Analyze the Delay Time by Data Mining for Network Intrusion Prevention System Using Bro

A Thesis submitted to
College of Arts and Sciences (Applied Sciences)
In partial fulfilment of the requirements for the degree
Master of Science (Information Technology)
Universiti Utara Malaysia

By

Kaled Hussain Azrane

KOLEJ SASTERA DAN SAINS
(College of Arts and Sciences)
Universiti Utara Malaysia

PERAKUAN KERJA KERTAS PROJEK
(Certificate of Project Paper)

Saya, yang bertandatangan, memperakukan bahawa
(I, the undersigned, certify that)

KALED HUSSAIN AZRANE
(800293)

calon untuk Ijazah
(candidate for the degree of) **MSc. (Information Technology)**

telah mengemukakan kertas projek yang bertajuk
(has presented his/ her project paper of the following title)

ANALYZE THE DELAY TIME BY DATA MINING FOR NETWORK INTRUSION PREVENSION SYSTEM USING BRO

seperti yang tercatat di muka surat tajuk dan kulit kertas projek
(as it appears on the title page and front cover of project paper)

bahawa kertas projek tersebut boleh diterima dari segi bentuk serta kandungan dan meliputi bidang ilmu dengan memuaskan.
(that the project paper acceptable in form and content, and that a satisfactory knowledge of the field is covered by the project paper).

Nama Penyelia Utama
(Name of Main Supervisor): **ASSOC. PROF. HATIM MOHAMED TAHIR**

Tandatangan
(Signature) : _____

Tarikh
(Date) : 28/4/09.

PERMISSION TO USE

In present this thesis in partial fulfillment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the University Library may make it freely available for inspection. I further agree that permission for copy of this thesis in any manner, in whole or in part, for scholarly purpose may be granted by me supervisor or, in their absence by the Dean of Research and Postgraduate Studies. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not allowed without my written permission. It is also understood that due recognition shall be given to me and to University Utara Malaysia FOR any scholarly use which may be made of any material from my thesis.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part should be addressed to:

Dean of Research and Postgraduate Studies

College of Arts and Sciences

Universiti Utara Malaysis

06010 UUM Sintok

Kedah Darul Aman

Malaysia

ABSTRACT

The important for using the network are increased day by day, and the important for the security for these networks are more important. To implement secure network, the network administrator use several type of security systems and software tools, the most focus systems used in this area are the firewalls and the intrusion detection and prevention systems. There are many features developed every year for these systems and there are many studies done to evaluate and develop these systems, this thesis focus on evaluate the performance for one of famous open free source intrusion detection and prevention system, which is Bro IDS, the thesis will test the performance for Bro in different situations to determine which conditions make Bro work with the minimum delay time for the packets, the thesis will use the data mining tool which it SPSS, to analyse the effects for the main policies on the delay time for the packets when the Bro work as intrusion prevention system.

ACKNOWLEDGMENT

All praise is due to Allah, Most Gracious, and Most Merciful. Without whose help and mercy, I would not have reached this far.

It would not have been possible for me to complete the course of my master without encourage and support of my family. My first expression of gratitude goes to my parents, big brother, other brothers, wife, and my three kids whose gave me the strength to complete this course.

I must convey my gratitude to my supervisor, Assoc. Prof. Hatim Mohamad Tahir for his support, guidance, critical remarks, and advices throughout this study. Also, I want to thank the people who contributed significantly to my work, and my deepest gratitude goes to all of them. Where they provided me with many hours of discussion and led me to ways of conducting data analysis.

I would like to thank my colleagues and friends to many moments of insight, inspiration, laughter, and support throughout my completion of the program.

DEDICATION

I would like to dedicate this thesis to my parents,

big brother, other brothers, wife,

and kids who lovely encouraged

and supported me through all my study.

The motivation for all I do.

CONTENTS

LIST OF FIGURES

ABBREVIATIONS

ASCII	American Standard Code For Information Interchange
CGI	Common Gateway Interface
CPU	Central Processing Unit
DIDS	Distributed Intrusion Detection System
DNS	Domain Name Service
DoS	Denial of Service
FIN	Freedom to Innovate Network
FTP	File Transfer Protocol
GNU	Government of National Unity
HIDS	Host Intrusion Detection System
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
ISP	Internet Service Provider
LAN	Local Area Network
NIC	Network Interface Card
NIDS	Network Intrusion Detection System
OS	Operating System
RFC	Request For Comments
SMB	Server Message Block
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SQL	Sequential Query Language
SSH	Secure Shell
SYN	Synchronize
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URI	Uniform Resource Identifier

CHAPTER ONE

INTRODUCTION

1 INTRODUCTION

Network security is not just about keeping people out of our network. Network security also provides access into our network in the way we want to provide it, allowing people to work together. Strong network security opens up pathways to let authorized people in to our business, regardless of where they are located physically or what kind of connection they have. To improve that and to detect unwanted attacks or even threads, we have to use Network Intrusion Detection System (NIDS). In general, Intrusion Detection System (IDS) is the process of monitoring for and identifying attempted unauthorized system access or manipulation. Most network administrators do IDS all the time without realizing it. Security administrators are constantly checking system and security log files for something suspicious. An antivirus scanner can be considered as IDS when it checks files and disks for known malware. An IDS is tool that can monitor host system changes or sniff network packets off the wire looking for signs of malicious intent.

The upgrading of computer security solutions is non-trivial: many types of intrusion exist, which are diverse in method of action and are constantly evolving. Programs and tools designed to disrupt or damage computer systems are collectively termed malware. The 'infection' of a computer network by malware can result in loss of confidentiality, integrity and availability of data, systems and services. Protection

against such attacks is becoming difficult, due to the increase in the range and volume of malware spreading via the internet. The increased availability of broadband networks means that computer viruses and worms can spread at a rate faster than ever before. Confidential data is now stored on servers worldwide and large scale fraud can be performed from remote locations (Greensmith, 2007).

An intrusion detection system defined by Sommer (2005) as "attempts to detect intrusions". In this study, the work will deal with network based systems, i.e., network intrusion detection systems (NIDS), which is defined by Sommer (2005), as a system, which monitor the network traffic for malicious activity, raising alerts when they detect attacks. And there is other type of intrusion detection system which is hosting based (HIDS) which rely on information gathered on individual hosts. Hybrid systems are both network- and host-based.

There are two types of NIDS: centralized or distributed in control. In centralized control mechanisms, a central entity is responsible for analyzing and processing the logged information provided by the various constituent IDSs. The constituent systems can also be HIDSs. On the other hand, NIDSs can be on distributed architectures. Corporate networks can be spread over great distances. Some attacks target an organization's entire network spread over such big dimensions. Distributed systems could be integrated for performance and operations under such environments. Many features from distributed theory (such as cooperative agents) could be applied to realize operations under such IDSs. Cooperative agents are one of the most important components of distributed intrusion detection architecture. An agent, in general, is an entity that acts for or represents another entity. In the software area, an agent is an autonomous or semi-autonomous piece of software that runs in the background and

performs useful tasks for another. Relative to IDSs, an agent is generally a piece of software that senses intrusions locally and reports attack information to central analysis servers. The cooperative agent's them-selves could form a network among themselves for data transmission and processing. The use of multiple agents across a network allows broader view of the network than may be possible with single IDS or centralized IDSs (Safiee, 2007). Figure 1.1 shows the architectural description of such distributed IDSs as example.

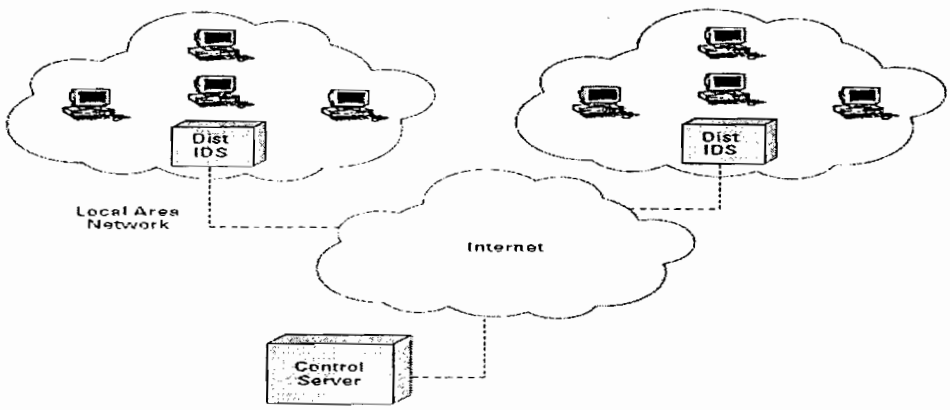

Fig. 1.1: Example of Distributed NIDS (Safiee, 2007).

Organizations and their information systems and networks are faced with security threats from a wide range of sources; including computer assisted fraud, espionage, sabotage, vandalism, as well as acts of God. Attack techniques have become more common, more ambitious, and increasingly sophisticated. Information security is important to businesses, public organizations, and to protect critical infrastructures at a national or global level, where the interconnection of public and private networks and the sharing of information resources increased the difficulty of achieving access control. When trend to distributed computing which also has weakened the effectiveness of central, specialist control. Security cannot be achieved through technical means alone, as it is an inherently human and social problem, and

therefore needs to be supported by appropriate management and procedures (Zanero, 2006).

IDSs typically use packet monitoring with the aid of a probabilistic model to determine whether or not a network is the subject of an intrusion. Primarily effective in determining intrusions in protocol-specific environments where the range of input data is unbounded4, traffic-based IDSs are not as applicable to the embedded systems paradigm. In this case, an applied computational scenario has been established and localized to a set of known behaviors and parameters (Lauf, 2007).

1.1 Work Background

The growing of the computing security is increasing continually, and the growing related with companies that have intranet connected to the internet. At the same time as the number of companies with computers and services accessible to the Internet increases, so does the number of attacks against companies. Furthermore, up to now there is no mechanism that can promise to totally secure a network. In order to maintain an acceptable level of security in this environment, network administrators deploy a variety of perimeter and the host-based tools, including firewalls, intrusion detection systems, patch and version managers, and anti-virus tools in order to deal with the constant threats. These tools form an integrated line of defense against network attacks.

Intrusion Detection has been defined by (Mukherjee, Heberlein, & Levitt, 1994) as "the problem of identifying individuals who are using a computer system without authorization (i.e., 'crackers') and those who have legitimate access to the system but are abusing their privileges (i.e., the 'insider threat')". Intrusion Detection Systems (IDSs) have evolved into a critical component in secure network architecture.

An IDS (refer to Figure 1.2) is any hardware, software, or combination of thereof that monitor a system or network of systems for malicious activity (Koziol, 2003).

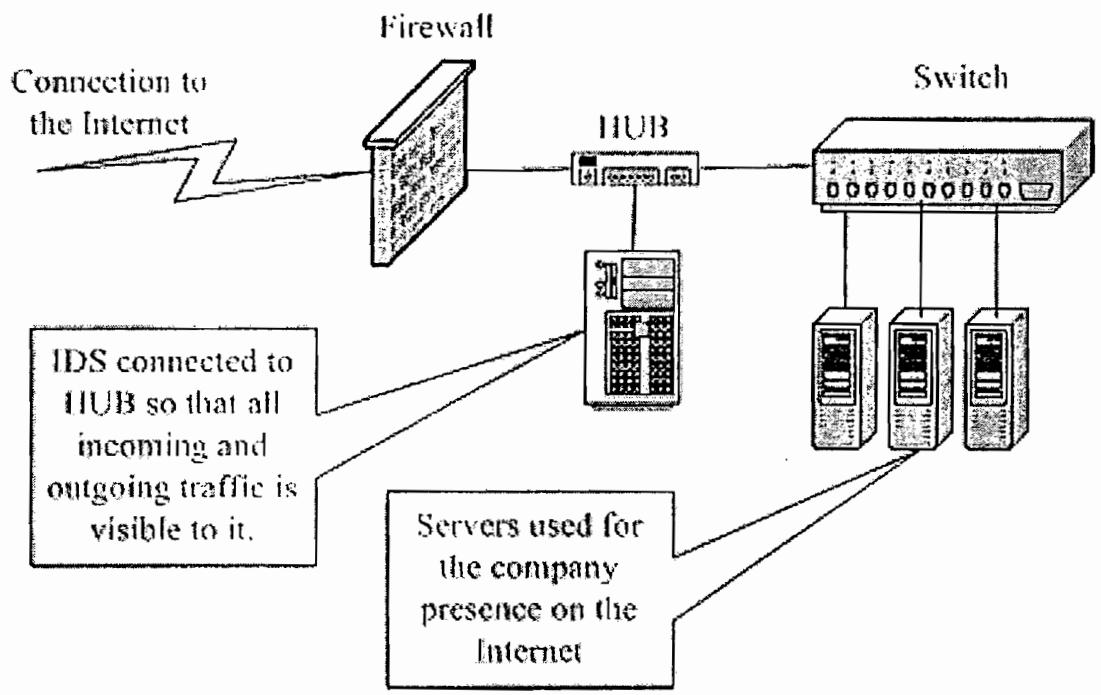

Figure 1.2: Connecting an IDS in a switched environment. (Rehman, 2003)

The classification for the Intrusion Detection Systems (IDSs) depends on the task which must be performed, could be divided in to three types: Host Intrusion Detection System (HIDS), Network Intrusion Detection System (NIDS) and Distributed Intrusion Detection System (DIDS). NIDS analyze the traffic as it flows throw a network; HIDS must be installed in specific host and monitors for intrusion attempts in this host; and DIDS is a combination of NIDSs, HIDSs, or both across the enterprise and all reporting to central correlation system (Baker, Caswell, & Poor, 2004).

The Network intrusion detection systems (NIDS) are a main security component in several network environments. These systems continuously monitor

network traffic for malicious activity, raising alerts when they detect attacks and also enable real-time detection of network attack (Attig & Lockwood, 2005). With the use of network intrusion detection systems, a network administrators need to ensure that network traffic is not being unduly delayed by overhead introduced from the real-time network intrusion detection system. Network administrators do not want to expose network security or add unneeded overhead to the already extremely busy networks by introducing a network intrusion detection system.

To get the most real result in testing the IDS, the evaluation should be with real traffic, where the simulated traffic are not enough, that mean the evaluation must carried out in real-world extreme conditions. This includes detection rate under a load, support for a high-speed network, protection against IDS evasion and, effect on the environment. The effect on the environment means that the IDS must impose minimal overhead on the environment that it monitors (Anttila, 2004).

In the past two decades of years, a number of intrusion detection systems have been developed both in the commercial and academic sectors (Roesch, 1999). Bro IDS is a lightweight network intrusion detection tool that can be deployed to monitor small TCP/IP networks and detect a wide variety of suspicious network traffic as well as outright attacks. Bro is available under the GNU General Public License and is free for use in any environment, making the employment of Bro as a network security system more of a network management and coordination issue than one of affordability that cost thousands of dollars at minimum.

1.2 Problem Statements

There are many studies done to evaluate the performance of the intrusion detection systems, most of this studies focus on the ability of the IDS to detect the

intrusions, but not on the ability of the system to ensure that continued network traffic performance levels are maintained when placed in active mode (Wagoner, 2007).

Typically, the passive mode for the network intrusion detection system (NIDS) residing on the edge of a network performs deep packet inspection on every packet that enters to the protected domain. While active mode or the inline network intrusion detection systems mean the system work as intrusion prevention when block unwanted packets, where obviously place some additional overhead into the network traffic path simply by the processing required to compare the traffic to the system signatures. How much overhead will be introduced into the network traffic by introduction of an inline network detection system and whether this overhead is significant needs to be determined.

This thesis offer the important of testing the performance of the network intrusion prevention system, by using simple method to measure the network delay which use Bro IPS, where the overhead will be determined when measuring the delay time in network traffic, also this thesis will determine which policies in Bro NIDS which may effect the delay time (decrease or increase the delay time).

Measuring network delay time introduced by the Bro Intrusion Prevention System is important for evaluation its performance and increasing the function of computer system security.

1.3 Research Questions

This thesis will attempt to answer the following questions:

1. What is the amount of the overhead, which determined by the delay time in network traffic, will be introduced by the implementation of a real-time

intrusion detection system? The null hypothesis for this question will be that the delay added by the intrusion detection system will not be noticeable?

2. What are processing policies available in Bro either individually or in combination will introduce the most delay, if any, to the network traffic? The null hypothesis for this question will be that the preprocessing policy that introduces the most delay will have when one of first selected policy or second or a combination of both pre-processors turned on.
3. Does the determined overhead, if any, by a real-time intrusion prevention system create a significant and practical delay in the overall end-to-end path of network traffic? The null hypothesis for this question will be that the delay introduced by a real-time intrusion prevention system will not be significant.

1.4 Research Objectives

Network intrusion prevention system is one of the major components used in network security to achieve the protection for specific network. So performance of the NIDS must be established enough to carry out real-time intrusion detection system (real-time means that an intrusion has to be detected before damage accrued).

The implement of this thesis will try to achieve the below objectives:

1. To determine the measured overhead or the delay time in network traffic when implement of a real-time intrusion prevention system.
2. To specify which process policies available in Bro will introduce the most delay, if any, to the network traffic.
3. To determine if the introduced overhead create a significant and practical delay in the overall end to end path on the network traffic.

1.5 Assumptions

There are some assumptions while evaluating the effect of the Bro intrusion prevention system, the scope of the evaluation is specific to certain indications of performance. The evaluation and performance measurements for the delay time must then be controlled and should not be effected by other processing components of the software. To ensure this the following assumptions will be made.

Assumption 1

Bro is an intrusion prevention system that is designed to prevent the network intrusions through both pattern matching and detection of anomalous network behavior. For this study, it will be assumed that Bro is capable of performing both functions efficiently and effectively.

Assumption 2

Bro can be installed and used within a few minutes of installation. However, there are many customizable components designed in the Bro system that will not be considered for this research. This study will use the default configuration assuming that it is sufficiently optimized for basic testing to determine the amount of delay or elapsed time from end-to-end traffic introduced by the intrusion prevention system.

Assumption 3

It will also be assumed that the sample traffic used for testing the intrusion prevention system will be representative of normal network traffic on the live network of Universiti Utara Malaysia.

1.6 Scope and Limitations

The evaluation will be implements in isolated local area network (LAN) using Ubuntu Linux operating system. Isolated LAN will use because many of the tests required direct control over the amount of computing activity in the environment.

This research will be limiting by the following boundaries:

- This research will be limited to the detection engine processing efficiency. This efficiency will be determine by the amount of delay that the engine introduces into the network traffic flow and will not be determine by the ability to accurately detect intrusion or by inaccurately detect intrusions.
- This study will be limited to the study of select pre-processor policies used by the Bro intrusion prevention system. There are several such policies available for bro; some have been tested and verified for enterprise usage, while others are still being developed and tested. We will limit this study to the frag.bro, netflow.bro, http.bro, ftp.bro and scan.bro preprocessor policies available in Bro.
- There are many different methods for alert output are also available in Bro. This study will limit the alert options to the standard default, which processes alerts to a basic log file. The system alert output will not be reviewed for this study, as this information would be use to determine detection accuracy.

1.7 Significant of Study

This study tries to evaluate Bro IPS effect on the environment and its performance in real-time intrusion prevention system. The evaluation is important in ensuring the strength of the network security. There are many of network intrusion detection system vendors market their products as being able to process traffic at

'wire speed' indicating there is no delay added to the network traffic from the detection system. This has not been tested in the prospective customer's production network so there is no indication of how the system will perform when implemented in a live environment. Developing a simple method of verifying the processing speed of the intrusion prevention system then becomes quite important to assist in product evaluation.

Also, as more products are made available, the customizable options available on these products are ever expanding. Without testing, a prospective customer only has the word of the vendor as to the performance impact of each option. Testing the system specifically for each option then provides valuable information assisting in product selection. Such testing should be completed on production traffic to ensure the results will be more relevant and applicable in the customer's environment. Such a simple method of performance testing can then be transferred to candidate intrusion prevention systems to be considered.

A final consideration is that a prime method of getting around an intrusion prevention system is to crush that system with excessive network traffic volume. This traffic does not have to be legitimate traffic, but can still make the intrusion prevention system of null effect. Testing will provide information indicating how much load the intrusion prevention should be able to maintain without serious network performance degradation. This in turn will allow the network administrator to be proactive in upgrading or switching out such a system.

Selecting an intrusion prevention system that is incapable of processing the volume of network traffic thrown at it will lead to poor network performance and eventually customer satisfaction will be negatively effected. Testing prior to implementation of a real time network intrusion prevention system should be

mandatory for anyone looking to use such a system. So, this research tries to prove the performance of Bro as Network Intrusion Prevention System.

1.8 Definition of Terms

For clarity of understanding the following terms will be defined. The definitions given will be operational definitions that are pertinent to this study and the software used to conduct the research.

- Intrusion: any unwanted access or attempted access to network resources through either malicious traffic such as viruses or Trojans or through intentional, directed system attacks.
- Intrusion Detection: any method used to pinpoint or locate an intrusion on either a host machine or in network traffic.
- Intrusion Prevention: use of an intrusion detection system or other specialized software to both monitor network traffic while detecting possible intrusions and is also able to dynamically respond to such intrusions to either block the traffic or quarantine such traffic.
- Network Intrusion Detection System (NIDS): an intrusion detection system that monitors network traffic scanning for anomalous behavior and matching defined patterns to detect intrusions.
- Network Intrusion Prevention System (NIPS): is like the NIDS with the ability to drop any suspicion packets in the traffic.
- Host Intrusion Detection System (HIDS): an intrusion detection system that runs on a host machine and detects intrusive behavior through monitoring and analysis of log file, security access policies, and user login information.

- Distributed Intrusion Detection System (DIDS): an intrusion detection system that is a combination of both a NIDS and a HIDS with the analysis completed in a central location.
- Engine: the primary detection mechanism used in most network intrusion detection systems. The software unit reads network traffic and compares it to defined patterns looking for matches indicating a possible intrusion attempt.
- Preprocessor Policy: any software plug-in component that is used to enhance the effectiveness of the Bro intrusion prevention system. These components are usually designed to detect a single, common attack activity and network traffic is run through these software components prior to running through the primary scan engine to reduce traffic through the scan engine and improve system performance.
- Console: the central repository of intrusion prevention alerts used for logging, analysis, and cross-referencing multiple alert devices.
- Sensor: an intrusion prevention device that does not analyze alerts, but reports alerts to a central location for analysis and cross reference. Bro running on a machine reporting to a central console would be considered a Sensor.
- Stealth Mode: an operational mode of network intrusion detection systems used to monitor network traffic while being undetectable to individuals attempting intrusion. Such machines either have one-way traffic from the outside network to the inside network or do not have a network address to allow communication back onto the network.

1.9 Organization of the Thesis

Chapter two gives an overview of intrusion detection and prevention systems, Bro components, and some related work. Chapter three describes a methodology that used to setup a test-bed environment and implements the experiments. Chapter four present the finding and analysis of the experiments results that done. Chapter five gives a conclusion and suggested for future works.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

With the growth of the Internet, large number of viruses and malicious probes spread every day. Many network users are vulnerable to attacks. Traditionally, networks have been protected using firewalls that monitor and filter network traffic. Firewalls usually examine the packet headers to determine whether or the packets are allowed to pass through or dropped. However, firewalls are not effective to protect networks from worms and viruses. Today, the most commonly used defense strategy is to use end-host based solutions that rely on security tools, such as antivirus software. The main drawback of these approaches is the inability to protect thousands of hosts in less than an hour (Chang, Tsai, & Chung, 2008).

Network intrusion detection systems (NIDS) are utilized to detect malicious attacks and protect Internet system. The intrusion detection systems are growing in popularity because they provide an efficient protection to attacks. The NIDS differs from a firewall in that it needs to scan both the headers and the payloads of each incoming packet for thousands of suspicious patterns (Song, Sproull, Attig, & Lockwood, 2005). By inspecting both packet headers and payloads to identify attack signatures, NIDS is able to discover whether malicious attacks or hackers are attempting to intrude (Kim, Jung, Lim, & Kim, 2007; Song, Sproull, Attig, & Lockwood, 2005). With this realization, '*Defense in-Depth*' (*or Security In-Depth*) is

a new concept has begun to develop depending on a military history that teaches us to never rely on a single defensive line or technique. Network security have tried to not rely on NIDS alone (Northcutt & Novak, 2003).

2.2 Bro Intrusion Prevention System

The approach to the protection of assets is most familiar to security practitioners and operatives. The strategy has an extensive history of application and success in the prevention of theft, destruction of facilities, and the protection of personnel and information. Probabilistic models of the defense in depth principle have been developed to optimize the protection of assets in an organization, and therefore the application of a range of barrier types to prevent unauthorized access is well understood. This approach to asset protection through a succession of barriers has been adopted as a strategy to restrict the penetration by unauthorized access to the assets, and hence gain time for the authorities to react to the penetration of the asset protection system (Smith, 2003).

The security of a computer network should occur in a deeply layered approach. The layers are each a specific application or tool to enhance security, prevent unauthorized access, and detect unauthorized access (Baker & Esler, 2007; McHugh, Christie, & Allen, 2000).

Such an approach should begin from the inside of an organization and then work outward. The first layer is the proper training of employees on the organization's security policies and information concerning what action the user should take in certain scenarios. This would be followed by the need for host machines to have some type of intrusion detection application on them as well as virus detection applications. Next, there should be some type of secure policy control

on the point of network connection or the network switch, which should also have a slightly advanced security policy on the uplink connection to the main router. The router should have secure policy access control lists created and implemented to assist in securing the internal network. The next layer would include a firewall and possibly a de-militarized zone enclosed by a second firewall. The final layer should include a network intrusion detection or intrusion prevention system (May, Hammerstein, Mattson, & Rush, 2006; Snyder).

2.3 Intrusion Detection System (IDS)

Intrusion detection has been an active field of research from 1980 (McHugh, Christie, & Allen, 2000) with the publication of John Anderson's related to computer auditing based on his research for US Air Force. Anderson's paper is widely considered the first work in the area of intrusion detection. Dorothy Denning's determining paper, "An Intrusion Detection Model," published in 1987, took intrusion detection one step further by suggesting a correlation between anomalous activities and computer misuse. Denning explained how anomalous activity could be used as an indicator of potential security incidents. This piece spawned research into intrusion detection (Crothers, 2003).

Current Intrusion detection systems come in three primary forms (Baker & Esler, 2007; Koziol, 2003) depending on its functionality : network-based intrusion detection systems, host-based intrusion detection systems, and distributed intrusion detection systems. An intrusion detection system is a necessary component to apply the concept of defense in-depth. Implementing any of these systems will require some type of processing power, even though it may be extremely limited, there is some overhead to the detection process, which may affect performance.

2.3.1 Network-Based Intrusion Detection Systems (NIDSs)

Network IDS monitors network traffic in real time and examines packets in order to detect signs of misuse. They do this by matching packets against database of attack signatures, or performing protocol decodes to detect anomalies (Dries, 2001). When they find suspicious activity, they could either raise an alert or terminate the suspicious connection. Some products can interact with firewalls with SNMP commands to block attacking hosts (NSSlabs, 2008a).

Usually network IDS work in promiscuous mode, listening to all the traffic on the network segment where the agent is installed. They are basically network sniffers. Depending on the amount of data on the network, this could be very resource consuming. For instance, most attacks are split to several packets, so IDS must "remember" several packets in order to track attack signatures. This requires heavy processing especially on fast networks. Because of that, network based IDS are usually installed on dedicated hosts. Normally there is one agent per network segment, because IDS cannot see the traffic on different segment without special configuration on switches or routers (Baker, Caswell, & Poor, 2004).

NIDS is more cost effective than an HIDS because it can protect a large area of network with one device. With NIDS, can monitor what is happening in and around the network. It is more popular than HIDS. A NIDS can be more secure and less prone to outages than an HIDS. The NIDS should be run on a single hardened host that supports only services related to intrusion detection, making it more difficult to disable.

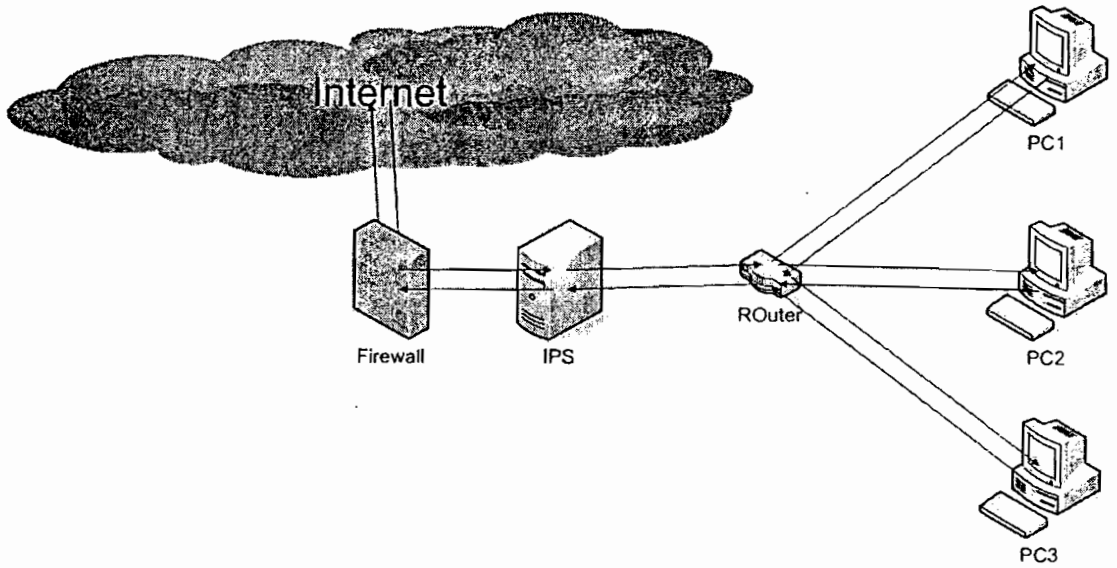

Figure 2.1: NIPS Schema

Refer to (Figure 2.1) depicts a network using three NIPS. The units have been placed on strategic network segments and can monitor network traffic for all devices on the segment. This configuration represents a standard perimeter security network topology where the screened subnets housing the public servers are protected by NIPS. When a public server is compromised on a screened subnet, the server can become a launching platform for additional exploits. Careful monitoring is necessary to prevent further damage.

The internal host systems are protected by an additional NIPS to mitigate exposure to internal compromise. The use of multiple NIPS within a network is an example of a defense-in-depth security architecture (Baker & Esler, 2007; Koziol, 2003).

2.3.2 Host-Based Intrusion Detection Systems (HIDSs)

In host-based solutions, there is an agent installed in every host to be monitored. This agent monitors auditable resources like event/system logs, kernel logs

and critical files for suspicious activity (Anttila, 2004). However, HIDS differ from NIDS in two ways. HIDS protects only the host system on which it resides (Dries, 2001), and its network card operates by default in nonpromiscuous mode. Nonpromiscuous mode of operation can be an advantage in some cases, because not all NICs are capable of promiscuous mode. In addition, promiscuous mode can be CPU intensive for a slow host machine. Due to their location on the host to be monitored, HIDS are private to all kinds of additional local information with security implications, including system calls, file system modifications, and system logs. In combination with network communications, this provides a robust amount of data to parse through in search of security events of possible concern. Another advantage of HIDS is the capability to tailor the ruleset very finely for each individual host. Consequently, the reduction in the number of relevant rules enhances performance and reduces processor overhead for each host.

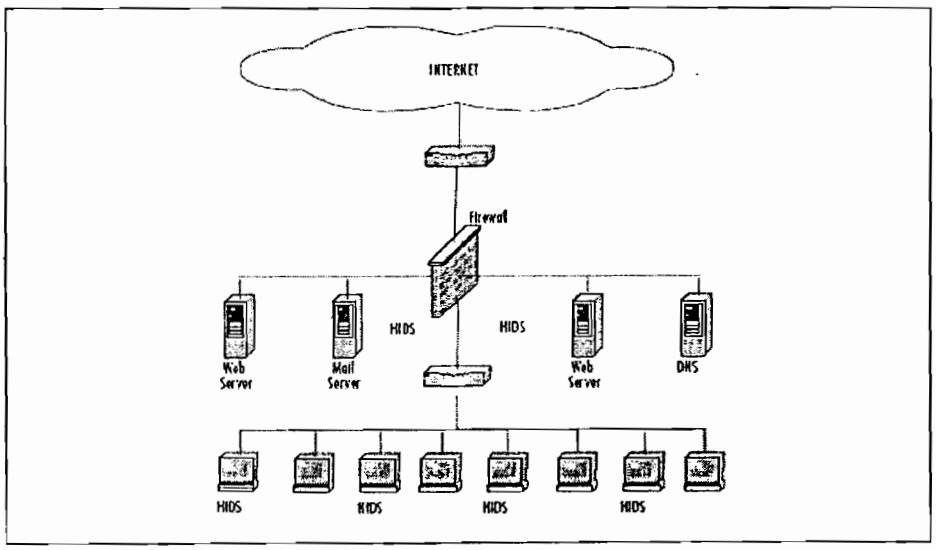

Figure 2.2: HIDS Schema

As previously mentioned, the ruleset for the HIDS (refer to Figure 2.2) on the mail server is customized to protect it from mail server exploits, and the Web server rules are tailored for Web exploits. During installation, individual host machines can be configured with a common set of rules. New rules can be loaded periodically to account for new vulnerabilities.

2.3.3 Distributed Intrusion Detection Systems (DIDS)

The standard DIDS functions in a Manager/Probe architecture. NIDS detection sensors are remotely located and report to a centralized management station (Dries, 2001). Attack logs are periodically uploaded to the management station and can be stored in a central database; new attack signatures can be downloaded to the sensors on an as-needed basis. The rules for each sensor can be tailored to meet its individual needs. Alerts can be forwarded to a messaging system located on the management station and used to notify the IDS administrator. The network transactions between sensor and manager can be on a private network, as depicted, or the network traffic can use the existing infrastructure.

Refer to (Figure 2.3) shows a DIDS composed of three sensors and a centralized management station.

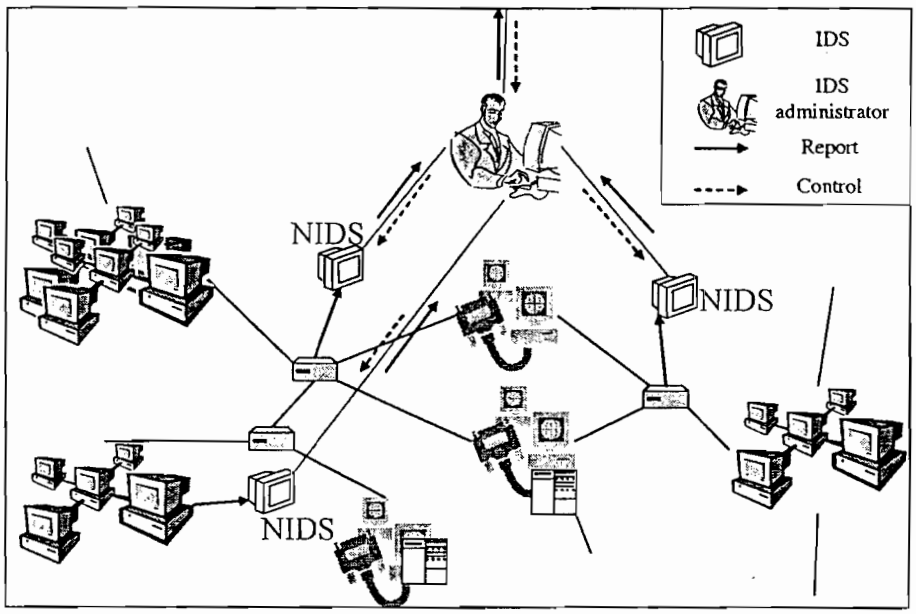

Figure 2.3: DIDS Schema

In a DIDS, the individual sensors can be NIDS, HIDS, or a combination of both. The sensor can function in promiscuous mode or nonpromiscuous mode. However, in all cases, the DIDS's single defining feature requires that the distributed sensors report to a centralized management station.

2.4 Intrusion Prevention Systems

There is a common misconception concerning the similarities and differences between intrusion detection systems and intrusion prevention systems. The primary roll of an intrusion detection system has is to detect intrusion or misuse. Network intrusion detection systems can be implemented as a real-time system or as a passive system (Chiu, Lin, Lee, & Lei, 2007). It is physically placed in the network traffic path and monitors and analyzes all traffic in a real-time live setting.

An intrusion prevention system (IPS) are proactive defense mechanisms designed to detect malicious packets within normal network traffic (something that

the current type of firewalls do not actually do, for example) and stop intrusions dead, blocking the offending traffic automatically before it does any damage rather than simply raising an alert as, or after, the malicious payload has been delivered (NSSLabs, 2008b). However, to be able to prevent such activity the system must first be able to detect such activity. This is where the misunderstanding arises. A network intrusion prevention system must be done in real time environment to provide the best possible measure of intrusion prevention (Baker & Esler, 2007).

A network intrusion detection system can detect a possible intrusion and alert on that detected possibility as instructed, but is limited to detection capabilities only. A network intrusion prevention system is capable of detecting a possible intrusion, alerting on that intrusion, and then taking a predefined action to prevent that traffic from passing on the network (Proctor, 2001). The operation of dynamic action in a network intrusion prevention system is the key element in differentiating the two systems.

2.5 IDS Detection Approaches

The detection approaches describes the characteristics of the analyzer (Debar, Dacier, & Wespi, 1999). When an intrusion-detection system uses information about the known attacks, representing it as misuse detection. In addition, when an intrusion-detection system uses information about the normal behavior of the system it monitors, representing it as anomaly detection.

2.5.1 Misuse Detection IDS

Misuse detection techniques apply the knowledge accumulated about specific attacks and system vulnerabilities. The intrusion-detection system contains information about these vulnerabilities and looks for attempts to exploit them. When such an attempt is detected, an alarm is triggered (Wu, Foo, Mei, & Bagchi, 2003). In other words, any action that is not explicitly recognized as an attack is considered acceptable. Essentially, the IDS look for a specific attack that has already been documented. Like a virus detection system, misuse detection software is only as good as the database of attack signatures that it uses to compare packets against. Therefore, the accuracy of misuse intrusion detection systems is considered good. However, their completeness requires that their knowledge of attacks be updated regularly.

Misuse detection provides various benefits. One of the first benefits is that the signature definitions are modeled on known intrusive activity. Furthermore, the user can examine the signature database, and quickly determine which intrusive activity the misuse detection system is programmed to alert on. Another benefit is that the misuse detection system begins protecting your network immediately upon installation. One final benefit is that the system is easy to understand. When an alarm fires, the user can relate this directly to a specific type of activity occurring on the network (Debar, Dacier, & Wespi, 1999).

Along with the numerous benefits, misuse detection systems also have their share of drawbacks. One of the biggest problems is maintaining state information for signatures in which the intrusive activity encompasses multiple discrete events (that is, the complete attack signature occurs in multiple packets on the network) (Newman, Manalo, & Tittel, 2004). Another drawback is that the misuse detection system must have a signature defined for all of the possible attacks that an attacker may launch against a network. This leads to the necessity for frequent signature updates to keep

the signature database of the misuse detection system up-to-date. One final problem with misuse detection systems is that someone may set up the misuse detection system in their lab and intentionally try to find ways to launch attacks that bypass detection by the misuse detection system (Debar, Dacier, & Wespi, 1999).

2.5.2 Anomaly Detection IDS

Anomaly detection techniques assume that an intrusion can be detected by observing a deviation from normal or expected behavior of the system or the users (Debar, Dacier and Wespi 1999). The model of normal or valid behavior is extracted from reference information collected by various means. The security manager defines the baseline, or normal, state of the network's traffic load, breakdown, protocol, and typical packet size. The intrusion-detection system later compares this model with the current activity. If a deviation is observed, an alarm is generated (Wu, Foo, Mei, & Bagchi, 2003). In other words, anything that does not correspond to a previously learned behavior is considered intrusive. Therefore, the intrusion-detection system might be complete, but its accuracy is a difficult issue. The anomaly detection technique is as good as its normal model definition.

Anomaly detection systems offer several benefits. First, they can detect insider attacks or account theft very easily. If a real user or someone using a stolen account starts performing actions that are outside the normal user profile, it generates an alarm. Second, because the system is based on customized profiles, it is very difficult for an attacker to know with certainty what activity he can do without setting off an alarm. Probably the largest benefit, however, is that intrusive activity is not based on specific traffic that represents known intrusive activity (as in a misuse IDS). An anomaly detection system can potentially detect an attack the first time it is used

(Debar, Dacier and Wespi 1999). The intrusive activity generates an alarm because it deviates from normal activity, not because someone conFigured the system to look for a specific stream of traffic.

Like every IDS, anomaly detection systems also suffer from several drawbacks. The first obvious drawback is that the system must be trained to create the appropriate user profiles. During the training period to define what normal traffic looks like on the network, it is will not be protected from attack. Just defining "normal" is a challenge in itself (Debar, Dacier and Wespi 1999). Maintenance of the profiles can also become time-consuming. Nevertheless, the biggest drawback to anomaly detection is probably the complexity of the system and the difficulty of associating an alarm with the specific event that triggered the alarm. Furthermore, there is no guarantee that a specific attack will even generate an alarm. If the intrusive activity is too close to normal user activity, then the attack will go unnoticed. It is also difficult to know which attacks will set off alarms unless actually test the attacks against the network using various user profiles.

2.6 Types of Computer Attacks Commonly Detected by IDSs

Three types of computer attacks are most commonly reported by IDSs: system scanning, denial of service (DOS), and system penetration. These attacks can be launched locally, on the attacked machine, or remotely, using a network to access the target (Bace & Mell, 2001).

2.6.1 Scanning Attacks

A scanning attack occurs when an attacker probes a target network or system by sending different kinds of packets. Using the responses received from the target,

the attacker can learn many of the system's characteristics and vulnerabilities. Thus, a scanning attack acts as a target identification tool for an attacker. Scanning attacks do not penetrate or otherwise compromise systems. Various names for the tools used to perform these activities include: network mappers, port mappers, network scanners, port scanners, or vulnerability scanners (Bace & Mell, 2001; Pfleeger & Pefleeger, 2007). Scanning attacks may yield:

- The topology of a target network
- The types of network traffic allowed through a firewall
- The active hosts on the network
- The operating systems those hosts are running
- The server software they are running
- The software version numbers for all detected software

With this information, an attacker can exactly identify victim systems on the target network along with specific attacks that can be used to penetrate those systems. Thus, attackers use scanning software to "case" a target before launching a real attack. Unfortunately for victims, just as it is legal for a person to enter a bank and to survey the visible security system. From the perspective of someone performing a scan, they are legally scouring the Internet to find publicly accessible resources.

The best IDS signatures for malicious scanning are usually able to differentiate between legitimate and malicious scanning. Scanning is likely the most common attack as it is the precursor to any serious penetration attempt.

2.6.2 Denial of Service Attacks

Denial of Service (DoS) attacks is a network attack that results in the denial of service by requested such as a web server. There are several mechanisms to generate a

DoS attack. This type of network DoS attack attempts to block the network pipe so that valid user traffic cannot get the network connection. However, this type of DoS typically need to be distributed because it usually requires more than one source to generate the attack (Capite, 2007; Pfleeger & Pefleeger, 2007).

A DoS attack tacks advantage of the fact that target systems such as servers must maintain state information and may have expected buffer size and network packet content for special application. DoS can exploit this vulnerability by sending packet size and data values that are not expected by the receiving application.

Several types of DoS attacks exist, including Teardrop attack and the Ping of Death, which send handcraft network packets that are different from those the application expects and may provoke the application and server to crash. These DoS attacks on unprotected server, such as ecommerce server, can cause the server to crash and prevent users from adding items to there shopping cart (Capite, 2007).

The term "distributed DoS" (DDoS) is a subset of DoS attacks. DDoS attacks are simply flooding DoS attacks where the attacker uses multiple computers to launch the attack. These attacking computers are centrally controlled by the attacker's computer and thus act as a single huge attack system(Thomas, 2004).

2.6.3 Penetration Attacks

Penetration attacks involve the unauthorized acquisition and/or alteration of system privileges, resources, or data. Consider these integrity and control violations as contrasted to DOS attacks that violate the availability of a resource and to scanning attacks, which don't do anything illegal. A penetration attack can gain control of a system by exploiting a variety of software flaws. The most common flaws and the security consequences of each are explained and enumerated below (Bace & Mell,

2001). While penetration attacks vary tremendously in details and impact, the most common types are:

- ***User to Root***: A local user on a host gains complete control of the target host.
- ***Remote to User***: An attacker on the network gains access to a user account on the target host.
- ***Remote to Root***: An attacker on the network gains complete control of the target host.
- ***Remote Disk Read***: An attacker on the network gains the ability to read private data files on the target host without the authorization of the owner
- ***Remote Disk Write***: An attacker on the network gains the ability to write to private data files on the target host without the authorization of the owner

2.7 General Architecture of a Network Intrusion Detection System

Since the publication of Anderson's seminal paper (Anderson, 1980), several intrusion detection systems have been invented. Today there exists a sufficient number of systems in the field for one to be able to form some sort of conception of a typical intrusion detection system, and its constituent parts. (Refer to Figure 2.4)

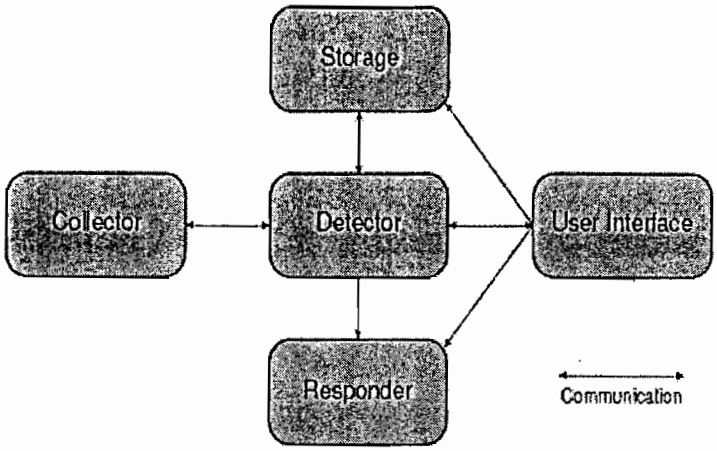

Figure 2.4: General Architecture of NIDS (Sommer, 2005)

Any generalized architectural model of an intrusion detection system (Axelsson, 2006; Sommer, 2005) would contain at least the following elements:

- ***Collector :*** Provides an interface for accessing data that is used by the detection process. For a NIDS, the primary kind of data collector is a network tap. A tap provides access to all raw network packets, which cross a particular position of a network. Other types of collectors include interfaces to host-based data or external databases. Typically, the raw data is stored somewhere, indefinitely for either later reference, or temporarily waiting processing. The volume of data is often exceedingly large; making this is a crucial element in any intrusion detection system, and leading some researchers to view intrusion detection as a problem in raw data reduction.
- ***Detector:*** Conducts the actual detection process. The detector is the brain of the NIDS. It is here that one or many algorithms are executed to find evidence (with some degree of certainty) in raw data of suspicious behavior. It accesses

data provided by collector and storage and it decides what should trigger an alert.

- ***User Interface***: Reports results to the user, and enables the user to control the NIDS. In particular, the user interface is needed to define the NIDS's device policy. In the simplest case, the user interface consists of ASCII files. Systems that are more sophisticated often include graphical user interfaces (GUIs).
- ***Storage:*** Stores persistent data required by the detector or the user interface. The reference data storage stores information about known intrusion signatures (for misuse systems) or profiles of normal behavior (for anomaly systems). In the latter case, the processing element updates the profiles as new knowledge about the observed behavior becomes available. This update is often performed at regular intervals in batches. The analysis of novel intrusions is a highly skilled task.
- ***Responder***: Reacts to detected intrusions in order to prevent future damage. Active responses may include dropping the connectivity to the potential attacker or even counter-attacks. A response may be triggered automatically or manually via the user interface.

The components can be combined into a single piece of software as well as be logically or physically separated. If separated, the components include communication sub-systems to exchange information. In principle, a detector may analyze input either in batches or in real-time. Batched processing was common in early NIDSs, which lacked the resources required for continuous monitoring. Nowadays, all major systems work in real-time (Axelsson, 2006). However, the

batched analysis is still common for host-based analysis; e.g., log files are often scanned for malicious activity at regular intervals.

2.8 Bro: Network Intrusion Prevention System

Bro (Roesch, 1999) is a free, cross-platform, lightweight network intrusion detection and prevention tool that can be used to monitor small TCP/IP networks, capable of performing real-time traffic analysis and packet logging on IP networks(Caruso, Guuindani, Schmitt, neycalazans, & Moraes, 2007). It can perform protocol analysis, content searching/matching, and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth portscans, CGI attacks, SMB probes, OS fingerprinting attempt, and much more (Hutchings, Franklin, & Carver, 2002; Roesch, 1999). Martin Roesch designed Bro in 1998, and his company Sourcefire Inc. now primarily develops it.

Bro is primarily a anomaly-based NIPS that uses a combination of rules and preprocessors to analyze traffic (Sourdis, Dimopoulos, Pnevmatikatos, & Vassiliadis, 2006). Bro uses a flexible rules language to describe traffic that it should collect or pass, as well as to detection engine that utilizes modular plugin architecture. The preprocessors code allows more extensive examination and manipulation of data that cannot be done via rules alone. Bro has a real-time alerting capability as well; he can provide a systems administrator with enough data to make decisions on the proper course of action in the face of suspicious activity. Bro can also be deployed rapidly to fill potential holes in a network's security coverage, such as when a new attack emerges and commercial security vendors are slow to release new attack recognition behaviors(Baker, Caswell, & Poor, 2004; Baker & Esler, 2007).

Bro can run in three modes (Guerrero & Cardenas, 2005) that make it very powerful: packet sniffing, packet logging, and intrusion prevention system. Packet sniffing mode simply reads the packets of the network and displays them in a continuous stream on the console. Packet logger mode logs the packets to the disk. Network intrusion prevention mode is the most complex and configurable; allowing Bro to analyze network traffic for matches against a user defined rule set and to perform several actions based upon what it sees.

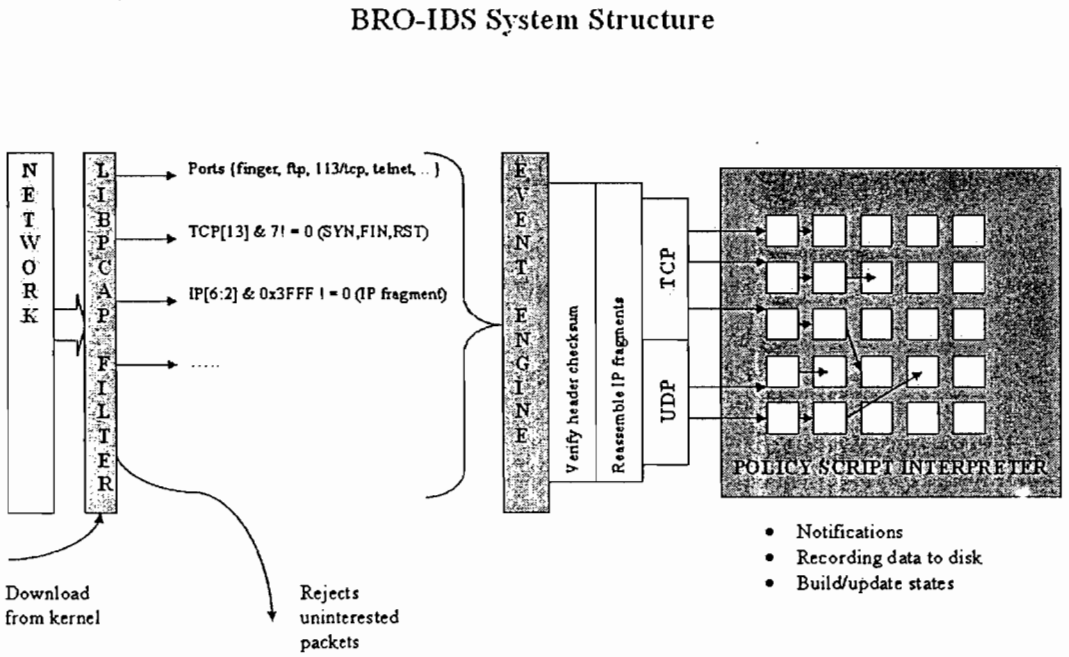

Figure 2.5 : Bro Architecture (Baker & Esler, 2007)

Bro is logically divided into multiple components. These components (refer to Figure 2.5) work together to detect particular attacks and to generate output in a required format from the detection system. Bro consists of the following major components:

- The sniffer
- The preprocessor
- The detection engine
- Logging and Alerting System
- The output

The most important feature is using Bro as IDS mode. Bro is a packet sniffer. However, it is designed to take packets and process them through the preprocessors. Each packet observed on the network is first passed through a set of preprocessors, which may extract information and/or modify the packet and then check those packets against a series of rules (through the detection engine). Then detection plug-ins matches the packet against signature conditions. If a match was found, sent through the alert system, it can be handled by whatever plug-ins have been chosen to handle alerting (Archibald, Ramirez, & Rathaus, 2005; Jeong, Jeon, Ryu, & Seo, 2006; Schwartz, Stoecklin, & Yilmaz, 2002).

2.8.1 The Packet Sniffer

A packet sniffer (refer to Figure 2.6) is device which is implemented either software or hardware that can be used to prepares the packets to be preprocessed or to be sent to the detection engine (Baker & Esler, 2007). It parses the packet and decodes the string of bytes into a packet structure that is formed of protocol fields and flags. Each subroutine in the decoder imposes order on the packet data by overlaying data structures on the raw network traffic (Rehman, 2003). These decoding routines are called in order through the protocol stack, from the data link layer up through the transport layer, finally ending at the application layer. During this decoding process, it validates the length and checksum fields. It then forwards the valid packets to the preprocessors.

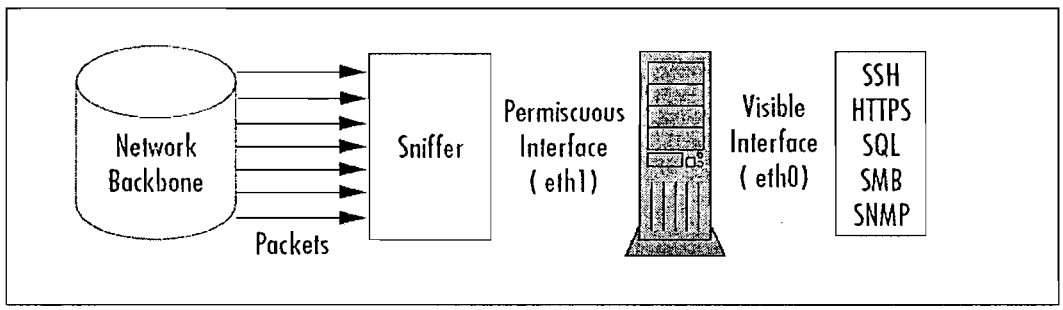

Figure 2.6 : Bro's Packet-Sniffing Functionality (Baker & Esler, 2007)

2.8.2 Preprocessors

When Bro receives a packet, it may not be ready for processing by the main Bro detection engine and application of Bro policies. For example, a packet may be fragmented. Before searching a string within the packet or determine its exact size, defragmentations are required by assembling all fragments of the data packet. On IPS, before applying any rules or try to find anomaly, the packets have to be reassembled (Rehman, 2003). The job of a preprocessor is to make a packet suitable for the detection engine to apply different rules to it. In addition, some preprocessors are used for other tasks such as detection of anomalies and obvious errors in data packets, decoding of HTTP URI. All enabled preprocessors operate on each packet. There is no way to bypass some of the preprocessors based upon some criteria (Baker & Esler, 2007). Refer to (Figure 2.7), illustrate the workings of the preprocessor within Bro.

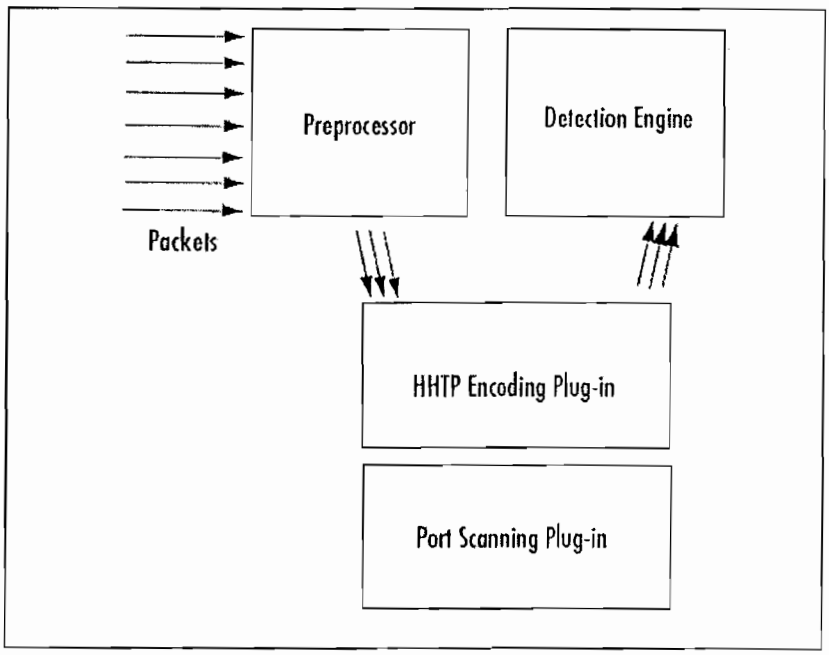

Figure 2.7: Bro's Preprocessor (Baker & Esler, 2007)

2.8.3 The Detection Engine

The detection engine (refer to Figure 2.8) is the most important part of Bro. Its responsibility is to detect if any intrusion activity exists in a packet. The detection engine employs Bro rules for this purpose (Baker & Esler, 2007). The rules are read into internal data structures or chains where they are matched against all packets. Bro organizes parts of packets to make the job of matching rules against them faster. It maintains detection rules in a two-dimensional linked list of what are termed Chain Headers and Chain Options. The commonalities are condensed into a single Chain Header and individual detection signatures are kept in Chain Option structures. If a packet matches any rule, appropriate action is taken; otherwise the packet is dropped (Asarcikli, 2005). Appropriate actions may be logging the packet or generating alerts.

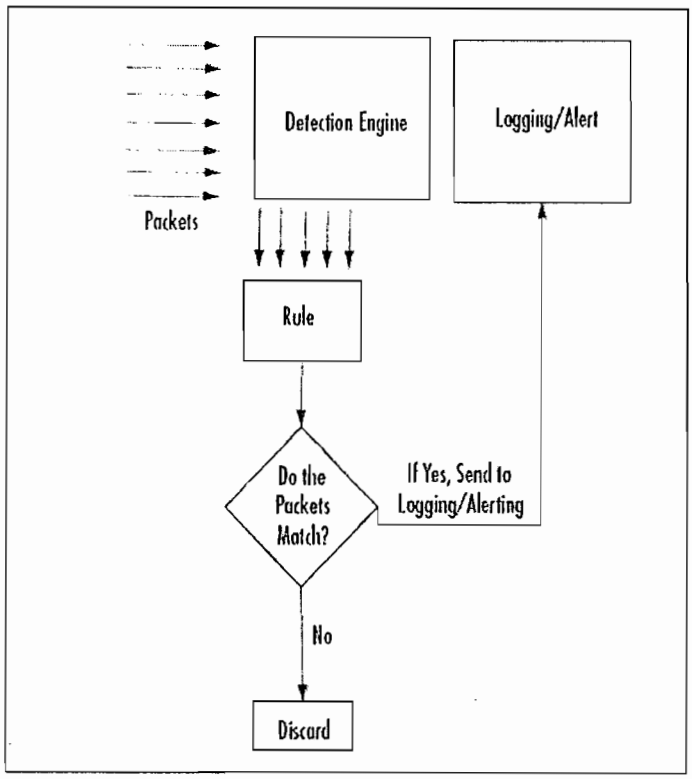

Figure 2.8 : Bro's Detection Engine (Baker & Esler, 2007)

2.8.4 Logging and Alerting System

This system is responsible from the generation of alerts and logging of packets and messages (refer to Figure 2.9). Depending upon what the detection engine finds inside a packet, the packet may be used to log the activity or generate an alert. All of the log files are stored under a preconfigured location by default. This location can be configured using command line options. There are many command line options to modify the type and detail of information that is logged by the logging and alerting system(Asarcikli, 2005).

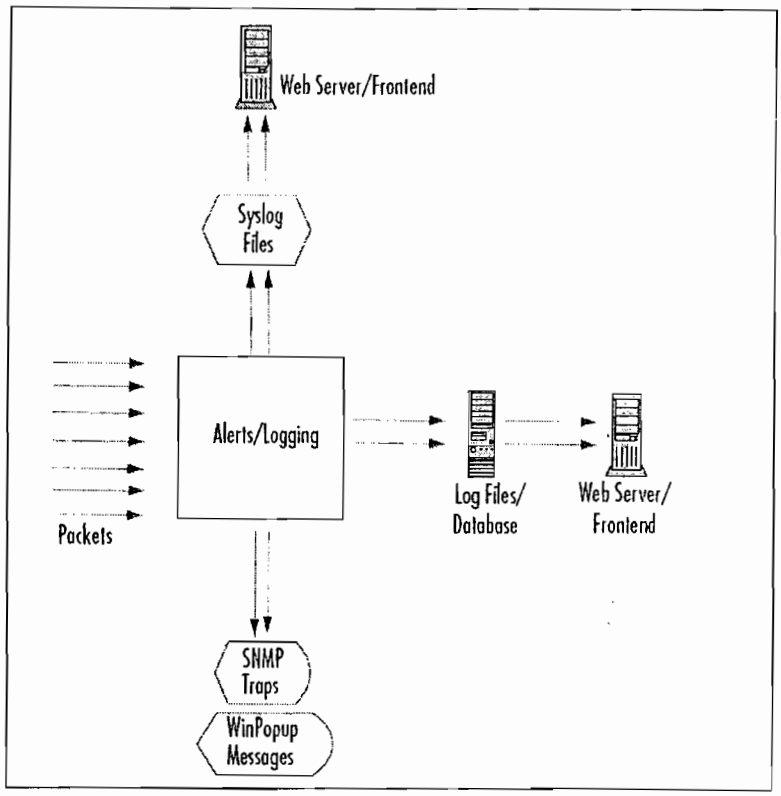

Figure 2.9 : Bro Alerting Component (Baker & Esler, 2007)

2.8.5 Output Modules

These modules control the type of output generated by the logging and alerting system. Depending on the configuration, output modules can send output messages to a number of other destinations (Baker, Caswell, & Poor, 2004; Baker & Esler, 2007; Rehman, 2003). Commonly used output modules are:

- The email module can be used to send Bro alerts in the form of traps to a management server,
- The syslog module logs messages to the syslog utility (using this module can log messages to a centralized logging server.),

2.9 Bro Preprocessors Policies

A preprocessor is a code, which compiled into the Bro engine upon build in order to normalize traffic and/or examine the traffic for attacks in a fashion beyond what can be done in normal rules. Although that might seem like an overly simplistic explanation for what these complex pieces of Bro do, it's important to realize their contribution to the overall whole of the intrusion detection system (IDS) (Baker & Esler, 2007; Beale & Foster, 2003).

Bro allows us to select which preprocessors should be enabled. From this standpoint, this is done through the Bro policies files ".bro" (Rehman, 2003). Bro has many preprocessors available. The Bro project team has certified some, while others are in testing and more yet are still in development. These preprocessors are what make Bro such a powerful and effective intrusion prevention system. The preprocessors that we are primarily concerned with this study are the scan.bro, netflow.bro, http.bro, frag.bro and ftp.bro. The detail descriptions for each one are following:

i. frag.bro

The frag.bro preprocessor is applying for reassembling packets as a target-based IP defragmentation policy for Bro. frag.bro rebuild all the fragments packets received into pseudo packets and then pushes them through the preprocessor and the detection engine. This nature makes frag.bro resource intensive and generates longer delay.

Target-based analysis is a relatively new concept in network-based intrusion detection. The idea of a target-based system is to model the actual targets on the network instead of only modeling the protocols and looking for attacks within them. The idea for "target-based IDS" came from where the attacker can determine what

style of IP defragmentation is being used on a particular target, the attacker can try to fragment packets such that the target will put them back together in a specific manner while any passive systems trying to model the host traffic have to guess which way the target OS is going to handle the overlaps and retransmits.

The basic idea behind target-based IDS is that we tell the IDS information about hosts on the network so that it can avoid Ptacek & Newsham style evasion attacks based on information about how an individual target IP stack operates.

ii. netflow.bro

The netflow.bro preprocessor is a target-based TCP reassembly policy for Bro. It is intended to replace both the Stream4 and flow preprocessors, and it is capable of tracking sessions for both TCP and UDP. Many attacks are spread across several packets and are undetectable to a non session-reassembling rule-matching IDS, that's the whole reason for stream reassembly(Baker & Esler, 2007; Beale & Foster, 2003; Novak & Sturges, 2007).

netflow.bro, like frag.bro, introduces target-based actions for handling of overlapping data and other TCP anomalies. The methods for handling overlapping data, TCP Timestamps, Data on SYN, FIN and Reset sequence numbers, etc. and the policies supported by netflow.bro are the results of extensive research with many target operating systems which make it resources intensive (Novak & Sturges, 2007).

iii. http.bro

http.bro has become one of the most widely and diversely used protocols on the Internet. Over time, researchers have found that Web servers will often take a

number of different expressions of the same URL as equivalent. For example, an IIS Web server will see these two URLs as being identical:

http://www.example.com/foo/bar/iis.html

http://www.example.com/foo\bar\iis.html

Unfortunately, a pattern matcher such as Bro will only match the pattern *foo/bar* against the first of these two. An attacker can use this "flexibility" in the Web server to attempt to hide his probes and attacks from the NIDS.

http.bro is stateless; it normalizes HTTP strings on a packet-by-packet (Baker & Esler, 2007) basis and will only process HTTP strings that have been reassembled by the netflow.bro preprocessor.

iv. ***Scan.bro***

This policy is designed to detect the first phase in a network attack: Reconnaissance. In the Reconnaissance phase, an attacker determines what types of network protocols or services a host supports. This is the traditional place where a portscan takes place. This phase assumes the attacking host has no prior knowledge of what protocols or services are supported by the target, otherwise this phase would not be necessary.

As the attacker has no beforehand knowledge of its intended target, most queries sent by the attacker will be negative (meaning that the services are closed). In the nature of legitimate network communications, negative responses from hosts are rare, and rarer still are multiple negative responses within a given amount of time. The primary objective in detecting portscans is to detect and track these negative responses(Baker & Esler, 2007).

One of the most common port scanning tools in use today is Nmap. Nmap encompasses many, if not all, of the current portscanning techniques. scan.bro was designed to be able to detect the different types of scans Nmap can produce.

v. ftp.bro

ftp.bro is can be stateful or stateless; it receives this data from the netflow.bro preprocessor, thus we need to turn it on when enable ftp.bro preprocessor (Baker & Esler, 2007). When FTP command channel buffers (on port 21) are used, ftp.bro will interpret the data, identifying FTP commands and parameters, as well as appropriate FTP response codes and messages. It will enforce the correctness of the parameters, determine when an FTP command connection is encrypted, and furthermore determine when an FTP data channel is opened.

The ftp.bro is extremely versatile, having the capability through the dynamic preprocessor to be able to configure every parameter, which makes for a very powerful emulation engine.

2.10 Advantages and Disadvantages of Bro

Bro is a very flexible application. Due to the modular design and ability to add or break in specialized software components Bro can be a powerful tool in a defense/security in-depth implementation. This design allows anyone capable of programming to build and implement their own preprocessor policies to customize Bro's operation to their specific environment. Customization can also be accomplished through specialized configurations of the existing preprocessor policies, as well as alert output operations (Baker & Esler, 2007).

Bro also has a large following and according to the Bro website Bro-IDS.org, Bro is the effectively standard in intrusion detection systems. There are many commercialized systems available, but many organizations use Bro because it is an effective intrusion detection system, and it can be obtained at no cost. Bro is a anomaly based detection system and with the large user base new behaviors are constantly being added. This large user and support base has led to what is described as a highly effective and efficient detection engine.

Bro does have some limited shortfalls when it comes to anomaly detection. The system was not designed for this type of operation, but some preprocessor policies attempt to add this functionality (Lippmann, Haines, Fried, Korba, & Das, 2000). Currently these policies are not considered effective in detection. There is also concern about how efficient the detection engine actually is in terms of processing performance. The base engine is considered quite efficient, but there is speculation as to how efficient the system becomes when used with the preprocessor policies. The added functionality is good, but what price do you have to pay for that functionality.

The main disadvantages of Bro are that the very few documentation and the short reference compare with other intrusion detection system like Snort. In addition, Bro does not use any database for outputs, and there are no graphic user interfaces. For some programmer, there other disadvantages which is Bro doesn't work in windows operating system.

2.11 Related Work

Network security has been researched for many years and will continue to be a key research topic until there are no network security breaches. According to (Puketza, 2000), there are three natural approaches to network security, which are:

protection through filtering, protection through assessment, and protection through detection. This work was done to show that by filtering known unwanted traffic, constantly testing for unknown vulnerabilities, and implementing measures to detect unwanted traffic a network can be secured.

The process of detection using an intrusion detection system was one of the steps suggested by (Puketza, 2000). Puketza's study used Network Security Monitoring (NSM), which is not a true intrusion detection system, but relies on components that operate like intrusion detection systems to create an integrated data collection and analysis suite. He considers detection one of the four key elements required to develop secure systems (refer to Figure 2.10).

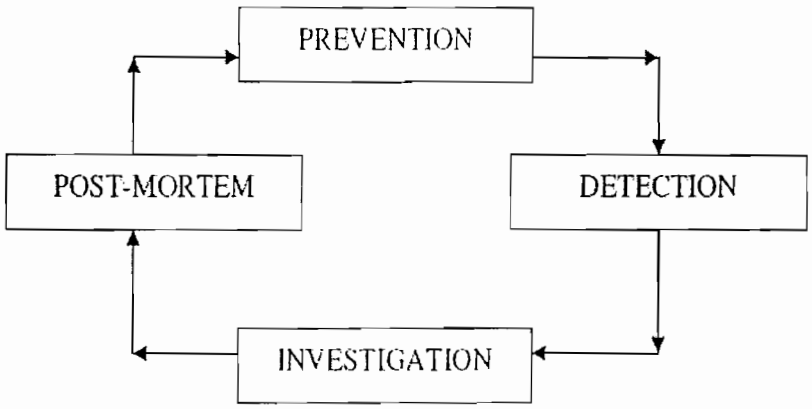

Figure 2.10: Key Element of a Secure Network Implementation

A high stress load on the intrusion detection system supposed by (Puketza, 2000) may effect the system's ability to monitor and detect intrusions . This indicates that the intrusion detection system may become a point of either extreme delay or packet loss. The test for this study was implemented by using software tools to build a level of stress on the intrusion detection system and then sending the system specific intrusion attacks. The system was then monitored to determine how well it detected

the attacks and at what level, system resources were being used. The test did not try to determine how much delay the system introduced into the network path. This study clearly indicated that there is a need for performance evaluation of intrusion detection systems.

An initial attempt to perform a comprehensive technical evaluation of intrusion detection technology was sponsored by the Defense Advanced Research Projects Agency (DARPA) conducted one of the most extensive IDS testing experiments in 1998 and 1999 (Lippmann, Haines, Fried, Korba, & Das, 2000). The network traffic of an Air Force base with thousands of machines and hundreds of users was simulated. Ad-hoc scripts were developed to recreate realistic behaviors of various classes of users: programmers, secretaries, managers, and system administrators. A large set of known and novel attacks were executed in the test-bed against different operating systems. The '99 experiment evaluated more than 18 research IDSs, measuring the detection rate, the false positive rate, and drawing the corresponding ROC curves. The evaluation of the various systems also took into account the amount of information reported for each attack (e.g., attack name, starting time, and intrusion category). Finally, the systems that performed better in the lab experiment were tested for false positives with the real Air Force traffic.

The performance of intrusion detection systems was discussed by (Zamboni, 2001). He was concerned with detection capability and suggested the implementation of internal sensors to perform intrusion detection in computer systems and determined that his proposed implementation could detect a far greater number of attempted intrusions. His study focused on the performance of the detection system itself. Also (Balzarotti, 2006) do so , in his study proposes a novel black-box technique to test and evaluate misuse detection models in the case of network-based intrusion detection

systems. The testing methodology is based on an automated mechanism to generate a large number of test cases by applying mutant operators to an attack template. Each operator implements a transformation function that is able to change the attack manifestation while preserving its functionality. This implies that all possible combinations of available transformations must be generated by analyzing the dynamic behavior of the intrusion detection system under test.

Utilizing Bro in the analysis of intrusion detection systems was a study done by (Yaacob, 2003) to evaluate Bro performance. The performance was measured in term of the resource usage such as main memory, CPU time, and disk space usage. IDS usage importantly depends on the disk storage in function stable. The storage usage space totally depends on alert that has been generated and log for the attack lunched. IDS performance must high be enough to carry out real-time intrusion detection before significant damage has occurred. According to this study, to have good IDS in detection, the system administrator needs to prepare the high resources usage for IDS host. A host that going to be used as IDS should have higher CPU processing, memory and file storage for the best performance. These will void the IDS performance from breaking down and lastly from being risky to over all computer system and organization.

The performance testing of a network and any applications that run on that network is important to ensure the proper level of service is being provided. Considering performance testing of a network one of the prime components often used for measurement is end-to-end delay (Cisco, 2007). When the end-to-end delay increases beyond a certain point, which must be determined on a network-by-network basis, the usability and reliability of the network begins to degrade. One method of introducing delay onto a network is through 'fixed delay components'. These

components add directly to the overall delay on the network. Using network intrusion detection system implemented in real-time manner could introduce fixed delay. Any component on the network whether it be routers, switches, or security appliances has the potential to introduce delay. How much delay is introduced and whether that value is significant or not is a question that must be answered (Cisco, 2007).

Generally the networks will similarly carry voice, video or multimedia traffic, as well as data, needs to be some standards for delay limits established. Such standards have been established to assist in determining when the delay value becomes unacceptable (refer to Table B-1, Appendix B). These standards suggest that overall per packet delay should remain below 150 milliseconds to ensure acceptable network performance.

A test bed to evaluate inline-Snort performance by determine the end-to-end delay time had been done by (Wagoner, 2007). In his study, he used inline-Snort as network intrusion detection system under FreeBSD operating system environment with considering many various combinations of preprocessors options configurations. The amount of end-to-end delay that was introduced by implementing an IDS will range from 1.913 ms to 1.35 ms. Also, he found the maximum delay time would introduced when using a combinations of the preprocessing options offered in inline-Snort. This option was running Snort with the Flow, Stream4, Telnet-negation and Http-inspect preprocessors turned on together.

2.12 Summery

This chapter gives us a wide view of the Intrusion Detection and Prevention Systems, its types and, the typical place for deployment. IDSs can serve many purposes in a defense-in-depth architecture. In addition to identifying attacks and

suspicious activity, can be use IPS data to prevent security vulnerabilities and weaknesses. The classical IPSs fall into two classes: *anomaly based*, and *misuse based.* An anomaly based IDS specify the normal behavior of users or applications and consider any pattern falling outside the defined behavior as an attack. A misuse based IDS specifies the signatures of attacks and parses audit files to detect any matches.

Bro is the popular as intrusion-detection system available and can work as intrusion-prevention. The system and its intrusion prevention ruleset are freely available, and both are regularly updated to account for the latest threats. This chapter also determines Bro in details, by describing its architecture, components and, the processing sequence for the packets inspection. The preprocessor options available in Bro that are chooses to evaluate its impacts or overhead add to the network traffic are discussed also the Bro rules that used by the detection engine. Some previous research related to Bro performance test are listed with brief description.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Introduction

This chapter discusses the methodology that will be used in this project to achieve the project objectives. It is more than just a collection of methods to handle the research; it is a semantic way proposed by the researcher to solve the research problems. The researcher depend on a simulation model that used by (Yaacob, 2003) to proposed a simulation test-bed IPS environment (refer to Figure 3.1) which contains six phases. It is involves the testing of specific values of the decision or uncontrollable variables in the model and observing the impact on the output values. It also involves setting up a model of a real system and conducting repetitive experiments on it.

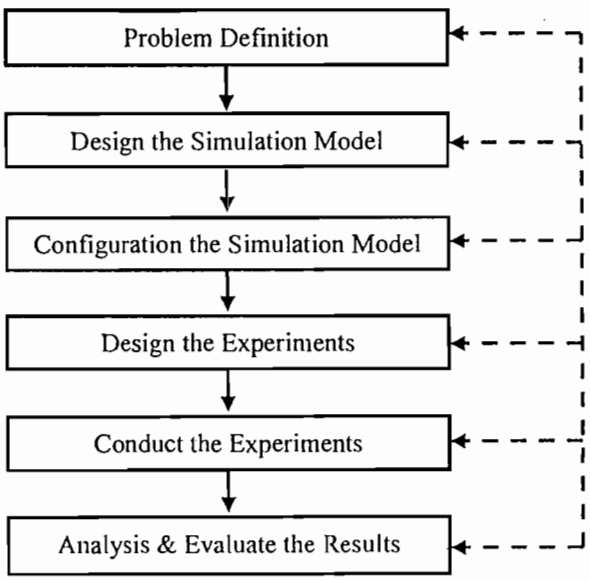

Figure 3.1: Simulation Test-Bed Model

3.2 Problem Definition

Problem definition is an important phase in every study. In this study, the researcher tried to study and collect the necessary information that related to the Intrusion Detection Systems environment requirements and its characteristics. This phase represent the backbone for this study. Information have been gathered and collected from books, journals, proceedings, white papers, reports.

This phase aim to specify the research problem(s), scope, domain, assumption, and the limitation after reviewing the collected information and the previous work done by other researchers to start from where they stopped.

3.3 Design the Simulation Model

To test the preprocessing effect, of the Bro intrusion prevention system, some network traffic will be needed to run through the detection engine. To get a better idea of how the detection engine operates in a live environment this traffic should be tested in a live environment. This can be quite disruptive to the network, so traffic needs to be obtained from inside a university network that can be used in a test environment. Better control of the testing environment can be maintained using this type of scenario.

There must be a laboratory to prepare a test bed to use for the test environment. For this laboratory setting three computer computers will be required. The machines being used in this study will be identical, one machine will have a second network interface card (NIC) installed that need to be configure as a router by forwarding the traffic (using IPTables in Linux) from the first card to the Bro analyzer and then to the second card. The machines contain Dual-Core 1.83GHz, 512 MB of RAM, and Realtek RTL8168/8111 PCI-E Gigabit network interface cards. The

additional NIC was Intel Pro/ 100+ management adapter. These machines will be called IPSSource which loaded with window XP SP2 operating system, IPSServer which loaded with Ubuntu Linux (8.10) operating system, and IPSDest which loaded same operation like IPSSource, depend on the machine's purpose on the simulation network (refer to Figure 3.2).

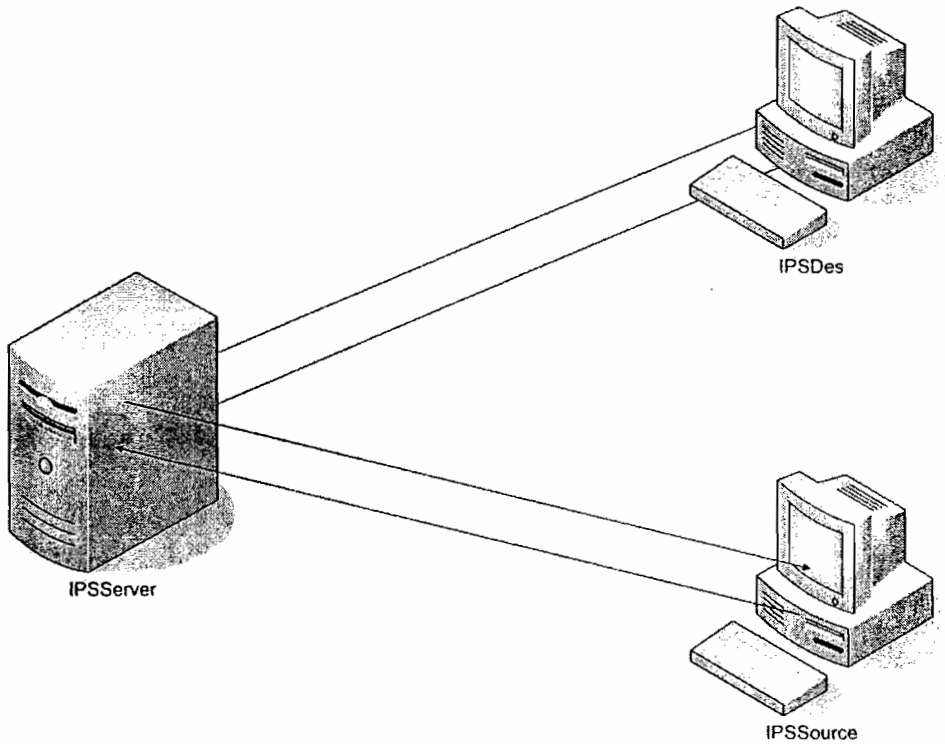

Figure 3.2: Intrusion Prevention Simulated Network

All the three machines will also require having the Wirshark, WinPcap softwares installed. IPSSource will have the Packet Builder Player application, while IPSServer will have the Bro version 1.4 software installed. Machine IPS will have two network interface cards installed, as it will be used for the network intrusion detection system engine. The experiment procedures are designed for testing Bro that monitors network computers. The best environment to use for these tests is an isolated area network because of the tests requires direct control over the amount of computing activity in the environment.

3.4 Configuration the Simulation Model

After the simulation model has been designed, the configuration starting. Bro performance testing need to configure Bro as an intrusion prevention system depending on the information gathered from Bro official website, books and, technical documentation manual using the default setting. To accomplish configuration a test bed simulation, the Universiti Utara Malaysia network would be used for capturing a portion of that traffic to be considered as a test sample. The sample will be captured using the Wireshark packet sniffer application. A computer that has Wireshark installed will be placed in a Postgraduate (M.Sc. IT) Research laboratory network, connected to the University network and will capture all traffic when login to UUM Portal and downloaded some lectures notes and also login to Sultanah Bahiyah Library to renewal some borrowed books. The captured information will be saved in a file that will be named Test.pcap. All experimental passes will use this file for data collection.

The captured data will be placed back on the network by using the Packet Builder Player loaded (refer to Figure 3.3) on the IPSSource computer. The program Packet Player will be used to read the file Test.pcap and then send the packets on a specified network interface. The captured traffic will be replayed onto the network using the exact timing recorded when the traffic was captured. This assists in ensuring that the data being used is as close to live traffic as possible. The data will then leave the IPSSource machine and it will be sent to IPSDest. However, it must cross IPSServer before reaching its destination. IPSServer computer set the Bro sensor's on the bridge connection where the packets will be processed and then it will be

forwarded on to IPSDest. Once the packet is received at IPSDest it will be read by the Wireshark application (refer to Figure 3.4), which will recapture the traffic along with new timestamps for each packet received.

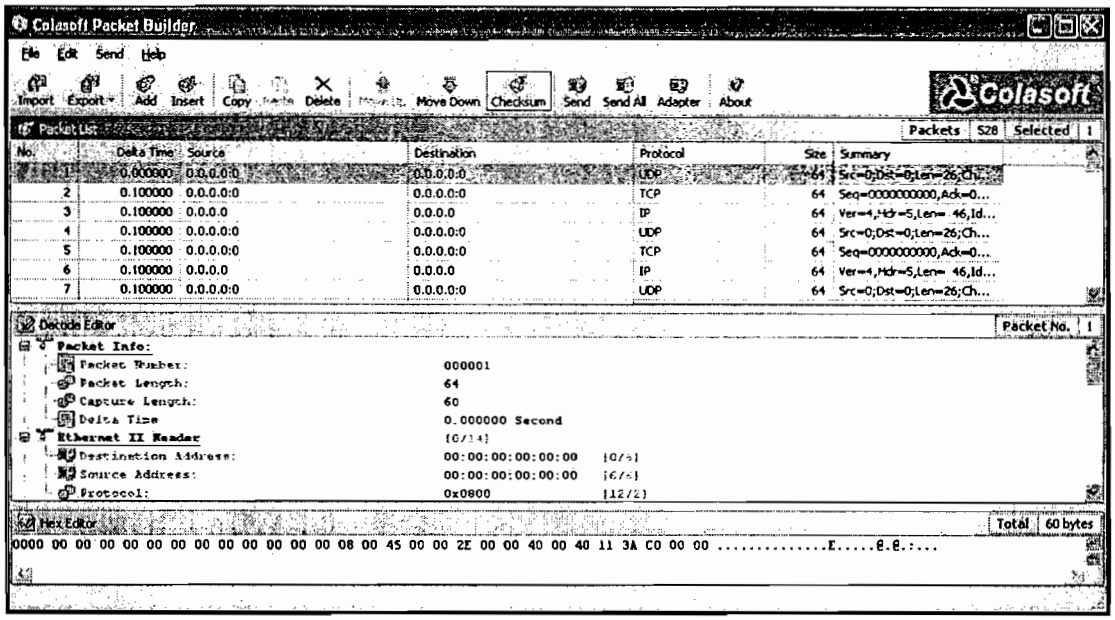

Figure 3.3: Packet Builder Player Interface.

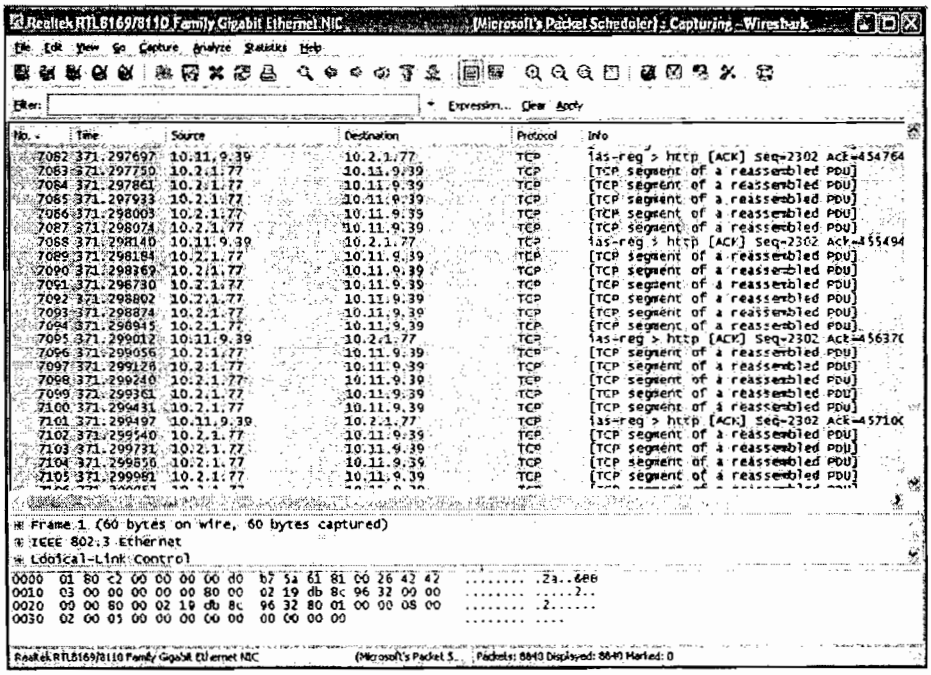

Figure 3.4: Wireshark Interface

3.5 Design the Experiments

Due to the numerous possible combinations of preprocessors that may be used, this study will use a full-factorial experimental design. Under consideration are five preprocessor policies and each policy can be turned either on or off. The preprocessor policies are frag.bro, netflow.bro, http.bro, ftp.bro and, scan.bro, the last three policies need to turn on with netflow.bro preprocessor (as explained in chapter 2). This indicates that there are 18 possible combinations, which require 18 test passes. Table B-2 (Appendix B) shows the design table with all possible combinations. To help ensure that unknown factors do not affect the results, the test will be performed with ten replications. The replications will allow a determination of how each preprocessor policy combination effects the total end-to-end delay of the network traffic by averaging multiple passes that may encounter noise effects, thus reducing the effect of the noise and providing more reliable results.

3.6 Conduct the Experiments

This phase of the methodology applying the experiments designed in the previous phase. By depending on Table B-2 (Appendix B), for each experiment need to change the Bro policy directory in order to turn the preprocessor(s) on or off. The experiment starting by use a Packet Builder Player on the IPSSource machine to send packets when a Wireshark activate to capture them in a sniffer mode on the IPSDest machine. The observation time get from calculate the difference in time of arriving the first packet and the final packet as captured by a Wireshark. This scenario replicate ten times for each experiment. At the end of this phase, the data set will be collected and point to begin the measurement and analysis method in order to evaluate the results.

3.7 Analysis & Evaluation the Results

The amount of delay introduced by the Bro intrusion prevention system will be used as an indicator of Bro effect on the envorinment. This delay will be measured as the elapsed time from receipt of the initial packet until receipt of the final packet on the IPSDest machine, as measured by Wireshark application. This simulated elapsed time will then be compared to the elapsed time of the original traffic capture. The elapsed time from the original traffic capture as measured by Wireshark is 29.999 seconds. This elapsed time was determined by the Wireshark application. The Wireshark application captured the initial packet and recorded this packet as time stamp 0:00. Each additional packet was then captured and time stamped. The time stamp placed on each of these packets was the amount of elapsed time from receipt of the first packet. The final packet captured recorded an elapsed time of 29.999. The effect of the intrusion prevention engine and associated policies will be determined by the difference of the measured elapsed time as described above compared to the original traffic time. These differences will then be collected and used for analysis.

Once the data has been collected, an analysis will be made of the obtained results following steps used by Wagoner in (2007). To start, a statistical analysis will be performed on the results obtained to determine significance. Significance will give indication of what factors or factorial combinations have the most statistically significant impact on the amount of overhead (Graham, 2000; Maxwell & Delaney, 2004) introduced into the end-to-end delay by the intrusion detection system. To begin with, a calculation of factor effect will be made, followed by a calculation of interaction effect between factors. Both the main effect and the interaction effect between factors will be analyzed for significance using analysis of variance (ANOVA). This will show what effect each individual factor had on end-to-end delay introduced by the intrusion detection system. Once the main effects and interactions

effects have been calculated; ANOVA will give indication of statistical significance of any factor or factor interaction.

Statistical significance will be determined by using the pooled estimate of variances analysis using the SPSS software application. A confidence interval of 95% and 99% will be used for this analysis. Significance factors calculated by SPSS, will give indication of statistically significant factorial effects.

Once statistical significance using ANOVA has been determined, a follow up analysis using a Tukey-Kramer method will be performed. This method will perform a pairwise comparison and t-test that will give a more detailed indication of which treatment factor combination adds the most end-to-end delay and which treatment factor combination has a statistically significant effect introduced by the Bro intrusion prevention system.

Once this is completed an analysis to determine practical significance will be completed. This will be determined using the information in Tab. B-1 (Appendix B). The total elapsed time determined for each experimental pass will be distributed to the individual packets being used from the file Test.pcap by dividing the amount of time into the total number of packets. The total number of packets in the file Test.pcap is 450. From the file Test.pcap the total elapsed time is 29.999 seconds, so the per-packet processing time for this file would be 29.999/450 seconds per packet. This per packet processing time value will be compared to Table B-1 to determine which values are practically significant and which have no practical effect.

Any per packet processing time value greater than 150 milliseconds will be considered practically significant. This is the value assumed to be the cutoff for noticeable network delay issues. Any elapsed time higher than this value can greatly effect multimedia applications and will be considered as excessive delay (Cisco,

2007). Once the statistical and practical significance is determined, an analysis will be completed to review the results and provide answers to the proposed research question.

3.8 Summary

This research applied the simulation model presented by Yaacob (2003) as a guideline for the research process. Six phases have been used listed below:

1. Problem Definition.
2. Design the Simulation Model.
3. Configuration the Simulation Model.
4. Design the Experiments
5. Conduct the Experiments
6. Analysis & Evaluating the results

In problem definition phase, a Bro deployed on the edge of the network when have been used as NIPS to perform a deep packets inspection. This scenario place additional overhead into the network traffic that wanted to study and analyze which option causes it. In designing the simulation model, test bed and machines are going to setup. In configuration the simulation model phase, Bro configured as IPS and a raw packets captured that are used in the experiments. Design the experiments phase determined the all experiments need to get the research objectives. In conduct the experiments, the experiment implements and the data set collected. The last phase, the outcomes of the experiments will be analyzes using a statistical methods.

CHAPTER FOUR

FINDING AND ANALYSIS OF DATA

4.1 Introduction

An Intrusion Prevention System (IPS), that monitors actively specific computing resources, and reports anomalous or intrusive activities, is becoming an important component in the security system of information infrastructure. IPSs can help security analysts find if an attack has taken place, either in real time or immediately after the attack. One of the challenges faced by the security administrators is how to identify network intrusions and how to evaluate the effectiveness of IPS. Studying and testing IPSs against a variety of the intrusive activities under background traffic is an interesting and challenging problem. These testing can be performed either in a real environment or an experimental environment. Testing in a Real Environment means that's testing can be conducted in a live environment where many real users produce significant background traffic by using a variety of network services, e.g., mail, Ftp, Telnet, etc. In this method, a test traffic sample is directly collected from the real environment, and intrusive activities are emulated by exploit scripts. Most researchers perform their tests in experimental environments, Due to the high risk to perform testing in a real environment. The challenge of this method is how to produce realistic a test traffic sample in an experimental environment. Three ways of generating a traffic sample : first, can be manually produced to perform whatever occurred in a real environment; second, using

a simulation scripts to generate data conforming to the statistical distribution in a real environment; third, collected from a real environment and replicated in the experimental environment as described by wan & yang in (2001).

Now, test traffic sample was captured from the real environment and testing in experimental environment was performed and the data set are ready to analyze.

4.2 Application to Research Questions

Question 1 asks how much end-to-end delay is introduced by the implementation of a real-time intrusion detection system. The observations made will assist in easily determining how much additional time, compared to the original capture, each treatment factor introduced. Question 2 asks which preprocessing options in Bro introduce the most delay. The observations compared to the original capture time value can be used to determine the amount of delay introduced by Bro. This value can then be ranked to show which treatment introduces the most delay. Question 3 then asks which treatment introduces either a statistically or practically significant delay. This answer will be determined using the ANOVA and Tukey-Kramer post hoc analyses.

4.3 Data Set Collected

The raw data from the simulation environment are shown in Table C-1 (Appendix C). This table also shows the average observation value of all test runs by treatment factor, along with the total average of all test runs for all treatment factors.

Observations were then entered into SPSS and were analyzed. This information was then processed with both the SPSS ANOVA options using the univariate analysis of the general linear model as declared in (Appendix D), and

where found to be statistically significant were further analyzed with the Tukey-Kramer post-hoc methodology. SPSS provides analysis for the Tukey-Kramer method, but the results are affected by the limitation of 3 significant decimal positions, while the values under analysis require 6 significant decimal positions. Therefore, an analysis of the Tukey-Kramer results was manually handled using MS-Excel. The output from the ANOVA analysis with 0.05% confidence interval is listed in Table D-1 (Appendix D) while with 0.01% confidence interval is listed in Table D-2 (Appendix D) and the Tukey-Kramer analysis results are listed in Tables D-4 through D-14 (Appendix D).

4.4 Method of Analysis

Beginning with the raw observations and average value per run from Table C-1 (Appendix C) we can easily determine the amount of time introduced by each treatment factor, as well as the overall average from use of the Real-Time Intrusion Prevention system. This information showing the average value per treatment run and the delay introduced is listed in Table C-2 (Appendix C). The final column shows the delay time introduced when compared to the original packet capture time of 29.999 seconds.

Reviewing these data shows that the time introduced by the Real-Time Intrusion Prevention System ranged from 0.014007seconds to 0.00725 seconds or from 14.007 μs to 7.25 μs. These numbers represent the total delay time for processing the entire packet capture and are not per packet values. The average delay introduced was 0.010277 seconds or 10.277 μs. The full range of introduced delay sorted from highest delay to lowest delay is shown in Table C-3 (Appendix C).

Now that the values for introduced delay time have been analyzed, the matter of statistical and practical significance can be considered. Screens captured of the setup configuration used for the ANOVA analysis of SPSS are shown in figures D-1 through D-4 (Appendix D). Using SPSS to process the ANOVA analysis gave indication that the overall model is significant by calculating a significance factor of 0.000. The analysis also indicated that all model had a significant impact at the 5% confidence interval, the confidence interval was decreased to 1% to be more accurate. The ANOVA analysis at 5% confidence level was shown in Table D-1 (Appendix D), also the ANOVA analysis at 1% confidence level was shown in Table D-2 (Appendix D). From Table D-2, the analysis indicate that 15 factors were statistical significant. The 15 interaction effects with significance factor are treatment factor 2 (netflow.bro), treatment factor 3 (frag.bro * netflow.bro), treatment factor 4 (netflow.bro * http.bro), treatment factor 5 (frag.bro * netflow.bro * http.bro), treatment factor 6 (netflow.bro * ftp.bro), treatment factor 7 (frag.bro * netflow.bro * ftp.bro), treatment factor 9 (frag.bro * netflow.bro * http.bro * ftp.bro), treatment factor 10 (netflow.bro * scan.bro), treatment factor 11(frag.bro * netflow.bro *scan.bro), treatment factor 12 (netflow.bro * http.bro * scan.bro), treatment factor 13 (netflow.bro * ftp.bro * scan.bro), treatment factor 14 (frag.bro * netflow.bro * http.bro * scan.bro), treatment factor 15 (frag.bro * netflow.bro * ftp.bro * scan.bro), treatment factor 16 (netflow.bro * http.bro * ftp.bro * scan.bro), treatment factor 17 (frag.bro * netflow.bro * http.bro * ftp.bro * scan.bro). The 15 interaction treatments are shown in Table D-3 (Appendix D) along with their treatment factor combination option mean values. An ad-hoc analysis of these 15 treatment factors is now performed using the Tukey-Kramer pairwise analysis. This ad-hoc analysis is

completed to assist in providing additional detail into the actual combinations of the significant interactions that provide the significant results.

Each treatment factor is compared to the other treatment factors using the Tukey-Kramer method through manual comparisons to assist in identifying the interaction combinations that may provide the difference between means, leading to an indication of significance from the ANOVA analysis, depending on the p-value comparisons to determine the greatest effect factor. The p-value is a numerical estimate of the reliability of our assumption that the difference in means on pre and post surveys is real and not due to chance. The results of these pairwise Tukey-Kramer comparisons are listed for the treatment factors indicating significance in the ANOVA analysis in Tables D-4 through D-14 (Appendix D). The summary results of the Tukey-Kramer analysis are listed in Table D-15 (Appendix D).

Table D-3 shows the mean values of combination differences for the significant treatment factor Frag.bro*Netflow.bro, while the Tukey-Kramer analysis is shown in Table D-4 with summery information in Table D-15. The pairwise comparisons show that there are three statistically significant treatment factors at 1% confidence factor. The first comparison occurs when both of Frag.bro and Netflow.bro are turned off compared to when Frag.bro is turned off, Netflow.bro is turned on. The p-value value is 2.79E-10 which is below the significant level of 0.01%, gave indicate that the Netflow.bro introduces significant delay when Frag.bro turned off. The second comparison occurred when both of Frag.bro and Netflow.bro are turned off compared to when Frag.bro is turned on, Netflow.bro is turned off. The p-value value is 3.06228E-20 which is below the significant level of 0.01%, gave indicate that the Frag.bro introduces significant delay when Netflow.bro turned off. The third comparison occurred when both of Frag.bro and Netflow.bro are turned on

compared to when Frag.bro is turned on, Netflow.bro is turned off. The p-value value is 4.25456E-11 which is below the significant level of 0.01%, gave indicate that the Frag.bro introduces significant delay when Netflow.bro turned off. Frag.bro has the greatest effect of this treatment factor combination because it had the lowest p-value.

Table D-3 shows the mean values of combination differences for the significant treatment factor Frag.bro*(Netflow.bro&Http.bro), while the Tukey-Kramer analysis is shown in Table D-5 with summery information in Table D-15. The pairwise comparisons show that there is a statistically significant treatment factor at 1% confidence factor in mean values when both of Frag.bro and (Netflow.bro&Http.bro) are turned on compared to Frag.bro is turned on and (Netflow.bro&Http.bro) is turned off. The p-value for this comparison is calculated at 0.000499. This gives indicate that Frag.bro introduced significant delay when (Netflow.bro&Http.bro) is turned off. This comparisons show that the Frag.bro has the greatest effect of the treatment factor because it had the lowest p-value.

Table D-3 shows the mean values of combination differences for the significant treatment factor Frag.bro*(Netflow.bro&Ftp.bro), while the Tukey-Kramer analysis is shown in Table D-6 with summery information in Table D-15. The pairwise comparisons show that there are two statistically significant treatment factors at 1% confidence factor in mean values. The first comparison occurs when both of Frag.bro and (Netflow.bro&Ftp.bro) are turned off when compared with the Frag.bro is turned on and (Netflow.bro&Ftp.bro) is turned off. The p-value for this treatment factor is 1.63E-07, this gives indicate that the Frag.bro has a significant delay when (Netflow.bro&Ftp.bro) turned off. The second comparison occurs when both of Frag.bro and (Netflow.bro&Ftp.bro) are turned on when compared with the Frag.bro is turned on and (Netflow.bro&Ftp.bro) is turned off. The p-value for this treatment

factor is 7.6E-05, this gives indicate that Frag.bro has a significant delay when (Netflow.bro&Ftp.bro) turned off. This comparisons show that the Frag.bro has the greatest effect of the treatment factor combinations because it had the lowest p-value.

Table D-3 shows the mean values of combination differences for the significant treatment factor Frag.bro*(Netflow.bro&scan.bro), while the Tukey-Kramer analysis is shown in Table D-7 with summery information in Table D-15. The pairwise comparisons show that there are two statistically significant treatment factors at 1% confidence factor in mean values. The first comparisons occurs when both of Frag.bro and (Netflow.bro& scan.bro) are turned off when compared to Frag.bro is turned off and (Netflow.bro&scan.bro) is turned on gave 0.00288 as p-value that indicate the (Netflow.bro& scan.bro) introduced significant delay when Frag.bro is turned off. The second comparisons occurred when both of Frag.bro and (Netflow.bro&scan.bro) are turned when compared with Frag.bro turned on and (Netflow.bro&scan.bro) are turned off. The p-value calculated at 5.73E-05. This give indicated that Frag.bro introduced significant delay when (Netflow.bro&scan.bro) is turned off. Because of the p-value of the second comparison is the lowest, this treatment factors combinations indicate that the Frag.bro has the greatest effect of the delay time because it had the lowest p-value.

Table D-3 shows the mean values of combination differences for the significant treatment factor (Netflow.bro&Http.bro) * (Netflow.bro&scan.bro), while the Tukey-Kramer analysis is shown in Table D-8 with summery information in Table D-15. The pairwise comparisons show that there are three statistically significant treatment factors at 1% confidence factor in mean values. The first comparison occurs when both of (Netflow.bro&Http.bro) and (Netflow.bro&scan.bro) are turned off when compare with (Netflow.bro&Http.bro) is turned off and (Netflow.bro&scan.bro)

is turned on. This comparison had a p-value of 0.000222. The second comparison occurs when both of (Netflow.bro&Http.bro) and (Netflow.bro&scan.bro) are turned off when compared with the (Netflow.bro&Http.bro) is turned on and (Netflow.bro&scan.bro) turned off. This comparison had a p-value of 0.002726. The third comparison occurs when both of the (Netflow.bro&Http.bro) and (Netflow.bro&scan.bro) are turned on when compare with the (Netflow.bro&Http.bro) is turned on and (Netflow.bro&scan.bro) is turned off. This comparison had a p-value of 0.00234. The greatest effect of these treatment factor comparisons is (Netflow.bro&scan.bro) because it had lowest p-value.

Table D-3 shows the mean values of combination differences for the significant treatment factor (Netflow.bro&Ftp.bro) * (Netflow.bro&scan.bro), while the Tukey-Kramer analysis is shown in Table D-9 with summery information in Table D-15. The pairwise comparisons show that there are two statistically significant treatment factors at 1% confidence factor in mean values. The first comparison occurs when both of the (Netflow.bro&Ftp.bro) and (Netflow.bro&scan.bro) are turned off when compared to (Netflow.bro&Ftp.bro) are turned off and (Netflow.bro&scan.bro) are turned on. This comparison had a p-value of 2.09E-13. The second comparison occurs when both of (Netflow.bro&Ftp.bro) and (Netflow.bro&scan.bro) are turned off when compared to (Netflow.bro&Ftp.bro) are turned on and (Netflow.bro&scan.bro) are turned off. This comparison had a p-value of 3.25E-16. The greatest effect of these treatment factor comparisons is (Netflow.bro&Ftp.bro) because it had lowest p-value.

Table D-3 shows the mean values of combination differences for the significant treatment factor Frag.bro * (Netflow.bro&Http.bro) * (Netflow.bro&Ftp.bro), while the Tukey-Kramer analysis is shown in Table D-10

with summery information in Table D-15. The pairwise comparisons show's that there are five statistically significant treatment factors at 1% confidence factor in mean values as listed bellow:

Combination 1	Combination 2	P-Value
F=off, H=off, FT=off	F=off, H=off, FT=on	0.000473
F=off, H=off, FT=off	F=off, H=on, FT=on	1.1E-05
F=on, H=on, FT=on	F=off, H=on, FT=off	0.000297
F=on, H=on, FT=on	F=on, H=off, FT=off	5.54E-06
F=on, H=on, FT=on	F=on, H=off, FT=on	0.006426

F = Frag.bro, H = Netflow.bro&Http.bro, FT = Netflow.bro&Ftp.bro

The greatest effect of this treatment factor comparisons is Frag.bro because it had lowest p-value, but it is influenced by both of (Netflow.bro&Http.bro) and (Netflow.bro&Ftp.bro).

Table D-3 shows the mean values of combination differences for the significant treatment factor Frag.bro * (Netflow.bro&Http.bro) * (Netflow.bro&scan.bro), while the Tukey-Kramer analysis is shown in Table D-11 with summery information in Table D-15. The pairwise comparisons show that there are eight statistically significant treatment factors at 1% confidence factor in mean values as listed below:

Combination 1	Combination 2	P-Value
F=off, H=off, SP=off	F=off, H=off, SP=on	6.3E-08
F=off, H=off, SP=off	F=off, H=on, SP=off	4.76E-06
F=off, H=off, SP=off	F=off, H=on, SP=on	0.000304
F=on, H=on, SP=on	F=off, H=off, SP=on	0.007051
F=on, H=on, SP=on	F=off, H=on, SP=off	2.7E-14
F=on, H=on, SP=on	F=on, H=off, SP=off	0.000152
F=on, H=on, SP=on	F=on, H=off, SP=on	2.6E-08
F=on, H=on, SP=on	F=on, H=on, SP=off	4.38E-05

F = Frag.bro, H = Netflow.bro&Http.bro, SF = Netflow.bro&scan.bro

The greatest effect of these treatment factor comparisons is (Netflow.bro&Http.bro) because it had lowest p-value, but it is influenced by both of Frag.bro and (Netflow.bro&scan.bro).

Table D-3 shows the mean values of combination differences for the significant treatment factor Frag.bro * (Netflow.bro&Ftp.bro) * (Netflow.bro&scan.bro) , while the Tukey-Kramer analysis is shown in Table D-12 with summery information in Table D-15. The pairwise comparisons show that there are nine statistically significant treatment factors at 1% confidence factor in mean values as listed below: .

Combination 1	Combination 2	P-Value
F=off, FT=off, SP=off	F=off, FT=off, SP=on	0.005245
F=off, FT=off, SP=off	F=off, FT=on, SP=off	0.000195
F=off, FT=off, SP=off	F=off, FT=on, SP=on	1.65E-08
F=off, FT=off, SP=off	F=on, FT=off, SP=on	9.75E-07
F=off, FT=off, SP=off	F=on, FT=on, SP=off	7.7E-07
F=on, FT=on, SP=on	F=on, FT=off, SP=off	0.000192
F=on, FT=on, SP=on	F=off, FT=on, SP=on	4.15E-05
F=on, FT=on, SP=on	F=on, FT=off, SP=on	0.003409
F=on, FT=on, SP=on	F=on, FT=on, SP=off	0.002639

F = Frag.bro, FT = Netflow.bro&Ftp.bro, SF = Netflow.bro&scan.bro

The greatest effect of these treatment factor comparisons is the interaction of (Netflow.bro&Ftp.bro) and (Netflow.bro&scan.bro) because it had lowest p-value. Review Table D-9 that indicated the (Netflow.bro&Ftp.bro) had the greatest effect in this interaction.

Table D-3 shows the mean values of combination differences for the significant treatment factor (Netflow.bro&Http.bro)*(Netflow.bro&Ftp.bro) * (Netflow.bro&scan.bro) , while the Tukey-Kramer analysis is shown in Table D-13 with summery information in Table D-15. The pairwise comparisons show that there are eleven statistically significant treatment factors at 1% confidence factor in mean values as listed below:

Combination 1	Combination 2	P-Value
H=off, FT=off, SP=off	H=off, FT=off, SP=on	2.03E-25
H=off, FT=off, SP=off	H=off, FT=on, SP=off	2.18E-20
H=off, FT=off, SP=off	H=on, FT=off, SP=off	2.58E-06
H=off, FT=off, SP=off	H=off, FT=on, SP=on	2.45E-08
H=off, FT=off, SP=off	H=on, FT=off, SP=on	4.12E-10
H=off, FT=off, SP=off	H=on, FT=on, SP=off	4.18E-21
H=on, FT=on, SP=on	H=off, FT=off, SP=on	5.83E-33
H=on, FT=on, SP=on	H=off, FT=on, SP=off	0.000636
H=on, FT=on, SP=on	H=on, FT=off, SP=off	1.14E-06
H=on, FT=on, SP=on	H=off, FT=on, SP=on	0.000484
H=on, FT=on, SP=on	H=on, FT=on, SP=off	5.16E-25

H = Netflow.bro&Http.bro, FT = Netflow.bro&Ftp.bro, SF = Netflow.bro&scan.bro

The greatest effect of this treatment factor comparisons is (Netflow.bro&scan.bro) because it had lowest p-value, but it is influenced by both of (Netflow.bro&Http.bro) and (Netflow.bro&Ftp.bro).

Table D-3 shows the mean values of combination differences for the significant treatment factor Frag.bro*(Netflow.bro&Http.bro)*(Netflow.bro&Ftp.bro) *(Netflow.bro& scan.bro) , while the Tukey-Kramer analysis is shown in Table D-14 with summery information in Table D-15. The pairwise comparisons show that there are twenty five statistically significant treatment factors at 1% confidence factor in mean values as listed below:

Combination 1	Combination 2	P-Value
F=off, H=off, FT=off, SP=off	F=off, H=off, FT=off, SP=on	4.9E-09
F=off, H=off, FT=off, SP=off	F=off, H=off, FT=on, SP=off	5.37E-07
F=off, H=off, FT=off, SP=off	F=off, H=on, FT=off, SP=off	3.66E-08
F=off, H=off, FT=off, SP=off	F=off, H=off, FT=on, SP=on	1.52E-10
F=off, H=off, FT=off, SP=off	F=off, H=on, FT=off, SP=on	0.00106
F=off, H=off, FT=off, SP=off	F=off, H=on, FT=on, SP=off	2.39E-09
F=off, H=off, FT=off, SP=off	F=on, H=off, FT=off, SP=on	5.28E-09
F=off, H=off, FT=off, SP=off	F=on, H=off, FT=on, SP=off	3.27E-10
F=off, H=off, FT=off, SP=off	F=off, H=on, FT=on, SP=on	2.09E-10
F=off, H=off, FT=off, SP=off	F=on, H=off, FT=on, SP=on	0.001195
F=off, H=off, FT=off, SP=off	F=on, H=on, FT=off, SP=on	2.26E-10
F=off, H=off, FT=off, SP=off	F=on, H=on, FT=on, SP=off	4.12E-09
F=on, H=on, FT=on, SP=on	F=off, H=off, FT=off, SP=on	1.15E-25
F=on, H=on, FT=on, SP=on	F=off, H=off, FT=on, SP=off	2.24E-08
F=on, H=on, FT=on, SP=on	F=off, H=on, FT=off, SP=off	4.77E-19
F=on, H=on, FT=on, SP=on	F=on, H=off, FT=off, SP=off	2.98E-20
F=on, H=on, FT=on, SP=on	F=off, H=off, FT=on, SP=on	2.32E-10
F=on, H=on, FT=on, SP=on	F=off, H=on, FT=off, SP=on	0.00391
F=on, H=on, FT=on, SP=on	F=off, H=on, FT=on, SP=off	2.16E-15
F=on, H=on, FT=on, SP=on	F=on, H=off, FT=off, SP=on	6.28E-25
F=on, H=on, FT=on, SP=on	F=on, H=on, FT=off, SP=off	0.000222
F=on, H=on, FT=on, SP=on	F=off, H=on, FT=on, SP=on	0.000463
F=on, H=on, FT=on, SP=on	F=on, H=off, FT=on, SP=on	2.34E-11
F=on, H=on, FT=on, SP=on	F=on, H=on, FT=off, SP=on	2.22E-06
F=on, H=on, FT=on, SP=on	F=on, H=on, FT=on, SP=off	3.13E-24

F = Frag.bro, H = Netflow.bro&Http.bro, FT = Netflow.bro&Ftp.bro, SF = Netflow.bro&scan.bro

The greatest effect of this treatment factor comparisons is (Netflow.bro&scan.bro) because it had lowest p-value, but it is influenced by both of Frag.bro and (Netflow.bro&Http.bro) and (Netflow.bro&Ftp.bro).

Next, examine the practical significance of the end-to-end delay time introduced by the Real-Time Intrusion Detection System will be made. This will be

determined by looking at the per-packet delay. This information is shown in Table C-4 (Appendix C). The delays calculated will be compared to the information in Table A-1 (Appendix A) to see if any treatment factor combination will generate a per packet delay greater than 150 µs. Using the information in Table C-4 gives us the treatment factor combination that provided the largest per packet delay increase. This treatment factor is frag.bro*netflow.bro, which had an increase in per packet delay of 0.066696 seconds or 66.696 µs. This increase is well below the 150 µs mark for acceptable delay in end-to-end traffic and all other treatment factor combinations provide even smaller per packet increases. The smallest increase in per packet delay of 0.066681 second or 66.681 µs came from the treatment factor combination of netflow.bro*http.bro.

CHAPTER FIVE

CONCLUSIONS

5.1 Summary of the Analysis Method

The data has been collected, the analysis has been done, and the answers to the research questions should now be available. To begin with, data has been gathered on end-to-end delay from ten different repetitions of each treatment factor. Averaging this information allowed a determination of how much end-to-end delay is introduced by each individual treatment factor combination. This information provided a very good idea of how a real time intrusion prevention system will perform.

Next, the gathered data mean values were compared to the elapsed time of the original packet capture to determine which processing options introduced the largest amount of end-to-end delay.

Finally, this gathered data was analyzed through an analysis of variance methodology to determine which configuration options of Bro introduced the largest amount of delay. The analysis of variance performed gave indication that 15 treatment factor combinations provided statistically significant results. These 15 combinations were then analyzed using a Tukey-Kramer method to gain some insight into what specific combinations generated the indication of significance from the analysis of variance. In conjunction with this, the data was also broken down into a per packet delay to look at the practical significance of each treatment factor combination. Even though the analysis of variance gave indication of statistically significant results, none

of the treatment factor combinations gave indication of having any practical significance on end-to-end delay.

5.2 Conclusions

Through the methodology followed, answers the research questions can be determined after the data has been obtained. Question 1 asks how much end-to-end delay is introduced by the implementation of a real-time intrusion prevention system. Using the Bro in a real-time network intrusion prevention system and considering the many various combinations of options and configurations the amount of end-to-end delay will range from 75.644 μs to 53.281 μs. This is a very small increase in relative terms and would be very difficult to notice in a normal operating environment. The null hypothesis for this situation was that the increase would not be noticeable. Considering this, the conclusion is to fail to reject the null hypothesis.

Question 2 asked which preprocessing options in Bro would introduce the most delay. This question can be easily answered by simply looking at the amount of delay introduced by each treatment factor option and finding the option with the largest delay. This option was running Bro with the Frag.bro and Netflow.bro preprocessor turned on while Http.bro, Ftp.bro, and scan.bro preprocessors are turned off. The null hypothesis for this question was that a treatment factor combination including the Frag.bro or Netflow.bro or a combination with both preprocessors turned on would generate the most end-to-end delay. For this question, the conclusion is to again, fail to reject the null hypothesis. The nature of the Frag.bro and Netflow.bro preprocessors make them system resource intensive and thus generate longer delays.

Question 3 then asked which treatment factor would introduce a statistically or practically significant delay. This analysis indicated that the overall model was significant by calculating a significance factor of 0.000 at a confidence level of 5%. To be more precise, this analysis repeated with confidence level 1% again the overall model was significant gave 0.000 significance factor. The analysis of variance performed provided fifteen treatment factor combinations that provided statistically significant results. These fifteen combinations with individual significance factors are:

Netflow.bro	.000
Frag.bro*Netflow.bro	.004
netflow.bro*Http.bro	.000
Frag.bro*Netflow.bro*Http.bro	.000
Netflow.bro*Ftp.bro	.000
Frag.bro*Netflow.bro*Ftp.bro	.000
Frag.bro*Netflow.bro*Http.bro*Ftp.bro	.000
Netflow.bro*scan.bro	.001
Frag.bro*Netflow.bro*scan.bro	.001
Netflow.bro*Http.bro*scan.bro	.010
Netflow.bro*Ftp.bro*scan.bro	.000
Frag.bro*Netflow.bro*Http.bro*scan.bro	.000
Frag.bro*Netflow.bro*Ftp.bro*scan.bro	.000
Netflow.bro*Http.bro*Ftp.bro*scan.bro	.001
Frag.bro*Netflow.bro*Http.bro*Ftp.bro*scan.bro	.000

The null hypothesis for the question of statistical significance was that the statistically significant treatment factors would have Frag.bro or Netflow.bro or a combination of both turned on. The results for this question are failing again to reject the null hypothesis. Overall the model treatment factors combinations include either Frag.bro or Netflow.bro or both. Five of the significant treatment factors combinations indicate that Frag.bro preprocessor added the greatest effect to the delay time. Three combinations indicate that scan.bro preprocessor added the greatest effect to the delay time when it associated with Netflow.bro. Two combinations indicate that Ftp.bro preprocessor added the greatest effect to the delay time when it associated

with Netflow.bro. Single combinations indicate that Ftp.bro preprocessor added the greatest effect to the delay time when it associated with Netflow.bro.

The final item to be studied was that of the practical effects of a real-time intrusion prevention system. From the analysis and data gathered, the highest per packet delay achieved was with the treatment factor of Frag.bro and Netflow.bro. This treatment factor achieved a per packet delay of 58.988883 μs. The null hypothesis for this portion of the study stated that the per packet delay would be less than 150 μs. It is clear that the per packet delay is considerably less than the 150 ms, which means that there is no practical significance added to the end-to-end delay. The conclusion is to fail to reject the null hypothesis.

Overall, there is a good indication that the Bro is capable of carrying the load of a production network with minimal statistical impact and virtually no practical impact. This is an unexpected result as much larger delays were thought to be introduced by the real-time intrusion prevention system associated with a variance between treatment factors. The Bro pre-processors obviously require processing cycles to complete their work, which in theory should increase the end-to-end delay of network traffic. However, the preprocessors save processor cycles by reducing the amount of traffic that must pass through the pattern matching engine, offsetting the increased processing cycles, creating a very operationally efficient security tool.

5.3 Future Works

While the results from this study are quite interesting and provide some useful information on Bro performance in a real-time Intrusion Prevention System from its effect on the environment viewpoint. A research question arising from these results is: would such a system still be able to maintain the performance, while also efficiently and accurately detecting intrusions, malware, and other malicious network activities?

This would required network traffic containing known attacks and malicious traffic, which is quite difficult to obtain and verify prior to use.

This study allowed Bro to use the basic logging system to track the alerts and prevention reports. Such logs are quite difficult to read carefully and analyze and can be time consuming. Bro provides several more user friendly options for logging of the engine alerts. However, when using these advanced options, how will they affect the processing performance as well as the prevention abilities of the Bro system?

These are some topics to be considered in future research projects. Each provides specific questions that could be answered following a similar methodology used for this study.

REFERENCES

Anderson, J. R. (1980). *Computer security threat monitoring and surveillance.*

Anttila, J. (2004). *Intrusion Detection in Critical E-business Environment.* Helsinki University of Technology, Finland.

Archibald, N., Ramirez, G., & Rathaus, N. (2005). *Nessus, Snort, & Ethereal Power Tools: Customizing Open Source security Application.* USA: Syngress Publishing, Inc.

Asarcikli, S. (2005). *Firewall Monitoring Using Intrusion Detection Systems.* Izmir Institute of Technology, Izmir.

Attig, M., & Lockwood, J. (2005). *SIFT:Snort Intrusion Filter for TCP.* Paper presented at the 13th IEEE Symposium on High Performance Interconnects.

Axelsson, S. (2006). *Understanding Intrusion Detection Through Visualization.* USA: Springer.

Bace, R., & Mell, P. (2001). Intrusion Detection Systems [Electronic Version] from http://www-cse.ucsd.edu/classes/fa01/cse221/projects/group10.pdf.

Baker, A. R., Caswell, B., & Poor, M. (2004). *Snort 2.1 Intrusion Detection* USA: Syngress Publishing, Inc.

Baker, A. R., & Esler, J. (2007). *Snort IDS and IPS Toolkit.* Burlington: Syngress Publishing, Inc.

Balzarotti, D. (2006). *Testing Network Intrusion Detection Systems.*

Politecnico di Milano, Italy.

Beale, J., & Foster, J. C. (2003). *Snort 2.0 Intrusion Detection*. USA: Syngress Publishing.

Capite, D. D. (2007). *Self-Defending Networks : The Next Genration of Network Security*. Indianapolis,USA: Cisco Press.

Caruso, L. C., Guuindani, G., Schmitt, H., Neycalazans, & Moraes, F. (2007). *SPP-NIDS - A Sea of Processors Platform for Network Intrusion Detection Systems*. Paper presented at the 18th IEEE/IFIP International Workshop on Rapid System Prototyping(RSP07).

Chang, Y. K., Tsai, M. L., & Chung, Y. R. (2008). *Multi-Character Processor Array for Pattern Matching in Network Intrusion Detection System.* Paper presented at the 22nd IEEE International Conference on Advanced Information Networking and Applications, AINA

Cisco. (2007). Understanding Delay in Packet Voice Networks [Electronic Version] from http://www.cisco.com/warp/public/788/voip/delay-details.html.

Crothers, T. (2003). *Implementing Intrusion Detection Systems*. Indiana: Wiley Publishing, Inc.

Dries, J. (2001). An Introduction to Snort: A Lightweight Intrusion Detection System [Electronic Version] from http://www.informit.com/articles/article.aspx?p=21777.

Graham, J. M. (2000). Interaction Effects: Their Nature and Some Post Hoc Exploration Strategies [Electronic Version] from http://ericae.net/ft/tamu/interaction.pdf.

Greensmith, J. (2007). *The Dendritic Cell Algorithm.* University of Nottingham.

Guerrero, J. H., & Cardenas, R. G. (2005). An example of communication between security tools: Iptables - Snort. *ACM, 39*(3), 34-43.

Hutchings, B. L., Franklin, R., & Carver, D. (2002). *Assisting Network Intrusion Detection with Reconfigurable Hardware.* Paper presented at the Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02).

Jeong, Y., Jeon, J., Ryu, J., & Seo, D. (2006). *A Developing of Signature-based Network Security Tester for NGSS.* Paper presented at the 8th IEEE International Conference Advanced Communication Technology, ICACT

Kim, Y., Jung, B., Lim, J., & Kim, K. (2007). *Processing of Multi-pattern Signature in Intrusion Detection System with Content Processor.* Paper presented at the 6th IEEE International Conference on Information, Communications & Signal Processing.

Korenek, J., & Kobiersky, P. (2007). *Intrusion Detection System Intended for Multigigabit Networks.* Paper presented at the Design and Diagnostics of Electronic Circuits and Systems, DDECS.

Koziol, J. (2003). *Intrusion Detection with Snort*: Sams Publishing.

Lauf, A. P. (2007). *Hybrids: Embeddable Hybrid Intrusion Detection System.* Vanderbilt University.

Lippmann, R., Haines, J., Fried, D., Korba, J., & Das, K. (2000). *The 1999 DARPA Off-Line Intrusion Detection Evaluation*: Lincoln Laboratory MIT.

Lussi, C. (2008). *Signature-based Extrusion Detection.* Swiss Federal Institute of Technology Zurich.

Maxwell, S., & Delaney, H. (2004). *Designing Experiments and Analyzing Data* (2nd ed.): Lawrence Erlbaum Associates.

May, C., Hammerstein, J., Mattson, J., & Rush, K. (2006). *Defense-in-Depth: Foundations for Secure and Resilient IT Enterprises*: Carnegie Mellon University.

McHugh, J., Christie, A., & Allen, J. (2000). Defending Yourself: The Role of Intrusion Detection Systems. *Software, IEEE, 17*(5).

Mukherjee, B., Heberlein, L., & Levitt, K. (1994). Network Intrusion Detection. *IEEE Network, 8*(4), 26-41.

Newman, D., Manalo, K., & Tittel, E. (2004). CSIDS Exam Cram 2 [Electronic Version] from http://www.informit.com/articles/article.aspx?p=174342&seqNum=1.

Northcutt, S., & Novak, J. (2003). *Network Intrusion Detection* (3rd ed.): New Riders.

Novak, J., & Sturges, S. (2007). Target-Based TCP Stream Reassembly [Electronic Version] from http://www.snort.org/docs/stream5-model-Aug032007.pdf.

NSSlabs. (2008). Gigabit Intrusion Detection Systems (IDS) [Electronic Version] from http://www.nsslabs.com/white-papers/gigabit-intrusion-detection-systems-ids.html.

NSSLabs. (2008). Intrusion Prevention Systems (IPS) [Electronic Version] from http://nsslabs.com/white-papers/intrusion-prevention-systems-ips.html.

Oksuz, A. (2007). *Unsupervised Intrusion Detection System.* Technical University of Denmark.

Papini, D. (2008). *An Anomaly based Wireless Intrusion Detection System.* Technical University of Denmark.

Perdisci, R. (2006). *Statistical Pattern Recognition Techniques for Intrusion Detection in Computer Networks: Challenges and Solutions.* Universita degli Studi di Cagliari, Cagliari, Italy.

Pfleeger, C., & Pefleeger, S. (2007). *Security in Computing* (4th ed.). USA: Pearson Education, Inc.

Pipa, D. (2008). *Intrusion Detection and Prevention: Immunologically Inspired Approaches.* University of London.

Puketza, N. (2000). *Approches to Computer Security: Filtering. Testing, and Detection.* University of California Davis.

Rehman, R. (2003). *Intrusion Detection Systems with Snort* (1st ed.). New Jersey Printice Hall PTR.

Roesch, M. (1999). Snort - Lightweight Intrusion Detection for Networks [Electronic Version] from http://www.snort.org/docs/lisapaper.txt.

Safiee, M. (2007). *An Intrusion Detection System (IDS) For Internet Network.* Universiti Teknologi Malaysia.

Schwartz, D., Stoecklin, S., & Yilmaz, E. (2002). *A Case-Based Approach to Network Intrusion Detection.* Paper presented at the 5th IEEE International Conference on Information Fusion.

Smith, C. (2003). *Understanding Concepts in the Defence in Depth Strategy.* Paper presented at the 37th Annual IEEE International Carnahan Conference on Security Technology.

Snyder, J. (2008). Six Strategies for Defense-in-depth: Securing the Network from the Inside Out [Electronic Version] from

http://www.arubanetworks.com/pdf/technology/whitepapers/wp_Defense-in-depth.pdf.

Sommer, R. (2005). *Viable Network Intrusion Detection in High-Performance Environments.* Technische Universitat Munchen.

Song, H., Sproull, T., Attig, M., & Lockwood, J. (2005). *Snort Offloader: A Reconfigurable Hardware NIDS Filter.* Paper presented at the IEEE International Conference on Field Programmable Logic and Applications.

Sourdis, I., Dimopoulos, V., Pnevmatikatos, D., & Vassiliadis, S. (2006). *Packet Pre-filtering for Network Intrusion Detection.* Paper presented at the 2006 ACM/IEEE symposium on Architecture for networking and communications systems, California,USA.

Tenhunen, T. (2008). *Implementing An Intrusion Detection System In The Mysea Architecture.* Naval Postgraduate School, Monterey, California.

Thomas, T. (2004). *Network Security: first-step.* Indianapolis, USA: Cisco Press.

Vallentin, M. (2006). *Transparent Load-Balancing for Network Intrusion Detection Systems.* Technische Universitat Munchen.

Wagoner, R. (2007). *Performance Testing An Inline Network Intrusion Detection System Using Snort.* Morehead State University.

Wagoner, R. (2007). *Performance Testing an Inline Network Intrusion Detection System Using Snort.* Morehead State University, Morehead.

Wan, T., & Yang, X. (2001). *IntruDetector: A Software Platform for Testing Network Intrusion Detection Algorithms.* Paper presented at the 17th Annual IEEE Computer Security Application Conferance.ACSAC.

Wu, Y., Foo, B., Mei, Y., & Bagchi, S. (2003). *Collaborative Intrusion Detection System (CIDS): A Framework for Accurate and Efficient IDS.* Paper presented at the 19th Annual IEEE Computer Security Applications Conference ACSAC

Yaacob, N. (2003). *Utilizing Snort in the analysis of intrusion Detection System.* University Utara Malaysia, Kedah.

Zamboni, D. (2001). *Using Internal Sensors for Computer Intrusion Detection.* Purdue University, Purdue

Zanero, S. (2006). *Unsupervised Learning Algorithms for Intrusion Detection.* Politecnico Milano University, Milano.

1. What is Bro?

Bro is a Unix-based Network Intrusion Detection System (IDS). Bro monitors network traffic and detects intrusion attempts based on the traffic characteristics and content. Bro detects intrusions by comparing network traffic against rules describing events that are deemed troublesome. These rules might describe activities (e.g., certain hosts connecting to certain services), what activities are worth alerting (e.g., attempts to a given number of different hosts constitutes a "scan"), or signatures describing known attacks or access to known vulnerabilities. If Bro detects something of interest, it can be instructed to either issue a log entry or initiate the execution of an operating system command. Bro targets high-speed (Gbit/second), high-volume intrusion detection. By judiciously leveraging packet filtering techniques, Bro is able to achieve the performance necessary to do so while running on commercially available PC hardware, and thus can serve as a cost effective means of monitoring a site's Internet connection.

2. Installation and Configuration

Download Bro from: http://www.bro-ids.org/ You can unpack the distribution anywhere except into the directory you plan to install in to. To untar the file, type: tar xvzf bro-0.9a6.6.tar.gz, figure A-1 illustrates the package on the desktop:

Figure A-1: The Bro package on the desktop

Bro is very easy to install. Just log in as root, and type:

sudo ./configure

or to install Bro in a location other than '/usr/local/bro', use:

sudo ./configure --prefix=/usr/local/bro --enable-shippedpcap

and then type:

sudo make

sudo make install

The Bro-Lite configuration script can be used to automatically configure Bro for you. It checks your system's BPF settings, creates a 'bro' user account, installs a script to start bro at boot time, and installs a number of cron jobs to checkpoint bro every night, run perioidic reports, and manage log files.

To run this configuration script type:

make install-brolite

This will run the script bro_config, which creates the file '$BROHOME/etc/bro.cfg'. bro_config will ask a number of simple questions.

After installing Bro, we can use it by navigate bash commander as illustrated in figure A-2 and A3

Figure A-2: Navigate the bash commander for the directory for Bro

Figure A-3: Commands to run Bro.

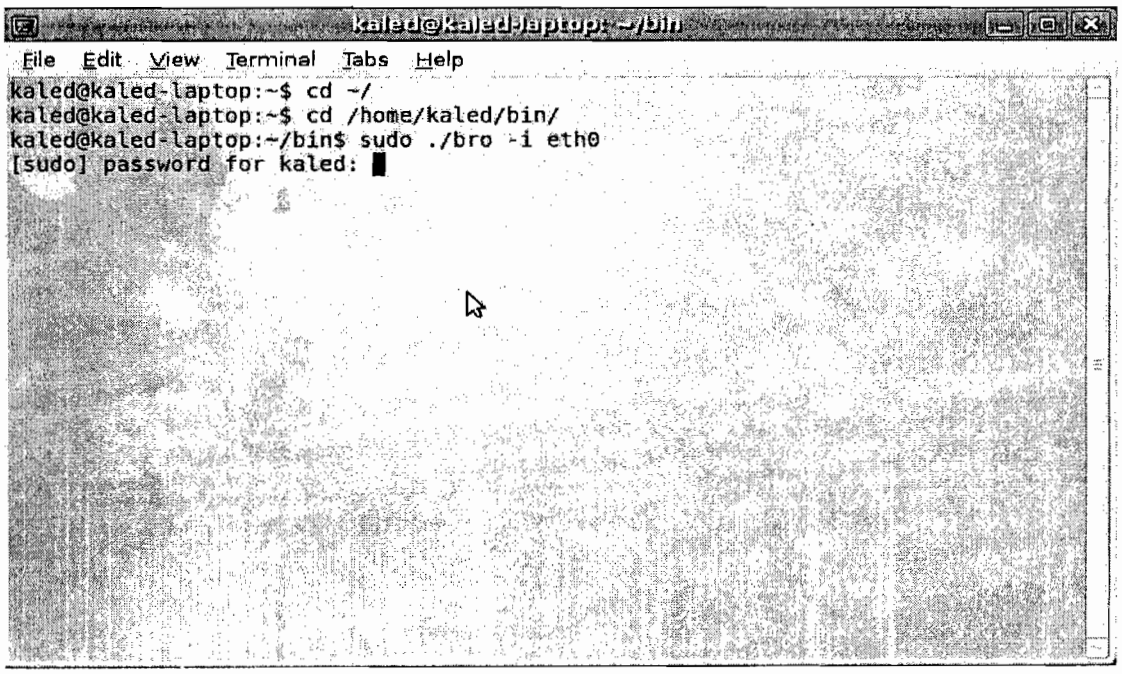

Table B-1 : Delay Specifications (Cisco, 2007).

RANGE IN MILLISECONDS	DESCRIPTION
0-150 ms	Acceptable for most user applications.
150-400 ms	Acceptable provided that administrators are aware of the transmission time and the impact it has on the transmission quality of user applications.
Above 400 ms	Unacceptable for general network planning purposes. However, it is recognized that in some exceptional cases this limit is exceeded.

Table B-2 : Experimental Design

A- frag.bro
B- netflow.bro
C- http.bro
D- ftp.bro
E- scan.bro

RUN	A	B	C	D	E	EXPERIAMENT
1	O	O	O	O	O	O
2	X	O	O	O	O	A
3	O	X	O	O	O	B
4	X	X	O	O	O	AB
5	O	X	X	O	O	BC
6	X	X	X	O	O	ABC
7	O	X	O	X	O	BD
8	X	X	O	X	O	ABD
9	O	X	X	X	O	BCD
10	X	X	X	X	O	ABCD
11	O	X	O	O	X	BE
12	X	X	O	O	X	ABE
13	O	X	X	O	X	BCE
14	O	X	O	X	X	BDE
15	X	X	X	O	X	ABCE
16	X	X	O	X	X	ABDE
17	O	X	X	X	X	BCDE
18	X	X	X	X	X	ABCDE

Table C-1: Experimental Design Observations and Averages Observation .

Experiments	Original	Observation	Delay Time	Average Observation	Average Delay Time
Run1	30.005307	30.007797	0.00249		
	30.00323	30.005434	0.002204		
	30.004825	30.005043	0.000218		
	30.005209	30.007278	0.002069		
	30.004634	30.005696	0.001062	30.0062496	0.0016086
Run2	30.00312	30.01335	0.01023		
	30.004264	30.013022	0.008758		
	30.008657	30.012292	0.003635		
	29.999569	30.010639	0.01107		
	30.006173	30.013525	0.007352	30.0125656	0.008209
Run3	30.006972	30.012142	0.00517		
	30.002772	30.012228	0.009456		
	30.001789	30.01483	0.013041		
	30.00094	30.012267	0.011327		
	30.011	30.013327	0.002327	30.0129588	0.0082642
Run4	30.005369	30.010807	0.005438		
	30.002402	30.00926	0.006858		
	30.00369	30.014179	0.010489		
	30.00043	30.011503	0.011073		
	30.002339	30.011882	0.009543	30.0115262	0.0086802
Run5	30.001515	30.007251	0.005736		
	30.006231	30.012812	0.006581		
	30.007151	30.007296	0.000145		
	30.006107	30.00941	0.003303		
	30.000706	30.00965	0.008944	30.0092838	0.0049418
Run6	30.007806	30.0081	0.000294		
	30.007341	30.010161	0.00282		
	30.006333	30.009213	0.00288		
	30.005245	30.007965	0.00272		
	30.006921	30.011695	0.004774	30.0094268	0.0026976
Run7	30.001233	30.015321	0.014088		
	30.005733	30.011094	0.005361		
	30.00447	30.01394	0.00947		
	30.009698	30.014355	0.004657		
	30.001539	30.010326	0.008787	30.0130072	0.0084726
Run8	30.007323	30.008337	0.001014		
	30.003324	30.009989	0.006665		
	30.003224	30.012679	0.009455		
	30.001342	30.014382	0.01304		
	30.000645	30.013236	0.012591	30.0117246	0.008553
Run9	30.005797	30.007898	0.002101		
	30.001005	30.012377	0.011372		
	29.999899	30.00823	0.008331		
	30.002199	30.0093	0.007101		
	29.999812	30.013543	0.013731	30.0102696	0.0085272
Run10	30.001625	30.012677	0.011052		
	30.002837	30.007689	0.004852		

	30.003384	30.007586	0.004202		
	30.00018	30.005094	0.004914		
	30.003756	30.006578	0.002822	30.0079248	0.0055684
Run11	30.001627	30.007687	0.00606		
	30.002876	30.006578	0.003702		
	30.001876	30.006578	0.004702		
	30.002765	30.007658	0.004893		
	30.004857	30.008635	0.003778	30.0074272	0.004627
Run12	30.007192	30.008768	0.001576		
	30.002874	30.008163	0.005289		
	30.004657	30.007568	0.002911		
	30.001746	30.007682	0.005936		
	30.001984	30.007687	0.005703	30.0079736	0.004283
Run13	30.004775	30.008752	0.003977		
	29.999499	30.008475	0.008976		
	30.005867	30.007683	0.001816		
	30.004958	30.007683	0.002725		
	30.005768	30.008573	0.002805	30.0082332	0.0040598
Run14	30.001758	30.008123	0.006365		
	30.002768	30.00856	0.005792		
	30.001874	30.008174	0.0063		
	30.002758	30.00715	0.004392		
	30.001785	30.00367	0.001885	30.0071354	0.0049468
Run15	30.005712	30.006787	0.001075		
	30.002758	30.008471	0.005713		
	30.001896	30.008461	0.006565		
	30.0018	30.006783	0.004983		
	30.001098	30.00678	0.005682	30.0074564	0.0048036
Run16	30.002912	30.007184	0.004272		
	30.00387	30.007613	0.003743		
	30.00297	30.008761	0.005791		
	30.00461	30.008561	0.003951		
	30.003287	30.008576	0.005289	30.008139	0.0046092
Run17	30.001098	30.006387	0.005289		
	30.001758	30.008275	0.006517		
	30.001764	30.002759	0.000995		
	30.001788	30.005768	0.00398		
	30.0019	30.002817	0.000917	30.0052012	0.0035396
Run18	30.001	30.004657	0.003657		
	30.001784	30.008934	0.00715		
	30.002819	30.00874	0.005921		
	30.003758	30.005785	0.002027		
	30.001773	30.003874	0.002101	30.006398	0.0041712
END					

Table C-2: End-to-End Delay Time Introduced by IDS Engine.

Preprocessors Options	Average Observe Time	Original Capture	Delay-Time Introduced
None	30.0062496	29.999	0.00725
frag.bro	30.0125656	29.999	0.013566
netflow.bro	30.0129588	29.999	0.013959
frag.bro*netflow.bro	30.0115262	29.999	0.012526
netflow.bro*http.bro	30.0092838	29.999	0.010284
frag.bro*netflow.bro*http.bro	30.0094268	29.999	0.010427
netflow.bro*ftp.bro	30.0130072	29.999	0.014007
frag.bro*netflow.bro*ftp.bro	30.0117246	29.999	0.012725
netflow.bro*http.bro*ftp.bro	30.0102696	29.999	0.01127
frag.bro*netflow.bro*http.bro *ftp.bro	30.0079248	29.999	0.008925
netflow.bro*scan.bro	30.0074272	29.999	0.008427
frag.bro*netflow.bro*scan.bro	30.0079736	29.999	0.008974
netflow.bro*http.bro*scan.bro	30.0082332	29.999	0.009233
netflow.bro* ftp.bro*scan.bro	30.0071354	29.999	0.008135
frag.bro*netflow.bro* http.bro *scan.bro	30.0074564	29.999	0.008456
frag.bro*netflow.bro*ftp.bro* scan.bro	30.008139	29.999	0.009139
netflow.bro*http.bro*ftp.bro* scan.bro	30.0052012	29.999	0.006201
frag.bro*netflow.bro*http.bro *ftp.bro*scan.bro	30.006398	29.999	0.007398

Table C-3: End-to-End Delay Introduced by IDS Engine Sorted by Delay-Time.

Preprocessors Options	Average Observe Time	Original Capture	Delay-Time Introduced
netflow.bro*ftp.bro	30.01301	29.999	0.014007
netflow.bro	30.01296	29.999	0.013959
frag.bro	30.01257	29.999	0.013566
frag.bro*netflow.bro*ftp.bro	30.01172	29.999	0.012725
frag.bro*netflow.bro	30.01153	29.999	0.012526
netflow.bro*http.bro*ftp.bro	30.01027	29.999	0.01127
frag.bro*netflow.bro*http.bro	30.00943	29.999	0.010427
netflow.bro*http.bro	30.00928	29.999	0.010284
netflow.bro*http.bro*scan.bro	30.00823	29.999	0.009233
frag.bro*netflow.bro*ftp.bro* scan.bro	30.00814	29.999	0.009139
frag.bro*netflow.bro*scan.bro	30.00797	29.999	0.008974
frag.bro*netflow.bro*http.bro *ftp.bro	30.00792	29.999	0.008925
frag.bro*netflow.bro* http.bro *scan.bro	30.00746	29.999	0.008456
netflow.bro*scan.bro	30.00743	29.999	0.008427
netflow.bro* ftp.bro*scan.bro	30.00714	29.999	0.008135
frag.bro*netflow.bro*http.bro *ftp.bro*scan.bro	30.0064	29.999	0.007398
None	30.00625	29.999	0.00725

Table C-4: End-to-End Delay Introduced by IDS Engine per-Packet Sorted by Delay.

Total Packets Number (T) = 8254

Delay (per Packet) = Average Observe Time / T

Preprocessors Options	Average Observe Time	Delay/packet
frag.bronetflow.bro	30.0130072	0.066696
netflow.bro	30.0129588	0.066695
frag.bro	30.0125656	0.066695
frag.bro*netflow.bro*ftp.bro	30.0117246	0.066693
netflow.bro*ftp.bro	30.0115262	0.066692
netflow.bro*http.bro*ftp.bro	30.0102696	0.066689
frag.bro*netflow.bro*http.bro	30.0094268	0.066688
frag.bro*netflow.bro*http.bro*ftp.bro* scan.bro	30.0092838	0.066687
netflow.bro*http.bro*scan.bro	30.0082332	0.066685
frag.bro*netflow.bro*ftp.bro*scan.bro	30.008139	0.066685
frag.bro*netflow.bro*scan.bro	30.0079736	0.066684
frag.bro*netflow.bro*http.bro*ftp.bro	30.0079248	0.066684
frag.bro*netflow.bro* http.bro *scan.bro	30.0074564	0.066683
netflow.bro*scan.bro	30.0074272	0.066683
netflow.bro* ftp.bro*scan.bro	30.0071354	0.066683
netflow.bro*http.bro	30.006398	0.066681
None	30.0062496	0.066681

Figure D-1: SPSS Univariate Factor Configurations

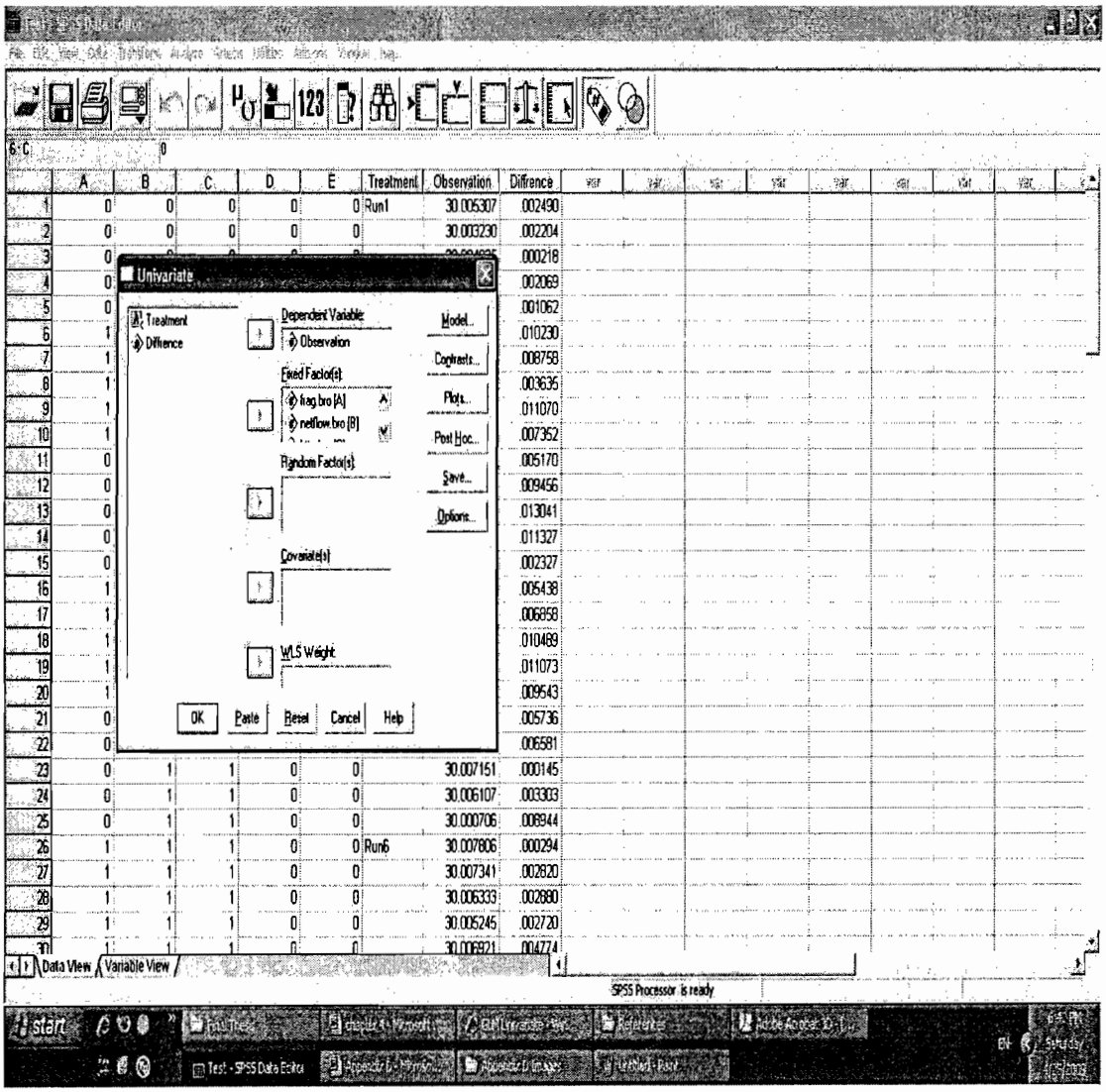

Figure D-2: SPSS Univariate Factor Configurations

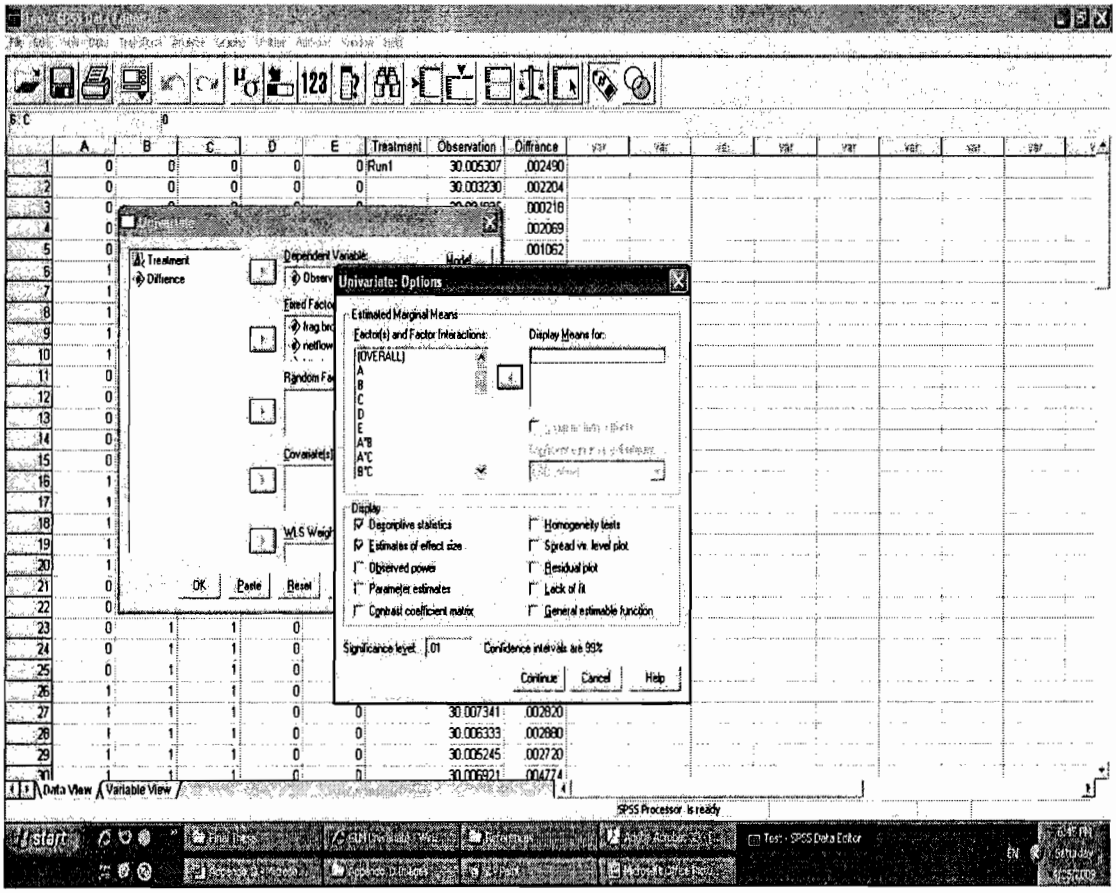

Figure D-3: SPSS Univariate Factor Configurations

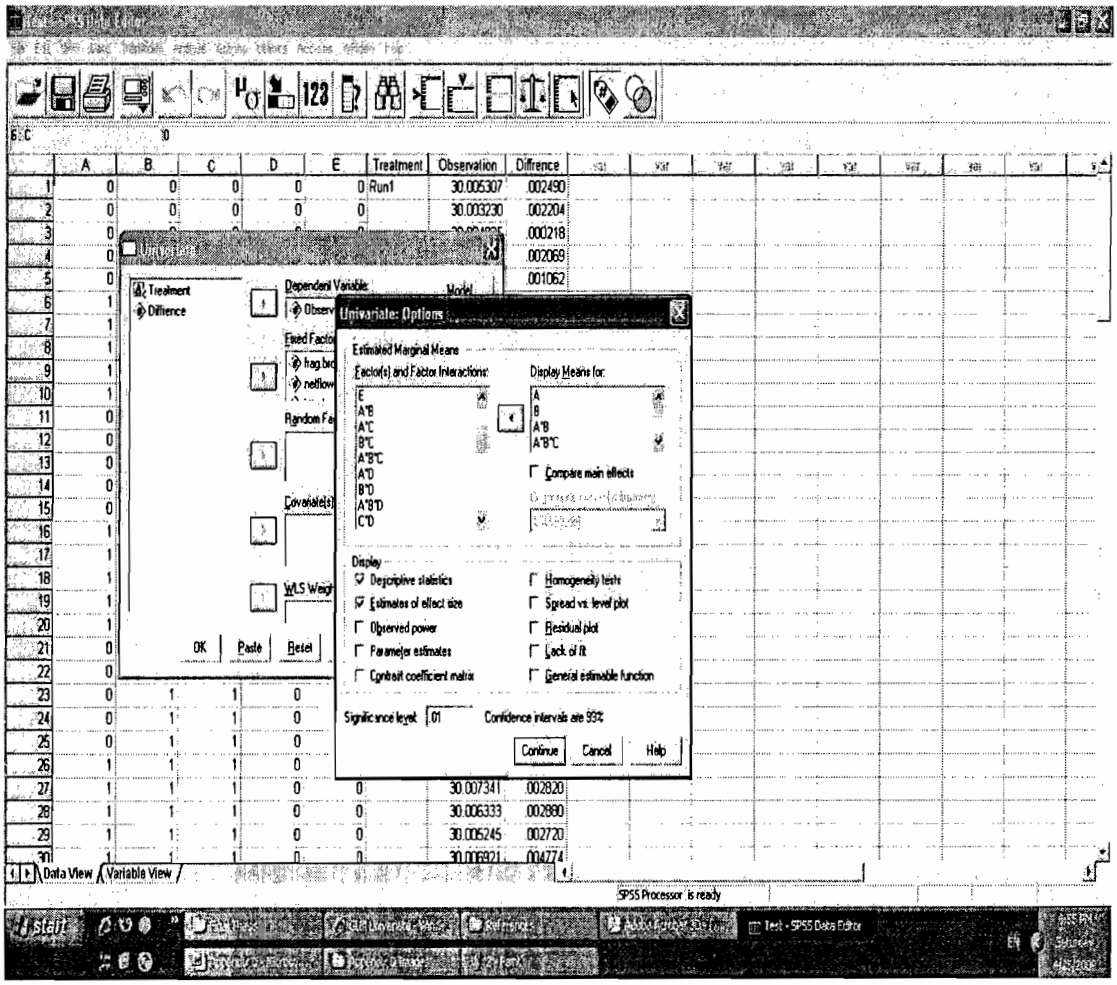

Figure D-4: SPSS Univariate Factor Configurations

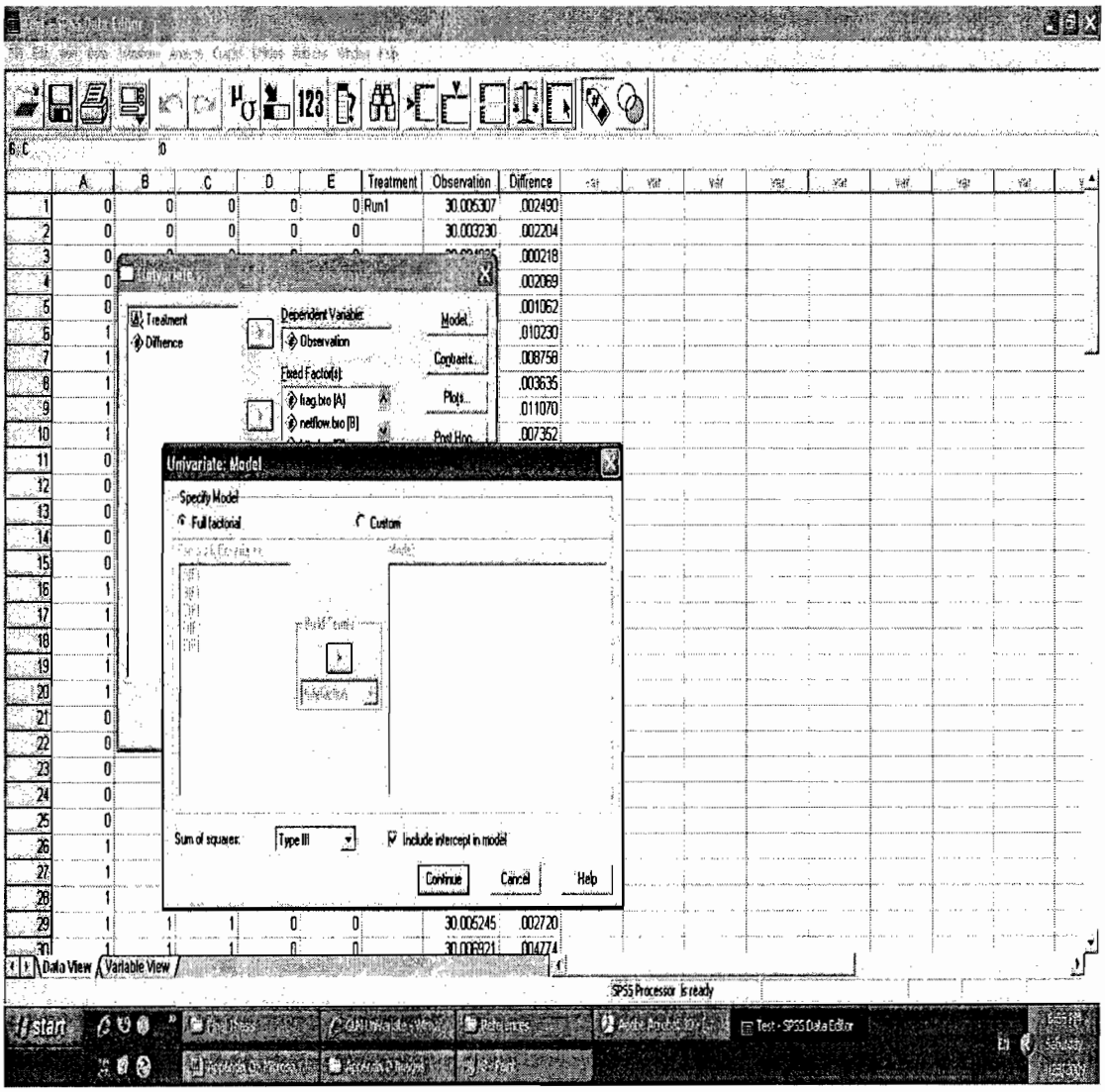

Table D-1: SPSS ANOVA Results with (5%) Confidence Interval.

Dependent Variable: Observation

Source	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	0.000144983	17	8.52844E-06	1.717067	0.05905194	0.288468219
Intercept	62060.03435	1	62060.03435	1.249E+10	2.257E-298	0.999999994
A	1.29944E-06	1	1.29944E-06	0.2616213	0.61057364	0.003620473
B	2.65356E-06	1	2.65356E-06	0.5342528	0.46719635	0.007365525
A * B	7.71656E-07	1	7.71656E-07	0.1553607	0.69462922	0.002153142
B * C	3.45648E-05	1	3.45648E-05	6.9590745	0.01021301	0.088135209
A * B * C	1.75453E-05	1	1.75453E-05	3.5324759	0.06422459	0.046767643
B * D	3.0584E-06	1	3.0584E-06	0.6157612	0.43520116	0.008479718
A * B * D	2.86108E-06	1	2.86108E-06	0.5760341	0.45034863	0.007936974
B * C * D	4.875E-07	1	4.875E-07	0.0981505	0.75496624	0.001361346
A * B * C * D	2.70479E-05	1	2.70479E-05	5.4456676	0.02241281	0.070315975
B * E	1.21602E-07	1	1.21602E-07	0.0244827	0.876101	0.000339921
A * B * E	6.90619E-07	1	6.90619E-07	0.1390452	0.71032758	0.001927461
B * C * E	6.28174E-07	1	6.28174E-07	0.126473	0.72315721	0.001753489
A * B * C * E	2.76113E-05	1	2.76113E-05	5.5591062	0.02110483	0.071675738
B * D * E	2.99654E-06	1	2.99654E-06	0.6033064	0.4398636	0.008309628
A * B * D * E	4.57016E-06	1	4.57016E-06	0.9201295	0.34065115	0.012618319
B * C * D * E	7.72711E-06	1	7.72711E-06	1.555732	0.21633314	0.021150385
A * B * C * D * E	4.73802E-06	1	4.73802E-06	0.9539267	0.33199204	0.013075742
Error	0.000357614	72	4.96686E-06			
Total	81018.70304	90				
Corrected Total	0.000502598	89				

a R Squared = .288 (Adjusted R Squared = .120)

Table D-2: SPSS ANOVA Results with (1%) Confidence Interval.

Dependent Variable: Observation

Source	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	0.000144983	17	8.52844E-06	1.717067018	0.059051944	0.288468219
Intercept	62060.03435	1	62060.03435	12494812142	2.2567E-298	0.999999994
A	1.29944E-06	1	1.29944E-06	0.261621254	0.610573636	0.003620473
B	2.65356E-06	1	2.65356E-06	0.534252836	0.467196352	0.007365525
A * B	7.71656E-07	1	7.71656E-07	0.155360725	0.694629223	0.002153142
B * C	3.45648E-05	1	3.45648E-05	6.959074548	0.010213008	0.088135209
A * B * C	1.75453E-05	1	1.75453E-05	3.532475888	0.064224589	0.046767643
B * D	3.0584E-06	1	3.0584E-06	0.615761165	0.435201155	0.008479718
A * B * D	2.86108E-06	1	2.86108E-06	0.57603408	0.450348626	0.007936974
B * C * D	4.875E-07	1	4.875E-07	0.098150523	0.754966236	0.001361346
A * B * C * D	2.70479E-05	1	2.70479E-05	5.445667599	0.022412814	0.070315975
B * E	1.21602E-07	1	1.21602E-07	0.024482653	0.876100997	0.000339921
A * B * E	6.90619E-07	1	6.90619E-07	0.139045199	0.710327577	0.001927461
B * C * E	6.28174E-07	1	6.28174E-07	0.126472961	0.723157214	0.001753489
A * B * C * E	2.76113E-05	1	2.76113E-05	5.559106157	0.021104832	0.071675738
B * D * E	2.99654E-06	1	2.99654E-06	0.603306438	0.439863605	0.008309628
A * B * D * E	4.57016E-06	1	4.57016E-06	0.920129459	0.340651151	0.012618319
B * C * D * E	7.72711E-06	1	7.72711E-06	1.555732027	0.216333138	0.021150385
A * B * C * D * E	4.73802E-06	1	4.73802E-06	0.953926739	0.331992038	0.013075742
Error	0.000357614	72	4.96686E-06			
Total	81018.70304	90				
Corrected Total	0.000502598	89				

a R Squared = .288 (Adjusted R Squared = .120)

Table D-3: Statistically Significant Treatment Factor.

1. frag.bro

Dependent Variable: Observation

frag.bro	Mean	Std. Error	95% Confidence Interval	
			Lower Bound	Upper Bound
0	30.003(a)	.000	30.003	30.004
1	30.004(a)	.000	30.003	30.004

a Based on modified population marginal mean.

2. netflow.bro

Dependent Variable: Observation

netflow.bro	Mean	Std. Error	95% Confidence Interval	
			Lower Bound	Upper Bound
0	30.004(a)	.001	30.003	30.006
1	30.003	.000	30.003	30.004

a Based on modified population marginal mean.

3. frag.bro * netflow.bro

Dependent Variable: Observation

frag.bro	netflow.bro	Mean	Std. Error	95% Confidence Interval	
				Lower Bound	Upper Bound
0	0	30.005(a)	.001	30.003	30.007
	1	30.003	.000	30.003	30.004
1	0	30.004(a)	.001	30.002	30.006
	1	30.003	.000	30.003	30.004

a Based on modified population marginal mean.

4. netflow.bro * http.bro

Dependent Variable: Observation

netflow.bro	http.bro	Mean	Std. Error	95% Confidence Interval	
				Lower Bound	Upper Bound
0	0	30.004(a)	.001	30.003	30.006
	1	.(b)	.	.	.
1	0	30.003	.000	30.003	30.004
	1	30.003	.000	30.003	30.004

a Based on modified population marginal mean.
b This level combination of factors is not observed, thus the corresponding population marginal mean is not estimable.

Table D-3: Statistically Significant Treatment Factor (con.).

5. frag.bro * netflow.bro * http.bro

Dependent Variable: Observation

frag.bro	netflow.bro	http.bro	Mean	Std. Error	95% Confidence Interval	
					Lower Bound	Upper Bound
0	0	0	30.005(a)	.001	30.003	30.007
		1	.(b)	.	.	.
	1	0	30.004	.000	30.003	30.005
		1	30.003	.000	30.002	30.004
1	0	0	30.004(a)	.001	30.002	30.006
		1	.(b)	.	.	.
	1	0	30.003	.000	30.002	30.004
		1	30.003	.000	30.002	30.004

a Based on modified population marginal mean.
b This level combination of factors is not observed, thus the corresponding population marginal mean is not estimable.

6. netflow.bro * ftp.bro

Dependent Variable: Observation

netflow.bro	ftp.bro	Mean	Std. Error	95% Confidence Interval	
				Lower Bound	Upper Bound
0	0	30.004(a)	.001	30.003	30.006
	1	.(b)	.	.	.
1	0	30.004	.000	30.003	30.005
	1	30.003	.000	30.002	30.003

a Based on modified population marginal mean.
b This level combination of factors is not observed, thus the corresponding population marginal mean is not estimable.

7. frag.bro * netflow.bro * ftp.bro

Dependent Variable: Observation

frag.bro	netflow.bro	ftp.bro	Mean	Std. Error	95% Confidence Interval	
					Lower Bound	Upper Bound
0	0	0	30.005(a)	.001	30.003	30.007
		1	.(b)	.	.	.
	1	0	30.004	.000	30.003	30.005
		1	30.003	.000	30.002	30.004
1	0	0	30.004(a)	.001	30.002	30.006
		1	.(b)	.	.	.
	1	0	30.004	.000	30.003	30.005
		1	30.003	.000	30.002	30.004

a Based on modified population marginal mean.
b This level combination of factors is not observed, thus the corresponding population marginal mean is not estimable.

Table D-3: Statistically Significant Treatment Factor (con.).

8. netflow.bro * http.bro * ftp.bro

Dependent Variable: Observation

netflow.bro	http.bro	ftp.bro	Mean	Std. Error	95% Confidence Interval	
					Lower Bound	Upper Bound
0	0	0	30.004(a)	.001	30.003	30.006
		1	.(b)	.	.	.
	1	0	.(b)	.	.	.
		1	.(b)	.	.	.
1	0	0	30.004	.000	30.003	30.005
		1	30.003	.000	30.002	30.004
	1	0	30.004	.000	30.003	30.005
		1	30.002	.000	30.001	30.003

a Based on modified population marginal mean.
b This level combination of factors is not observed, thus the corresponding population marginal mean is not estimable.

9. frag.bro * netflow.bro * http.bro * ftp.bro

Dependent Variable: Observation

frag.bro	netflow.bro	http.bro	ftp.bro	Mean	Std. Error	95% Confidence Interval	
						Lower Bound	Upper Bound
0	0	0	0	30.005(a)	.001	30.003	30.007
			1	.(b)	.	.	.
		1	0	.(b)	.	.	.
			1	.(b)	.	.	.
	1	0	0	30.004	.001	30.002	30.005
			1	30.003	.001	30.002	30.005
		1	0	30.004	.001	30.003	30.006
			1	30.002	.001	30.000	30.003
1	0	0	0	30.004(a)	.001	30.002	30.006
			1	.(b)	.	.	.
		1	0	.(b)	.	.	.
			1	.(b)	.	.	.
	1	0	0	30.003	.001	30.002	30.005
			1	30.003	.001	30.002	30.005
		1	0	30.005	.001	30.003	30.006
			1	30.002	.001	30.001	30.004

a Based on modified population marginal mean.
b This level combination of factors is not observed, thus the corresponding population marginal mean is not estimable.

Table D-3: Statistically Significant Treatment Factor (con.).

10. netflow.bro * scan.bro

Dependent Variable: Observation

netflow.bro	scan.bro	Mean	Std. Error	95% Confidence Interval	
				Lower Bound	Upper Bound
0	0	30.004(a)	.001	30.003	30.006
	1	.(b)	.	.	.
1	0	30.004	.000	30.003	30.005
	1	30.003	.000	30.002	30.004

a Based on modified population marginal mean.
b This level combination of factors is not observed, thus the corresponding population marginal mean is not estimable.

11. frag.bro * netflow.bro * scan.bro

Dependent Variable: Observation

frag.bro	netflow.bro	scan.bro	Mean	Std. Error	95% Confidence Interval	
					Lower Bound	Upper Bound
0	0	0	30.005(a)	.001	30.003	30.007
		1	.(b)	.	.	.
	1	0	30.004	.000	30.003	30.005
		1	30.003	.000	30.002	30.004
1	0	0	30.004(a)	.001	30.002	30.006
		1	.(b)	.	.	.
	1	0	30.004	.000	30.003	30.005
		1	30.003	.000	30.002	30.004

a Based on modified population marginal mean.
b This level combination of factors is not observed, thus the corresponding population marginal mean is not estimable.

12. netflow.bro * http.bro * scan.bro

Dependent Variable: Observation

netflow.bro	http.bro	scan.bro	Mean	Std. Error	95% Confidence Interval	
					Lower Bound	Upper Bound
0	0	0	30.004(a)	.001	30.003	30.006
		1	.(b)	.	.	.
	1	0	.(b)	.	.	.
		1	.(b)	.	.	.
1	0	0	30.004	.000	30.003	30.005
		1	30.003	.000	30.002	30.004
	1	0	30.004	.000	30.003	30.005
		1	30.003	.000	30.002	30.004

a Based on modified population marginal mean.
b This level combination of factors is not observed, thus the corresponding population marginal mean is not estimable.

Table D-3: Statistically Significant Treatment Factor (con.).

13. frag.bro * netflow.bro * http.bro * scan.bro

Dependent Variable: Observation

frag.bro	netflow.bro	http.bro	scan.bro	Mean	Std. Error	95% Confidence Interval	
						Lower Bound	Upper Bound
0	0	0	0	30.005(a)	.001	30.003	30.007
			1	.(b)	.	.	.
		1	0	.(b)	.	.	.
			1	.(b)	.	.	.
	1	0	0	30.005	.001	30.003	30.006
			1	30.002	.001	30.001	30.004
		1	0	30.003	.001	30.002	30.004
			1	30.003	.001	30.002	30.004
1	0	0	0	30.004(a)	.001	30.002	30.006
			1	.(b)	.	.	.
		1	0	.(b)	.	.	.
			1	.(b)	.	.	.
	1	0	0	30.003	.001	30.002	30.004
			1	30.004	.001	30.002	30.005
		1	0	30.005	.001	30.003	30.006
			1	30.002	.001	30.001	30.004

a Based on modified population marginal mean.
b This level combination of factors is not observed, thus the corresponding population marginal mean is not estimable.

14. netflow.bro * ftp.bro * scan.bro

Dependent Variable: Observation

netflow.bro	ftp.bro	scan.bro	Mean	Std. Error	95% Confidence Interval	
					Lower Bound	Upper Bound
0	0	0	30.004(a)	.001	30.003	30.006
		1	.(b)	.	.	.
	1	0	.(b)	.	.	.
		1	.(b)	.	.	.
1	0	0	30.005	.000	30.004	30.006
		1	30.003	.000	30.002	30.004
	1	0	30.003	.000	30.002	30.004
		1	30.002	.000	30.001	30.003

a Based on modified population marginal mean.
b This level combination of factors is not observed, thus the corresponding population marginal mean is not estimable.

Table D-3: Statistically Significant Treatment Factor (con.).

15. frag.bro * netflow.bro * ftp.bro * scan.bro

Dependent Variable: Observation

frag.bro	netflow.bro	ftp.bro	scan.bro	Mean	Std. Error	95% Confidence Interval	
						Lower Bound	Upper Bound
0	0	0	0	30.005(a)	.001	30.003	30.007
			1	.(b)	.	.	.
		1	0	.(b)	.	.	.
			1	.(b)	.	.	.
	1	0	0	30.005	.001	30.003	30.006
			1	30.003	.001	30.002	30.005
		1	0	30.003	.001	30.002	30.005
			1	30.002	.001	30.001	30.003
1	0	0	0	30.004(a)	.001	30.002	30.006
			1	.(b)	.	.	.
		1	0	.(b)	.	.	.
			1	.(b)	.	.	.
	1	0	0	30.005	.001	30.003	30.006
			1	30.003	.001	30.002	30.005
		1	0	30.003	.001	30.001	30.004
			1	30.003	.001	30.001	30.004

a Based on modified population marginal mean.
b This level combination of factors is not observed, thus the corresponding population marginal mean is not estimable.

16. netflow.bro * http.bro * ftp.bro * scan.bro

Dependent Variable: Observation

netflow.bro	http.bro	ftp.bro	scan.bro	Mean	Std. Error	95% Confidence Interval	
						Lower Bound	Upper Bound
0	0	0	0	30.004	.001	30.003	30.006
			1	.(a)	.	.	.
		1	0	.(a)	.	.	.
			1	.(a)	.	.	.
	1	0	0	.(a)	.	.	.
			1	.(a)	.	.	.
		1	0	.(a)	.	.	.
			1	.(a)	.	.	.
1	0	0	0	30.004	.001	30.002	30.005
			1	30.003	.001	30.002	30.005
		1	0	30.004	.001	30.002	30.005
			1	30.003	.001	30.001	30.004
	1	0	0	30.006	.001	30.004	30.007
			1	30.003	.001	30.002	30.005
		1	0	30.002	.001	30.001	30.003
			1	30.002	.001	30.001	30.003

a This level combination of factors is not observed, thus the corresponding population marginal mean is not estimable.

Table D-3: Statistically Significant Treatment Factor (con.).

17. frag.bro * netflow.bro * http.bro * ftp.bro * scan.bro

Dependent Variable: Observation

frag.bro	netflow.bro	http.bro	ftp.bro	scan.bro	Mean	Std. Error	95% Confidence Interval	
							Lower Bound	Upper Bound
0	0	0	0	0	30.005	.001	30.003	30.007
				1	.(a)	.	.	.
			1	0	.(a)	.	.	.
				1	.(a)	.	.	.
		1	0	0	.(a)	.	.	.
				1	.(a)	.	.	.
			1	0	.(a)	.	.	.
				1	.(a)	.	.	.
	1	0	0	0	30.005	.001	30.003	30.007
				1	30.003	.001	30.001	30.005
			1	0	30.005	.001	30.003	30.007
				1	30.002	.001	30.000	30.004
		1	0	0	30.004	.001	30.002	30.006
				1	30.004	.001	30.002	30.006
			1	0	30.002	.001	30.000	30.004
				1	30.002	.001	30.000	30.004
1	0	0	0	0	30.004	.001	30.002	30.006
				1	.(a)	.	.	.
			1	0	.(a)	.	.	.
				1	.(a)	.	.	.
		1	0	0	.(a)	.	.	.
				1	.(a)	.	.	.
			1	0	.(a)	.	.	.
				1	.(a)	.	.	.
	1	0	0	0	30.003	.001	30.001	30.005
				1	30.004	.001	30.002	30.006
			1	0	30.003	.001	30.001	30.005
				1	30.004	.001	30.002	30.006
		1	0	0	30.007	.001	30.005	30.009
				1	30.003	.001	30.001	30.005
			1	0	30.002	.001	30.000	30.004
				1	30.002	.001	30.000	30.004

a This level combination of factors is not observed, thus the corresponding population marginal mean is not estimable.