture Interfaces";
//
// _lblIFaceName
```

```
//
_lblIFaceName.AutoSize = true;
_lblIFaceName.Location = new System.Drawing.Point(6, 32);
_lblIFaceName.Name = "_lblIFaceName";
_lblIFaceName.Size = new System.Drawing.Size(38, 13);
_lblIFaceName.TabIndex = 3;
_lblIFaceName.Text = "Name:";
//
// _btnCloseIFace
//
this._btnCloseIFace.Enabled = false;
this._btnCloseIFace.Location = new System.Drawing.Point(206, 65);
this._btnCloseIFace.Name = "_btnCloseIFace";
this._btnCloseIFace.Size = new System.Drawing.Size(75, 23);
this._btnCloseIFace.TabIndex = 2;
this._btnCloseIFace.Text = "Close";
this._btnCloseIFace.UseVisualStyleBackColor = true;
this._btnCloseIFace.Click += new System.EventHandler(this._btnCloseIFace_Click);
//
// _btnOpenIFace
//
this._btnOpenIFace.Location = new System.Drawing.Point(61, 65);
this._btnOpenIFace.Name = "_btnOpenIFace";
this._btnOpenIFace.Size = new System.Drawing.Size(75, 23);
this._btnOpenIFace.TabIndex = 1;
this._btnOpenIFace.Text = "Open";
this._btnOpenIFace.UseVisualStyleBackColor = true;
this._btnOpenIFace.Click += new System.EventHandler(this._btnOpenIFace_Click);
//
// _cmbInterfaces
//
this._cmbInterfaces.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;
this._cmbInterfaces.FormattingEnabled = true;
this._cmbInterfaces.Location = new System.Drawing.Point(61, 29);
this._cmbInterfaces.Name = "_cmbInterfaces";
this._cmbInterfaces.Size = new System.Drawing.Size(291, 21);
this._cmbInterfaces.TabIndex = 0;
//
```



```
// _grpLogger
//
_grpLogger.Controls.Add(this._btnClearLog);
_grpLogger.Controls.Add(this._rtbLogWindow);
_grpLogger.Location = new System.Drawing.Point(2, 130);
_grpLogger.Name = "_grpLogger";
_grpLogger.Size = new System.Drawing.Size(358, 347);
_grpLogger.TabIndex = 1;
_grpLogger.TabStop = false;
_grpLogger.Text = "Logger";
//
// _btnClearLog
//
this._btnClearLog.Location = new System.Drawing.Point(134, 29);
this._btnClearLog.Name = "_btnClearLog";
this._btnClearLog.Size = new System.Drawing.Size(118, 23);
this._btnClearLog.TabIndex = 1;
this._btnClearLog.Text = "Clear Log Window";
this._btnClearLog.UseVisualStyleBackColor = true;
this._btnClearLog.Click += new System.EventHandler(this._btnClearLog_Click);
//
// _rtbLogWindow
//
this._rtbLogWindow.Location = new System.Drawing.Point(0, 58);
this._rtbLogWindow.Name = "_rtbLogWindow";
this._rtbLogWindow.Size = new System.Drawing.Size(358, 289);
this._rtbLogWindow.TabIndex = 0;
this._rtbLogWindow.Text = "";
//
// _grpCtrlPanel
//
_grpCtrlPanel.Controls.Add(_grpProtocols);
_grpCtrlPanel.Controls.Add(_grpSafeValue);
_grpCtrlPanel.Location = new System.Drawing.Point(366, 1);
_grpCtrlPanel.Name = "_grpCtrlPanel";
_grpCtrlPanel.Size = new System.Drawing.Size(397, 476);
_grpCtrlPanel.TabIndex = 2;
_grpCtrlPanel.TabStop = false;
```



```
_grpCtrlPanel.Text = "Control Panel";
//
// _grpProtocols
//
_grpProtocols.Controls.Add(this._txtAllowedProtos);
_grpProtocols.Controls.Add(this._btnRemoveProto);
_grpProtocols.Controls.Add(this._lstAllowedProtos);
_grpProtocols.Controls.Add(this._btnAddProto);
_grpProtocols.Location = new System.Drawing.Point(6, 187);
_grpProtocols.Name = "_grpProtocols";
_grpProtocols.Size = new System.Drawing.Size(385, 227);
_grpProtocols.TabIndex = 5;
_grpProtocols.TabStop = false;
_grpProtocols.Text = "Allowed Protocols";
//
// _txtAllowedProtos
//
this._txtAllowedProtos.Enabled = false;
this._txtAllowedProtos.Location = new System.Drawing.Point(112, 28);
this._txtAllowedProtos.Name = "_txtAllowedProtos";
this._txtAllowedProtos.Size = new System.Drawing.Size(120, 20);
this._txtAllowedProtos.TabIndex = 4;
//
// _btnRemoveProto
//
this._btnRemoveProto.Enabled = false;
this._btnRemoveProto.Location = new System.Drawing.Point(255, 114);
this._btnRemoveProto.Name = "_btnRemoveProto";
this._btnRemoveProto.Size = new System.Drawing.Size(75, 23);
this._btnRemoveProto.TabIndex = 3;
this._btnRemoveProto.Text = "Remove";
this._btnRemoveProto.UseVisualStyleBackColor = true;
this._btnRemoveProto.Click += new System.EventHandler(this._btnRemoveProto_Click);
//
// _lstAllowedProtos
//
this._lstAllowedProtos.Enabled = false;
this._lstAllowedProtos.FormattingEnabled = true;
```



```
this._lstAllowedProtos.Location = new System.Drawing.Point(112, 55);
this._lstAllowedProtos.Name = "_lstAllowedProtos";
this._lstAllowedProtos.Size = new System.Drawing.Size(120, 147);
this._lstAllowedProtos.TabIndex = 2;
//
// _btnAddProto
//
this._btnAddProto.Enabled = false;
this._btnAddProto.Location = new System.Drawing.Point(255, 26);
this._btnAddProto.Name = "_btnAddProto";
this._btnAddProto.Size = new System.Drawing.Size(75, 23);
this._btnAddProto.TabIndex = 1;
this._btnAddProto.Text = "Add";
this._btnAddProto.UseVisualStyleBackColor = true;
this._btnAddProto.Click += new System.EventHandler(this._btnAddProto_Click);
//
// _grpSafeValue
//
_grpSafeValue.Controls.Add(this._btnUpdateSValue);
_grpSafeValue.Controls.Add(this._nudSafeValue);
_grpSafeValue.Location = new System.Drawing.Point(94, 29);
_grpSafeValue.Name = "_grpSafeValue";
_grpSafeValue.Size = new System.Drawing.Size(220, 135);
_grpSafeValue.TabIndex = 4;
_grpSafeValue.TabStop = false;
_grpSafeValue.Text = "Packet Safe Value Control";
//
// _btnUpdateSValue
//
this._btnUpdateSValue.Location = new System.Drawing.Point(51, 60);
this._btnUpdateSValue.Name = "_btnUpdateSValue";
this._btnUpdateSValue.Size = new System.Drawing.Size(118, 23);
this._btnUpdateSValue.TabIndex = 2;
this._btnUpdateSValue.Text = "Update Safe Value";
this._btnUpdateSValue.UseVisualStyleBackColor = true;
this._btnUpdateSValue.Click += new System.EventHandler(this._btnUpdateSValue_Click);
//
// _nudSafeValue
```



```
//
this._nudSafeValue.DecimalPlaces = 2;
this._nudSafeValue.Increment = new decimal(new int[] {
1,
0,
0,
131072});
this._nudSafeValue.Location = new System.Drawing.Point(82, 34);
this._nudSafeValue.Maximum = new decimal(new int[] {
100,
0,
0,
131072});
this._nudSafeValue.Name = "_nudSafeValue";
this._nudSafeValue.Size = new System.Drawing.Size(62, 20);
this._nudSafeValue.TabIndex = 3;
this._nudSafeValue.Value = new decimal(new int[] {
44,
0,
0,
131072});
//
// MainPage
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(762, 480);
this.Controls.Add(_grpCtrlPanel);
this.Controls.Add(_grpLogger);
this.Controls.Add(_grpInterfaces);
this.Name = "MainPage";
this.Text = "Form1";
this.FormClosing += new System.Windows.Forms.FormClosingEventHandler(this.Form_Closing);
_grpInterfaces.ResumeLayout(false);
_grpInterfaces.PerformLayout();
_grpLogger.ResumeLayout(false);
_grpCtrlPanel.ResumeLayout(false);
_grpProtocols.ResumeLayout(false);
```



```
            _grpProtocols.PerformLayout();
            _grpSafeValue.ResumeLayout(false);
            ((System.ComponentModel.ISupportInitialize)(this._nudSafeValue)).EndInit();
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.Button _btnCloseIFace;
        private System.Windows.Forms.Button _btnOpenIFace;
        private System.Windows.Forms.ComboBox _cmbInterfaces;
        private System.Windows.Forms.RichTextBox _rtbLogWindow;
        private System.Windows.Forms.Button _btnClearLog;
        private System.Windows.Forms.Button _btnUpdateSValue;
        private System.Windows.Forms.NumericUpDown _nudSafeValue;
        private System.Windows.Forms.Button _btnRemoveProto;
        private System.Windows.Forms.ListBox _lstAllowedProtos;
        private System.Windows.Forms.Button _btnAddProto;
        private System.Windows.Forms.TextBox _txtAllowedProtos;

    }
}
using System;
using ZeroMQ;
using PcapDotNet.Packets;
using System.Threading;
using System.Collections.Generic;

namespace Valkyrie_IDS.Core
{
    class ExtensibleQueueProcessor : QueueProcessor
    {
        private Func<Packet, bool> QualifierFunction;
        private string AffirmativeSocketName;
        private string NegativeSocketName;

        private ZmqSocket AffirmativeSocket = null;
```



```
    private ZmqSocket NegativeSocket = null;
    public ExtensibleQueueProcessor(List<string> _queues, ZmqContext _baseContext, Func<Packet, bool>
_qualifier, string _affirmative, string _negative)
      : base(_queues, _baseContext)
    {
      QualifierFunction = _qualifier;
      AffirmativeSocketName = _affirmative;
      NegativeSocketName = _negative;
    }
    protected override void WorkerMethod()
    {
      while (this.Active)
      {
        int result = 0;
        try
        {
          result = ListenSocket.Receive(buffer, SocketFlags.DontWait);
        }
        catch (Exception e)
        {
          continue; //or return?
        }
     if (result > 0)
        {

          //byte[] intermediateBuffer = new byte[result];
          //buffer.BlockCopy(0, intermediateBuffer, 0, result);
          //Packet _thisPacket = new Packet(intermediateBuffer, DateTime.Now, DataLinkKind.Ethernet);

          Packet _thisPacket = new Packet(buffer, DateTime.Now, DataLinkKind.Ethernet);
          if (!_thisPacket.IsValid) continue; //safety catch
          if (QualifierFunction(_thisPacket))
          {
#warning Maybe decorate all <Socket>.Send methods with exception blocks?
            try
            {
              AffirmativeSocket.Send(_thisPacket.Buffer);
            }
```

```
            catch (Exception positiveSocketException)
            {
               continue; //something's wrong..but continue in the loop and reevaluate the expression.
            }
         }
         else
         {
            if (NegativeSocket != null)

               try
               {
                  NegativeSocket.Send(_thisPacket.Buffer);
               }
               catch (Exception negativeSocketException)
               {
                  continue; //something's wrong..but continue in the loop and reevaluate the expression.
               }
         }
         for (int i = 0; i < buffer.Length; i++)
         {
            buffer[i] = 0;
         }

      }
      else
      {
         try
         {
            Thread.Sleep(new Random(DateTime.Now.Millisecond).Next(0, 5000));
         }
         catch (Exception e)
         {
            continue;
         }
      }
   }//end worker while loop
   //when it is no longer active, this thread will die.
}
```



```
public override void BeginProcessing()
{
   ListenSocket = BaseContext.CreateSocket(SocketType.PULL);
   ListenSocket.Linger = TimeSpan.Zero;
   Inputs.ForEach(n => ListenSocket.Connect(n));
   AffirmativeSocket = BaseContext.CreateSocket(SocketType.PUSH);
   AffirmativeSocket.Linger = TimeSpan.Zero;
   if (NegativeSocketName != null)
   {
      NegativeSocket = BaseContext.CreateSocket(SocketType.PUSH);
      NegativeSocket.Linger = TimeSpan.Zero;
   }
   try
   {
      AffirmativeSocket.Bind(AffirmativeSocketName);
     if (NegativeSocket != null) NegativeSocket.Bind(NegativeSocketName);
   }
   catch (Exception exc)
   {
      AffirmativeSocket.Dispose();
      if (NegativeSocket != null) NegativeSocket.Dispose();
      ListenSocketCleanup();
      throw new Exception(exc.Message, exc.InnerException);
   }
   this.Active = true;
   _workerThread = new Thread(new ThreadStart(WorkerMethod))
   {
      Name = this.FriendlyName.Replace(" ", string.Empty) + "Thread",
   };
   _workerThread.Start();
}
public override void StopProcessing()
{
   if (this.Active == false) return; //this fix saves problems with disposing of the context later.
   this.Active = false;
   if (_workerThread.ThreadState == System.Threading.ThreadState.WaitSleepJoin)
   {
      //don't try to abort it, rather, wake it up.
```

```
            _workerThread.Interrupt();
        }
        else
        {
            _workerThread.Join(2000);
          _workerThread = null; //maybe force the GC's hand?
        }
        QualifierFunction = null; //mark your object for garbage collection?
        AffirmativeSocket.Dispose();
        if (NegativeSocket != null) NegativeSocket.Dispose();
        ListenSocketCleanup();
    }
    public string GetAffirmativeEndpoint()
    {
        return this.AffirmativeSocketName;
    }
    public string GetNegativeEndpoint()
    {
        return this.NegativeSocketName;
    }
  }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Drawing;
namespace Valkyrie_IDS.Helpers
{
  public static class ExtensionMethods
  {
    public static void AppendColoredText(this RichTextBox box, string text, Color newColor)
    {
      box.SelectionStart = box.TextLength;
        box.SelectionLength = 0;
        box.SelectionColor = newColor;
        box.AppendText("[" + DateTime.Now.ToLongTimeString() + "]: " + text + Environment.NewLine);
```



```
        }
    }

}
using System;
using System.Windows.Forms;
using System.Drawing;
namespace Valkyrie_IDS.Helpers
{
    #region Logger
    public class Logger
    {
        private delegate void UpdateMethodDelegate(string s, Color c);
        UpdateMethodDelegate _myUpdateMethod;
        private RichTextBox _reference;
        public Logger(RichTextBox _rtb)
        {
            _reference = _rtb;
            _myUpdateMethod = UpdateMethod;
        }
        public void Error(string message)
        {
            //return;
            if (_reference.InvokeRequired)
            {
                try
                {
                    _reference.Invoke(_myUpdateMethod, message, Color.Red);
                }
                catch
                {
                    return;
                }
                return;
            }
            _reference.AppendColoredText(message, Color.Red);
        }
        public void Info(string message)
```

```
    {
        if (_reference.InvokeRequired)
        {
            _reference.Invoke(_myUpdateMethod, message, Color.Black);
            return;
        }
        _reference.AppendColoredText(message, Color.Black);
    }
    public void Success(string message)
    {
        if (_reference.InvokeRequired)
        {
            _reference.Invoke(_myUpdateMethod, message, Color.Green);
            return;
        }
        _reference.AppendColoredText(message, Color.Green);
    }
    private void UpdateMethod(string st, Color clr)
    {
        _reference.AppendColoredText(st, clr);
    }

  }
  #endregion
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Valkyrie_IDS.Core;
namespace Valkyrie_IDS
{
  public partial class MainPage : Form
  {
    #region Non-GUI Helpers
    private void InitializeProcessors()
    {
```



```
        LoggingQueues.Clear();
        WorkerQueue.Clear();

        //var factory = new ValkyrieWorkFactory(Log);
        ExtensibleQueueProcessor _thisEQP;
        TerminalQueueProcessor _thisTQP;
        for (int i = 0; i < N_WORKERS * 2 ; i++)
        {
           if ((i == 0) || (i % 2 == 0))
           {
              _thisEQP = new ExtensibleQueueProcessor(KnownPacketQueues, _baseContext,
factory.GetKnownPacketComparer(), QUEUE_NAME_LOGGER + i.ToString(), null);
              _thisEQP.FriendlyName = i + "_Known_To_Logger";
           }
           else
           {
              _thisEQP = new ExtensibleQueueProcessor(UnknownPacketQueues, _baseContext,
factory.GetUnknownPacketComparer(), QUEUE_NAME_LOGGER + i.ToString(), null);
              _thisEQP.FriendlyName = i + "_Unknown_To_Logger";
           }
           LoggingQueues.Add(_thisEQP.GetAffirmativeEndpoint());
           WorkerQueue.Add(_thisEQP);
        }
        _thisTQP = new TerminalQueueProcessor(LoggingQueues, _baseContext, factory.GetLoggerTask());
        _thisTQP.FriendlyName = "One_To_Logger";
        WorkerQueue.Add(_thisTQP);
        _thisTQP = new TerminalQueueProcessor(LoggingQueues, _baseContext, factory.GetLoggerTask());
        _thisTQP.FriendlyName = "Two_To_Logger";
        WorkerQueue.Add(_thisTQP);
     }
     private void StartProcessors()
     {
        WorkerQueue.ForEach(n => n.BeginProcessing());
     }
     private void StopProcessors()
     {
        //WorkerQueue.Reverse();
        WorkerQueue.ForEach(n => n.StopProcessing());
```



```
    }
    private void InitPublishers()
    {
      KnownPacketQueues.Clear();
      UnknownPacketQueues.Clear();
      _knownPacketsQueue = _baseContext.CreateSocket(ZeroMQ.SocketType.PUSH);
      _unknownPacketsQueue = _baseContext.CreateSocket(ZeroMQ.SocketType.PUSH);
      _knownPacketsQueue.Linger = TimeSpan.Zero;
      _unknownPacketsQueue.Linger = TimeSpan.Zero;
#warning Experimental HWM code..regarding that potential memory leak
      _knownPacketsQueue.SendHighWatermark = 1000;
      _unknownPacketsQueue.SendHighWatermark = 1000;
      _knownPacketsQueue.Bind(QUEUE_NAME_KNOWN);
      _unknownPacketsQueue.Bind(QUEUE_NAME_UNKNOWN);
      KnownPacketQueues.Add(QUEUE_NAME_KNOWN); //store this endpoint so that sockets know how
many to connect to
      UnknownPacketQueues.Add(QUEUE_NAME_UNKNOWN); //same
    }
    private void StopPublishers()
    {
      _knownPacketsQueue.Dispose();
      _unknownPacketsQueue.Dispose();
    }
    private void DoCleanup()
    {
      StopProcessors();
      StopPublishers();
      try
      {
        _baseContext.Dispose();
      }
      catch
      {
        return;
      }
    }
    #endregion
   #region GUI Helpers
```



```
        private void DisableInterfaceSelectors()
        {
            _btnOpenIFace.Enabled = false;
        }
        private void Open_Interface_GUIRoutine()
        {
            _btnOpenIFace.Enabled = false;
            _btnCloseIFace.Enabled = true;
            //_btnUpdateSValue.Enabled = true;
            //_nudSafeValue.Enabled = true;
            _txtAllowedProtos.Enabled = true;
            _btnAddProto.Enabled = true;
            _lstAllowedProtos.Enabled = true;
            _btnRemoveProto.Enabled = true;
        }
        private void Close_Interface_GUIRoutine()
        {
            _btnOpenIFace.Enabled = true;
            _btnCloseIFace.Enabled = false;
            //_btnUpdateSValue.Enabled = false;
            //_nudSafeValue.Enabled = false;
            _txtAllowedProtos.Enabled = false;
            _btnAddProto.Enabled = false;
            _lstAllowedProtos.Enabled = false;
            _btnRemoveProto.Enabled = false;
        }
        #endregion
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using PcapDotNet.Core;
using PcapDotNet.Packets;
namespace Valkyrie_IDS.Helpers
{
    class PcapUtility
```

```
  {
    public PcapUtility()
    {
      AvailableInterfaces = LivePacketDevice.AllLocalMachine;
    }
    public LivePacketDevice GetDeviceByIndex(int index)
    {
      return AvailableInterfaces[index];
    }
    public string[] GetDeviceNames(bool forceEthernet = true)
    {
      IEnumerable<LivePacketDevice> _dataSet;
      if (forceEthernet)
      {
        _dataSet = AvailableInterfaces.Where(x => x.Description.ToLower().Contains("ethernet")).AsParallel();
      }
      else
      {
        _dataSet = AvailableInterfaces;
      }
      return _dataSet.Select(n =>
      {
        string x = n.Description;
        int startIndex = x.IndexOf('\'') + 1;
        int length = x.LastIndexOf('\'') - startIndex;
        return x.Substring(startIndex, length);
      }).ToArray();
    }
    private IList<LivePacketDevice> AvailableInterfaces { get; set; }
  }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using ZeroMQ;
using System.Diagnostics;
```

```
namespace Valkyrie_IDS.Core
{
   class QueueProcessor
   {
      protected Thread _workerThread;
      protected ZmqContext BaseContext { get; set; }
      protected ZmqSocket ListenSocket { get; set; }
      public string FriendlyName { get; set; }
      protected bool Active { get; set; }
      protected byte[] buffer;
      protected List<string> Inputs;
      public QueueProcessor(List<string> _queues, ZmqContext _parentContext)
      {
         Inputs = _queues;
         this.Active = false;
         BaseContext = _parentContext;
         buffer = new byte[3500];
         this.FriendlyName = "QueueProcessor";
      }
      public virtual void BeginProcessing()
      {
         //always ensure the endpoint exists before connecting.
         ListenSocket = BaseContext.CreateSocket(SocketType.PULL);
#warning Experimental HWM code..regarding that memory leak..
         ListenSocket.ReceiveHighWatermark = 1000;
         Inputs.ForEach(n => ListenSocket.Connect(n));
         this.Active = true;

         _workerThread = new Thread(new ThreadStart(WorkerMethod))
         {
            Name = this.FriendlyName.Replace(" ", string.Empty) + "Thread",
         };
         _workerThread.Start();
      }
      public virtual void StopProcessing()
      {
         if (this.Active == false) return; //this fix saves problems with disposing of the context later.
         this.Active = false;
```

```
        if (_workerThread.ThreadState == System.Threading.ThreadState.WaitSleepJoin)
        {
           //don't try to abort it, rather, wake it up.
           _workerThread.Interrupt();
        }
        else
        {
#warning Watch this space..if anything goes wrong with killing the threads uncomment this out and remove the joining call below.
           //try
           //{
           //    //is there explicit need to abort??
           //   // _workerThread.Abort();
           //}
           //catch (Exception e)
           //{
           //    //Console.WriteLine("CLIENT: Caught exception, aborting thread.");
           //}
#warning Experimental code about Thread.Join() here
           _workerThread.Join(2000);
           _workerThread = null; //maybe force the GC's hand?
        }
        ListenSocketCleanup();
       // Console.WriteLine("CLIENT " + FriendlyName + " shutting down.");
     }
     protected virtual void WorkerMethod()
     {
        while (this.Active)
        {
           var result = ListenSocket.Receive(buffer, SocketFlags.DontWait);
           if (result > 0)
           {
              //do some work with the contents of the buffer variable.
              for (int i = 0; i < buffer.Length; i++)
              {
                 buffer[i] = 0;
              }
```

```
            }
            else
            {
               //sleep for around 5 seconds for now.
               try
               {
                  Thread.Sleep(new Random(DateTime.Now.Millisecond).Next(0, 5000));
               }
               catch (Exception e)
               {
                  //our sleep was interrupted. Something's up.
                  continue;
               }
            }
         }//end while loop
         //when it is no longer active, this thread will die.
      }
      protected void ListenSocketCleanup()
      {
         Inputs.ForEach(n => ListenSocket.Disconnect(n));
         ListenSocket.Close();
         ListenSocket.Dispose();
      }
   }
}
using System;
using ZeroMQ;
using PcapDotNet.Packets;
using System.Threading;
using System.Collections.Generic;
namespace Valkyrie_IDS.Core
{
   class TerminalQueueProcessor:QueueProcessor
   {
      private Action<Packet> Task;
      public TerminalQueueProcessor(List<string> _queues,ZmqContext _baseContext, Action<Packet>
_task):base(_queues,_baseContext)
      {
```



```
        Task = _task;
    }

    protected override void WorkerMethod()
    {
        while (this.Active)
        {
#warning Perhaps decorate all ListenSocket.Receive calls with exception blocks?
            int result = 0;
            try
            {
                result = ListenSocket.Receive(buffer, SocketFlags.DontWait);
            }
            catch (Exception e)
            {
                continue; //or return?
            }
            if (result > 0)
            {
                //do some work with the contents of the buffer variable.
#warning Check that the packet ctor is using ipv4 not ethernet in terminalQueueProcessor
                Packet _thisPacket = new Packet(buffer, DateTime.Now, DataLinkKind.Ethernet);
                if (!_thisPacket.IsValid) continue; //safety catch
                Task(_thisPacket);
                for (int i = 0; i < buffer.Length; i++)
                {
                    buffer[i] = 0;
                }
            }
            else
            {
                try
                {
                    Thread.Sleep(new Random(DateTime.Now.Millisecond).Next(0, 5000));
                }

                catch (Exception e)
                {
```



```
                continue;
              }
            }
          }//end worker while loop
          Task = null; //mark this for garbage collection.
          //when it is no longer active, this thread will die.
      }
      }
  }
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using PcapDotNet.Packets;
using PcapDotNet.Core;
using Valkyrie_IDS.Helpers;
namespace Valkyrie_IDS.Core
{
  class ValkyrieWorkFactory
  {
      private Logger _logInstance;
      private const double SAFE_VALUE = 0.44;
      private const int KNOWN_PACKET_SIZE = 1500;
      private const int UNKNOWN_PACKET_SIZE = 84;
      private Func<Packet,bool> _KnownPacketClassifier;
      private Func<Packet,bool> _UnknownPacketClassifier;
      private Action<Packet> _LoggerTask;
     private double SafeValue { get; set; }
      public ValkyrieWorkFactory(Logger debugConsole)
      {
        _logInstance = debugConsole;
        this.SafeValue = 0.44;
        LoadFunctionObjects();
      }
/*
      public Action<Packet> GetLoggerTask()
      {
        //return new Action<Packet>((p) =>
```

```
        //{
        //    #warning Snapshot length used here
        //    //_logInstance.Error("Called!");
        //    _logInstance.Error("Suspicious packet detected");
        //   // PacketDumpFile.Dump("./Logs/" + p.Timestamp.ToLongTimeString().Replace("
",string.Empty).ToString(), DataLinkKind.Ethernet, 3000, new Packet[] {p});
        //    PacketDumpFile.Dump("./Logs/" + p.Timestamp.Ticks.ToString(), DataLinkKind.Ethernet, 3000, new
Packet[] { p });
        //});
        return _LoggerTask;
    }
    public Func<Packet, bool> GetKnownPacketComparer()
    {
//        return new Func<Packet, bool>((p) =>
//        {
//#warning Experimental Code in Known Packet Comparer
//            double packetScore = (KNOWN_PACKET_SIZE - p.Ethernet.IpV4.Payload.Length) /
(double)KNOWN_PACKET_SIZE;
//            if (packetScore > SAFE_VALUE) return true;
//            return false;
//            //if (p.Ethernet.IpV4.Transport.SourcePort != 80) return true;
//            //return false;
//        });
        return _KnownPacketClassifier;
    }
    public Func<Packet, bool> GetUnknownPacketComparer()
    {
#warning Experimental Code in Unknown Packet Comparer
        //return new Func<Packet, bool>((p) =>
        //{
        //    double packetScore = (UNKNOWN_PACKET_SIZE - p.Ethernet.IpV4.Payload.Length) /
(double)UNKNOWN_PACKET_SIZE;
        //    if (packetScore > SAFE_VALUE) return true;
        //    return false;
        //    //if (p.Ethernet.IpV4.Transport.SourcePort != 80) return true;
        //    //return false;
        //});
        return _UnknownPacketClassifier;
```

```
    }

    private void LoadFunctionObjects()
    {
        _LoggerTask = new Action<Packet>((p) =>
      {
        _logInstance.Error("Suspicious packet detected");
        PacketDumpFile.Dump("./Logs/" + p.Timestamp.Ticks.ToString(), DataLinkKind.Ethernet, 3000, new
Packet[] { p });
      });
      _UnknownPacketClassifier = new Func<Packet, bool>((p) =>
      {
        double packetScore = (UNKNOWN_PACKET_SIZE - p.Ethernet.IpV4.Payload.Length) /
(double)UNKNOWN_PACKET_SIZE;
        Console.WriteLine("Safevalue of: " + this.getSafeValue().ToString());
        if (packetScore > this.getSafeValue()) return true;
        return false;
      });
      _KnownPacketClassifier = new Func<Packet, bool>((p) =>
      {
        double packetScore = (KNOWN_PACKET_SIZE - p.Ethernet.IpV4.Payload.Length) /
(double)KNOWN_PACKET_SIZE;
        Console.WriteLine("Safevalue of: " + this.getSafeValue().ToString());
        if (packetScore > this.getSafeValue()) return true;
        return false;
      });
    }
    private double getSafeValue()
    {
      return this.SafeValue;
    }
    public void setSafeValue(double newSafeValue)
    {
      this.SafeValue = newSafeValue;
    }
  }
}
```

APPENDIX IV

Captured Packets on a Small LAN

[08:43:28]: UDP 10.255.2.101:52076 --> 224.0.0.252:5355
[08:43:28]: TCP 254.210.10.255:1 --> 2.94.0.0:2048
[08:43:29]: UDP 10.255.1.138:58858 --> 255.255.255.255:10007
[08:43:29]: UDP 10.255.2.100:17500 --> 255.255.255.255:17500
[08:43:29]: UDP 10.255.1.237:137 --> 10.255.3.255:137
[08:43:29]: UDP 10.255.2.100:17500 --> 10.255.3.255:17500
[08:43:29]: UDP 10.255.2.100:17500 --> 255.255.255.255:17500
[08:43:29]: UDP 10.255.2.80:137 --> 10.255.3.255:137
[08:43:29]: UDP 10.255.2.80:137 --> 10.255.3.255:137
[08:43:29]: UDP 10.255.2.100:17500 --> 255.255.255.255:17500
[08:43:29]: UDP 10.255.2.82:137 --> 10.255.3.255:137
[08:43:29]: UDP 10.255.2.95:138 --> 10.255.3.255:138
[08:43:29]: UDP 10.255.2.56:137 --> 10.255.3.255:137
[08:43:29]: UDP 10.255.2.102:54850 --> 239.255.255.250:1900
[08:43:29]: UDP 0.0.0.0:5678 --> 255.255.255.255:5678
[08:43:29]: UDP 10.255.2.88:137 --> 10.255.3.255:137
[08:43:29]: UDP 10.255.2.80:137 --> 10.255.3.255:137
[08:43:30]: UDP 10.255.2.47:68 --> 255.255.255.255:67
[08:43:30]: UDP 10.255.2.82:137 --> 10.255.3.255:137
[08:43:30]: UDP 10.255.2.43:68 --> 10.255.0.1:67
[08:43:30]: UDP 10.255.1.138:58860 --> 255.255.255.255:10007
[08:43:30]: UDP 10.255.1.138:58861 --> 255.255.255.255:10019
[08:43:30]: UDP 10.255.2.95:60040 --> 239.255.255.250:1900
[08:43:30]: UDP 10.255.2.95:138 --> 10.255.3.255:138
[08:43:30]: UDP 10.255.2.88:137 --> 10.255.3.255:137
[08:43:30]: UDP 10.255.2.94:138 --> 10.255.3.255:138
[08:43:30]: UDP 10.255.2.80:137 --> 10.255.3.255:137
[08:43:30]: UDP 10.255.2.80:137 --> 10.255.3.255:137
[08:43:30]: UDP 10.255.2.82:137 --> 10.255.3.255:137
[08:43:30]: UDP 10.255.2.95:137 --> 10.255.3.255:137

[08:43:31]: UDP 10.255.2.95:1900 --> 239.255.255.250:1900

[08:43:31]: UDP 10.255.2.95:1900 --> 239.255.255.250:1900

[08:43:31]: UDP 10.255.2.95:1900 --> 239.255.255.250:1900

[08:43:31]: UDP 10.255.2.95:1900 --> 239.255.255.250:1900

[08:43:31]: UDP 10.255.2.95:60040 --> 239.255.255.250:1900

[08:43:31]: UDP 10.255.2.88:137 --> 10.255.3.255:137

[08:43:31]: UDP 10.255.2.97:138 --> 10.255.3.255:138

[08:43:31]: UDP 10.255.2.82:137 --> 10.255.3.255:137

[08:43:31]: TCP 10.255.2.100:53675 --> 10.0.0.10:110

[08:43:31]: TCP 10.0.0.10:110 --> 10.255.2.100:53675

[08:43:31]: TCP 10.0.0.10:110 --> 10.255.2.100:53675

[08:43:31]: TCP 10.255.2.100:53675 --> 10.0.0.10:110

[08:43:31]: TCP 10.0.0.10:110 --> 10.255.2.100:53675

[08:43:31]: TCP 10.0.0.10:110 --> 10.255.2.100:53675

[08:43:31]: TCP 10.255.2.100:53675 --> 10.0.0.10:110

[08:43:31]: Suspicious packet detected

[08:43:31]: TCP 10.0.0.10:110 --> 10.255.2.100:53675

[08:43:31]: TCP 10.0.0.10:110 --> 10.255.2.100:53675

[08:43:31]: TCP 10.255.2.100:53675 --> 10.0.0.10:110

[08:43:31]: TCP 10.0.0.10:110 --> 10.255.2.100:53675

[08:43:31]: TCP 10.0.0.10:110 --> 10.255.2.100:53675

[08:43:32]: TCP 10.255.2.100:53675 --> 10.0.0.10:110

[08:43:32]: TCP 10.0.0.10:110 --> 10.255.2.100:53675

[08:43:32]: TCP 10.0.0.10:110 --> 10.255.2.100:53675

[08:43:32]: TCP 10.255.2.100:53675 --> 10.0.0.10:110

[08:43:32]: TCP 10.0.0.10:110 --> 10.255.2.100:53675

[08:43:32]: TCP 10.255.2.100:53631 --> 2.18.19.235:80

[08:43:32]: UDP 10.255.2.95:137 --> 10.255.3.255:137

[08:43:32]: TCP 10.0.0.10:110 --> 10.255.2.100:53675

[08:43:32]: TCP 10.255.2.100:53675 --> 10.0.0.10:110

[08:43:32]: TCP 2.18.19.235:80 --> 10.255.2.100:53631

[08:43:32]: TCP 2.18.19.235:80 --> 10.255.2.100:53631

[08:43:32]: TCP 10.255.2.100:53631 --> 2.18.19.235:80

[08:43:32]: TCP 2.18.19.235:80 --> 10.255.2.100:53631
[08:43:32]: UDP 10.255.2.95:138 --> 10.255.3.255:138
[08:43:32]: TCP 2.18.19.235:80 --> 10.255.2.100:53631
[08:43:32]: TCP 10.255.2.100:53631 --> 2.18.19.235:80
[08:43:32]: TCP 2.18.19.235:80 --> 10.255.2.100:53631
[08:43:32]: UDP 10.255.2.88:137 --> 10.255.3.255:137
[08:43:32]: UDP 10.255.2.61:137 --> 10.255.3.255:137
[08:43:32]: TCP 10.255.2.100:53631 --> 2.18.19.235:80
[08:43:32]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:33]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:33]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:33]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:33]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:33]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:33]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:33]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:33]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:33]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:33]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:33]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:33]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:33]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:33]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:33]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:33]: UDP 10.255.1.138:58864 --> 255.255.255.255:10007
[08:43:34]: UDP 10.255.2.82:137 --> 10.255.3.255:137
[08:43:34]: UDP 10.255.1.138:58865 --> 255.255.255.255:10019
[08:43:34]: UDP 10.255.2.95:137 --> 10.255.3.255:137
[08:43:34]: UDP 10.255.2.88:137 --> 10.255.3.255:137
[08:43:34]: UDP 10.255.2.92:138 --> 10.255.3.255:138
[08:43:34]: UDP 10.255.2.61:137 --> 10.255.3.255:137
[08:43:34]: UDP 10.255.2.82:137 --> 10.255.3.255:137
[08:43:34]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:34]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:34]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:34]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:34]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:34]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:34]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:34]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:34]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:34]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:34]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:34]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:34]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:34]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:34]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:34]: UDP 10.255.2.95:60040 --> 239.255.255.250:1900
[08:43:35]: UDP 10.255.1.138:58866 --> 255.255.255.255:10007
[08:43:35]: UDP 10.255.2.102:54850 --> 239.255.255.250:1900
[08:43:35]: UDP 10.255.2.88:137 --> 10.255.3.255:137
[08:43:35]: UDP 10.255.2.61:137 --> 10.255.3.255:137
[08:43:35]: UDP 10.255.2.101:53864 --> 224.0.0.252:5355
[08:43:35]: UDP 10.255.2.101:53864 --> 224.0.0.252:5355
[08:43:35]: UDP 10.255.2.95:138 --> 10.255.3.255:138
[08:43:35]: UDP 10.255.2.101:137 --> 10.255.3.255:137
[08:43:35]: UDP 10.255.2.83:53211 --> 224.0.0.252:5355
[08:43:35]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:35]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:35]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:35]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:35]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:35]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:35]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:35]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:35]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:35]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:35]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:35]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:35]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:35]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:35]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:35]: Suspicious packet detected

[08:43:35]: UDP 10.255.2.95:60040 --> 239.255.255.250:1900

[08:43:35]: Suspicious packet detected

[08:43:35]: UDP 10.255.1.138:58868 --> 255.255.255.255:10007

[08:43:35]: UDP 10.255.2.88:137 --> 10.255.3.255:137

[08:43:36]: UDP 10.255.1.138:58869 --> 255.255.255.255:10019

[08:43:36]: UDP 10.255.2.80:63810 --> 224.0.0.252:5355

[08:43:36]: UDP 10.255.2.95:138 --> 10.255.3.255:138

[08:43:36]: UDP 10.255.2.82:137 --> 10.255.3.255:137

[08:43:36]: UDP 10.255.2.101:137 --> 10.255.3.255:137

[08:43:36]: UDP 10.255.2.85:138 --> 10.255.3.255:138

[08:43:36]: UDP 10.255.2.88:137 --> 10.255.3.255:137

[08:43:36]: UDP 10.255.2.76:49233 --> 239.255.255.250:1900

[08:43:36]: UDP 10.255.2.95:138 --> 10.255.3.255:138

[08:43:36]: UDP 10.255.1.138:58871 --> 255.255.255.255:10019

[08:43:36]: UDP 10.255.2.101:137 --> 10.255.3.255:137

[08:43:36]: UDP 10.255.2.61:137 --> 10.255.3.255:137

[08:43:36]: UDP 10.255.2.100:63152 --> 40.0.0.2:53

[08:43:36]: UDP 40.0.0.2:53 --> 10.255.2.100:63152

[08:43:36]: UDP 10.255.2.88:137 --> 10.255.3.255:137

[08:43:36]: TCP 10.255.2.100:53705 --> 199.7.55.72:80

[08:43:36]: TCP 199.7.55.72:80 --> 10.255.2.100:53705

[08:43:36]: Suspicious packet detected

[08:43:36]: TCP 10.255.2.100:53705 --> 199.7.55.72:80

[08:43:36]: TCP 10.255.2.100:53705 --> 199.7.55.72:80

[08:43:36]: UDP 10.255.2.61:138 --> 10.255.3.255:138

[08:43:37]: TCP 199.7.55.72:80 --> 10.255.2.100:53705

[08:43:37]: TCP 199.7.55.72:80 --> 10.255.2.100:53705



[08:43:37]: TCP 199.7.55.72:80 --> 10.255.2.100:53705

[08:43:37]: TCP 10.255.2.100:53705 --> 199.7.55.72:80

[08:43:37]: TCP 199.7.55.72:80 --> 10.255.2.100:53705

[08:43:37]: TCP 199.7.55.72:80 --> 10.255.2.100:53705

[08:43:37]: TCP 10.255.2.100:53705 --> 199.7.55.72:80

[08:43:37]: UDP 10.255.2.43:68 --> 10.255.0.1:67

[08:43:37]: TCP 199.7.55.72:80 --> 10.255.2.100:53705

[08:43:37]: TCP 10.255.2.100:53705 --> 199.7.55.72:80

[08:43:37]: UDP 10.255.1.138:58872 --> 255.255.255.255:10007

[08:43:37]: UDP 10.255.2.61:137 --> 10.255.3.255:137

[08:43:37]: UDP 10.255.2.80:58159 --> 224.0.0.252:5355

[08:43:37]: UDP 10.255.2.80:56522 --> 224.0.0.252:5355

[08:43:37]: UDP 10.255.2.80:60297 --> 224.0.0.252:5355

[08:43:37]: UDP 10.255.2.80:59222 --> 224.0.0.252:5355

[08:43:37]: UDP 10.255.2.80:65108 --> 224.0.0.252:5355

[08:43:37]: UDP 10.255.2.80:56522 --> 224.0.0.252:5355

[08:43:37]: UDP 10.255.2.80:58159 --> 224.0.0.252:5355

[08:43:38]: UDP 10.255.2.80:60297 --> 224.0.0.252:5355

[08:43:38]: UDP 10.255.2.80:65108 --> 224.0.0.252:5355

[08:43:38]: UDP 10.255.2.80:59222 --> 224.0.0.252:5355

[08:43:38]: Suspicious packet detected

[08:43:38]: UDP 10.255.1.138:58873 --> 255.255.255.255:10019

[08:43:38]: Suspicious packet detected

[08:43:38]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:38]: Suspicious packet detected

[08:43:38]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:38]: Suspicious packet detected

[08:43:38]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:38]: Suspicious packet detected

[08:43:38]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:38]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:38]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:38]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:38]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:38]: UDP 10.255.2.102:54850 --> 239.255.255.250:1900

[08:43:38]: UDP 10.255.2.88:137 --> 10.255.3.255:137

[08:43:38]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:38]: Suspicious packet detected

[08:43:38]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:38]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:38]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:39]: Suspicious packet detected

[08:43:39]: UDP 10.255.2.95:138 --> 10.255.3.255:138

[08:43:39]: Suspicious packet detected

[08:43:39]: UDP 10.255.2.83:53211 --> 224.0.0.252:5355

[08:43:39]: UDP 10.255.2.61:137 --> 10.255.3.255:137

[08:43:39]: UDP 0.0.0.0:68 --> 255.255.255.255:67

[08:43:39]: UDP 10.255.0.1:67 --> 10.255.2.99:68

[08:43:39]: UDP 10.255.2.95:137 --> 10.255.3.255:137

[08:43:39]: UDP 10.255.2.88:137 --> 10.255.3.255:137

[08:43:39]: UDP 10.255.2.83:63725 --> 224.0.0.252:5355

[08:43:39]: Suspicious packet detected

[08:43:39]: UDP 10.255.2.83:63725 --> 224.0.0.252:5355

[08:43:39]: Suspicious packet detected

[08:43:39]: UDP 10.255.1.169:49974 --> 239.255.255.250:1900

[08:43:39]: UDP 10.255.1.138:58875 --> 255.255.255.255:10019

[08:43:39]: Suspicious packet detected

[08:43:39]: UDP 10.255.2.95:60040 --> 239.255.255.250:1900

[08:43:39]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:39]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:39]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:40]: Suspicious packet detected

[08:43:40]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:40]: Suspicious packet detected

[08:43:40]: UDP 0.0.0.0:68 --> 255.255.255.255:67

[08:43:40]: Suspicious packet detected

[08:43:40]: UDP 10.255.2.80:64103 --> 239.255.255.250:1900

[08:43:40]: Suspicious packet detected

[08:43:40]: UDP 10.255.2.102:49152 --> 255.255.255.255:1947

[08:43:40]: Suspicious packet detected

[08:43:40]: UDP 10.255.2.88:137 --> 10.255.3.255:137

[08:43:40]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:40]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:40]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:40]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:40]: UDP 10.255.1.138:58876 --> 255.255.255.255:10007

[08:43:40]: UDP 10.255.2.83:1900 --> 239.255.255.250:1900

[08:43:40]: UDP 10.255.2.83:1900 --> 239.255.255.250:1900

[08:43:40]: UDP 10.255.2.83:1900 --> 239.255.255.250:1900

[08:43:40]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:40]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:40]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:40]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:40]: UDP 10.255.2.95:137 --> 10.255.3.255:137

[08:43:40]: UDP 10.255.2.95:137 --> 10.255.3.255:137

[08:43:40]: UDP 10.255.1.138:58878 --> 255.255.255.255:10007

[08:43:40]: UDP 10.255.2.102:54850 --> 239.255.255.250:1900

[08:43:40]: UDP 10.255.2.95:137 --> 10.255.3.255:137

[08:43:40]: UDP 10.255.2.95:137 --> 10.255.3.255:137

[08:43:40]: TCP 10.255.2.100:53707 --> 2.18.19.235:443

[08:43:41]: TCP 2.18.19.235:443 --> 10.255.2.100:53707

[08:43:41]: TCP 10.255.2.100:53707 --> 2.18.19.235:443

[08:43:41]: TCP 10.255.2.100:53707 --> 2.18.19.235:443

[08:43:41]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:41]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:41]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:41]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:41]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:41]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:41]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:41]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:41]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:41]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:41]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:41]: Suspicious packet detected

[08:43:41]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:41]: Suspicious packet detected

[08:43:41]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:41]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:41]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:42]: TCP 2.18.19.235:443 --> 10.255.2.100:53707

[08:43:42]: TCP 2.18.19.235:443 --> 10.255.2.100:53707

[08:43:42]: TCP 2.18.19.235:443 --> 10.255.2.100:53707

[08:43:42]: TCP 10.255.2.100:53707 --> 2.18.19.235:443

[08:43:42]: UDP 10.255.2.61:137 --> 10.255.3.255:137

[08:43:42]: TCP 2.18.19.235:443 --> 10.255.2.100:53707

[08:43:42]: TCP 10.255.2.100:53707 --> 2.18.19.235:443

[08:43:42]: UDP 10.255.2.95:137 --> 10.255.3.255:137

[08:43:42]: UDP 10.255.1.169:49974 --> 239.255.255.250:1900

[08:43:42]: UDP 10.255.2.80:137 --> 10.255.3.255:137

[08:43:42]: TCP 2.18.19.235:443 --> 10.255.2.100:53707

[08:43:42]: UDP 10.255.2.95:137 --> 10.255.3.255:137

[08:43:42]: UDP 10.255.2.95:60040 --> 239.255.255.250:1900

[08:43:42]: UDP 10.255.1.138:58881 --> 255.255.255.255:10019

[08:43:42]: TCP 10.255.2.100:53707 --> 2.18.19.235:443

[08:43:42]: UDP 10.255.2.95:137 --> 10.255.3.255:137

[08:43:42]: UDP 10.255.2.80:64103 --> 239.255.255.250:1900

[08:43:42]: UDP 10.255.2.61:137 --> 10.255.3.255:137

[08:43:42]: UDP 10.255.2.95:137 --> 10.255.3.255:137

[08:43:42]: UDP 10.255.2.95:137 --> 10.255.3.255:137

[08:43:42]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:43]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900

[08:43:43]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:43]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:43]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:43]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:43]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:43]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:43]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:43]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:43]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:43]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:43]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:43]: UDP 192.168.1.1:1900 --> 239.255.255.250:1900
[08:43:43]: UDP 10.255.2.95:137 --> 10.255.3.255:137
[08:43:43]: UDP 10.255.2.95:137 --> 10.255.3.255:137
[08:43:43]: UDP 10.255.2.95:137 --> 10.255.3.255:137
[08:43:43]: UDP 10.255.2.95:137 --> 10.255.3.255:137
[08:43:43]: UDP 0.0.0.0:5678 --> 255.255.255.255:5678
[08:43:43]: UDP 10.255.2.95:137 --> 10.255.3.255:137
[08:43:43]: UDP 10.255.2.95:137 --> 10.255.3.255:137
[08:43:43]: TCP 10.255.2.100:53675 --> 10.0.0.10:110
[08:43:43]: TCP 10.0.0.10:110 --> 10.255.2.100:53675
[08:43:43]: UDP 10.255.2.100:17500 --> 255.255.255.255:17500
[08:43:44]: UDP 10.255.2.95:137 --> 10.255.3.255:137
[08:43:44]: UDP 10.255.2.95:137 --> 10.255.3.255:137
[08:43:44]: UDP 10.255.2.95:137 --> 10.255.3.255:137
[08:43:44]: UDP 10.255.2.95:137 --> 10.255.3.255:137
[08:43:44]: UDP 10.255.2.61:137 --> 10.255.3.255:137
[08:43:44]: UDP 10.255.2.100:17500 --> 10.255.3.255:17500
[08:43:44]: UDP 10.255.2.100:17500 --> 255.255.255.255:17500
[08:43:44]: UDP 10.255.2.100:17500 --> 255.255.255.255:17500
[08:43:44]: UDP 10.255.2.102:49152 --> 10.255.3.255:1947
[08:43:44]: Suspicious packet detected

[08:43:44]: UDP 10.255.2.95:137 --> 10.255.3.255:137

[08:43:44]: Suspicious packet detected

[08:43:44]: TCP 10.255.2.100:53707 --> 2.18.19.235:443

[08:43:44]: Suspicious packet detected

[08:43:44]: TCP 10.255.2.100:53675 --> 10.0.0.10:110

[08:43:44]: UDP 10.255.2.101:62749 --> 224.0.0.252:5355

[08:43:44]: Suspicious packet detected

[08:43:44]: TCP 2.18.19.235:443 --> 10.255.2.100:53707

[6:38:47 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:38:47 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:38:47 PM]: UDP 192.168.5.241:17500 --> 192.168.5.255:17500

[6:38:47 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:38:48 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:38:48 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:38:48 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:38:48 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:38:48 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:38:48 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:38:49 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:38:49 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:38:49 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:38:49 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:38:49 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:38:49 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:38:49 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:38:52 PM]: TCP 192.168.5.241:56260 --> 74.125.129.125:5222

[6:38:52 PM]: TCP 74.125.129.125:5222 --> 192.168.5.241:56260

[6:38:53 PM]: Suspicious packet detected

[6:38:54 PM]: Suspicious packet detected

[6:38:58 PM]: UDP 192.168.5.241:138 --> 192.168.5.255:138

[6:39:01 PM]: UDP 192.168.5.1:5678 --> 255.255.255.255:5678

[6:39:09 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:09 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:09 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:09 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:09 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:09 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:09 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:10 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:10 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:10 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:10 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:10 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:10 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:12 PM]: TCP 108.160.163.39:80 --> 192.168.5.241:56303
[6:39:12 PM]: TCP 192.168.5.241:56303 --> 108.160.163.39:80
[6:39:12 PM]: TCP 108.160.163.39:80 --> 192.168.5.241:56303
[6:39:16 PM]: Suspicious packet detected
[6:39:16 PM]: Suspicious packet detected
[6:39:16 PM]: Suspicious packet detected
[6:39:17 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500
[6:39:17 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500
[6:39:17 PM]: UDP 192.168.5.241:17500 --> 192.168.5.255:17500
[6:39:17 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500
[6:39:22 PM]: TCP 192.168.5.241:56260 --> 74.125.129.125:5222
[6:39:22 PM]: TCP 74.125.129.125:5222 --> 192.168.5.241:56260
[6:39:28 PM]: Suspicious packet detected
[6:39:28 PM]: Suspicious packet detected
[6:39:30 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:30 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:30 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:30 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:30 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:30 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:31 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:39:31 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:31 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:31 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:31 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:31 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:31 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:32 PM]: Packet Safe Value set to 0.45

[6:39:38 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:39:38 PM]: UDP 192.168.5.241:60003 --> 224.0.0.252:5355

[6:39:38 PM]: UDP 192.168.5.241:61294 --> 224.0.0.252:5355

[6:39:39 PM]: UDP 192.168.5.241:61294 --> 224.0.0.252:5355

[6:39:39 PM]: UDP 192.168.5.241:60003 --> 224.0.0.252:5355

[6:39:39 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:39:40 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:39:40 PM]: Suspicious packet detected

[6:39:40 PM]: Suspicious packet detected

[6:39:41 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:39:41 PM]: UDP 192.168.5.241:58695 --> 224.0.0.252:5355

[6:39:41 PM]: UDP 192.168.5.241:52894 --> 224.0.0.252:5355

[6:39:41 PM]: UDP 192.168.5.241:52894 --> 224.0.0.252:5355

[6:39:41 PM]: UDP 192.168.5.241:58695 --> 224.0.0.252:5355

[6:39:41 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:39:42 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:39:42 PM]: Suspicious packet detected

[6:39:42 PM]: Suspicious packet detected

[6:39:43 PM]: Suspicious packet detected

[6:39:43 PM]: Suspicious packet detected

[6:39:44 PM]: Suspicious packet detected

[6:39:46 PM]: Suspicious packet detected

[6:39:47 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:39:47 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:39:47 PM]: UDP 192.168.5.241:17500 --> 192.168.5.255:17500

[6:39:47 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:39:49 PM]: TCP 74.125.129.125:5222 --> 192.168.5.241:56260

[6:39:49 PM]: TCP 192.168.5.241:56260 --> 74.125.129.125:5222

[6:39:51 PM]: Suspicious packet detected

[6:39:51 PM]: Suspicious packet detected

[6:39:51 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:51 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:51 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:51 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:51 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:51 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:52 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:52 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:52 PM]: TCP 192.168.5.241:56260 --> 74.125.129.125:5222

[6:39:52 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:52 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:52 PM]: TCP 74.125.129.125:5222 --> 192.168.5.241:56260

[6:39:52 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:52 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:52 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:39:55 PM]: Suspicious packet detected

[6:39:58 PM]: Suspicious packet detected

[6:40:01 PM]: UDP 192.168.5.1:5678 --> 255.255.255.255:5678

[6:40:07 PM]: TCP 108.160.163.39:80 --> 192.168.5.241:56303

[6:40:07 PM]: TCP 192.168.5.241:56303 --> 108.160.163.39:80

[6:40:07 PM]: TCP 108.160.163.39:80 --> 192.168.5.241:56303

[6:40:08 PM]: Suspicious packet detected

[6:40:12 PM]: Suspicious packet detected

[6:40:12 PM]: Suspicious packet detected

[6:40:12 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:12 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:12 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:12 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:12 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:12 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:12 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:13 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:13 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:13 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:13 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:13 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:13 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:17 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:40:17 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:40:17 PM]: UDP 192.168.5.241:17500 --> 192.168.5.255:17500

[6:40:17 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:40:22 PM]: TCP 192.168.5.241:56260 --> 74.125.129.125:5222

[6:40:22 PM]: TCP 74.125.129.125:5222 --> 192.168.5.241:56260

[6:40:25 PM]: Suspicious packet detected

[6:40:28 PM]: Suspicious packet detected

[6:40:33 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:33 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:33 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:33 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:33 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:33 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:34 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:34 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:34 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:34 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:34 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:34 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:34 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:38 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:40:38 PM]: UDP 192.168.5.241:53803 --> 224.0.0.252:5355

[6:40:38 PM]: UDP 192.168.5.241:49306 --> 224.0.0.252:5355

[6:40:39 PM]: UDP 192.168.5.241:53803 --> 224.0.0.252:5355

[6:40:39 PM]: UDP 192.168.5.241:49306 --> 224.0.0.252:5355



[6:40:39 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:40:40 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:40:40 PM]: Suspicious packet detected

[6:40:40 PM]: Suspicious packet detected

[6:40:41 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:40:41 PM]: UDP 192.168.5.241:65108 --> 224.0.0.252:5355

[6:40:41 PM]: UDP 192.168.5.241:52086 --> 224.0.0.252:5355

[6:40:41 PM]: UDP 192.168.5.241:65108 --> 224.0.0.252:5355

[6:40:41 PM]: UDP 192.168.5.241:52086 --> 224.0.0.252:5355

[6:40:41 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:40:42 PM]: Suspicious packet detected

[6:40:42 PM]: Suspicious packet detected

[6:40:42 PM]: Suspicious packet detected

[6:40:42 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:40:46 PM]: Suspicious packet detected

[6:40:46 PM]: Suspicious packet detected

[6:40:46 PM]: Suspicious packet detected

[6:40:47 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:40:47 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:40:47 PM]: UDP 192.168.5.241:17500 --> 192.168.5.255:17500

[6:40:47 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:40:52 PM]: TCP 192.168.5.241:56260 --> 74.125.129.125:5222

[6:40:52 PM]: TCP 74.125.129.125:5222 --> 192.168.5.241:56260

[6:40:54 PM]: Suspicious packet detected

[6:40:54 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:54 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:54 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:54 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:54 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:54 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:55 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:55 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:55 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:55 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:55 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:55 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:55 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:40:56 PM]: Suspicious packet detected

[6:41:01 PM]: UDP 192.168.5.1:5678 --> 255.255.255.255:5678

[6:41:03 PM]: TCP 108.160.163.39:80 --> 192.168.5.241:56303

[6:41:03 PM]: TCP 192.168.5.241:56303 --> 108.160.163.39:80

[6:41:03 PM]: TCP 108.160.163.39:80 --> 192.168.5.241:56303

[6:41:04 PM]: Suspicious packet detected

[6:41:04 PM]: Suspicious packet detected

[6:41:07 PM]: Suspicious packet detected

[6:41:15 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:41:15 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:41:15 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:41:15 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:41:15 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:41:15 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:41:15 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:41:16 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:41:16 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:41:16 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:41:16 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:41:16 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:41:16 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:41:17 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:41:17 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:41:17 PM]: UDP 192.168.5.241:17500 --> 192.168.5.255:17500

[6:41:17 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:41:22 PM]: TCP 192.168.5.241:56260 --> 74.125.129.125:5222

[6:41:22 PM]: TCP 74.125.129.125:5222 --> 192.168.5.241:56260

[6:41:26 PM]: Suspicious packet detected

[6:41:26 PM]: Suspicious packet detected

[6:41:36 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:36 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:36 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:36 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:36 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:36 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:37 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:37 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:37 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:37 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:37 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:37 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:37 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:37 PM]: UDP 192.168.0.1:49683 --> 239.255.255.250:1900
[6:41:37 PM]: UDP 192.168.0.1:49683 --> 239.255.255.250:1900
[6:41:37 PM]: UDP 192.168.0.1:49683 --> 239.255.255.250:1900
[6:41:37 PM]: UDP 192.168.0.1:49683 --> 239.255.255.250:1900
[6:41:37 PM]: UDP 192.168.0.1:49683 --> 239.255.255.250:1900
[6:41:37 PM]: UDP 192.168.0.1:49683 --> 239.255.255.250:1900
[6:41:37 PM]: UDP 192.168.0.1:49683 --> 239.255.255.250:1900
[6:41:38 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137
[6:41:38 PM]: UDP 192.168.5.241:58669 --> 224.0.0.252:5355
[6:41:38 PM]: UDP 192.168.5.241:65533 --> 224.0.0.252:5355
[6:41:39 PM]: UDP 192.168.5.241:58669 --> 224.0.0.252:5355
[6:41:39 PM]: UDP 192.168.5.241:65533 --> 224.0.0.252:5355
[6:41:39 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137
[6:41:40 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137
[6:41:41 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137
[6:41:41 PM]: UDP 192.168.5.241:62458 --> 224.0.0.252:5355
[6:41:41 PM]: UDP 192.168.5.241:58326 --> 224.0.0.252:5355
[6:41:41 PM]: UDP 192.168.5.241:62458 --> 224.0.0.252:5355
[6:41:41 PM]: UDP 192.168.5.241:58326 --> 224.0.0.252:5355

[6:41:41 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137
[6:41:42 PM]: Suspicious packet detected
[6:41:42 PM]: Suspicious packet detected
[6:41:42 PM]: Suspicious packet detected
[6:41:42 PM]: Suspicious packet detected
[6:41:42 PM]: Suspicious packet detected
[6:41:42 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137
[6:41:42 PM]: Suspicious packet detected
[6:41:45 PM]: Suspicious packet detected
[6:41:45 PM]: Suspicious packet detected
[6:41:47 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500
[6:41:47 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500
[6:41:47 PM]: UDP 192.168.5.241:17500 --> 192.168.5.255:17500
[6:41:47 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500
[6:41:52 PM]: TCP 192.168.5.241:56260 --> 74.125.129.125:5222
[6:41:52 PM]: TCP 74.125.129.125:5222 --> 192.168.5.241:56260
[6:41:56 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137
[6:41:56 PM]: UDP 192.168.5.241:58267 --> 224.0.0.252:5355
[6:41:56 PM]: UDP 192.168.5.241:58267 --> 224.0.0.252:5355
[6:41:57 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137
[6:41:57 PM]: Suspicious packet detected
[6:41:57 PM]: Suspicious packet detected
[6:41:57 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:57 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:57 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:57 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:57 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137
[6:41:57 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:57 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:58 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:58 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:58 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:41:58 PM]: Suspicious packet detected

[6:41:58 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:41:58 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:41:58 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:41:58 PM]: TCP 108.160.163.39:80 --> 192.168.5.241:56303

[6:41:58 PM]: TCP 192.168.5.241:56303 --> 108.160.163.39:80

[6:41:58 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:41:58 PM]: TCP 108.160.163.39:80 --> 192.168.5.241:56303

[6:42:01 PM]: UDP 192.168.5.1:5678 --> 255.255.255.255:5678

[6:42:01 PM]: Suspicious packet detected

[6:42:02 PM]: Suspicious packet detected

[6:42:05 PM]: Suspicious packet detected

[6:42:05 PM]: Suspicious packet detected

[6:42:14 PM]: UDP 192.168.5.241:62854 --> 192.168.5.1:53

[6:42:14 PM]: UDP 192.168.5.1:53 --> 192.168.5.241:62854

[6:42:14 PM]: TCP 192.168.5.241:56077 --> 108.160.165.84:443

[6:42:14 PM]: TCP 108.160.165.84:443 --> 192.168.5.241:56077

[6:42:14 PM]: TCP 192.168.5.241:56077 --> 108.160.165.84:443

[6:42:14 PM]: TCP 192.168.5.241:56077 --> 108.160.165.84:443

[6:42:14 PM]: TCP 108.160.165.84:443 --> 192.168.5.241:56077

[6:42:14 PM]: TCP 108.160.165.84:443 --> 192.168.5.241:56077

[6:42:14 PM]: TCP 108.160.165.84:443 --> 192.168.5.241:56077

[6:42:14 PM]: TCP 192.168.5.241:56077 --> 108.160.165.84:443

[6:42:14 PM]: TCP 108.160.165.84:443 --> 192.168.5.241:56077

[6:42:14 PM]: TCP 192.168.5.241:56077 --> 108.160.165.84:443

[6:42:15 PM]: TCP 108.160.165.84:443 --> 192.168.5.241:56077

[6:42:15 PM]: TCP 192.168.5.241:56077 --> 108.160.165.84:443

[6:42:15 PM]: Suspicious packet detected

[6:42:15 PM]: Suspicious packet detected

[6:42:15 PM]: TCP 108.160.165.84:443 --> 192.168.5.241:56077

[6:42:15 PM]: TCP 192.168.5.241:56077 --> 108.160.165.84:443

[6:42:15 PM]: TCP 108.160.165.84:443 --> 192.168.5.241:56077

[6:42:15 PM]: TCP 192.168.5.241:56077 --> 108.160.165.84:443

[6:42:16 PM]: TCP 108.160.165.84:443 --> 192.168.5.241:56077

[6:42:16 PM]: TCP 192.168.5.241:56077 --> 108.160.165.84:443
[6:42:16 PM]: Suspicious packet detected
[6:42:16 PM]: Suspicious packet detected
[6:42:16 PM]: Suspicious packet detected
[6:42:16 PM]: Suspicious packet detected
[6:42:17 PM]: Suspicious packet detected
[6:42:17 PM]: Suspicious packet detected
[6:42:17 PM]: Suspicious packet detected
[6:42:17 PM]: Suspicious packet detected
[6:42:17 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500
[6:42:17 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500
[6:42:17 PM]: UDP 192.168.5.241:17500 --> 192.168.5.255:17500
[6:42:17 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500
[6:42:18 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:42:18 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:42:18 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:42:18 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:42:18 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:42:18 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:42:18 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:42:19 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:42:19 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:42:19 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:42:19 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:42:19 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:42:19 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900
[6:42:19 PM]: Suspicious packet detected
[6:42:19 PM]: Suspicious packet detected
[6:42:19 PM]: Suspicious packet detected
[6:42:19 PM]: Suspicious packet detected
[6:42:20 PM]: Suspicious packet detected
[6:42:22 PM]: TCP 192.168.5.241:56260 --> 74.125.129.125:5222
[6:42:22 PM]: TCP 74.125.129.125:5222 --> 192.168.5.241:56260

[6:44:03 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:03 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:03 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:04 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:04 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:04 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:04 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:04 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:04 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:04 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:17 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:44:17 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:44:17 PM]: UDP 192.168.5.241:17500 --> 192.168.5.255:17500

[6:44:17 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:44:22 PM]: TCP 192.168.5.241:56260 --> 74.125.129.125:5222

[6:44:23 PM]: TCP 74.125.129.125:5222 --> 192.168.5.241:56260

[6:44:24 PM]: Suspicious packet detected

[6:44:24 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:24 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:24 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:24 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:24 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:24 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:24 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:25 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:25 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:25 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:25 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:25 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:25 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:27 PM]: Suspicious packet detected

[6:44:38 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:44:38 PM]: UDP 192.168.5.241:58061 --> 224.0.0.252:5355

[6:44:38 PM]: UDP 192.168.5.241:57842 --> 224.0.0.252:5355

[6:44:39 PM]: UDP 192.168.5.241:57842 --> 224.0.0.252:5355

[6:44:39 PM]: UDP 192.168.5.241:58061 --> 224.0.0.252:5355

[6:44:39 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:44:40 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:44:40 PM]: Suspicious packet detected

[6:44:40 PM]: Suspicious packet detected

[6:44:40 PM]: Suspicious packet detected

[6:44:41 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:44:41 PM]: UDP 192.168.5.241:63136 --> 224.0.0.252:5355

[6:44:41 PM]: UDP 192.168.5.241:52174 --> 224.0.0.252:5355

[6:44:41 PM]: UDP 192.168.5.241:52174 --> 224.0.0.252:5355

[6:44:41 PM]: UDP 192.168.5.241:63136 --> 224.0.0.252:5355

[6:44:41 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:44:42 PM]: UDP 192.168.5.241:137 --> 192.168.5.255:137

[6:44:42 PM]: Suspicious packet detected

[6:44:43 PM]: Suspicious packet detected

[6:44:43 PM]: Suspicious packet detected

[6:44:45 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:45 PM]: TCP 108.160.163.39:80 --> 192.168.5.241:56303

[6:44:45 PM]: TCP 192.168.5.241:56303 --> 108.160.163.39:80

[6:44:45 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:45 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:45 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:45 PM]: TCP 108.160.163.39:80 --> 192.168.5.241:56303

[6:44:45 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:45 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:46 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:46 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:46 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:46 PM]: Suspicious packet detected

[6:44:46 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:46 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:46 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:46 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:44:47 PM]: Suspicious packet detected

[6:44:47 PM]: Suspicious packet detected

[6:44:47 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:44:47 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:44:47 PM]: UDP 192.168.5.241:17500 --> 192.168.5.255:17500

[6:44:47 PM]: UDP 192.168.5.241:17500 --> 255.255.255.255:17500

[6:44:47 PM]: Suspicious packet detected

[6:44:49 PM]: Suspicious packet detected

[6:44:52 PM]: TCP 192.168.5.241:56260 --> 74.125.129.125:5222

[6:44:53 PM]: TCP 74.125.129.125:5222 --> 192.168.5.241:56260

[6:44:53 PM]: TCP 74.125.129.125:5222 --> 192.168.5.241:56260

[6:44:53 PM]: TCP 192.168.5.241:56260 --> 74.125.129.125:5222

[6:44:54 PM]: Suspicious packet detected

[6:44:54 PM]: Suspicious packet detected

[6:44:58 PM]: Suspicious packet detected

[6:44:58 PM]: Suspicious packet detected

[6:45:01 PM]: UDP 192.168.5.1:5678 --> 255.255.255.255:5678

[6:45:06 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:45:06 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:45:06 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:45:06 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:45:06 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:45:06 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:45:07 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:45:07 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:45:07 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:45:07 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:45:07 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:45:07 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

[6:45:07 PM]: UDP 192.168.0.1:36529 --> 239.255.255.250:1900

APPENDIX V

Log Files

635077060423189783	635077060423539803	635077060423619807	635077060443470943
635077060443840964	635077060443890967	635077060443950970	635077060444010974
635077060444060977	635077060444170983	635077060444240987	635077060444340993
635077060444400996	635077060458371795	635077060458491802	635077060458561806
635077060487593467	635077060488333509	635077060489113553	635077060489923600
635077060490743647	635077060491543692	635077060492133726	635077060493053779
635077060493863825	635077060494583866	635077060529375856	635077060562997779
635077060563727821	635077060564507866	635077060610710508	635077060649492727
635077060664483584	635077060665613649	635077060666803717	635077060667923781
635077060669843891	635077060676074247	635077060677314318	635077060678924410
635077060680154480	635077060681074533	635077060717256603	635077060718666683
635077060719936756	635077060721296834	635077060756438844	635077060757918928
635077060758678972	635077060759419014	635077060760149056	635077060760929100
635077060766939444	635077060767699488	635077060768499533	635077060769349582
635077060770269635	635077060771109683	635077060771999734	635077060773219803

635077061066036552	635077061068886715	635077061069716762	635077061308290408
635077061308340411	635077061337692089	635077061338062111	635077061338232120
635077061338332126	635077061338422131	635077061338522137	635077061338612142
635077061338672145	635077061343362414	635077061343412417	635077061343482421
635077061343732435	635077061343852442	635077061344452476	635077061346222577
635077061346422589	635077061349112743	635077061349472763	635077061350112800
635077061355033081	635077061356243150	635077061388715008	635077061388775011
635077061388835015	635077061388895018	635077061388965022	635077061400685692
635077061426817187	635077061426867190	635077061426947194	635077061427027199
635077061646819770	635077061660210536	635077061660440550	635077061660500553
635077061660560556	635077061660620560	635077061675501411	635077061675551414
635077061685281970	635077061685341974	635077061685391977	635077061685471981
635077061685521984	635077061685661992	635077061715953725	635077061717193796
635077061717263800	635077061888193576	635077061888243579	635077061931396047
635077061931446050	635077061940806586	635077061940876590	635077061956517484
635077061956577488	635077061956657492	635077062551811533	635077062552581577

635077466118123257	635077466158455564	635077466159005596	635077466168526140
635077466181066858	635077466182056914	635077466182746954	635077466183627004
635077466184407049	635077466188697294	635077466189847360	635077466190597403
635077466195327673	635077466196377733	635077466197237782	635077466199887934
635077466200717982	635077466201548029	635077466202348075	635077466203158121
635077466216198867	635077466217188924	635077466243640437	635077466244080462
635077466244950511	635077466245760558	635077466250240814	635077466250780845
635077466270851993	635077466277172354	635077466278152411	635077466292063206
635077466292993259	635077466293883310	635077466294283333	635077466295303392
635077466298323564	635077466299053606	635077466299813649	635077466300803706
635077466329285335	635077466329795364	635077466331115440	635077466331365454
635077466332565523	635077466333425572	635077466334775649	635077466334785650
635077466337595811	635077466338695873	635077466342816109	635077466343986176
635077466345606269	635077466347736391	635077466375437975	635077466410809998
635077466412690106	635077466414690220	635077466415130245	635077466415750281
635077466416710336	635077466416740337	635077466417910404	635077466418400432

635077466419250481	635077466419860516	635077466446572044	635077466447022069
635077466447522098	635077466467973268	635077466469853375	635077466471033443
635077466472313516	635077466476323745	635077466477323802	635077466478603876
635077466481254027	635077466490874578	635077466495234827	635077466495974869
635077466497654965	635077466501905208	635077466505325404	635077466508925610
635077466512985842	635077466513025845	635077466514765944	635077466514875950
635077466515695997	635077466516006015	635077466518986185	635077466519146195
635077466519316204	635077466519386208	635077466519746229	635077466520366264
635077466520476271	635077466521526331	635077466521706341	635077466521916353
635077466523266430	635077466523276431	635077466526496615	635077466526576620
635077466527386666	635077466529536789	635077466529546789	635077466531216885
635077466532626966	635077466535087106	635077466535547133	635077466536477186
635077466536777203	635077466537377237	635077466537437241	635077466537497244
635077466538207285	635077466538807319	635077466539247344	635077466539897381
635077466542147510	635077466542927555	635077466543677598	635077466545167683
635077466545247688	635077466546557762	635077466546727772	635077466546777775

635077466549227915	635077466550868009	635077466552138082	635077466552628110
635077466552658111	635077466552868123	635077466553398154	635077466553648168
635077466553738173	635077466557738402	635077466560048534	635077466562328664
635077466564108766	635077466566218887	635077466570409127	635077466571439186
635077466573329294	635077466575799435	635077466592230375	635077466592950416
635077466597160657	635077466597730689	635077466598900756	635077466599550793
635077466600160828	635077466601790922	635077466602220946	635077466602660971
635077466603090996	635077466603150999	635077466605081110	635077466605141113
635077466605741148	635077466606751205	635077466607751263	635077466608151285
635077466608241291	635077466609441359	635077466610121398	635077466610191402
635077466610351411	635077466612121512	635077466612621541	635077466613471590
635077466614501649	635077466615131685	635077466615301694	635077466616801780
635077466620281979	635077466620551995	635077466620772007	635077466623272150
635077466625322268	635077466626012307	635077466626282322	635077466627222376
635077466627492392	635077466628612456	635077466640293124	635077466640533138
635077466640603142	635077466640673146	635077466648433589	635077466648483592

635077466648543596	635077466651003736	635077466651143744	635077466651293753
635077466669724807	635077466671364901	635077466672204949	635077466673895046
635077466674345071	635077466684395646	635077466687675834	635077466691136032
635077466691146032	635077466691266039	635077466692646118	635077466692776126
635077466693036141	635077466693086143	635077466713827330	635077466713877333
635258906592522524	635258906592562518	635258906592612562	635258906592812674
635258906592862713	635258906592912748	635258906592952775	635258906593002810
635258906581404408	635258906581494450	635258906581594487	635258906581704575
635258906581794612	635258906581844651	635258906562336009	635258906562386040
635258906562456089	635258906541070743	635258906541150784	635258906541743719
635258906541783737	635258906541848822	635258906549507893	635258906445917248
635258906447955877	635258906410893296	635258906410973279	635258906419569571
635258906419639422	635258906380888361	635258906348833632	635258906339722176
635258906328004787	635258906328084741	635258906294632113	635258906294782222
635258906286003217	635258906278316653	635258906278396538	635258906278476550
635258906278576617	635258906260563436	635258906260633476	635258906260953697
635258906261033742	635258906254141720	635258906254221744	635258906254371858

635258906254451870	635258906254511906	635258906229998298	635258906230078310
635258906229778111	635258906229858164	635258906229928229	635258906214251167
635258906214311202	635258906214431311	635258906214521340	635258906214591396
635258906214671445	635258906203639298	635258906203719372	635258906203809417
635258906172065906	635258906175213220	635258906151346631	635258906120761434
635258906101737589	635258906102235273	635258906102335303	635258906102425397
635258906102515426	635258906097327033	635258906097397319	635258906097477323
635258906097557372	635258906029513632	635258905882110731	635258905869314398
635258905830274124	635258905739142165	635258905669925565	635258905670047746
635258905670117795	635258905677882544	635258905677962416	635258905678052478
635258905678132490	635258905678202563	635258905678252566	635258905678302601
635258905678362637	635258905678442714	635258905678492741	635258905678722880
635258905678762914	635258905678812937	635258905678852960	635258905678902995
635258905678943022	635258905678993057	635258905679043100	635258905679093123
635258905669055018	635258905669135071	635258905669225096	635258905669315186
635258905669395203	635258905669485293	635258905669525307	635258905669575318
635258905669625378	635258905669675389	635258905669715436	635258905669765471

635258905669805510	635258905669845512	635258905669885538	635258905630626822
635258905630696862	635258905630876940	635258905631139186	635258905631239253
635258905631549456	635258905631699525	635258905631749560	635258905631859631
635258905631899687	635258905631949722	635258905631999724	635258905632039771
635258905632261350	635258905632311377	635258905632351404	635258905632391414
635258905632491493	635258905632581554	635258905632621581	635258905632671637
635258905632711647	635258905632741669	635258905632781695	635258905632821722
635258905632861748	635258905632901763	635258905632931793	635258905632971819
635258905633021846	635258905633061865	635258905633101900	635258905633141934
635258905633374565	635258905633414591	635258905633454618	635258905633494644
635258905633524666	635258905633837029	635258905633927091	635258905633967118
635258905634007148	635258905634047175	635258905623683656	635258905623763709
635258905623853775	635258905623923811	635258905623973875	635258905624033911
635258905624093955	635258905624163974	635258905624224006	635258905624274062
635258905624314088	635258905624354115	635258905624414134	635258905624474195
635258905624524205	635258905624574249	635258905624624280	635258905624724338
635258905624774385	635258905624814396	635258905624874440	635258905624914466

635258905624974502	635258905625014541	635258905625054567	635258905625154621
635258905625194656	635258905625254692	635258905625294722	635258905625344770
635258905625394781	635258905625444824	635258905625494847	635258905625534878
635258905625574904	635258905625624939	635258905625684996	635258905625735010
635258905625775041	635258905625825068	635258905625865095	635258905625905121
635258905625955156	635258905626005188	635258905626045214	635258905626095270
635258905626135276	635258905626175303	635258905589762531	635258905590062718
635258905590142763	635258905590192807	635258905590232837	635258905590272868
635258905590322916	635258905590372922	635258905590422953	635258905579997986
635258905580048013	635258905580088040	635258905580128070	635258905580178085
635258905580218112	635258905580258130	635258905580308161	635258905580358200
635258905580398219	635258905580448254	635258905580528315	635258905580578342
635258905580648399	635258905580698418	635258905580758474	635258905580808493
635258905580858528	635258905580898567	635258905580958599	635258905580998625
635258905581038656	635258905581088679	635258905581128710	635258905587711210
635258905587781209	635258905587861311	635258905587961337	635258905588051431
635258905588141448	635258905588201479	635258905588261548	635258905588311591



635258905588371619	635258905588421695	635258905588471665	635258905588521692
635258905588561743	635258905588611877	635258905588771856	635258905588821895
635258905588871951	635258905588921966	635258905588961984	635258905589012023
635258905589052054	635258905589092081	635258905589142108	635258905589192143
635258905589232169	635258905589272204	635258905589332236	635258905589392276
635258905589432302	635258905589482346	635258905589522356	635258905589592396
635258905589632435	635258905589682466	635258905589722501	635258905578952484
635258905579032488	635258905579122579	635258905579517027	635258905579587071
635258905579707151	635258905579757186	635258905579827822	635258905579867869
635258905579907920	635258905579957955	635258905555224263	635258905555294287
635258905555394354	635258905555645458	635258905555685476	635258905555735524
635258905555785538	635258905555835574	635258905555895773	635258905555935640
635258905555995680	635258905556045715	635258905556105755	635258905556155802
635258905545771379	635258905545811381	635258905545851432	635258905545891467
635258905545941502	635258905532004798	635258905532074843	635258905539062617
635258905526521370	635258904341051924	635258904284830049	635258904243211322
635257417597130779	635257417581967288	635257417539219143	635257417539283621

635257417539363690 635257417491308172 635253746480862278 635253746480922425

635253746480982420 635253746481042501 635253746481082479 635253746481112509

635253746464657313 635253746464717304 635253746464777381 635253746444136559

635253746444197614 635253746425959566 635253746426019651 635253746400519756

635253746400910314 635253746401572184 635253746377436844 635253746377576223

635253746369736164 635253746369796241 635253746325813548 635253746325868034

635253746325918143 635253746325988192 635253746326058200 635253746326118244

635253746326168267 635253746326446024 635253746315741360 635253746299449376

635253746300203504 635253746299079245 635253746299129125 635253746299199124

635253746299279182 635253746276147191 635253746276197279 635253746276257324

635253746276307392 635253746267515887 635253746267576428 635253746267646502

635253746267787174 635253746267847173 635253746267907209 635253746227676565

635253746227726428 635253746227786460 635253746227850355 635253746196599969

635253746188886308 635253746165564106 635253746165613965 635253746146439004

635253746146491211 635253746146682617 635253746146752711 635253746146822702

635253746146882779 635253746146972809 635253746137534620 635253746137933902

635253746137993946 635253746138054027 635253746138094009 635253746138134064

635253746111431536	635253746111487771	635253746111547758	635253746111617815
635253746111707897	635253746111780590	635253746111840671	635253746111900707
635253746111950743	635253746091159157	635253746091218081	635253746091278129
635253746091564926	635253746091748269	635253746091798333	635253746091858390
635253746091903462	635253746091943554	635253746091973535	635253746092013562
635253746092063622	635253746092103624	635253746092143647	635253746092173689
635253746092203691	635253746092233730	635253746092263752	635253746092293770
635253746092323771	635253746092363810	635253746092383824	635253746092413850
635253746092443847	635253746092473890	635253746092503892	635253746074064331
635253746074116513	635253746074176504	635253746074236544	635253746074296588
635253746074356624	635253746074561750	635253746074621745	635253746074681830
635253746074741829	635253746074791869	635253746074831908	635253746075136916
635253746075186931	635253746075236954	635253746075277013	635253746075317016
635253746075357067	635253746075538049	635253746043684068	635253746043752323
635253746043782423	635253746043832455	635253746043842455	635253746043912508
635253746044360060	635253746020739956	635253746020790045	635253746020861523
635253746021244757	635253746021447305	635253746021507328	635253746021577340

635253746021643575	635253746021683655	635253746021713657	635253746021743679
635253746021773697	635253746021812488	635253746021852925	635253746021882972
635253746021913015	635253746021953005	635253746021983019	635253746022013045
635253746022043050	635253746022093086	635253746022123092	635253746026608688
635253746026658802	635253746026728810	635253746026788850	635253746026848931
635253746026898971	635253746004161741	635253746004222098	635253746004282175
635253746004342219	635253746004412268	635253746004471113	635253746004541158
635253746004918546	635253746004978562	635253746005233432	635253746005303554
635253746005353549	635253746005494258	635253746005537955	635253746005567965
635253746005597995	635253746005628017	635253745982356384	635253745982408607
635253745972200690	635253745972253263	635253745972323340	635253745972383421
635253745967770580	635253745967820476	635253745967890480	635253745967950520
635253745968010560	635253745968070637	635253745968251460	635253745968331492
635253745968391508	635253745968461586	635253745968519524	635253745968579593
635253745968618379	635253745968658406	635253745968698433	635253745968728422
635253745968768482	635253745968808500	635253745968848511	635253745968878529
635253745968908567	635253745968928572	635253745968975738	635253745969005797

635253745969056091	635253745969076105	635253745969104427	635253745952294845
635253745952354668	635253745952424712	635253745952539390	635253745952661663
635253745952691738	635253745952731736	635253745952963086	635253745952993109
635253745953043144	635253745953083183	635253745953118513	635253745953158515
635253745930005833	635253745930065865	635253745930135922	635253745930185941
635253745930229273	635253745930289317	635253745930329344	635253745930359386
635253745930399389	635253745930439432	635253745930479438	635253745930509460
635253745930539494	635253745930579513	635253745930619540	635253745930659570
635253745930689588	635253745930729611	635253745920017653	635253745920073862
635253745920133972	635253745920539108	635253745920599153	635253745920666332
635253745920736414	635253745920806508	635253745920866519	635253745920896516
635253745920936629	635253745920966651	635253745920996649	635253745921026671
635253745921046709	635253745921086707	635253745921106725	635253745921136751
635253745921176790	635253745921206791	635253745921236813	635253745921266852
635253745921296849	635253745921336901	635253745921366915	635253745921396916
635253745921436943	635253745924244176	635253745924350470	635253745924400543
635253745924460558	635253745924520594	635253745924600648	635253745924660725

635253745924736989	635253745924777073	635253745925015218	635253745925057158
635253745925087201	635253745925117223	635253745925157266	635253745925187267
635253745925227298	635253745925353639	635253745925383690	635253745925413708
635253745925453735	635253745925503786	635253745925533809	635253745925563818
635253745925593836	635253745925633851	635253745925685393	635253745925724771
635253745925763546	635253745925794405	635253745925824435	635253745925871100
635253745925892785	635253745926185226	635253745926217334	635253745926243727
635253745926273753	635253745926412295	635253745926452317	635253745926482335
635253745929413754	635253745929761510	635253745929831547	635253745929891546
635253745872041679	635253745859892921	635253745860309577	635253745860389712
635253745860631728	635253745860691773	635253745859691720	635253745859752786
635253745859812913	635253745843449857	635253745843519979	635253745843570011
635253745843650023	635253745843700059	635253745843760099	635253745843790113
635253745843840148	635253745843890204	635253745814925649	635253745794122339
635253745794471676	635253745794541724	635253745796701436	635253745798677319
635253745798729338	635253745771617293	635253745751821286	635253745751881059
635253745751951141	635253745752021190	635253745754696798	635253745732658469

635253745732718895	635253745688220046	635253745643845974	635253745643916113
635253745644126243	635253745645131874	635253745645181491	635253745645251540
635253745645311580	635253745645371633	635253745488194342	635253745488243044
635253745488373170	635253745488423205	635253745471756919	635253745471806774
635253745471876810	635253745471936863	635253745441096379	635253745441151074
635253745429742622	635253745429805441	635253745400995265	635253745403978230
635253745396610080	635253745375260732	635253745375323317	635253745375408740
635253745375458850	635253745375528907	635253745375588898	635253745375628916
635253745366153878	635253745366207387	635253745366267460	635253745366327541
635253745366397598	635253745341995121	635253745342054935	635253745321867746
635253745321897727	635253745321927753	635253745329653853	635253745301826890
635253745301877028	635253745305947422	635253745306006247	635253745261173715
635253745242517985	635253745242578103	635253745237010183	635253745228642576
635253745228762734	635253745228832738	635253745175407509	635253745175470197
635253745175530188	635253745175580223	635253745175640300	635253745175768328
635253745175828471	635253745176019815	635253745176039923	635253745176099902
635253745176119919	635253745176179922	635253745176199936	635253745143092662

635253745143122639	635253745143162703	635253745143202701	635253745143242728
635253745143272729	635253745115610816	635253745115670811	635253745092808222
635253745097197443	635253745055362494	635253745057292612	635253745031197427
635253745031337340	635253745031397393	635253745034212294	635253745034266061
635253744984899329	635253744984963935	635253744938068507	635253744901562393
635253744891796034	635253744885701190	635253744862429475	635253744862479588
635253744862539628	635253744862595826	635253744862648944	635253744862699062
635253744867217148	635253744867277221	635253744867337261	635253744867397351
635253744867457341	635253744867517382	635253744795207716	635253744752708853
635253744745582005	635253744720941679	635253744719713453	635253744699049387
635253744671562853	635253744671632951	635253744671723017	635253744671808087
635253744674025942	635253744674106053	635253744658369040	635253744658439171
635253744626646084	635253744612844714	635253744614070887	635253744614146108
635253744584333388	635253744550530738	635253744550598106	635253744536606266
635253744517263066	635253744485415797	635253744485425909	635253744486493149
635253744486559860	635253744486599928	635253744486698361	635253744486758438
635253744486808469	635253744486868559	635253744486898531	635253744486968580

635253744487008611 635253744487058642 635134493807134224 635134493807184079

635134493775004873 635134493775050142 635134493725636403 635134493691271102

635134493695982610 635134493696042683 635134493696092751 635134493617072386

635134493603114335 635134493603819809 635134493603859832 635134493603889854

635134493594566780 635134493594616175 635134493570736260 635134493559957752

635134493560007828 635134493559847907 635134493559907733 635134493524132148

635134493524182249 635134493529522131 635134493529572166 635134493497003721

635134493497055550 635134493441088630 635134493445804609 635134493396267448

635134493396316769 635134493351533016 635134493351597727 635134493311851295

635134493313084678 635134493291546604 635134493291596172 635134493267119531

635134493267170486 635134493257414776 635134493257463883 635134493257513968

635134493230577836 635134493230618483 635134493194192258 635134493140686003

635134493140752496 635134493120266281 635134493085486740 635134493067661442

635134493067712898 635134493028840192 635134493028890034 635134492982993852

635134492963817691 635134492950691746 635134492926494709 635134492926541908

635134492927536818 635134492861072241 635134492861122909 635134492866559158

635134492859528601 635134492859575606 635134492859625708 635134492859675698

635134492859725729 635134492844388688 635134492844436289 635134492844486353

635134492844546385 635134492787684892 635134492787732760 635134492787782816

635134492769057785 635134492769107824 635134492751316330 635134492751356426

635134492751406462 635134492756072295 635134492756121698 635134492759310051

635134492738391787 635134492738440328 635134492738490388 635134492738540427

635134492738904859 635134492738924901 635134492738954923 635134492739004954

635134492739064994 635134492739115033 635134492739175065 635134492662193410

635134492662243351 635134492657313018 635134492618543067 635134492561333379

635134492555659864 635134492555721239 635134492555781348 635134492525123130

635134492525678371 635098353466305607 635098353466335666 635098353453227450

635098353436787325 635098353427133606 635098353395066641 635098353380527554

635098353358761096 635098353358791171 635098353347300445 635098353324194495

635098353324224636 635098353324274613 635098353324309410 635098353309373854

635098353309413910 635098353309453981 635098353294976875 635098353295006951

635098353296320887 635098353296350967 635098353262539081 635098353262619212

635098353211988258 635098353212018338 635098353212068373 635098353213010568

635098353213040615 635098353195714291 635098353195824390 635098353195864413

635098353195894435 635098353197985838 635098353198015884 635258906591271688

635258906591541849 635258906591641944 635258906591711976 635258906591761987

635258906591822031 635258906591872062 635258906591912105 635258906591962140

635258906592032164 635258906592092204 635258906592142239 635258906592202279

635258906592262319 635258906592312350 635258906592352377 635258906592392407

635258906592432430 635258906592472456