ANALYSIS AND EVALUATION OF NETWORK INTRUSION DETECTION METHODS TO UNCOVER DATA THEFT

A thesis submitted in partial fulfilment of
the requirements of Napier University
for the degree of Master of Science in Advanced Networking
in the Faculty of Engineering, Computing & Creative Industries

2009

By
Julien Corsini
School of Computing

Authorship Declaration

I, Julien Corsini, confirm that this dissertation and the work presented in it are my own achievement.

Where I have consulted the published work of others this is always clearly attributed;

Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;

I have acknowledged all main sources of help;

If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself;

I have read and understand the penalties associated with Academic Misconduct.

I also confirm that I have obtained informed consent from all people I have involved in the work in this dissertation following the School's ethical guidelines.

Signed:

Date: 09/01/2009

Matriculation no: 05007968

Data Protection Declaration

Acknowledgements

My first thanks goes to Lionel Saliou, my supervisor, for all of the help and assistance he has provided with this project. I will be forever grateful to him for pushing me to use LaTeX and BibTeX. This detail helped me tremendously with my thesis, especially in terms of time saving and formatting.

Additional thanks goes to Professor William J. Buchanan for acting as internal supervisor for this thesis, and to Jamie Graves for his invaluable help concerning setting up the experimental testbed.

I also have to thank Joel Sommers, Vinod Yegneswaran, and Paul Barford for providing me with the exploit generator they developed during their research, and for providing great support.

Last but not least, I would like to thank Becca Thornton for the proof reading of this thesis and for her moral support throughout the duration of this project.

Contents

List of Tables

List of Figures

Abstract

NOWADAYS, the majority of organisations use signature-based intrusion detection to detect intruders on their intranets. This trend is partly due to the fact that signature detection is a well-known technology, as opposed to anomaly detection which is actively being researched. Along with this, anomaly **I**ntrusion **D**etection **S**ystem are known to generate many alerts, the majority of which being false alarms. Hence, organisations need concrete comparisons between different tools in order to choose which is best suited for their needs. This thesis aims at comparing anomaly with signature detection methods in order to establish which is best suited to uncover threats, such as data theft. The main difference between anomaly and signature-based detection is that an anomaly **I**ntrusion **D**etection **S**ystem needs to be trained, hence another aim is to establish the influence of the training period length of an anomaly **I**ntrusion **D**etection **S**ystem on its detection rate. Thus, this thesis presents a **N**etwork-based **I**ntrusion **D**etection **S**ystem evaluation testbed setup, and it shows the setup for two of these using the signature detector Snort and the anomaly detector **S**tatistical **P**acket **A**nomaly **D**etection **E**ngine. The evaluation testbed is then used to recreate a data theft scenario that includes the following stages: reconnaissance; gaining unauthorised access; and data theft. Therefore, it offers the opportunity to compare both detection methods with regards to that threat.

This thesis thus acts as documentation for setting up a network **I**ntrusion **D**etection **S**ystem evaluation testbed, and it could also be considered as documentation for the anomaly detector **S**tatistical **P**acket **A**nomaly **D**etection **E**ngine. Indeed, there is no centralised documentation for **S**tatistical **P**acket **A**nomaly **D**etection **E**ngine, and no research paper could be identified that documents the configuration of an evaluation testbed for **I**ntrusion **D**etection **S**ystem.

Standards for evaluating **I**ntrusion **D**etection **S**ystem could not identified, and thus this required the creation of a bespoke evaluation testbed which,

in turn, limited the time dedicated to evaluating the threat scenario itself. Along with this, results show that configuration, testing and verification of the anomaly detection system is highly error-prone. Hence, without a "*plug-and-play*" evaluation environment for **I**ntrusion **D**etection **S**ystem it is challenging to address these configuration issues.

Chapter 1

Introduction

1.1 Background

Nowadays, the use of the Internet has increased considerably. Its use has spread to the core of most of the businesses. The connectivity it provides allows corporations to extend their activity and increase productivity. Since the Internet developed recently at such a fast rate, its availability increased greatly. Connections to the Internet are now available everywhere, and at relatively low prices. This means that virtually anybody can access it and access any network. This ease of accessibility introduced a new kind of criminality: cyber-crime [1]. This type of crime developed exponentially during the past decade, mainly due to the democratisation of the Internet [2].

Data is the most important asset in an organisation [3]. This highlights the crucial need for network security in order to keep data secure. Computer network security is often deployed in two ways. The first security application tries to establish a strong outside barrier in order to prevent unauthorised users gaining access to a network Ingham and Forrest [4]. Since internal users still need to access resources outside the local network, this barrier has to let some communications go through. Intruders usually take advantage of these characteristics to carry out exploits. In order to address this security issue, the second type of security applied is monitoring the network for traces of exploits [5]. Tools to achieve computer network security are numerous and often corporations do not know what to invest in. This thesis aims to address this issue by providing a direct comparison between a signature **I**ntrusion **D**etection **S**ystem (IDS) and an anomaly IDS in order for organisations to choose the proper technology to mitigate for data theft.

1.2 Aim and objectives

The overall aim of this project is to provide an analysis and evaluation of **N**etwork-based **I**ntrusion **D**etection **S**ystem (NIDS). In order to achieve this overall aim, there are intermediary objectives to achieve. These steps are as follow:

- Carry out a critical appraisal of network security.
- Provide a theoretical analysis of signature-based and anomaly-based IDS.
- Investigate the current IDS evaluation methodologies.
- Establish the availability of tools required to create an evaluation testbed.
- Setup an evaluation testbed and **D**evice **U**nder **T**est (DUT), with regards to the results of the previous investigation.
- Evaluate the detection systems selected with the testbed implemented for a data theft scenario.

1.3 Thesis structure

The remaining of this thesis is organised as follows.

- **Chapter 2 - Literature Review**: reviews the current literature research on the themes of security issues and common practice. Some of these issues include security management and IDS testing. It also includes a presentation of computer networks threats, the tools used by **I**nformation **T**echnologies (IT) employees to protect organisation networks and with an emphasis on the theoretical analysis of NIDS technologies.
- **Chapter 3 - Design and methodology**: provides a design for NIDS evaluation testbed and experiments. It presents two experiments which involve two technologies of anomaly-detection IDS and aim at comparing the manner in which they operate, as well as comparing their respective effectiveness.
- **Chapter 4 - Implementation**: implements the design from Chapter 3 with a collection of tools, reviewed in chapter 2.
- **Chapter 5 - Evaluation**: provides the results of the two experiments, as well as an analysis.

- **Chapter 6 - Conclusions**: concludes this project, presents the successful points and limitations of this project and includes as well suggestions for future work.

Chapter 2

Literature review

2.1 Introduction

This chapter presents a review of the literature research in computer network security. This chapter is structured around three main themes. The first theme is related to security management. Section 2.2 presents a definition of security management and the current security situation in organisations. It describes how IT staff perceive security and how it is managed. The second theme is the reason why network security is highly important: computer systems threats. Section 2.3 analyses the different threats that organisations might be the target of, and shows the risks they face. In order to keep their assets secure and mitigate eventual intrusions, organisations apply security with the help of specific tools.Section 2.4, Section 2.5 and Section 2.6 present the security tools commonly used in order to achieve computer network security. Finally, Section 2.7 investigates the current methods for IDS evaluation.

2.2 Security management

It is not possible to start discussing security devices and computer systems threats without having an overall view of what information systems security involves, as well as analysing what is at stake for enterprises. This section defines security management and outlines the computer network security threats to organisations.

2.2.1 Definition

Siponen and Oinas-Kukkonen [6] define security management as "a mean of maintaining secure information system in organisations, including information system planning and evaluation". In other words, security management should cover design, implementation and testing of processes and devices aiming at keeping secure a company's assets and keeping at a minimum security risks [7]. In the past, information security involved the security department and the network management side of an organisation, and the main problem was that both sides were not always clearly defined or present in an organisation. Even in the best case, both departments did not interact and were unaware of each other [8]. The security department was producing security technologies setups, such as software or hardware dedicated to one task in order to solve specific challenges, while the other department was trying to standardise management solutions. The leader groups in this domain were the **I**nternational **O**rganization for **S**tandardization (ISO)/**I**nternational **T**elegraph and **T**elephone **C**onsultative **C**ommittee (CCITT) (now **I**nternational **T**elecommunication **U**nion - **T**elecommunication (ITU-T)) and the **I**nternet **E**ngineering **T**ask **F**orce (IETF) [8]. Furthermore, security management and network have been sharing the same goals [8], such as availability for example, and only technical approaches were used to solve security problems. Baker and Wallace [9] showed that such an approach cannot work since information security is not limited to technical problems. Indeed, they show that the close relationship between technology and business functionality has always been the source of costly information security incidents. Security was considered as a risk management however Hale and Brusil [8] state that it is now recognised by most organisations as a competitive and economic advantage.

2.2.2 Evolution of security management

Nowadays, organisations realise that security information is an organisation-wide concern and a management and technology challenge [10]. The US **N**ational **I**nstitute of **S**tandards and **T**echnology (NIST) defines a classification of levels of control that security management should cover and these levels are [9]:

- Technical: this includes security devices used to protect information systems.
- Operational: this includes threats recovery plans.
- Management: this includes security policies, employees training.

The increased importance of security management arose from the central role of the Internet and the repercussions of security breaches [2]. The consequences of security breaches are numerous for organisations. The main causes of damage include: money loss; data loss and public image deterioration. Money loss can be direct, as for some banking societies [3] or indirect, in terms of disruption of service, or staff employed to mitigate attacks [11]. Hale and Brusil [8] shows that marketing image deterioration due to information security breaches can be more destructive than financial loss. Indeed, the time needed to rebuild reputation and regain consumers confidence can prove lengthy. Miller [12] shows the different prices offer for vulnerabilities discoveries, hence showing how important threats are for organisations. Data breaches is another main issue. It is increasingly becoming a target for intruders since it is valuable. People victim of data theft may also sue organisations because it is their responsibility to keep sensitive information secure [2]. For instance, up to 100 million records were reported compromised over the year 2006 [8]. This awareness is also due to the fact that companies realised the necessity of having a structured methodology for security, allowing a better use of computer forensics in case of security breach [13, 14].

Although IT professionals try to bring security more to the attention of management personnel, the situation in most organisations is that security standards are inadequate [15]. Possible explanations include: security is not a matter of one person in am organisation; this task is most often shared between different services within an organisation; and creating management conflicts due to different mindsets or priorities [16]. Indeed, a network professional will see security in term of system vulnerabilities whereas another business-minded employee who will see security in term of costs [2]. Along with this, a survey [17] shows that there is no real security administrator, per se, who are solely dedicated to security tasks. Instead, security is achieved by IT professionals in general. The survey also shows that these professionals do not spend as much time on security as they should.

Fung et al. [10] highlight that "the scale of the threats has escalated more rapidly than the commitment to combat it". Fung et al. [10] also show that the data collection of risk assessment proves time consuming, disruptive and expensive, hence IT professionals tend to work with over-simplified models. Organisations seem to be actively trying to control information security, but are following different paths to achieve so. This proves inconsistent and superficial, although some sectors of the industry manage better control quality, such as in the finance industry [9]. This could be explained by the important consequences of threats in terms of financial loss, corporation image and data confidentiality.

2.3 Computer systems threats

This section discusses computer systems threats and vulnerabilities, and presents a classification of threats, followed by different examples of possible applications of threats. It also highlights the consequences of successful computer systems intrusions.

2.3.1 Information systems threats

It is essential to provide a clear definition of computer systems threats. For example, Newman [18] defines a threat as:

> *... any potential occurrence, either accidental or malicious, that can have an undesirable effect on the assets and resources of the organisation.*

Threats can have a harmful effect by trying to break the three main goals of computer security: confidentiality, integrity and availability [19]. Thus, computer systems threats include:

- gaining access to private information (breach of confidentiality).
- tampering with or accessing private information (breach of integrity).
- disrupting access to information or services (breach of availability).

Intrusions can also be sub-classified into two categories: active and passive intrusions. Active intrusions involve direct action on data, resources or hardware, whereas passive intrusions do not interfere directly in computer systems, such as eavesdropping for example [18]. As well as passive or active, attacks can be classified as insider attack or outsider attack, providing the source of the attack being from inside the targeted network or outside from it. Computer intrusions generally follow the five following stages [3]:

- Reconnaissance: Collection of information (structure of network, equipment, etc.) about the targeted system.
- Scanning: Scan for vulnerabilities to exploit.
- Gaining access: Take control of a user account, access to equipment, etc.
- Maintaining access: Escalate privileges.
- Covering tracks: Hide traces of intrusion.

The NIST [1], Buzzard [20] and Newman [18] provide a list of threats categories which sums up the different types of existing threats:

- Software flaws and configuration errors: the main source of vulnerabilities within computer systems. The fact that software programmers give priority to functionality rather than security [18] leads to common program flaws, such as buffer overflows, which are the most common source of exploits.
- Brute force attack: aims at gaining unauthorised access [18]. This type of attack tries all the possible combinations of a password for a given username in order to gain access to the corresponding account.
- File alteration: breaches the integrity characteristic of computer security. It involves changing data in a data collection, or changing data exchanged between two persons. For instance, an intruder could change health, police or banking records for his/her own benefit [21].
- Data theft: another main source of concern for security professionals. Many organisations rely heavily on computer network systems and protecting personal information has become critical [21]. Sabotage usually comes from an inside intruder. Often, the damage is caused by an employee and directed to hardware equipment.
- Sabotage: usually involves an employee of the organisation who tries to disrupt either the network system or components, such as network equipment, servers and so on. The reasons for sabotage are multiple, although greed and injustice are the more common.
- Social engineering: becoming a common type of threat. It usually involves impersonation of an authorised body in order to retrieve login

information from users (most often usernames and passwords). Another type of social engineering threat is phishing, which aims at getting critical data from users, such as credit card numbers, bank account details, social security numbers and so on [18, 22].
- Industrial espionage: can involve concurrent companies trying to steal industrial intelligence from each other. Such espionage can also take larger proportion and become governmental espionage for use in "cyber war" [1].
- Malicious software: includes worms, viruses, Trojans and logic bombs and can have diverse sorts of devastating effects [1]. This type of software often exploits **O**perating **S**ystem (OS) weaknesses or software flaws.

Since business relies on the Internet, a new form of threat has emerged: cyber terrorism. These intruders use secure communication means, which source is often hard to trace back to, in order to blackmail businesses for instance. The threat mainly consists of taking down a server, usually representing a company online, which can prove loss-making for an e-commerce organisation, for instance [21].

2.3.2 Threats applications

The most common applications of computer systems threats include:

- Viruses
- Worms
- Trojans
- **D**enial-**o**f-**S**ervice (DoS) attacks
- Reconnaissance / Scanning

Viruses are pieces of malicious software, often attached to legitimate documents which require human intervention to be activated [1]. These types of software or malware can have various actions, from simply showing advertisements windows to erasing or changing the content of files [18, 23, 24]. User tend to keep the spread of viruses going on by sharing files or sending emails.

Worms are considered as a sub-class of viruses [18] but differ from the them because they can replicate themselves and spread across a network without any human intervention [25]. They can have destructive effects on a host system, but can also affect networks by replicating and sending multiple copies of themselves across a network thus creating a DoS, such as with the "Witty" worm [26].

Trojans are also malicious pieces of software attached to legitimate files. Where they differ from viruses is that they are usually attached to applications which seem useful, but Trojans add hidden functions, such as remote shell access, thus compromising the security of the host. Such functions can enable remote access for an external intruder, hence creating a backdoor, or sending valuable data outside the organisation [18, 24].

DoS attacks aim at depriving legitimate users from access to resource [18]. Such resources could be a server, a printer or any other type of device providing a service. There are three basic types of DoS [27]: consumption of limited resource, destruction/modification of configuration files and physical modification of network infrastructure. Attention will be drawn to the first type of DoS since it is more closely related to network traffic. The second type is more related to host intrusion, and the last one meets the idea of sabotage that Section 2.3.1 covers.

Consumption of resources can target three main components: server memory; network bandwidth; and disk space [27]. Memory starvation is often achieved through SYN packets flood. A client sends numerous bogus SYN packets to a server in a short period of time, forcing the server to reserve memory resources for each session initiated, leading to total memory usage or system crash [27]. Bandwidth starvation often happens following to a **U**ser **D**atagram **P**rotocol (UDP) or **I**nternet **C**ontrol **M**essage **P**rotocol (ICMP) packets flood, such as Ping of Death or Smurf attacks for instance. Finally, disk space starvation can be achieved by forcing the system to produce a large amount of logs for example [27].

A variation of DoS attacks is **D**istributed **D**enial-**o**f-**S**ervice (DDoS) [28]. Such attacks involve several sources, rather than a single one, flooding the target, resulting in a larger number of packet sent across a network to a single target. DoS is the most common tool used by cyber terrorists in order to blackmail and take down businesses servers in exchange for a ransom [21].

Reconnaissance tools and scanners are used in the first step of an attack, in order to gain information about the target. Different types of information can be gathered which these tools, such as open port numbers, network addressing scheme and topology, web applications security [20].

2.3.3 Threats consequences

Organisations which are victims of computer systems security breaches can experience loss or degradation mainly in three specific domain: performance; public image and monetary. The first domain is computer systems related and technical, whereas the other two are not.

[29] shows that performance can be heavily affected by computer systems attacks. Through their experiment, Lan et al. [29] show that a DDoS attack can increase the mean latency for DNS lookup by 230% and the web latency by 30%. They also measure the spread of the Slapper worm in a simulation, and state that if all the infected host would launch a coordinated DDoS, a network could be taken offline in a matter of minutes. Worms, viruses and trojans can have different effects, depending on the designer's intentions. For instance, the Blaster worm would reboot infected machines, spread to other vulnerable systems and was programmed to launch a DDoS against Microsoft's Windows update website [11]. This type of attack could thus also induce hardware damage on the targeted devices. [1] shows that employee sabotage can also damage devices if access to hardware equipment is possible.

The other business domains which suffer from attacks are the corporation image and financial aspect of organisations under attack. As Hughes and Delone [24] state, "advanced, post-industrial societies and economies are critically dependent on linked computer information and communication systems", which makes it essential to have a secure and operating network in order to keep information and business running. Organisations which survive attacks usually lose credibility if strictly private information is released. For instance, Labib [3] shows this with a bank from which credit card number and accounts information were stolen. Such an intrusion can considerably diminish the security credibility of a bank.

Finally, the financial loss possible following a security breach is the domain which drives computer systems security. Table 2.1 shows how much attacks have spread between 1987 and 1998, and the increasing financial loss they incurred [21]. Table 2.1 thus correlates the findings of a recent survey [24] which reports "$130,104,542 in losses from 13 different types of computer security incidents, with the greatest amounts attributed to viruses ($42,787,767), unauthorised access ($31,233,100), and theft of proprietary information ($30,933,000)".

Table 2.1 – Reported incidents of computer security breaches

	1987	**1990**	**1994**	**1998**
Total abuse incidents reported	118	180	537	510
No. hacking incidents	35	26	15	56
Hacking as % of total	30	14	3	11
Resulting loss (£)	100	31,500	16,220	360,860

2.4 Firewalls

Section 2.3 outlined some of the threats computer networks and systems can be exposed to. It also showed how much was at stake for companies which are relying heavily on corporation networks. In order to maintain security in an organisation, IT professionals employ a diverse range of security tools. Among the most common are firewalls, IDSs and other host security oriented tools, such as antiviruses. Section 2.5 describes IDSs characteristics, Section 2.6 presents other common tools, and this section highlights firewalls as well as their respective strengths and weaknesses.

The most common computer networks security tools consist of firewalls. They can be found in most of all corporate networks and form the first barrier against intrusions [30]. A firewall is a dedicated system or software application that inspects traffic against a set of rules [4]. Without good configuration, firewalls are useless. Unfortunately, as well as being very popular they are also often mis-configured, allowing any traffic by default rather than denying all of it.

Ingham and Forrest [4] provide a definition of firewalls as a device or group of devices which separates corporation assets from potentially dangerous external environments, which could be the Internet, for instance. [4] also gives a few criteria that have to be fulfilled in order to class a device as a firewall:

- A firewall should be at the boundary of two networks
- All traffic entering or leaving the intranet should cross the firewall
- A firewall should have the capability of allowing and dropping traffic, in order to enforce a security policy
- A firewall should be resistant to direct attacks and have no direct user access

Firewalls also offer the possibility to create a **D**e**M**ilitarized **Z**one (DMZ): it is possible to place machines which offer services to the Internet in a DMZ. By doing so, the rest of the network remains protected in case of a breach in the DMZ [4]. Ingham and Forrest [4] also show that firewalls do not offer perfect security and list what firewalls do not protect against. For instance, firewalls cannot protect the network from insider attacks, from attacks targeting services provided by internal servers, hence the use of a DMZ, or from intrusions embedded in legitimate packets. It often comes down to a trade-off between ease of use and security: the tighter the security is, the less a user will be able to do [31].

Different types of firewall exists and can be classified as either packet filters, application layer gateway or stateful packet filters [30, 32].

2.4.1 Packet filters

Packet filters are the most popular and the simplest form of firewalls [30]. They analyse traffic flowing through and allow or drop traffic based on network and transport layers packet information such as source and destination **I**nternet **P**rotocol (IP) addresses and ports, direction, type of packets, and so on. This type of firewall is often implemented on edge routers with **A**ccess **C**ontrol **L**ist (ACL)s applied on ingoing and outgoing interfaces, allow flexibility and complexity in term of rules, as well as processing packets at a fast rate [32]. Unfortunately, it is not the most secure device because it does not keep tracks of connections history.

The Internet mainly consists of **T**ransmission **C**ontrol **P**rotocol (TCP) flows between clients and servers. These flows are made of requests and replies between computers [33]. For instance, if a FTP session is considered, a client will send a request to the destination port of a server. The server will then reply to the client. The first TCP/IP packet will have a destination port of 21 and the reply will have a source port of 21. This means that the packet filter should have two specific rules to enable FTP sessions to cross the firewall. The problem here is that an intruder with knowledge of the target system can easily forge fake FTP packets that the firewall will allow, thinking that they are legitimate FTP responses. This example shows the need of a method to check the legitimacy of incoming replies, with regards to request priorly made [34].

2.4.2 Application layer gateway

Application layer gateways act as a relay for connections between inside machines and extranets. It sits between users inside and servers outside. A user who tries to connect to an Internet server will actually connect to the gateway, which will in turn carry out the request on behalf of the user to the external server. The server replies to the gateway, which will then forward the reply back to the user. For external users, a network using an application layer gateway appears as a single machine [4]. Advantages of gateways are that they can filter based on packets content, include a user level authentication and hide the structure of the network from external potential intruders [30]. The main drawbacks of application layer gateways are that not all services have usable proxies already existing and that they are relatively slow to process packets [4].

2.4.3 Stateful packet filters

These systems are a refinement of the packet filtering technology. It acts as traditional packet filters but also monitors connections for increased security [30]. Stateful packet filters operate at the network and application layers of the **O**pen **S**ystems **I**nterconnection (OSI) model. Stateful firewalls monitoring initial connections (SYN flag on) and allows replies to cross the firewall until the connection is close (FIN flag on) [4]. A similar method can be used for ICMP and UDP packets, although these protocols are not connection oriented.

In most cases, different types of firewalls are combined together in order to increase the overall perimeter security. Application layer gateways or stateful filters are often used as primary firewall and traditional packet filters are added after the firewall to avoid inexistent security in case of main firewall failure.

2.5 Intrusion Detection Systems

2.5.1 Definition and purpose

Intrusion detection refers to the monitoring of events and the analysis for signs of intrusions. Intrusion detection systems (IDSs) are software applications which automate these monitoring and analysis processes [5].

IDSs are typically used to detect attacks or violations not detected by other security means, to detect reconnaissance attempts preceding attacks such as with probes and scans. They can also be used to control the quality of an existing security design and administration, or to help diagnosis, recovery and correction of breaches in case of a current attack occurred [5]. Figure 2.1 shows the different component of an IDS:

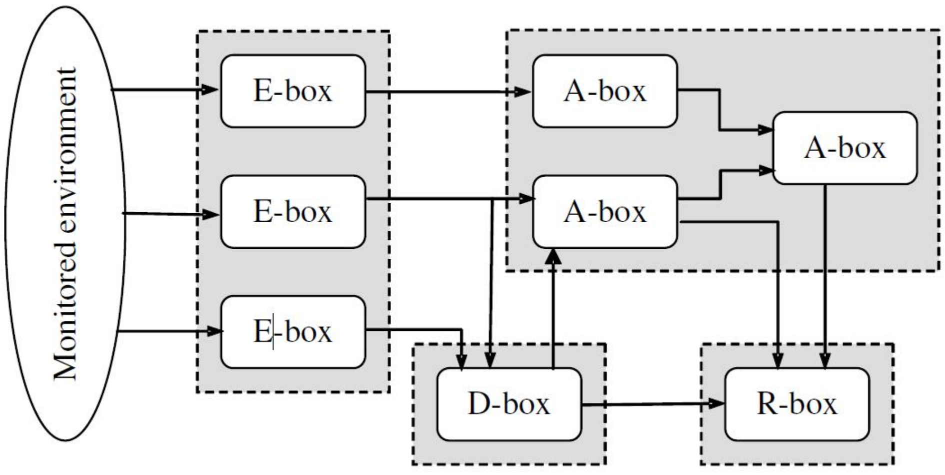


Figure 2.1 – IDS Architecture outlined in [35]

An IDS captures monitored data through its sensors ("E-box"). It then compares this data in the analysis module ("A-box") and stores it ("D-box"). It can finally react to a detected intrusion via a reaction component ("R-box") [35].

IDSs can also be sub-classified into many different categories, but one of the main difference include the following two categories: host-based IDSs and network-based IDSs.

2.5.2 Host-based IDS and Network-based IDS

Host-based **I**ntrusion **D**etection **S**ystem (HIDS) were the first type of intrusion detection systems to appear [36]. They are typically installed on the host they are monitoring and have access to the operating system information [37]. HIDSs prove useful because they can detect encrypted attacks, by checking traffic before being sent or just received, and also because they can detect attacks targeted to the specific system and undetectable in network traffic, such as Trojans. Another advantage of HIDSs is that they can access system information, generating more accurate alerts and more detailed logfiles. Disadvantages include that they can monitor the single host they are running on, and have to be specifically set up for each host. Scalability is the main issue for HIDSs They also use resources on the target host [5].

On the other hand, NIDS can monitor a segment of network to a large section of a network, depending on their placement [37]. They function in promiscuous mode in order to capture network packets, thus they have very little impact on the overall network performances. Unfortunately they have a few disadvantages, including the fact that they cannot process encrypted packets and require the use of SPAN ports if attached to a switch in order to monitor all traffic going through the switch [5]. Another main disadvantages of NIDSes is that they can have difficulties processing large amount of network packets if they are set up to monitor a large and/or busy section of the network [38]. Figure 2.2 highlights the main differences between HIDSs and NIDSs:

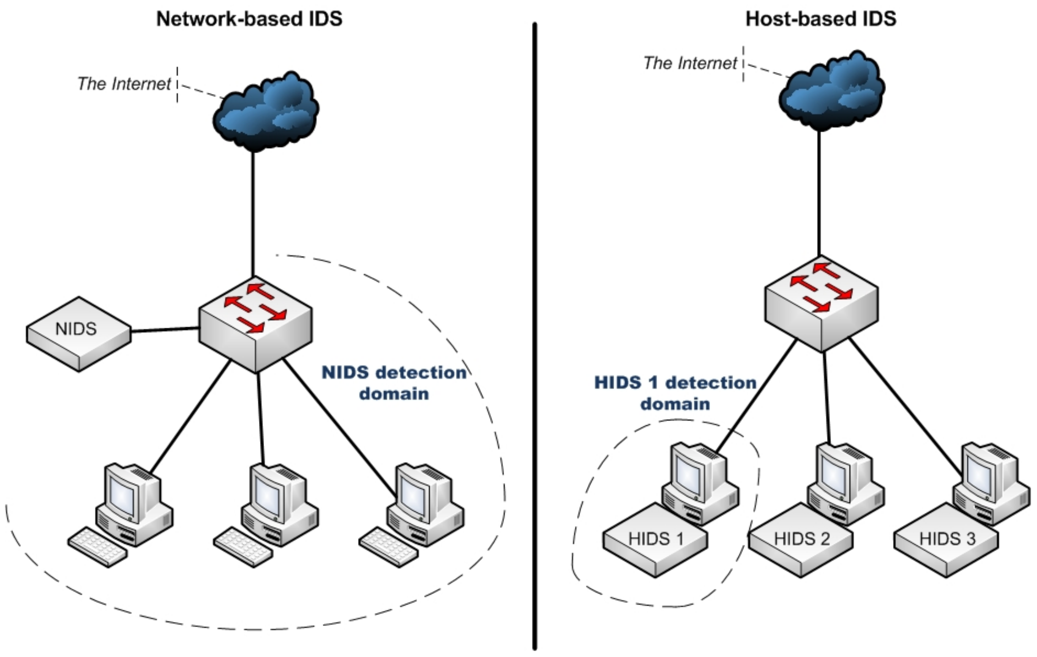


Figure 2.2 – NIDS/HIDS Differences

Debar et al. [37] also show how it is possible to measure the efficiency of an IDS. IDSs are often evaluated in terms of accuracy, performance and completeness. Accuracy is the ability of the IDS to flag as intrusive only packets that are part of an attack. The performance of an IDS is the rate at which events are processed, thus a good performance measure makes real time detection possible. Finally, the completeness of a system is the ability of detecting all the attacks that occurred in a given time. This measure is often the hardest to establish in a live environment because it is impossible to know exactly how many attacks were carried out and at which time. Arguably, if such was possible, then there would be no need for IDSs.

There are some key concepts related to IDSs: false-positives, false-negatives and true positives. Table 2.2 provides definitions for these terms [39].

Table 2.2 – Alarm types

Alarm Type	**Definition**
True-positive	IDS rightfully flags an attack as such
False-positive	IDS triggers an alarm although no attack is actually happening
False-negative	Real attack that the IDS does not flag as intrusion
True-negative	IDS does not flag legitimate events as attacks (most common situation)

Measurement of IDSs efficiency will be explored in more details in Section 2.7.

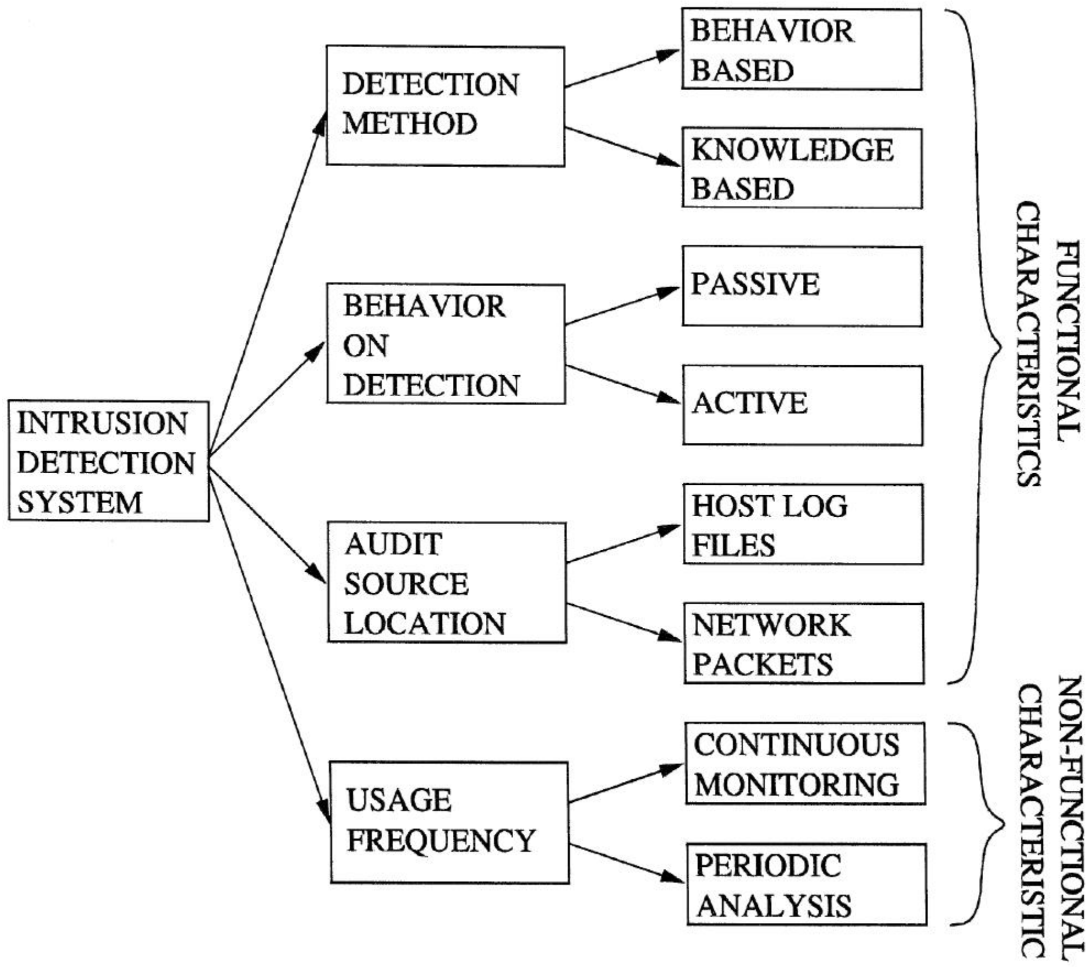


Figure 2.3 – IDS Characteristics [37]

The **audit source location** highlights the difference between HIDS ("host log files") and NIDS ("network packets") shown in Figure 2.3. The **usage frequency** is not an important characteristic because modern IDSs run in continuous monitoring [35]. The **behavior on detection** defines the system as an IDS ("passive") or an **I**ntrusion **P**revention **S**ystem (IPS), which reaction module includes active effect on the network or devices rather than simple logging (Section 2.6). The **detection method** is the second main characteristic that influences heavily on the way an IDS operates. A behaviour-based system, also called anomaly detection system, will analyse packets in a very different manner than a knowledge-based system (also called misuse detection). Section 2.5.3 details signature-based IDSs and Section 2.5.4 details anomaly-based

IDSs.

2.5.3 Misuse detection

Misuse detection, also called knowledge-based detection, is the most popular commercial type of IDSs [39]. Misuse detection systems use knowledge of known attacks, exploits and vulnerabilities and look for matching attacks patterns in network traffic or system events [5, 37, 36, 39]. Such knowledge is also referred as signatures. The accuracy of such systems is considered to be very good because they tend to have a low rate of false-positive alarms (see Table 2.2) since they flag only events that are real attacks. This type of systems can detect known attacks reliably [40].

As well as having a low false-positive rate, these systems produce detailed data about the attacks. Since the signature is known and detected, the attack is clearly recognisable, making the network administrator's work easier. In order to keep a good completeness standard, the signatures database has to be maintained up to date very frequently [37]. The main drawbacks of misuse detection are that signatures can be easily escaped with morphs of known attacks [41] and that these systems can only detect attacks related to their knowledge database [5]. In their taxonomy, Debar et al. [37] show that different methodologies can be used to achieve the same misuse detection goal. Among these methods are expert systems, signature analysis, petri nets or state-transition analysis. The most commonly applied to commercial IDSs is the signature analysis method, which reduces patterns of attacks to the lowest level of semantics. Examples of well-known misuse detection IDSs include Snort [42] (open source tool) and Bro (commercial tool) [25].

2.5.4 Anomaly detection

Anomaly detection methodology applied to computer systems was born in 1986 when Denning proposed the idea that it could be possible to identify abnormal unusual behaviour (anomalies) by comparing current behaviour to a known normal state [43]. This statement was based on the assumption that attacks are clearly different from normal traffic. This "normal traffic" states are recorded in profiles. These profiles can either be generated via offline learning or the system can learn by analysis traffic in an online way

[44]. Anomaly detection systems prove useful at detecting insiders attacks, as well as previously unknown attacks, known as "zero day" [40]. Such include intrusions between the time a vulnerability is made public and a patch is being released to fix it.

Anomaly-based IDSs are useful when it comes to detecting new threats, or different versions of known threats [44]. Where signature-based IDSs prove very useful for detecting known attacks, it has been proved that evading such security systems can be accomplished relatively easily [45]. Unfortunately, these advantages do not come without intrinsic drawbacks: the system must go through a training phase before any intrusion detection in order to build profiles for normal traffic. Another issue with anomaly detection IDSs is that they usually produce a large number of false-positive alarms, overwhelming the system administrator with hundreds of alerts to process [40]. Anomaly detection IDSs rely on several methodologies [46, 47, 48, 49, 50, 51, 52, 53]. Figure 2.4 shows the main methodologies anomaly detection IDSs use:

Statistical based anomaly detection is one of the "simplest" and oldest method, modelling statistics from different parameters [35]. For example, **S**tatistical **P**acket **A**nomaly **D**etection **E**ngine (SPADE) [42] uses the time series model form of statistical approach, using timers, counters and order of arrival of events. The other anomaly detection approaches involve different more or less complicated methodologies [54].

As Gates and Taylor [23] state, modern anomaly detectors are often based on Denning's assumptions [43] which were valid at the time for HIDSs, but not anymore in the context of current networks and NIDSes. Such assumptions are for example that attacks are anomalous, that attacks are rare, that attack-free training data is available or that the false alarm rate should be under 1% to be acceptable. The literature analysis in [23] shows that all of the above assumptions can easily be challenged. Nowadays, attacks are more and more common with the increase use of the Internet, and intruders can manage to craft intrusive traffic like normal traffic. The attack-free training data remains one of the main issues with anomaly detection. Training a detector in a "live" environment might include attacks as normal behavior. However this issue is currently being actively researched. [55] presents a profile data sanitisation through "simple weighted voting schemes" which improves the

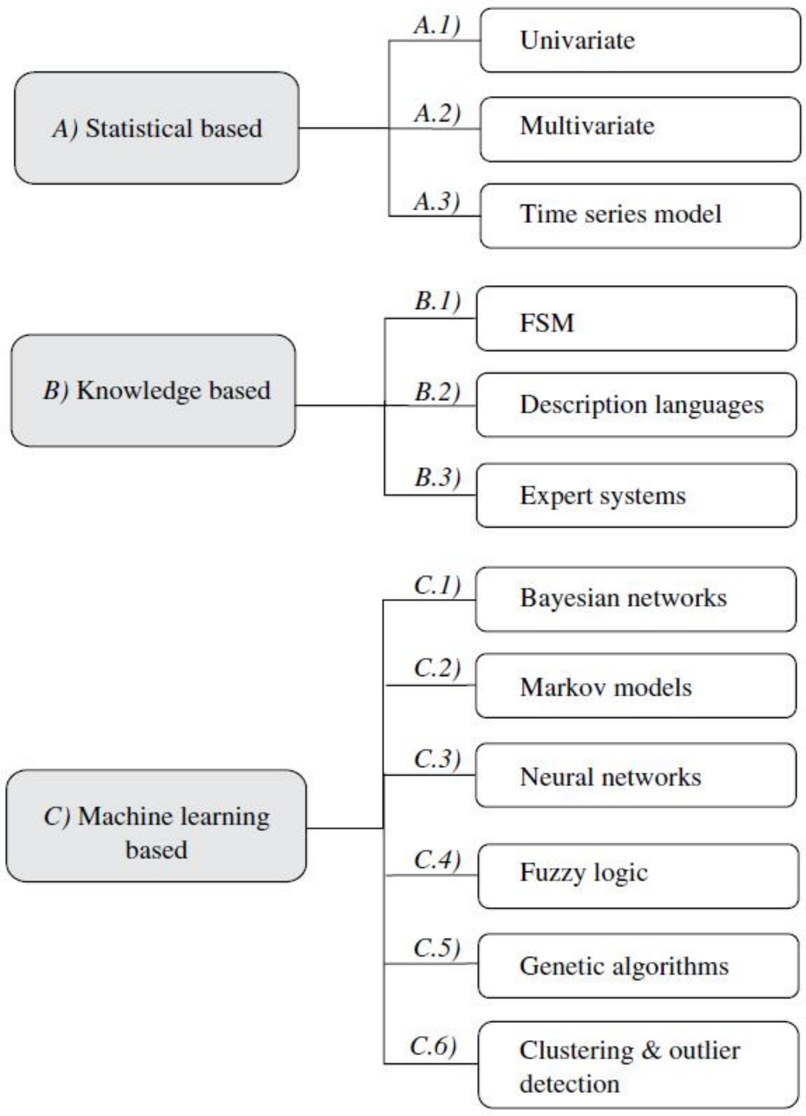


Figure 2.4 – Anomaly detection technologies [35]

anomaly detector detection rate by far (five times more than without sanitization).

2.5.5 Hybrid systems

[40] shows that the detection capabilities of IDSs can be improved by taking a hybrid approach, taking the best of both signature and anomaly detection. **E**vent **M**onitoring **E**nabling **R**esponses to **A**nomalous **L**ive **D**isturbances (EMERALD) [56] use this approach, by the means of statistical analysis engines and expert systems.

2.6 Other security tools

Sections 2.4 and 2.5 showed the main tools used by IT professionals to achieve network security. This section highlights other tools more host-oriented or used to enhance the capabilities of IDSs.

2.6.1 IDS-related tools

One of the main issues with anomaly detection IDSs is the large amount of alarms reported, mainly false-positives. Such problem can affect the judgement of the system administrator who has to process all these alarms and might the attacks among the load of logs. In [57], Julisch presents a method aiming at grouping alarms generated by IDSs into clusters, which have the same root cause or source attack. Solving the problems generating these alarms, which are typically false-negatives, helps the network administrator do his job more efficiently. The benefits of this method are that the alarm bulk is reduced by 90%, leaving the human analyse fewer alarms. Another work by Colombe and Stephens [58] presents a "visualisation technique to effectively filter out false positives". This method provides the system administrator with a visual representation of alarms in terms of vertical bars, with different sizes and thicknesses depending on their frequencies: "the idea of the display is to tap into the user's visuospatial pattern recognition skills". This method is presented for host-based detectors, but could probably be extended to network IDSs. The input could easily be alarm logs of network packets flagged as intrusive, with the same system of vertical bars related to the importance of the intrusion.

2.6.2 Antiviruses

Antiviruses are very common in any organisation. Installed on each machine of a network, they provide local defence against a wide range of malware (worms, Trojans, viruses, root-kits, etc.). To do so, they operate in a similar way as signature based IDSs do: they scan the host system for matching patterns of threats with a database of threats signatures. Another type of common host-based security tool is host-based firewalls. Such firewalls place the trust boundary on the machine network adaptor and act as a traditional packet filter, analysing packets based on IP header fields. Finally, the last security tool could be seen as more abstract than the previous ones: patches for system vulnerabilities can improve greatly the effectiveness of other security measures [59].

2.7 Testing of Intrusion detection systems

While the Section 2.4, Section 2.5 and Section 2.6 presented the different types of security tools and their characteristics, this section focuses on IDSs evaluation as they constitute the main topic of this dissertation. Arguably, the main challenge in IDSs deployment is assessing and comparing performances of their systems with other IDSs [35]. These evaluations are needed and driven by the fact that security systems have to prove what they are capable of detecting, and how well they operate compared to the each other. Also, Woloch [16] states that testing of intrusion detection systems is not as advanced as one would hope. This section thus presents the different current methods of intrusion detection evaluation and testing and their characteristics.

2.7.1 Evaluation metrics for IDSs

[3, 60, 54] mention detection rate and false alarm rate as the best suited metrics. The detection rate is equivalent to the the number of intrusions detected divided by total intrusions injected in the traffic. The false alarm rate is equivalent to the false-positive rate of the IDS (as seen in section 2.4, a false-positive occurs when the IDS flags legitimate traffic as intrusive or abnormal).

Sommers et al. [38] use efficiency, effectiveness, packet loss and **C**entral **P**rocessing **U**nit (CPU) utilisation as metrics. The first two metrics are equivalent to the two rates presented above, whereas CPU utilisation and packet loss are new measures, useful to determine how a system copes under traffic load. The work done by Graves et al. [7] emphasises on the necessity of this latter packet loss measure. The efficiency of a system is in fact the false-positive alarms occurrence ($Efficiency = \frac{True_{positives}}{All_{alarms}}$). The closer to 1 it is, the better the system can flag real attacks only. The effectiveness produces the false-negative alarm rate of the IDS ($Effectiveness = \frac{True_{positives}}{All_{positives}}$). This metric shows the events missed by the IDS. It has to be noted that these metrics apply to any type of IDS.

2.7.2 Offline evaluation

Offline evaluation consists of recreating datasets of network traffic including attacks without recreating the whole network topology. The use of tcpdumps and replay tools allow such type of evaluation [61]. The most commonly used datasets were created by **D**efense **A**dvanced **R**esearch **P**rojects **A**gency (DARPA) / **M**assachusetts **I**nstitute of **T**echnology (MIT) Lincoln Labs in 1998 and 1999, called 1998 DARPA set and 1999 DARPA set, and also sometimes called **I**ntrusion **D**etection **Eval**uation (IDEVAL) datasets [61, 62, 63, 64]. The DARPA sets are simulations of network traffic based on observation of real network traffic including common attacks, which aim at providing blind evaluation material for researchers [62]. These datasets were captured at the edge of a network, at the border router. Figure 2.5 presents the structure and services characteristics used in the DARPA datasets network [61]:

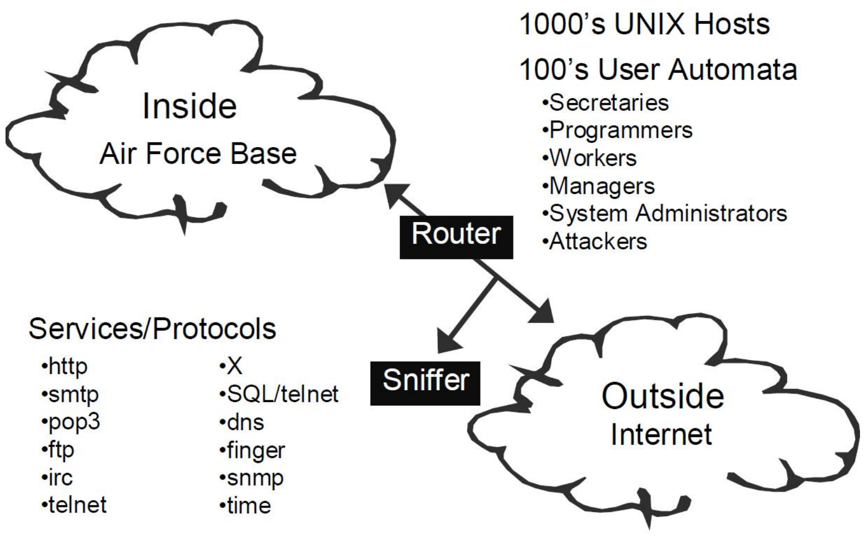


Figure 2.5 – DARPA experimental setup

The 1998 DARPA set includes 7 weeks of training data with labelled test data and 2 weeks of unlabelled test data [61]. During the first test competition, 8 IDSs were tested. The data set includes also over 300 instances of 38 attacks. The 1999 DARPA set presents over 5 million connections over 5 weeks: 2 were attack-free and 3 weeks included attacks. Another data set was created in 1999, based on the 1998 DARPA set: the 1999 **K**nowledge **D**iscovery and **D**ata mining (KDD) Cup, created for a machine learning evaluation competition.

Table 2.3 [61] lists by categories the different types of attacks present in the DARPA sets.

Table 2.3 – DARPA attack types

	Solaris	SunOS	Linux	Cisco Router
Denial of Service	**apache2**	**apache2**	**apache2**	
	back	back	back	
	mailbomb	land	**mailbomb**	
	neptune	**mailbomb**	neptune	
	ping of death	neptune	ping of death	
	process table	ping of death	**process table**	
	smurf	**process table**	smurf	
	syslogd	smurf	teardrop	
	udp-storm	**udp-storm**	**udp-storm**	
Remote to Local	dictionary	dictionary	dictionary	**snmp-get**
	ftp-write	ftp-write	ftp-write	
	guest	guest	guest	
	http-tunnel	phf	imap	
	phf	**xlock**	**named**	
	xlock	**xsnoop**	phf	
	xsnoop		**sendmail**	
			xlock	
			xsnoop	
User to Root	**at**	loadmodule	perl	
	eject		**xterm**	
	ffbconfig			
	fdformat			
	ps			
Surveillance/Probing	ip sweep	ip sweep	ip sweep	ip sweep
	mscan	**mscan**	**mscan**	**mscan**
	nmap	nmap	nmap	nmap
	saint	**saint**	**saint**	**saint**
	satan	satan	satan	satan

The main advantages of the DARPA sets are that they allow fast identical trial runs for IDSs evaluation. The fact that the sets are free to use allows many researchers to carry out the same experiments and thus compare IDSs between each other [60]. Unfortunately, many critical papers showed that these sets are flawed [3, 60, 63], the main shortcomings being:

- Simple, limited network topology

- Low background traffic and linear attacks distribution
- Limited number of victim target systems
- Simulated traffic include unlikely IP header attribute values

Mahoney and Chan [62] present a way to make the traffic more realistic by injecting real traffic into the sets and by removing simulation artifacts, but this method still does not offer a perfect testbed. This is due to the fact that this modified dataset still holds the main DARPA sets limitations [63].

2.7.3 Online evaluation

After seeing the shortcomings of current offline evaluation, there is a critical need for realistic traffic and attack generators, as well as data sets mixing both type of traffic in a realistic manner [40].

Current researchers focus their work on simulation testbeds and attacks generators [45, 41, 60, 65]. Lincoln Labs' work aiming at creating an online testbed resulted in the **L**incoln **A**daptable **R**eal-time **I**nformation **A**ssurance **T**estbed (LARIAT) tool [41]. LARIAT is capable of generating realistic background user traffic and real network attacks. It was created to overcome the issues inherent to the DARPA sets, in order to create a next generation of testbed. The main two goals of LARIAT are supporting real-time evaluation and creating easily deployable and configurable testbed [41]. It simulates an internal and external networks: it is thus possible to evaluate IDSs "plugged" in between both simulated networks. The main issue with LARIAT is that its use is limited to the US military and to "some academic organisations under special circumstances" [35].

Another two tools, which used together achieve similar goals as LARIAT are **M**alicious tr**A**ffic **C**omposition **E**nvironment (MACE) [38] and Harpoon [66], both developed by Sommers et al. Figure 2.6 [38] shows how these two tools can be used to achieve LARIAT's goals:

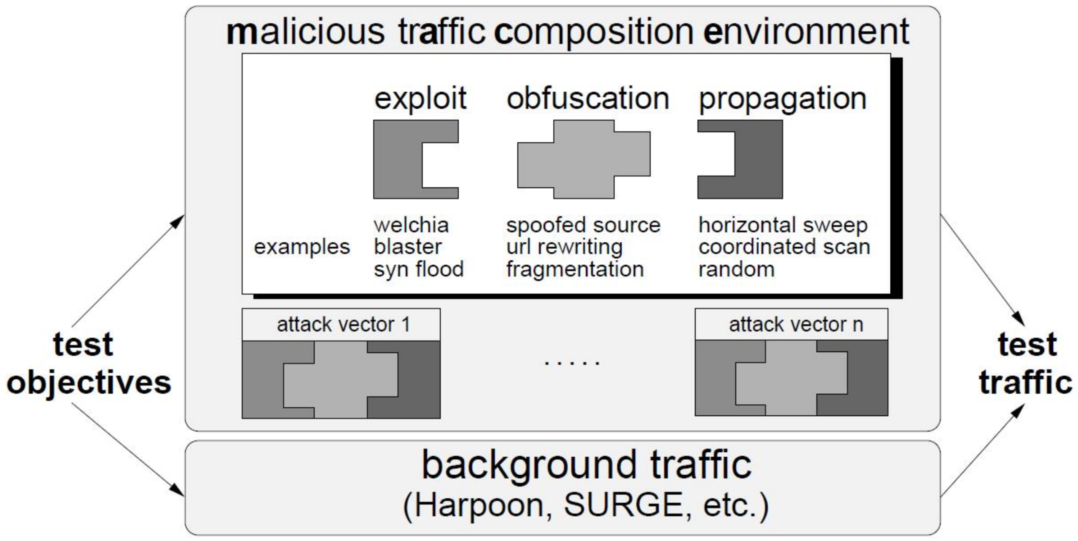


Figure 2.6 – Trident framework

Harpoon is a flow-level traffic generator, used to create benign realistic traffic based on real network packets traces [66]. It allows to modulate a mixture of benign and malicious traffic in a realistic way, as well as controlling the temporal arrival of each type. MACE is a performance benchmarking tool and malicious traffic generator. Table 2.4 [65] shows the different attacks that MACE is capable of generating:

Table 2.4 – Taxonomy of MACE exploits

Host Based					Network Based
Application Level			Transport Level		
Worms	Backdoors	DoS	Fragmentation	Other DoS	
Welchia	mydoom	winnuke	rose	synflood	smurf
Nimda	sdbot		teardrop1	pod	fraggle
CodeRed2			teardrop2	land	
Blaster			bonk	jolt	
Dameware			nestea		
Sasser			oshare		

Sommers et al. released an new tool called Trident [65, 67], which includes MACE and Harpoon as well as extra novel features such as DARPA attacks recreation for instance.

The second type of online evaluation tools include exploits generators for signature based IDSs evaluation, but which can be easily adapted to anomaly based IDSs when used together with a traffic generator, such as Harpoon. [45]

presents an evaluation of two open source signature based IDSs (Snort and ISS Secure) with a framework generating mutant exploits. This tool is "an automated mechanism to generate functional variations of exploits by applying mutant operators to exploit templates" [45]. The evaluation carried out in [45] uses ten common exploits, including DoS attacks, buffer overflows, targeted to different operating systems, including OpenBSD, Linux distributions and Windows OS, and different common services such as FTP, **H**yper**T**ext **T**ransfer **P**rotocol (HTTP) and **S**ecure **S**ockets **L**ayer (SSL)). It shows that 10 out of 10 basic exploits were detected, against 1 out of 10 for mutated exploits. This shows that it is relatively easy to evade signature detection by using mutant exploits.

Another similar framework is presented in [41]: **P**olymorphic **B**lending **A**ttacks (PBA). This paper provides a formal framework for creation of mutants, like the previous tool. It tests the efficiency of this tool against an anomaly based IDS. The outcome of this evaluation shows that it is also relatively easy to evade anomaly detection by using morphs of attacks.

Finally, [68] presents a framework for defining test cases scenarios. The authors prove that the existing classifications of possible attacks does not match all the needs of IDS evaluation and testing. Thus, they provide a framework covering all the characteristics of attacks in order to create a complete scenario evaluation. Gadelrab et al. [68] conclude that research should still be done in order to match real attacks to each categories.

Evaluation of IDSs is as much of a challenge as designing efficient algorithms for intrusion detection. As Garcia-Teodoro et al. [35] state that "the considerable research effort made to date in the field of NIDS assessment is proof of its importance. However, it remains an open issue and a significant challenge". The legal aspect of testing can slow down advances in evaluation. Furnell and Papadaki [22] show that the use of security tools in order to test network defences can come against the UK Computer Misuse Act 1990.

Another main issue with testing IDSs is that they are often only tested between different systems using the same detection methodology, in other words, signature detection with signature detection and anomaly detection with anomaly detection. Several researchers [69, 62, 54] evaluate offline anomaly detection systems. [38] evaluates online signature detection systems. Only the initial DARPA test mixes anomaly and signature based systems, although using an offline methodology evaluation. After analysing the different test scenarios realised by researches, it seem that nobody tried to experiment an online evaluation between a signature detection system and an anomaly detection system.

2.8 Conclusion

Section 2.2 showed the current situation related to security in organisations. It showed that security should be more structured, as well as being given more time and importance. Most corporations do not have dedicated employees for computer network security tasks and security is not always well coordinated. Section 2.3 analysed main threats that organisations face. It appears that the main computer systems threats include data theft, software flaws vulnerabilities and malicious software use. These threats can be classified as most serious because they involve the greatest financial impact. Applications used to carry out such intrusions are typically viruses, worms, Trojans and DoS tools. Social engineering and phishing is also becoming increasingly common. These threats can have a consequent impact on businesses in terms of financial loss, corporation image being worn or equipment damaged. It is thus vital to find a way of preventing such attacks from happening maintaining a secure network.

Section 2.4 highlighted that firewalls cannot provide adequate network security on their own. Ingham and Forrest [4] state that firewalls tend to offer a false sense of security in companies using them. Relying on firewalls only would be dangerous since they cannot block all attacks. In order to catch the threats that firewalls miss out, techniques such as intrusion detection are often implemented in enterprises. In a way, firewalls rely on signature analysis, or anomaly detection, each having advantages and inconveniences.

Most IDSs use signature based detection because it is more efficient for known threats and is also better understood than anomaly based detection (Section 2.5). However, anomaly based detection currently represents an important research effort [40] due to its capabilities in detecting unknown attacks for instance. [48] also shows how the use of a two-tiered adaptive IDS, combining in sequence a signature filter and an anomaly filter could improve greatly the detection rate, while keeping the false-positives rate low. Research has also been carried out in the field of distributed IDSs, trying to solve scalability and manageability of traditional IDSs [19].

Finally, section 2.7 presented the current trends and methods used in IDS evaluation. It showed that the current most popular methods, such as using the DARPA set is actually now obsolete and that there is a real need for updating IDS evaluation. A new data set including data from real network could solve the issues by allowing testing against current attacks trends, although this seems hard to achieve since it could include risks for the company providing data [40]. An alternative to offline evaluation is online evaluation, relying on traffic simulation rather than traffic datasets. The main issue with online evaluation is that realistic simulated traffic is a hard task to achieve. The literature reviewed in this chapter showed the need for testing anomaly and signature detectors against a scenario including a realistic simulated background traffic and exploits.

Chapter 3

Design and methodology

3.1 Introduction

This chapter details the basic design, as well as the main goals of the experiment carried out as part of this dissertation. As seen in Section 2.7, there are some limitations to existing literature concerning IDS evaluation. First of all, there is a need for testing directly two different types of NIDS, in other words anomaly and signature detection. There is also a need for online evaluation using realistic simulated traffic generation, as opposed to offline evaluation using network traffic traces.

This Chapter presents experiments which attempt to take into consideration both of these needs. Section 4.2 shows to what extent this experiment achieves these goals. It has to be noted that the different types of DUT in these experiments are network based.

The initial objective of this experiment was to set up a testbed for two different types of NIDS and generate simulated background traffic as well as range of exploits. Such an experiment proved too generic since the choice of exploits ready to use was relatively small compared to the amount of existing exploits. Instead, the experiment was split in two: a first experiment on the learning window variation of an anomaly IDS, and a second experiment testing two different types of IDS in a specific, well-defined scenario. [70] states that a valid computer security experiments should consist of only one varying component. The experiments carried out in this paper meet this criteria. The following sections define an overview of the testbeds used.

3.2 Network architecture

The basic network architecture is composed of a router and switch. Since the background traffic is split into two IP address ranges according to whether it is client or server (see section 4.2), two **V**irtual **L**ocal **A**rea **N**etwork (VLAN) are needed to mimic internal and external traffic. The router is used in this configuration in order to route traffic between both VLANs. Appendix E presents the router configuration used in Chapter 4.

Another essential piece of configuration is setting up the **S**witched **P**ort **AN**alyser (SPAN) port on the switch in order to send all traffic crossing the switch to the IDS station for analysis. Appendix F presents the switch configuration used in Chapter 4 (VLANs and SPAN port).

3.3 Training window experiment

This experiment aims at demonstrating any effects that a variation of training window length could have on an anomaly-based IDS. Figure 3.1 presents a high level diagram of the testbed setup for this experiment.

The DUT is represented by the station running the IDS. This station is linked to a switch and monitors all network traffic crossing this network device. The traffic generator is used to produce benign background traffic for anomaly system profile creation. Ideally, this station should produce this type of traffic with a traffic generation simulation tool such as Harpoon [66] for instance. The exploit generator is used after the profile generation phase has been completed. How well the IDS detects the exploit generated will help compare each different learning window and allow extracting conclusions from these observations. Finally, the router is used in this testbed in order to route or discard the generated traffic and make all the connections appear real to the IDS.

To sum up, the anomaly-based IDS will be subjected to different learning periods. For each period, the profile created will be stored for the next experimental phase, being the attack detection. After this profile generation phase, the IDS will be subjected to a mix of benign background traffic and malicious traffic. The amount of malicious traffic injected is known, thus the different types of alarms shown in Table 2.2 can be known, and measures like

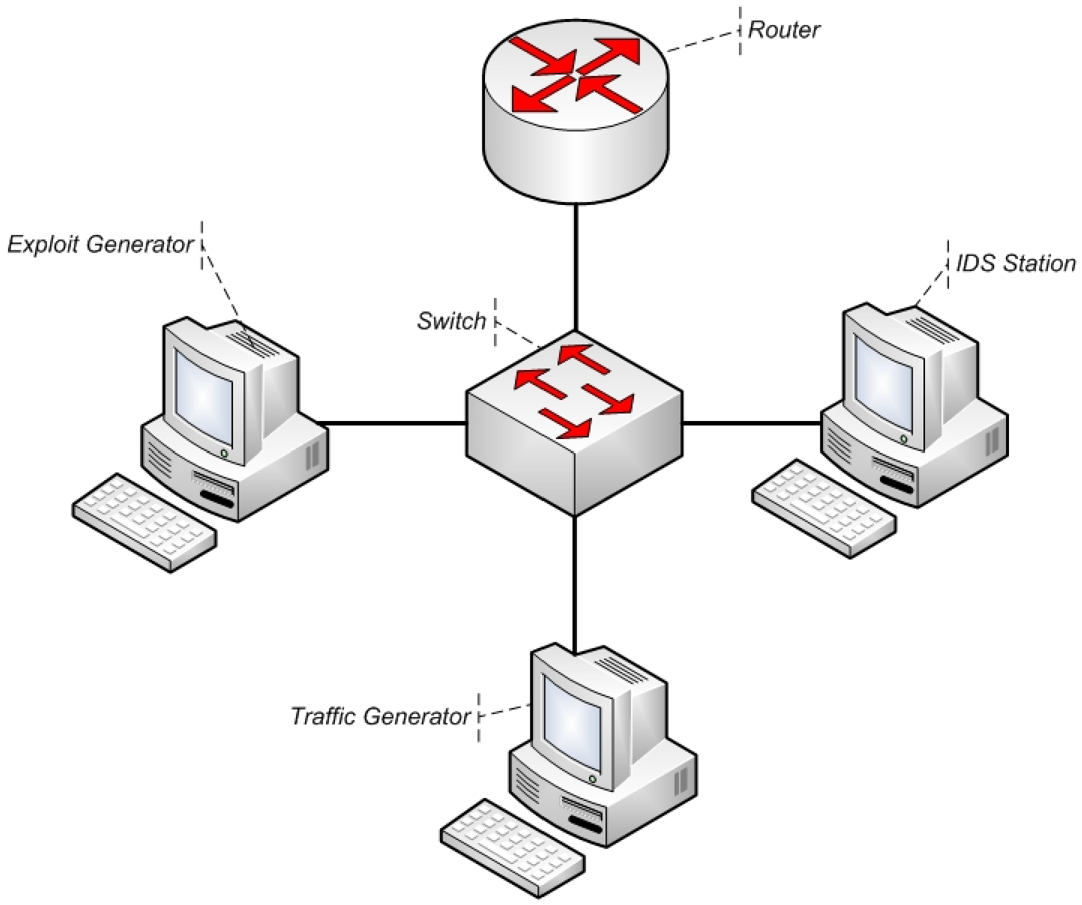


Figure 3.1 – Training window experiment design

effectiveness and efficiency shown in Section 2.7.1 can be evaluated.

3.4 Scenario

This experimental scenario is realised in order to focus this research on a specific type of threat rather than only available threats. For this scenario, the following background is to be considered.

A renowned bank branch computer networks system includes an FTP server hosting highly sensitive data, such as bank account details for example. Currently, the corporation uses the following tools as part of its security system: a firewall at the boundary with the untrusted network, antiviruses on local machines and a built-in IDS on the gateway router analysing traffic going in and out of the trusted network, such as on a Cisco router. Figure 3.2 shows a high level representation of such a computer system. The security at the boundary of the corporate network is optimum, but the security staff is worried about threats present on the inside of their network. Insiders threats are multiple (Section 2.3.1), although here the main concern is data theft from the FTP server, only protected by a username and password combination. In order to protect the branch from such a threat, the security staff would like to know which type of NIDS would be best suited in this case.

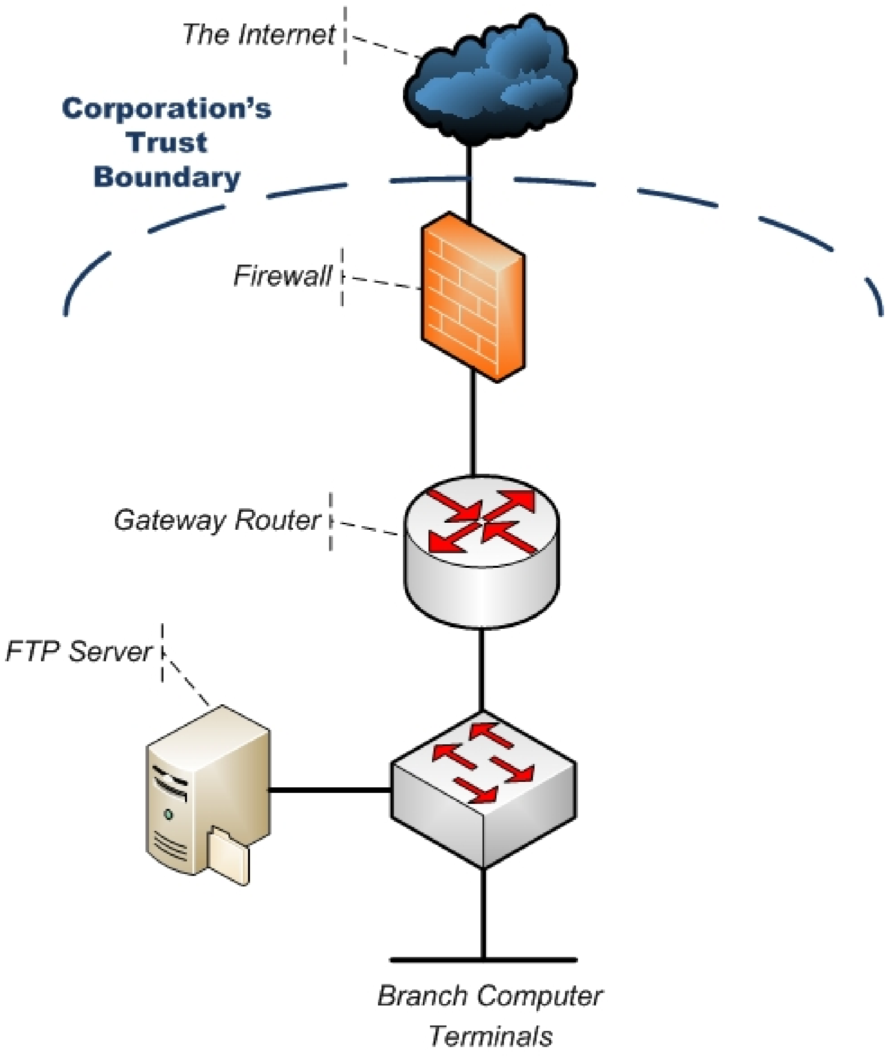


Figure 3.2 – Scenario bank computers system

There are some considerations to take into account with regards to this scenario. Sensitive data would probably not be stored on a simple FTP server in a real case environment, and the access to such a server would probably be more securely controlled (see Section 4.5.2). The simplistic approach used in this scenario is chosen due to time considerations and testing focus: a FTP is faster to breach than a more secure server, and this experiment is focused on NIDS rather than server security. This scenario can show which IDS is best for such an environment. Adding additional security measures could not do any harm but make the whole system more secure.

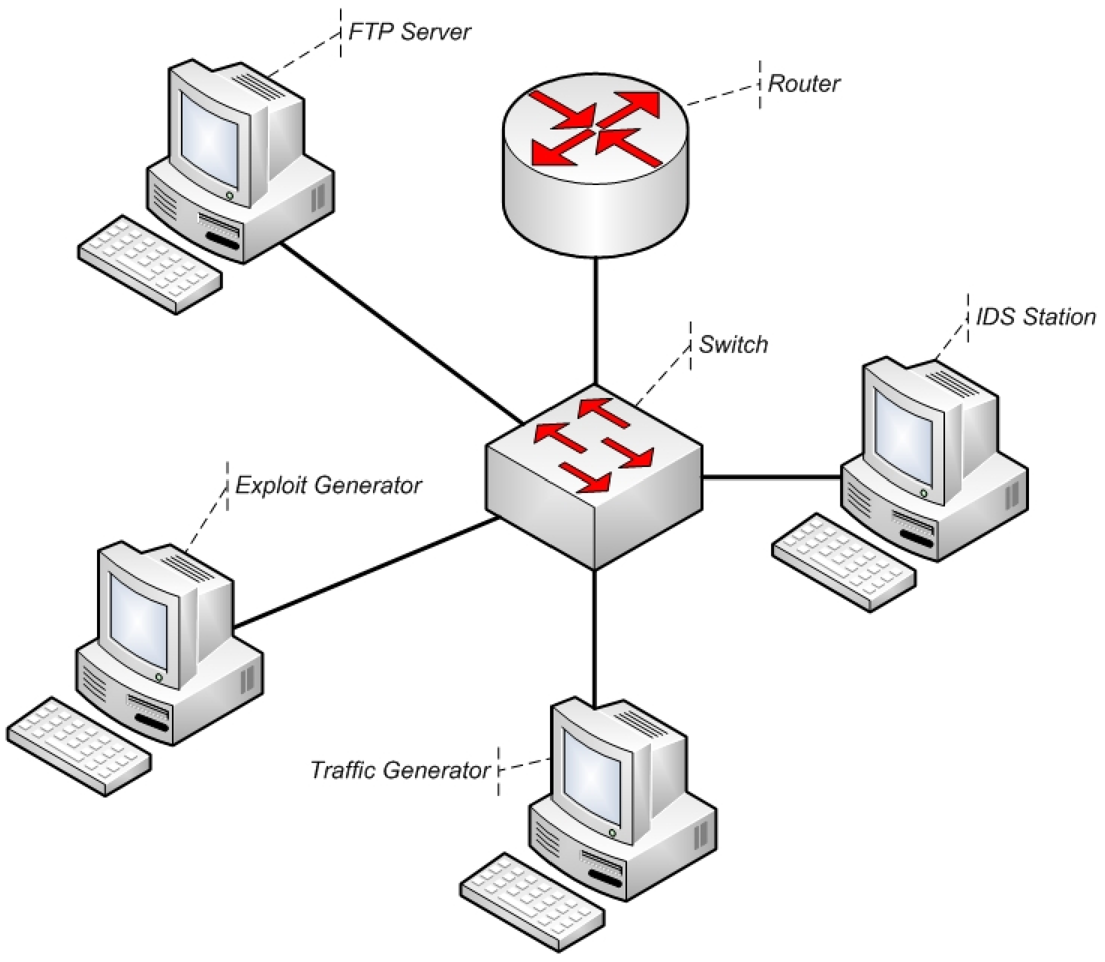


Figure 3.3 – Scenario experiment design

Figure 3.3 shows how this scenario is translated into a usable testbed for NIDS evaluation. The testbed is very similar to the one used in the learning window experiment, with the difference of an extra machine running an FTP server. The scenario threat is data theft. Data theft is usually composed of a collection of exploits following the steps highlighted in Section 2.3.1. In this case, the data theft consists of:

- Live IP addresses scan
- Portscan on live addresses

- Brute force attack on FTP username/password
- Data theft

The exploit generation station will carry out every step of a data theft threat.

3.5 Conclusion and hypothetical results

It is possible to draw hypothetical results from the normal behaviour of IDSs and exploits generated. With regards to the training window experiment, the anomaly-based IDS should flag all packets which are part of the attack generation if they are very different from background traffic, and the same goes for all different learning window length. Any new packet different from the traffic seen by the IDS should be flagged anomalous. This hypothesis depends on the method used by the anomaly-based IDS to create normal traffic profile, and as above-mentioned, is dependent on the traffic similarity.

With regards to the scenario experiment, the hypothetical results are clearer. The signature-based IDS should flag the steps for which it has signatures. Portscans, IP address scans and FTP attacks detection are commonly implemented in such IDSs. On the other hand, the signature-based IDS cannot detect the data theft since this is considered normal traffic from a signature point-of-view (Section 2.5.3). With regards to the anomaly detection, the anomaly-based IDS should flag any packet that is very different from the profile as anomalous. Thus, it should flag any new scan try, repetitive fast FTP connections (brute force attack) and unusual data transfer to odd stations as anomalous.

The experimental designs demonstrated in this chapter are in their simplest forms and allow a reliable performance analysis of different types of IDSes. The first experiment will explore the impact of different learning window lengths on anomaly detection, while the scenario experiment will determine which type of IDS is best suited to uncover data theft.

Chapter 4

Implementation

4.1 Introduction

This chapter implements the experimental designs presented in Chapter 3 and describes the tools used for each experiment, as well as the procedure to install and configure each of them. It is split into four main sections. Section 4.2 and Section 4.3 detail the background traffic generation and the IDSs chosen, common to both experiments. Section 4.4 describes the specificities of the training window experiment and Section 4.5 describes the extra tools used to implement the scenario experiment.

4.2 Background traffic generation

As Section 3.1 describes, the experiment should have a tool producing simulated realistic background traffic. This would be the ideal scenario. Only one traffic simulation tool proved available at the time the experiment took place: Harpoon (Section 2.7.3). Unfortunately, Harpoon proved to be not realistic enough. It always uses the same port number for its communication. Despite the issues related to its use shown in Section 2.7.2, there was no other option than to use the DARPA set in order to create benign background traffic.

Recreating the DARPA set for the purpose of IDS testing is not trivial. There could a possibility of feeding the traffic traces directly to the IDS but in the end, this was not tested. Instead, this experiment recreates the DARPA set as live traffic through a network composed of a couple of network devices. To achieve this, the traffic traces have to be prepared prior to being sent across the network.

The TCPReplay website provides an example of a similar scenario [71]. This is a good starting point, although this example becomes quite obscure and difficult to follow from time to time. Graves and Saliou [72] provides a more in-depth tutorial about recreating the DARPA set, available in appendix D.

Briefly, the PCAP files have to be prepared according to three steps: separation of inside and outside traffic, rewriting of IP and **M**edia **A**ccess **C**ontrol (MAC) addresses and finally, replay of traffic source file. The `PCAP` file used for the experiments is the `tcpdump` of the Monday Week 1 from the 1998 DARPA dataset [73].

The “tcpprep” command separates packets into two categories, client packets and server packets.

```
tcpprep -a bridge -o OUT.prep -i inside.tcpdump
```

After this is done, the use of the “tcprewrite” command is needed in order to rewrite the layer 2 and 3 information of every packet. This command requires the router interfaces MAC addresses and the subnet IP addresses in order to route packets properly.

```
tcprewrite --dmac=00:18:18:9d:3f:69,00:18:18:9d:3f:68 --smac
    =00:18:18:9d:3f:68,00:18:18:9d:3f:69 -e
    172.16.1.64/26:192.168.1.64/26 -C -c OUT.prep -i inside.tcpdump
    --o output.pcap
```

Finally, the “tcpreplay” command allows replaying the PCAP file with different parameters, such as an infinite loop, a speed of 11.6 Mbps and the network interfaces used to send the packets.

```
tcpreplay -i eth0 -I eth1 -M30 -l0 -c OUT.prep output.pcap
```

4.3 Intrusion Detection Systems

This section shows which IDSs where chosen to be tested. With regards to the anomaly-based IDS, the choice was very limited. Some of the IDSs mentioned in Section 2.5 were available to use, but not appropriate for “live” conditions. The work of Mahoney and Chan [62] like **P**acket **H**eader **A**nomaly **D**etector (PHAD) or **NET**work **A**nomaly **D**etector (NETAD), for example, uses `tcpdumps` of saved traffic as traffic input. It is not possible to have the

IDS analysing all packets with a **N**etwork **I**nterface **C**ard (NIC) in promiscuous mode. There was only one open-source anomaly detector satisfying the requirements of these experiments: SPADE, presented in Section 4.3.2. Appendix C.1 provides the hardware characteristics of this system.

4.3.1 Signature-detection

The signature-based IDS chosen for both experiments is Snort [74]. This IDS was chosen since it is free, extremely powerful and widely used by researchers [67, 45, 42]. Since signature detectors are only as good as the signatures they use, Snort uses the "Sourcefire VRT Certified Rules" version 2.4 for unregistered users [75]. This set of rules contains a large number of signatures used to detect diverse threats, such as DoS attacks, worms and viruses, web servers attacks, and so on.

On top of these rules, one specific rule was added to the "ftp.rules" file in order to make sure that the signature-based IDS flags the brute force attack on the FTP server in the scenario (Section 3.4). This rule is as follow:

```
alert tcp any any -> \$HOME_NET 21 (msg:\"FTP brute force failed
    login unicode attempt\"; flow:to_server,established; content:\"
    PASS\"; reference:url; threshold:type threshold, track by_src,
    count 15, seconds 1200; priority: 1; classtype:attempted-user;
    sid:2000001;)
```

This rule sets up a threshold of 15 occurrences in 1200 seconds for attempts to send passwords to an FTP server. If the number of tries is over this threshold, then Snort will report the next tries as intrusions. This rule functions in a similar way to a simple connection security setting of any FTP server.

4.3.2 Anomaly-detection

As mentioned in Section 4.3, open source anomaly detectors are not commonly found on the Internet. The only anomaly-based IDS found suiting the needs of this experiment is SPADE. SPADE is a preprocessor plugin for Snort. It achieves anomaly detection by assigning anomaly scores to every packet analysed. This anomaly score is based on the probability of the event, calculated following a combination of parametres such as source and destination ports and IP addresses [76].

The first issue with SPADE is finding the source files because this project has been discontinued since 2003 [42]. All the links to the installation files of SPADE link to the Silicon Defense website, which does not contain any information about SPADE anymore. These files are available through the tool **O**pen **S**ource **S**ecurity **I**nformation **M**anagement (OSSIM), in the "ossim/contrib/snort/" folder of this distribution.

The second issue comes with the installation and the configuration of SPADE: the official documentation is inexistent, and general literature on it is rare and sometimes even contradictory. For instance, Orebaugh et al. [42] states that to install SPADE with Snort, one should move the SPADE source files into the top directory of Snort and execute the usual Unix/Linux installation process (./configure, make, make install). Unfortunately, this did not seem to work. Farshchi [77] states that SPADE is integrated to all versions of Snort above version 1.7. His instructions were followed but SPADE source was found in the Snort install or on Snort website in the contribution section.

The method used in this experiment to install SPADE was found in a post on the Snort forum [78]. The first step is to copy the file with the .diff extension into the top directory of Snort. Then, run the following command and install Snort as usual:

```
patch -p1 < snort-spade-VERSION.diff
```

To run SPADE exclusively, run Snort in the normal way but specify the "spade.conf" file as parameter rather than the usual "snort.conf" file.

Running SPADE with its default "spade.conf" configuration file generates 235 alerts for 1 run of the DARPA dataset. 48 alerts are generated by the "type=dead-dest proto=tcp" preprocessor, and 187 alerts are generated by the "type=dead-dest proto=icmp icmptype=noterr" preprocessor. In order to start the experiments with a clean state, SPADE uses the following configuration file:

```
preprocessor spade: dest=alert logfile=/var/log/spade/spade.log
    statefile=/var/log/spade/spade.rcv 3 5000
preprocessor spade-homenet: 192.168.1.0/24
preprocessor spade-detect: type=closed-dport tcpflags=synonly wait=3
preprocessor spade-detect: type=closed-dport tcpflags=weird thresh
    =0.5
preprocessor spade-detect: type=closed-dport tcpflags=synack
```

```
    preprocessor spade-detect: type=closed-dport tcpflags=established
    preprocessor spade-detect: type=closed-dport tcpflags=teardown
    preprocessor spade-detect: type=dead-dest tcpflags=synonly wait=2
    preprocessor spade-detect: type=dead-dest tcpflags=weird wait=2
    preprocessor spade-detect: type=dead-dest tcpflags=synack wait=2
    preprocessor spade-detect: type=dead-dest tcpflags=setup wait=2
    preprocessor spade-detect: type=dead-dest tcpflags=established wait
        =5
    preprocessor spade-detect: type=dead-dest tcpflags=teardown wait=2
    preprocessor spade-detect: type=dead-dest proto=udp wait=2 Xsports
        =520 Xdports=520
    preprocessor spade-detect: type=odd-dport proto=tcp wait=2 Xsports
        =20
    preprocessor spade-detect: type=odd-dport from=nothome proto=tcp
    preprocessor spade-detect: type=odd-typecode
    preprocessor spade-detect: type=odd-typecode to=nothome
    preprocessor spade-detect: type=odd-port-dest from=nothome proto=tcp
         Xdports=80
```

This configuration is close to the default SPADE configuration file. Only the "type=dead-dest proto=icmp icmptype=noterr" preprocessor was removed and alerts restrictions for port 20 were added to the "type=dead-dest proto=tcp" preprocessor. The `homenet` IP address range was also added, as well as the location of the logfiles.

Details on the use of each preprocessor can be found in the "Usage.spade" file from SPADE source or in [42]. [77] also provides configuration advice, different from the previous two papers. For instance, Farshchi [77] advises to add a reporting threshold on the first preprocessor line to turn it on, as in "preprocessor spade: 4 dest=alert ...". The user should be warned that by doing so, the packet drop rate increases from 0% to 30% on average.

4.4 Training window experiment

As seen in Section 3.3, this experiment aims at testing the impact of different learning window lengths for anomaly-based IDSs. Appendix A presents a detailed diagram of this experiment.

4.4.1 Exploit generation

The exploit type used in this experiment is a basic SYN flood DoS attack between the exploit generator and the local router interface. This exploit is generated with the tool MACE (Figure 2.6). Each attack run lasts for 30 seconds at a speed rate of 0.5 Mbps. Each attack produces 3032 TCP SYN packets. Appendix C.2 provides the hardware characteristics of this system. The basic configuration file of MACE used (mace_config.xml) is as follow:

```
<mace_config>
    <python_declarations>
      srcap = AddressPool(AddressPool.Random,
          '192.168.1.3/32:1-65536')
      dstap = AddressPool(AddressPool.Random,
          '192.168.1.1/32:1-65536')
      broadsrc = AddressPool(AddressPool.Random,
          '10.52.255.255/32:1-65536')
      broadsrc.disableChecks()
    </python_declarations>
  <exploit_delivery interval_length="30" num_intervals="1" rate_unit
      ="mbps">
  <exploit source_pool="srcap" dest_pool="dstap" vector="DoS"
      threads="1" rate_profile="0.5" />
  </exploit_delivery>
</mace_config>
```

4.4.2 Experimental parametres

The only experimental variation in this experiment is the training window length. Before launching any attack, the anomaly-based IDS is trained for 2 minutes, 10 minutes and 30 minutes. Straight after the training period, the first attack is launched. There is then 3 minutes of only benign traffic before a second identical attack is launched. It has to be noted that background traffic is produced on a continuous basis throughout the experiment. This experiment last 4 minutes for each run (30 seconds attack + 3 minutes + 30 seconds attack). Figure 4.1 shows the chronology of these events.

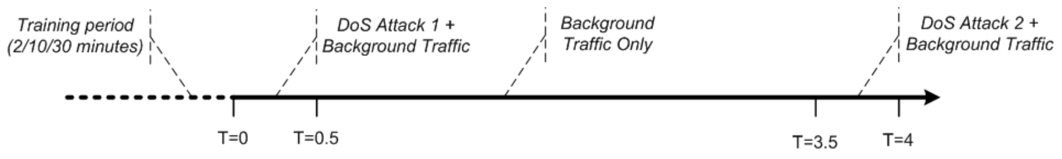


Figure 4.1 – Training Window - Experimental parametres

4.5 Scenario

As seen in section 3.4, this experiment aims at testing the detection difference between an anomaly-based IDS and a signature-based IDS. Appendix B presents a detailed diagram of this experiment.

4.5.1 Exploit generation

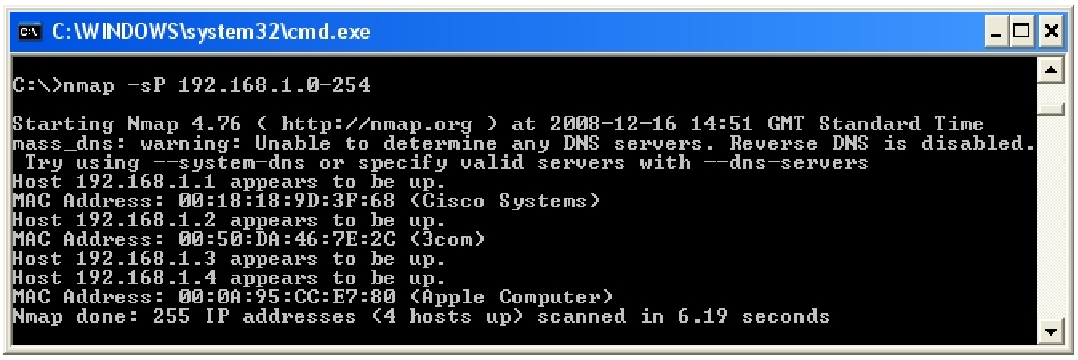
Appendix C.3 provides the hardware characteristics of this system. As shown in Section 3.4, the data theft scenario attack consists of the following exploits:

- Live IP addresses scan.
- Portscan on live addresses.
- Brute force attack on FTP username/password.
- Data theft.

The first two scans are run with the Nmap v4.76 software [79]. The command used to run the live IP addresses scan is:

```
nmap -sP 192.168.1.0-254
```

Figure 4.2 represents a screenshot of this reconnaissance step. It is possible to see that the IP address 192.168.1.1 is the Cisco router local interface, and that the IP address of the FTP server is 192.168.1.4 (Apple Computer), for example.



```
C:\>nmap -sP 192.168.1.0-254

Starting Nmap 4.76 ( http://nmap.org ) at 2008-12-16 14:51 GMT Standard Time
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
 Try using --system-dns or specify valid servers with --dns-servers
Host 192.168.1.1 appears to be up.
MAC Address: 00:18:18:9D:3F:68 (Cisco Systems)
Host 192.168.1.2 appears to be up.
MAC Address: 00:50:DA:46:7E:2C (3com)
Host 192.168.1.3 appears to be up.
Host 192.168.1.4 appears to be up.
MAC Address: 00:0A:95:CC:E7:80 (Apple Computer)
Nmap done: 255 IP addresses (4 hosts up) scanned in 6.19 seconds
```

Figure 4.2 – Reconnaissance with Nmap - Step 1

The second reconnaissance scan, the portscan of the server is executed with the following command:

```
nmap -sS -sV -P 0 -T5 -O 192.168.1.1
```

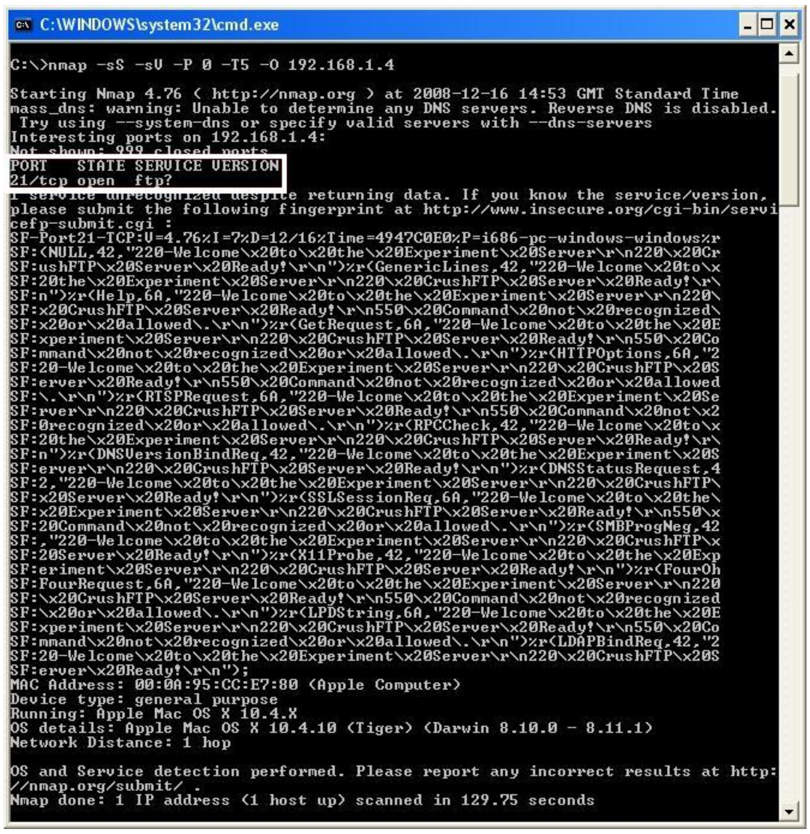
Figure 4.3 shows the output of this command. It is possible to notify the TCP port 21 being open on the server, as well as other details about the OS. The large paragraph in this screenshot is the fingerprint of the CrushFTP server. Nmap does not recognise this software as actual FTP, hence the question mark at "ftp?". It is still able to tell that it seems to be FTP running on the server.

The next step of this data theft attack is using brute force in order to gain unauthorised access to the server password. In order to achieve this task, Hydra v5.4 for Microsoft Windows is used. Hydra is a fast network login cracker which supports FTP [80].

The following command is used to launch Hydra:

```
hydra -l admin2 -P passlist.txt -t 5 192.168.1.1 ftp
```

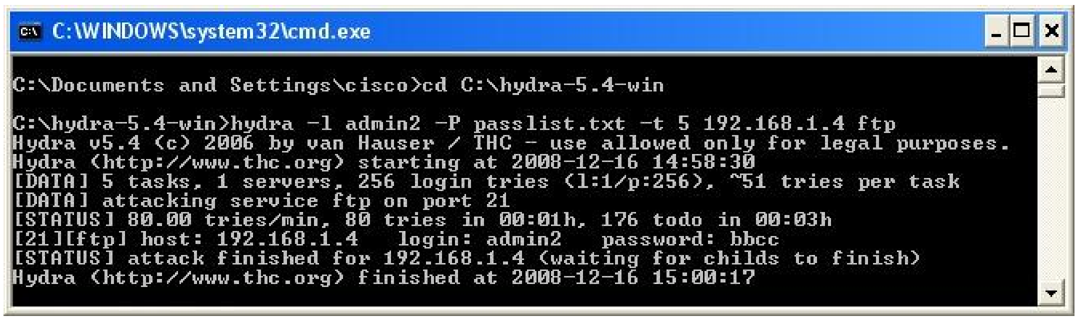
The passlist.txt file holds 256 passwords (Section 4.5.2 provides additional information about the password choice) among which only one is valid. The "-t 5" option limits Hydra to use only 5 concurrent threads rather than the 16 default. This limitation of Hydra was identified while testing the tool. With the default configuration, Hydra did not return the correct password. The same observation was made for any thread setting higher than 5. This technical difficulty slowed the password cracking process slightly, given the fact that the password possibilities were relatively short. Indeed, with 16 threads, the password can be obtained in tenths of seconds, while with 5 threads, it takes



```
C:\>nmap -sS -sV -P 0 -T5 -O 192.168.1.4

Starting Nmap 4.76 ( http://nmap.org ) at 2008-12-16 14:53 GMT Standard Time
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
 Try using --system-dns or specify valid servers with --dns-servers
Interesting ports on 192.168.1.4:
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
21/tcp open  ftp?
1 service unrecognized despite returning data. If you know the service/version,
please submit the following fingerprint at http://www.insecure.org/cgi-bin/servi
cefp-submit.cgi :
SF-Port21-TCP:V=4.76%I=7%D=12/16%Time=4947C0E0%P=i686-pc-windows-windows%r
SF:(NULL,42,"220-Welcome\x20to\x20the\x20Experiment\x20Server\r\n220\x20Cr
SF:ushFTP\x20Server\x20Ready!\r\n")%r(GenericLines,42,"220-Welcome\x20to\x
SF:20the\x20Experiment\x20Server\r\n220\x20CrushFTP\x20Server\x20Ready!\r\
SF:n")%r(Help,6A,"220-Welcome\x20to\x20the\x20Experiment\x20Server\r\n220\
SF:x20CrushFTP\x20Server\x20Ready!\r\n550\x20Command\x20not\x20recognized\
SF:x20or\x20allowed\.\r\n")%r(GetRequest,6A,"220-Welcome\x20to\x20the\x20E
SF:xperiment\x20Server\r\n220\x20CrushFTP\x20Server\x20Ready!\r\n550\x20Co
SF:mmand\x20not\x20recognized\x20or\x20allowed\.\r\n")%r(HTTPOptions,6A,"2
SF:20-Welcome\x20to\x20the\x20Experiment\x20Server\r\n220\x20CrushFTP\x20S
SF:erver\x20Ready!\r\n550\x20Command\x20not\x20recognized\x20or\x20allowed
SF:\.\r\n")%r(RTSPRequest,6A,"220-Welcome\x20to\x20the\x20Experiment\x20Se
SF:rver\r\n220\x20CrushFTP\x20Server\x20Ready!\r\n550\x20Command\x20not\x2
SF:0recognized\x20or\x20allowed\.\r\n")%r(RPCCheck,42,"220-Welcome\x20to\x
SF:20the\x20Experiment\x20Server\r\n220\x20CrushFTP\x20Server\x20Ready!\r\
SF:n")%r(DNSVersionBindReq,42,"220-Welcome\x20to\x20the\x20Experiment\x20S
SF:erver\r\n220\x20CrushFTP\x20Server\x20Ready!\r\n")%r(DNSStatusRequest,4
SF:2,"220-Welcome\x20to\x20the\x20Experiment\x20Server\r\n220\x20CrushFTP\
SF:x20Server\x20Ready!\r\n")%r(SSLSessionReq,6A,"220-Welcome\x20to\x20the\
SF:x20Experiment\x20Server\r\n220\x20CrushFTP\x20Server\x20Ready!\r\n550\x
SF:20Command\x20not\x20recognized\x20or\x20allowed\.\r\n")%r(SMBProgNeg,42
SF:,"220-Welcome\x20to\x20the\x20Experiment\x20Server\r\n220\x20CrushFTP\x
SF:20Server\x20Ready!\r\n")%r(X11Probe,42,"220-Welcome\x20to\x20the\x20Exp
SF:eriment\x20Server\r\n220\x20CrushFTP\x20Server\x20Ready!\r\n")%r(FourOh
SF:FourRequest,6A,"220-Welcome\x20to\x20the\x20Experiment\x20Server\r\n220
SF:\x20CrushFTP\x20Server\x20Ready!\r\n550\x20Command\x20not\x20recognized
SF:\x20or\x20allowed\.\r\n")%r(LPDString,6A,"220-Welcome\x20to\x20the\x20E
SF:xperiment\x20Server\r\n220\x20CrushFTP\x20Server\x20Ready!\r\n550\x20Co
SF:mmand\x20not\x20recognized\x20or\x20allowed\.\r\n")%r(LDAPBindReq,42,"2
SF:20-Welcome\x20to\x20the\x20Experiment\x20Server\r\n220\x20CrushFTP\x20S
SF:erver\x20Ready!\r\n");
MAC Address: 00:0A:95:CC:E7:80 (Apple Computer)
Device type: general purpose
Running: Apple Mac OS X 10.4.X
OS details: Apple Mac OS X 10.4.10 (Tiger) (Darwin 8.10.0 - 8.11.1)
Network Distance: 1 hop

OS and Service detection performed. Please report any incorrect results at http:
//nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 129.75 seconds
```

Figure 4.3 – Reconnaissance with Nmap - Step 2

more than 2 minutes. This was not an issue for this scenario, but it can slow down the process significantly for longer password combinations. Figure 4.4 shows the output of the command:



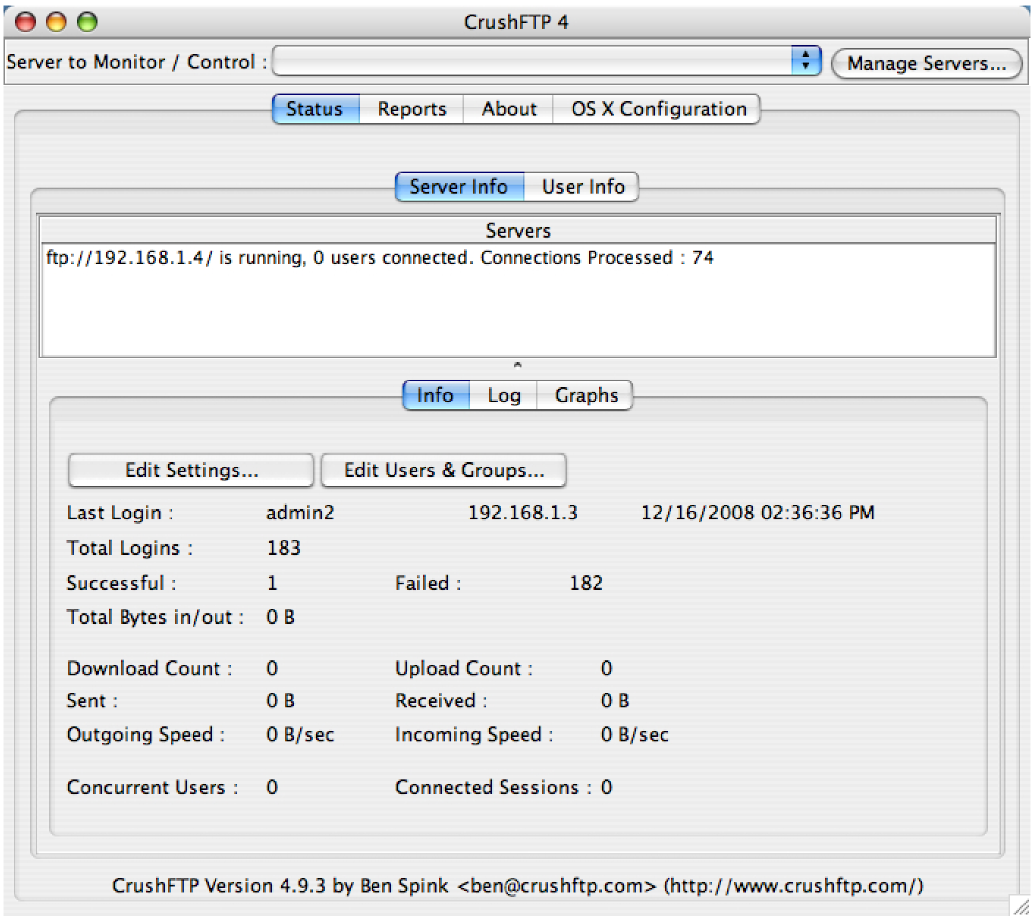
```
C:\Documents and Settings\cisco>cd C:\hydra-5.4-win

C:\hydra-5.4-win>hydra -l admin2 -P passlist.txt -t 5 192.168.1.4 ftp
Hydra v5.4 (c) 2006 by van Hauser / THC - use allowed only for legal purposes.
Hydra (http://www.thc.org) starting at 2008-12-16 14:58:30
[DATA] 5 tasks, 1 servers, 256 login tries (l:1/p:256), ~51 tries per task
[DATA] attacking service ftp on port 21
[STATUS] 80.00 tries/min, 80 tries in 00:01h, 176 todo in 00:03h
[21][ftp] host: 192.168.1.4   login: admin2   password: bbcc
[STATUS] attack finished for 192.168.1.4 (waiting for childs to finish)
Hydra (http://www.thc.org) finished at 2008-12-16 15:00:17
```

Figure 4.4 – Password Cracking with Hydra

It is possible to see the valid username/password combination on the line beginning with "[21]".

4.5.2 FTP server

Appendix C.4 provides the hardware characteristics of this system. The machine acting as the FTP server runs a dedicated FTP server software rather than using the built-in FTP server tool. This piece of software is called CrushFTP v4.9.3 [81]. This software was chosen because it offers a centralised method of management, from usernames and passwords to log files. It also offers many "live" details on the ongoing FTP sessions, such as the number of active connections, the number of failed and successful logins, and so on (Figure 4.5).



Servers

ftp://192.168.1.4/ is running, 0 users connected. Connections Processed : 74

```
Last Login :          admin2          192.168.1.3     12/16/2008 02:36:36 PM
Total Logins :        183
Successful :          1          Failed :            182
Total Bytes in/out :  0 B

Download Count :      0          Upload Count :      0
Sent :                0 B        Received :          0 B
Outgoing Speed :      0 B/sec    Incoming Speed :    0 B/sec

Concurrent Users :    0          Connected Sessions : 0
```

CrushFTP Version 4.9.3 by Ben Spink <ben@crushftp.com> (http://www.crushftp.com/)

Figure 4.5 – FTP server

The target files of the data theft on the FTP server is a large video file renamed to "ScenarioDatabase" in order to mimic a large database.

There are three considerations to take into account with the setup of the FTP server. First of all, as mentioned in Section 3.4, FTP is not a secure protocol. In a real environment, a more secure protocol such as **S**ecure **SH**ell (SSH) for instance would certainly be preferred.

The second security weakness to take into account is the weakness of the username/password combination used to log into the FTP server. This pair is set to "admin2" as username and "bbcc" as password. The password is chosen among the list of all possible combinations of four characters (a, b, c and d). This means that there is $4^4 = 256$ possible combinations, which is not highly secure.

Finally, the FTP server is setup in a way that allows a password brute force attack. The settings limiting the rate of failed login attempts in a period of time is set high on purpose in order to allow Hydra to operate at reasonable speed without getting blocked by the server security.

These intentional security flaws were implemented to allow the experiment to fit in the given project timescale. With these parameters, it is possible to achieve data theft within five minutes. If SSH and server login limitation were implemented, it would take longer to achieve the same result. The second reason for these flaws to be present is the fact that this experiment aims at testing NIDSes rather than achieving the best server security possible.

4.5.3 Experimental parameters

The experimental parameters for the scenario experiment are as follow:

- FTP server username: admin2.
- FTP server valid password: bbcc.
- Training window length: 10 minutes.
- Data theft length: 5 minutes.

4.6 Conclusion

This section has shown the tools used in order to implement the designs from chapter 3. The tools used in this experiment are SPADE and Snort for the IDSs, Tcpreplay for the background traffic generation, Nmap, Hydra and MACE for the exploits generation and CrushFTP for the FTP server. It has also explained the steps needed to install and configure them when needed. Section 4.3.2 and Section4.2 provide detailed documentation for the set up of the anomaly detector SPADE and for replaying the DARPA evaluation set. This documentation fills in the lack of material currently available about these specific tools. Table 4.1 shows a summary of these tools, the platforms they run on and versions.

Table 4.1 – Experiment tools summary

Tool	Version	OS
Snort	2.3.3 (Build 14)	FreeBSD 7
SPADE	2.3.0	FreeBSD 7
TCPReplay	3.0.beta7	Ubuntu 6.06
Nmap	4.76	WinXP
Hydra	5.4	WinXP
CrushFTP	4.9.3	MAC OSX
MACE	0.3	FreeBSD 5.4

Chapter 5

Evaluation

5.1 Introduction

This chapter presents an analysis of the detection systems presented in Section 4.3 with the testbed described in Chapter 3 and implemented in Chapter 4. Sections 5.2.1, 5.2.2, 5.2.1 and 5.3.2 respectively present the results and findings of the training window experiment and of the scenario. Section 5.4 shows the limitations of these experiments, as well as their weak points. Finally, section 5.5 concludes with this evaluation chapter.

5.2 Training window experiment

Section 2.7.1 provides a set of metrics adapted to the evaluation of IDSs: efficiency and effectiveness. Efficiency is equivalent to the false positive occurrence: in this experiment, the false-positive count is null, thus it is pointless using this metric with regards to this experiment. The second metric, effectiveness, is the main metric used for this experiment, giving the false-negative occurrence (Table 2.2 provides a definition of the different types of alerts). In this experiment, effectiveness and detection rate are the same.

5.2.1 Results

Figures 5.1 and 5.2 present respectively the count of true-positives and the effectiveness for SPADE during the overall experiment. It is possible to notice an increase of true-positives detection when the learning window length increases. This augmentation is nevertheless minimal when talking in terms of detection rates: Figure 5.2 shows that the increase is less than 4%.

Figures 5.3 and 5.4 show the detection rate of SPADE for the first attack

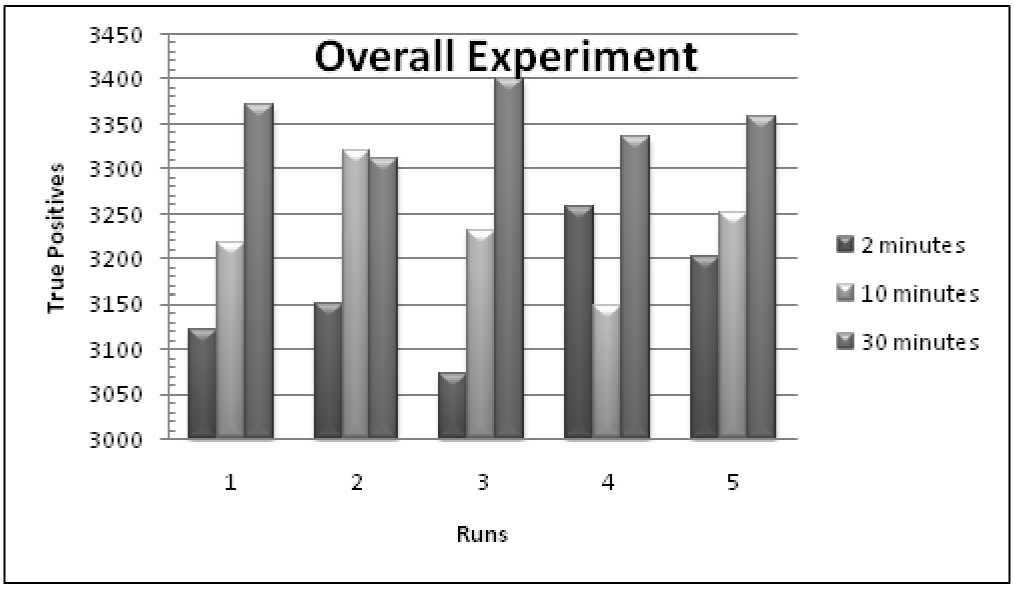


Figure 5.1 – SPADE True-positives results

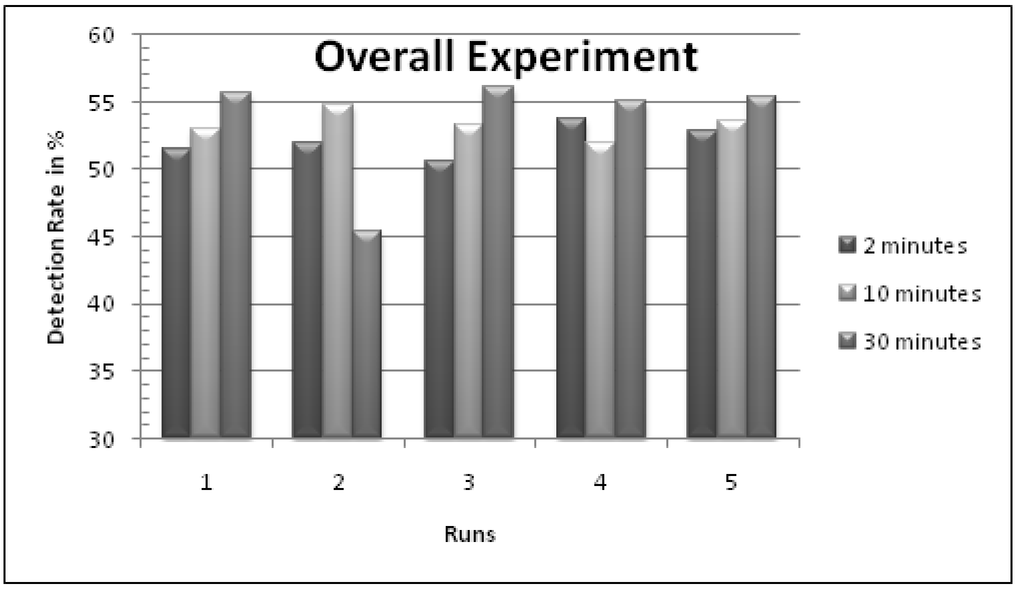


Figure 5.2 – SPADE detection Rate

and for the second attack separated. It is possible to see that the detection rate for the second series of attacks is much higher than for the first series of attacks, and so on for all different learning window lengths. This increase is around 10%.

Table 5.1 presents the averages of the detection rates for all training window lengths, for the overall attack and for each attack series.

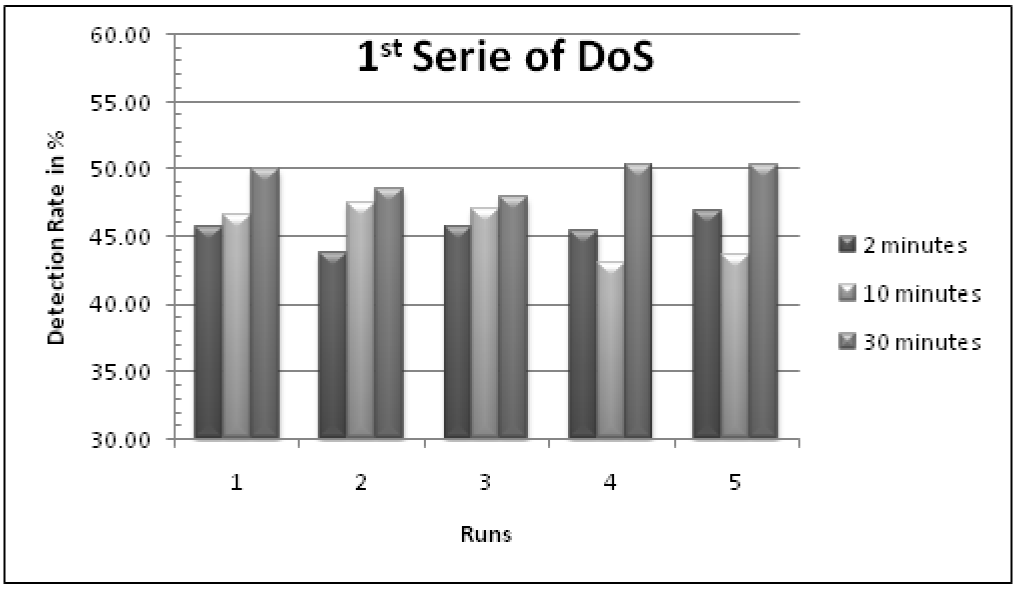


Figure 5.3 – 1^{st} attack detection Rate

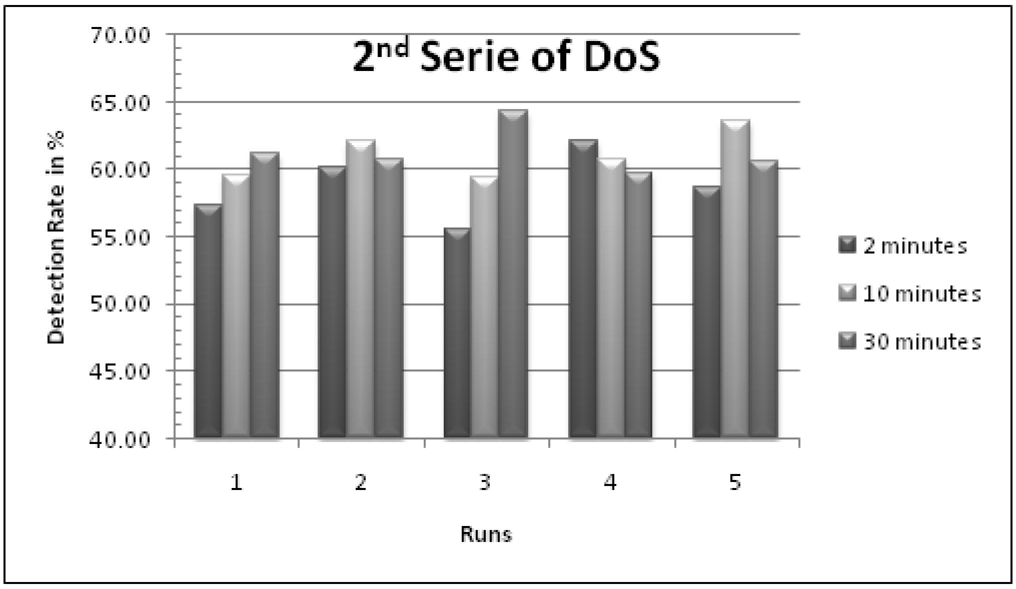


Figure 5.4 – 2^{nd} attack detection rate

Table 5.1 – Training window experimental results - Averages

Learning period length (minutes)	2	10	30
Overall detection rate	52.12	53.32	53.49
1^{st} **attack detection rate**	45.47	45.55	49.38
2^{nd} **attack detection rate**	58.77	61.09	61.30

5.2.2 Findings

Table 5.1 shows that the anomaly-based IDS during the second set of attack has learnt from the previous attack since its detection rate for the second attack is improved by 10 to 15%.

The aim of this experiment was to test if the learning window length of the IDS would have an impact on its detection rate. The detection rate has improved by only 1% between a 2 minutes and a 10 minutes learning window. The improvment is only 0.3% between 10 minutes and 30 minutes learning windows. In light of these results, it is possible to affirm that SPADE does not need a specific training period length to achieve better detection rates.

The last finding from this experiment is the fact that the anomaly detector does not produce any false-positive alarms and detects only two thirds of the packets of a same attack. Section 5.4 provides hypothesis concerning this phenomenon.

5.3 Scenario

After some informal tests following the same exploits steps as the scenario, the behaviour of the anomaly detector seemed odd compared to the expected results. Because of this, the scenario experiment was carried out in a more informal manner than the experiment from section 5.2. This odd behaviour was characterised by the fact that SPADE did not produce any false-positive alarms during all the tests and had an unsatisfactory detection rate.

5.3.1 Results

Table 5.2 presents an overview of the detection capabilities of each system observed during five runs of each distinct exploit.

Table 5.2 – Scenario experimental results

Exploit	Snort	SPADE
IP scan	Not detected	Not detected
Portscan	Detected	Detected
Brute force	Detected	Not detected
Data theft	Not detected	Not detected

5.3.2 Findings

With regards to the signature-based IDS, Snort, the detector flagged all the exploits for which it had a signature. It detected the portscan and the brute force attack on the FTP server. It did not detect the two other exploits, IP scan and data theft, for several reasons. The reason why it did not detect the IP scan is that, in order to achieve this, Snort needs all the MAC addresses of the subnet in order to detect an **A**ddress **R**esolution **P**rotocol (ARP) scan on all the machines it knows. This difficulty is due to the fact that ARP packets are layer 2 (from the OSI model) packets. Snort works mainly on the Network (3) and the Transport (4) layers, with other methods detecting anomalies in the Data Link Layer and the Application Layer Protocols. The data theft exploit cannot be detected by Snort because it consists of a legitimate FTP file transfer.

With regards to the anomaly-based IDS, SPADE, the detector only flagged one exploit out of the four in total: the portscan. It detected this exploit with its closed destination preprocessor. These results for the anomaly detector are odd. Given the fact that no FTP activity has be done in the background traffic between the exploit generator and the FTP server prior to carrying out the data theft attack (all the exploits), the anomaly detector should in theory detect the brute force traffic and the data theft traffic as anomalous since the percentage of FTP traffic is increasing when these events occur.

5.4 Critique of implementation

There are two main points of the implementation presented in Chapter 4 which deserve further discussion: the anomaly-based IDS SPADE and the background traffic.

The first point concerning the anomaly detector SPADE is that using it with its default configuration proved useless. As Liston [76] and Farshchi [77] state, SPADE needs to be fine tuned to the network it is monitoring in order to prove worthy. This specific tuning can be achieved either by changing the thresholds manually or by using the automatic observation mechanism built with SPADE. The first method would probably be very time exhausting, while the second automatic method seems more adequate. It has not been used here in these experiments for two reasons. The first reason is that the automatic observation period spans over 24 hours and this was not possible to achieve in the experimental conditions available. The second reason is that the packet capture used to create background traffic might have been too short. This leads to the second point to discuss: the background traffic.

Another reason why the anomaly detector SPADE performed in an unsatisfactory manner in both experiments could be that the tcpdump (Monday of the first week of the DARPA set) might be too short. The packet capture file used could be replayed in about five minutes. This could lead to erroneous profile generation by SPADE. Another factor linked with the length of the packet capture is the replay speed. As mentioned in section 4.2, the background traffic speed was of 11.6 Mbps. Although this could be an explanation, the SPADE fine tuning necessity seems more likely to explain the results observed in both experiments.

5.5 Conclusions

With regards to the training window experiment, the results showed that a learning period variation does not heavily influence the detection rate of SPADE. These results seem odd however. The fact that the anomaly detector SPADE did not generate any false-positives alarms and only detected two thirds of similar intrusive packets is different from any expected results. As shown in Section 5.4, this could be due to SPADE configuration or to the fact that the background traffic is not lengthy enough to mimic realistic background traffic; or simply, SPADE might not need a *traditional* learning period like other anomaly detectors.

The results of the scenario experiment proved inconclusive as well. With regards to the signature-based IDS Snort, the results confirmed the results expected in Section 3.5: a signature detector is detecting the intrusions it is set up for. Snort configuration had rules about portscans and FTP brute force attacks, hence it detected both steps of a data theft only. The fact that the experiment failed comes from the point-of-view of the anomaly detector. SPADE only detected the second reconnaissance step of the data theft. Given the fact that any traffic between the exploit generator and the FTP server is new to the anomaly detector, it should be flagged as anomalous. The same possible explanations apply for this experiment (5.4).

To conclude this chapter, it is possible to say that the results of both experiments are inconclusive since the anomaly-based IDS SPADE should be fine tuned to the network monitored. Although the results are inconclusive, the training window experiment showed that SPADE does not need a pure dedicated training period like other anomaly detectors do. The results of the scenario experiment for the signature-based IDS Snort confirmed the fact that signature detectors are as good as their signatures. With regards to the background traffic, a more realistic approach needs to be found in order to run the anomaly detector with traffic over a longer period of time.

Chapter 6

Conclusions

6.1 Project management

Appendices G and H respectively show the initial project schedule and the final one. The main difference between both Gantt char is that the literature review and testbed setup completion took more time than initially planned. Writing the literature review took almost two months rather than one because it needed many changes and several reviews by the thesis supervisor. Setting up the testbed also took more time then planned for several reasons. The main reasons are that the availability of tools, such as the anomaly detector and the background traffic generator, were hard to establish, and that the configuration of the aforementioned tools was non trivial. Setting up a testbed recreating the DARPA set and setting up the anomaly detector SPADE prove very time consuming. The second factor, setting up SPADE, needed too much time to be set up ideally in order to carry out the experiments.

Appendix I provides the diaries, highlighting meetings with the supervisor.

6.2 Appraisal of achievement

The overall aim of this project is to provide an analysis and evaluation of NIDS. In order to achieve this overall aim, there are intermediary goals to achieve. These steps are as follow:

- Carry out a critical appraisal of network security.
- Provide a theoretical analysis of IDS.
- Investigate the current IDS evaluation methodologies.
- Establish availability of tools.
- Setup an evaluation testbed and DUTs.

- Evaluate the detection systems with the testbed implemented.

6.2.1 Objective 1

The first objective has been met in the first half of the literature review, in Sections 2.2, 2.3 and 2.4. From the literature review, it appears that network security is currently not under control in the majority of organisations. Most of the time, there is no security dedicated staff, and the employees in charge of security do not spend as much time as they should on it. This is critical since cyber criminality is rising and organisations are increasingly at risk. Such risks include data theft, financial loss, corporation image deterioration, and so on. In order to keep the organisation assets secure, network security is often implemented via antiviruses, firewalls and IDSs - IPSs. It also appears that a combination of all these tools highly improves the overall network security, rather than relying on a couple of tools uniquely, for example, firewalls. The literature review shows as well that tools like anomaly detectors are actively researched in the current intrusion detection domain.

6.2.2 Objective 2

The second objective was met in Section 2.5 of the literature review. The literature showed why signature-detection IDSs are more popular than anomaly-detection IDSs. Signature detection relies on signature analysis of known threats, kept in a database. On the other hand, anomaly detection relies on creating profiles of normal traffic, and then the analysis of current traffic against these profiles in order to find anomalous traffic. Signature detection is a well known, precise methodology while anomaly detection is still heavily under research. This last type of detection is mainly not implemented due to the fact that it produces a large amount of alerts in general. Security employees spend the minimum amount of time they can spare on security, so it is easily possible to understand the issue with anomaly detection. On the other hand, the main advantage of anomaly detection is that it can potentially detect zero day attacks, as opposed to signature detection which needs prior knowledge of an attack to detect it.

6.2.3 Objective 3

The third objective was met in section 2.7 of the literature review chapter. It has been shown that there are currently two methods for evaluating NIDSs: offline and online evaluation. Offline evaluation is often implemented with the use of the well known DARPA set. This type of evaluation consists of recreating previous traffic traces as opposed to online evaluation, aiming at using simulation tools in order to generate realistic new traffic. The literature showed that offline evaluation with the DARPA set is no longer appropriate to current network characteristics. On the other hand, the main issue with online evaluation is that the simulated traffic has to be as realistic as a real-environment network. Such an issue has been experienced when trying to setup the Harpoon tool for the experimental side of this project. The conclusion for this objective is that testing of NIDSs should be implemented with realistic traffic simulation wherever possible.

6.2.4 Objective 4

The forth objective was partially met while concluding with the literature review. The availability of tools like MACE and Harpoon was established by contacting their creators, Sommers et al. [67]. The main issues were finding an anomaly detector suitable for the experimental section of this project, as well as a realistic traffic simulator. Freely usable anomaly-detection NIDSs are relatively hard to find. Systems such as PHAD or NETAD are available but not suitable for the testbed implemented. The only other anomaly detector available was the one used: SPADE. Even though it was available for use, more issues arose when trying to install it, configure it and optimise it. These issues are mainly due to the fact that the SPADE project has been discontinued since 2003 and the documentation is sparse and sometimes contradictory [42] (Section 4.3.2).

This objective has only been partially met since no open source realistic traffic generator has been found. Harpoon [66] has been tested but not deemed usable. In order to overcome this issue, the offline evaluation methodology of the DARPA set had to be used to carry on with the experiments. The conclusion for this objective is that there is a lack of research, communication and sharing in this specific area of topic.

6.2.5 Objective 5

As mentioned in section 6.2.4, it has not been possible to setup a traffic simulator. Nevertheless, a testbed was setup with a core common to both experiments. The background traffic generation used the DARPA set and was implemented with the TCPReplay tool, once again, not without difficulties. This is also due to the fact that recreating such a task is not well documented. Only one tutorial by Graves and Saliou [72] was found on the topic. The setup of the DUTs was relatively easier. The scenario experiment involved setting up an FTP server and a station acting as an exploit generator in order to mimic a data theft attack. Such an attack involved the setup of a reconnaissance tool (Nmap), a remote dictionary attack tool (Hydra) and data theft.

6.2.6 Objective 6

The experimental results did not prove as successful as the other objectives aforementioned. This is mainly due to the fact that SPADE, the anomaly detector, did not operate as well as planned. The first experiment, the training window experiment, was conducted without issue but the results were unsatisfactory. SPADE does not seem to be relying on a pure training period to create a normal traffic profile. Another observation is that it did not produce any false-positive alarm during both experiment, although the theory on the anomaly detector concluded that they produce large amount of alerts (Section 2.5). The detection rate of SPADE also proved relatively poor: 50% for the training window experiment and only 25% of exploits detected during the scenario experiment. Possible explanations include the fact that SPADE requires to be fine tuned to the network monitored, either manually or with automatic thresholds generation. A second possible reason for these results might be that the packet capture file was too short or replayed at an inappropriate speed.

6.3 Suggestions for future work

6.3.1 Experiment

With regards to the experiments conducted in Chapter 5, future work can be organised in two categories. If the aim is to keep the testbed and DUTs as they are defined in Chapter 4, then three improvements can be brought in:

- Run longer traffic captures.
- Run traffic capture at lower speed.
- Run SPADE automatic thresholds configuration.

This first improvement would be running all the packet captures from the DARPA set in the first training week or more (there are seven training weeks data available). This would extend the background traffic cycle considerably. Currently, only the Monday traffic of the first week is used. The second improvement would not prove useful on its own, but extending the length of the background traffic cycle and slowing the speed down could change the results of the experiments. Finally, the last change to the current experimental setup is the most likely to have an important impact on the results. Since SPADE requires to be well tuned to the network it monitors, running its automatic thresholds configurations might adapt the detector to the testbed better than with an "off the box" configuration.

If the aim is to change part of the experimental setup, then replacing the captured traffic generation by a realistic traffic simulator is crucial. The other system that would be worth replacing as well is the anomaly detector, SPADE. The only issue with replacing these devices is that the experiment has been implemented with the only currently available tools.

6.3.2 Area of research

With regards to the general area of research (IDSs and evaluation), the future work to be considered mainly concerns the evaluation side of IDSs. As shown in Section 2.8, a new way of generating background traffic is needed. Since the DARPA set is now obsolete, a new project involving releasing free recent real network packet captures would bring forward IDS evaluation.

The second future work needed is a way of simplifying testbeds creation for IDS evaluation. Currently, any researcher trying to evaluate an IDS often has to setup a complex testbed, composed of diverse tools in order to achieve the evaluation. The problem is that documentation on some of these necessary systems is rare. Often, this setup process is at least as time consuming as the evaluation process itself. The need for a centralised IDS framework is much needed. Figure 6.1 presents a "plug and play" framework that could be designed:

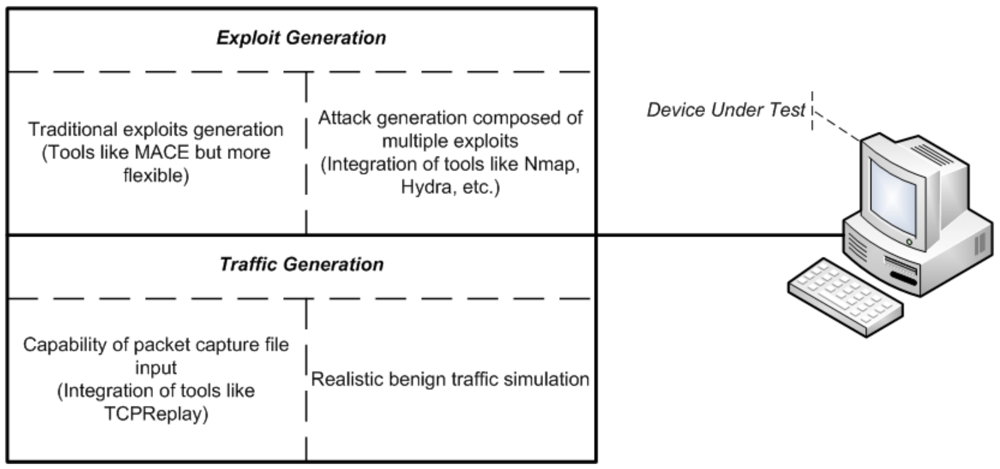

Exploit Generation
Traditional exploits generation (Tools like MACE but more flexible)
Attack generation composed of multiple exploits (Integration of tools like Nmap, Hydra, etc.)
Traffic Generation
Capability of packet capture file input (Integration of tools like TCPReplay)
Realistic benign traffic simulation
Device Under Test

Figure 6.1 – "Plug and play" evaluation framework

Such a framework would provide researchers with all the tools they need to create exploits and different kind of background traffic generation in a centralised device. This would solve the issue of setting up testbeds with many independent tools and would centralise all configuration into one device.

References

[1] N. I. of Standards & Technology, *An Introduction to Computer Security: The NIST Handbook*, NIST, Ed. U.S. Department of Commerce, 2006.

[2] J. Grossklags, N. Christin, and J. Chuang, "Security and insurance management in networks with heterogeneous agents," in *9th ACM conference on Electronic commerce*. New York, NY, USA: ACM, 2008, pp. 160–169.

[3] K. Labib, "Computer security and intrusion detection," *Crossroads*, vol. 11, no. 1, pp. 2–2, 2004.

[4] K. Ingham and S. Forrest, "A history and survey of network firewalls," University of New Mexico, Tech. Rep., 2002.

[5] R. Bace and P. Mell, "Nist special publication on intrusion detection systems," National Institute of Standards and Technology, Tech. Rep., 2001.

[6] M. T. Siponen and H. Oinas-Kukkonen, "A review of information security issues and respective research contributions," *SIGMIS Database*, vol. 38, no. 1, pp. 60–80, 2007.

[7] J. Graves, W. J. Buchanan, L. Saliou, and J. L. Old, "Performance analysis of network based forensic systems for in-line and out-of-line detection and logging," in *5th European Conference on Information Warfare and Security (ECIW)*, 2006.

[8] J. Hale and P. Brusil, "Secur(e/ity) management: A continuing uphill climb," *J. Netw. Syst. Manage.*, vol. 15, no. 4, pp. 525–553, 2007.

[9] W. H. Baker and L. Wallace, "Is information security under control?: Investigating quality in information security management," *IEEE Security and Privacy*, vol. 5, no. 1, pp. 36–44, 2007.

[10] P. Fung, L. for Kwok, and D. Longley, "Electronic information security documentation," in *ACSW Frontiers '03: Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2003, pp. 25–31.

[11] J. Morrison, "Blaster revisited," *Queue*, vol. 2, no. 4, pp. 34–43, 2004.

[12] C. Miller, "The legitimate vulnerability market," in *6th Workshop on the Economics of Information Security*, The Heinz School and CyLab at Carnegie Mellon University Pittsburgh, PA, USA, June 7- 8 2007.

[13] S. Peisert, M. Bishop, and K. Marzullo, "Computer forensics in forensis," *SIGOPS Operating Systems Review*, vol. 42, no. 3, pp. 112–122, 2008.

[14] S. Perry, "Network forensics and the inside job," *Network Security*, vol. 2006, pp. 11–13, 2006.

[15] D. K. Smetters and R. E. Grinter, "Moving from the design of usable security technologies to the design of useful secure applications," in *Proceedings of the 2002 workshop on New security paradigms*. New York, NY, USA: ACM, 2002, pp. 82–89.

[16] B. Woloch, "New dynamic threats require new thinking - "moving beyond compliance"," *Computer law & security report*, vol. 22, pp. 150 – 156, 2006.

[17] D. Botta, R. Werlinger, A. Gagné, K. Beznosov, L. Iverson, S. Fels, B. Fisher, and B. Fisher, "Towards understanding it security professionals and their tools," in *SOUPS '07: Proceedings of the 3rd symposium on Usable privacy and security*. New York, NY, USA: ACM, 2007, pp. 100–111.

[18] R. C. Newman, "Cybercrime, identity theft, and fraud: practicing safe internet - network security threats and vulnerabilities," in *InfoSecCD '06: Proceedings of the 3rd annual conference on Information security curriculum development*. New York, NY, USA: ACM, 2006, pp. 68–78.

[19] Z.-Q. Wang, H.-Q. Wang, Q. Zhao, and R.-J. Zhang, "Research on distributed intrusion detection system," in *2006 International Conference on Machine Learning and Cybernetics*, 2006.

[20] K. Buzzard, "Computer security - what should you spend your money on?" *Computers & Security*, vol. 18, pp. 322 – 334, 1999.

[21] S. M. Furnell and M. J. Warren, "Computer hacking and cyber terrorism: The real threats in the new millennium?" *Computers & Security*, vol. 18, pp. 28 – 34, 1999.

[22] S. Furnell and M. Papadaki, "Testing our defences or defending our tests: the obstacles to performing security assessment references," *Computer Fraud and Security*, vol. 1, pp. 8 – 12, 2008.

[23] C. Gates and C. Taylor, "Challenging the anomaly detection paradigm: a provocative discussion," in *2006 workshop on New security paradigms*. New York, NY, USA: ACM, 2007, pp. 21–29.

[24] L. A. Hughes and G. J. Delone, "Viruses, worms, and trojan horses: Serious crimes, nuisance, or both?" *Soc. Sci. Comput. Rev.*, vol. 25, no. 1, pp. 78–98, 2007.

[25] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "A taxonomy of computer worms," in *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode*. New York, NY, USA: ACM, 2003, pp. 11–18.

[26] C. Shannon and D. Moore, "The spread of the witty worm," *IEEE Security and Privacy*, vol. 2, no. 4, pp. 46–50, 2004.

[27] K. J. Houle and G. M. Weaver, "Trends in denial of service attack technology," Carnegie Mellon University's Computer Emergency Response Team Coordination Center, Tech. Rep., 2001.

[28] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[29] K. Lan, A. Hussain, and D. Dutta, "Effect of malicious traffic on the network," in *Passive and Active Measurement Workshop*, April 2003.

[30] M. J. Edwards, *Internet Security with Windows NT*. 29th Street Press, 1998.

[31] E. Roop, "Caught between security & usability," *For The Record*, vol. Vol. 19 No. 14, p. P. 24, 2007.

[32] B. Morgan and N. Lovering, *CCNP ISCW Official Exam Certification Guide*. Cisco Press, 2008.

[33] R. Braden, "Requirements for internet hosts - communication layers," United States, 1989.

[34] H. Bidgoli, *Handbook of Information Security*, H. Bidgoli, Ed. Wiley, 2005.

[35] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 30, pp. 1 – 11, 2008.

[36] S. Axelsson, "Intrusion detection systems: A survey and taxonomy," Department of Computer Engineering, Chalmers University of Technology, Tech. Rep., 2000.

[37] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Computer Network*, vol. 31, no. 9, pp. 805–822, 1999.

[38] J. Sommers, V. Yegneswaran, and P. Barford, "A framework for malicious workload generation," in *4th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2004, pp. 82–87.

[39] H. Zhengbing, L. Zhitang, and W. Junqi, "A novel network intrusion detection system (nids) based on signatures search of data mining," in *1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 1–7.

[40] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, pp. 3448 – 3470, 2007.

[41] P. Fogla and W. Lee, "Evading network anomaly detection systems: formal reasoning and practical techniques," in *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2006, pp. 59–68.

[42] A. Orebaugh, S. Biles, and J. Babbin, *Snort Cookbook: Solutions and Examples for Snort Administrators*, O'Reilly, Ed. O'Reilly, 2005.

[43] D. E. Denning, "An intrusion detection model," in *Seventh IEEE Symposium on Security and Privacy*, 1987, pp. 119–131.

[44] F. Gong, "Deciphering detection techniques: Part ii anomaly-based intrusion detection," Network Associates, March 2003.

[45] G. Vigna, W. Robertson, and D. Balzarotti, "Testing network-based intrusion detection signatures using mutant exploits," in *11th ACM conference on Computer and Communications Security*. New York, NY, USA: ACM, 2004, pp. 21–30.

[46] W. Lee and S. J. Stolfo, "Data mining approaches for intrusion detection," in *7th USENIX Security Symposium*, 1998.

[47] M. V. Mahoney, "Network traffic anomaly detection based on packet bytes," in *2003 ACM symposium on Applied computing*. New York, NY, USA: ACM, 2003, pp. 346–350.

[48] B. D. Caulkins, J. Lee, and M. Wang, "A dynamic data mining technique for intrusion detection systems," in *43rd annual Southeast regional conference*. New York, NY, USA: ACM, 2005, pp. 148–153.

[49] M. V. Mahoney and P. K. Chan, "Learning nonstationary models of normal network traffic for detecting novel attacks," in *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2002, pp. 376–385.

[50] ——, "Learning models of network traffic for detecting novel attacks," Florida Institute of Technology, Tech. Rep., 2002.

[51] ——, "Phad: Packet header anomaly detection for identifying hostile network traffic," Florida Institute of Technology, Tech. Rep., 2001.

[52] Y. Li, B. Fang, L. Guo, and Y. Chen, "Network anomaly detection based on tcm-knn algorithm," in *2nd ACM symposium on Information, computer and communications security*. New York, NY, USA: ACM, 2007, pp. 13–19.

[53] L. Kuang and M. Zulkernine, “An anomaly intrusion detection method using the csi-knn algorithm,” in *2008 ACM symposium on Applied computing*. New York, NY, USA: ACM, 2008, pp. 921–926.

[54] A. Lazarevic, A. Ozgur, L. Ertoz, J. Srivastava, and V. Kumar, “A comparative study of anomaly detection schemes in network intrusion detection,” in *In Proceedings of the Third SIAM International Conference on Data Mining*, 2003.

[55] G. F. Cretu, A. Stavrou, M. E. Locasto, S. J. Stolfo, and A. D. Keromytis, “Casting out demons: Sanitizing training data for anomaly sensors,” in *SP '08: Proceedings of the 2008 IEEE Symposium on Security and Privacy (sp 2008)*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 81–95.

[56] P. A. Porras and P. G. Neumann, “EMERALD: event monitoring enabling responses to anomalous live disturbances,” in *1997 National Information Systems Security Conference*, oct 1997. [Online]. Available: http://www.csl.sri.com/papers/emerald-niss97/

[57] K. Julisch, “Clustering intrusion detection alarms to support root cause analysis,” *ACM Transactions on Information and System Security*, vol. 6, pp. 443–471, 2003.

[58] J. B. Colombe and G. Stephens, “Statistical profiling and visualization for detection of malicious insider attacks on computer networks,” in *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*. New York, NY, USA: ACM, 2004, pp. 138–142.

[59] R. Lippmann, S. Webster, and D. Stetson, “The effect of identifying vulnerabilities and patching software on the utility of network intrusion detection,” in *Recent Advances in Intrusion Detection, 5th International Symposium, RAID 2002*, 2002.

[60] L. M. Rossey, R. K. Cunningham, D. J. Fried, J. C. Rabek, J. W. Haines, and M. A. Zissman, “Lariat: Lincoln adaptable realtime information assurance testbed,Ť submitted for publication,” in *In IEEE Proc. Aerospace Conference*, 2002, pp. 2671–2682.

[61] D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. Mcclung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, "Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation," in *in Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, 2000, pp. 12–26.

[62] M. V. Mahoney and P. K. Chan, "An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection," in *In Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection*. Springer-Verlag, 2003, pp. 220–237.

[63] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," in *ACM Transactions on Information and Systems Security*, vol. 3, no. 4, 2000.

[64] R. Durst, T. Champion, B. Witten, E. Miller, and L. Spagnuolo, "Testing and evaluating computer intrusion detection systems," *Commun. ACM*, vol. 42, no. 7, pp. 53–61, 1999.

[65] J. Sommers, V. Yegneswaran, and P. Barford, "Toward comprehensive traffic generation for online ids evaluation," University of Wisconsin, Tech. Rep., August 2005.

[66] J. Sommers, H. Kim, and P. Barford, "Harpoon: a flow-level traffic generator for router and network tests," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 392–392, 2004.

[67] J. Sommers, V. Yegneswaran, and P. Barford, "Recent advances in network intrusion detection systems tuning," in *CISS '06: Proceedings of the 40th IEEE Conference on Information Sciences and Systesm*, March 2006.

[68] M. S. Gadelrab, A. A. E. Kalam, and Y. Deswarte, "Defining categories to select representative attack test-cases," in *QoP '07: Proceedings of the 2007 ACM workshop on Quality of protection*. New York, NY, USA: ACM, 2007, pp. 40–42.

[69] R. A. Maxion and K. M. C. Tan, "Benchmarking anomaly-based detection systems," in *In International Conference on Dependable Systems and Networks*. IEEE Computer Society Press, 2000, pp. 623–630.

[70] S. Peisert and M. Bishop, *How to Design Computer Security Experiments*. Springer, 2007, ch. IFIP International Federation for Information Processing, Volume 237. Fifth World Conference on Information Security Education, p. 141Ű148.

[71] A. Turner, "Tcpreplay 3.x online manual," 2008. [Online]. Available: http://tcpreplay.synfin.net/trac/wiki/usage#PassingTrafficThroughaFirewallRouterNon-TransparentDevice

[72] J. Graves and L. Saliou, "Using tcpreplay - a practical tutorial," 2008, 2008.

[73] M. L. Laboratory, "Darpa intrusion detection data sets," 1998. [Online]. Available: http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1998/training/week1/monday.tar

[74] SourceFire, "Snort.org," 2008. [Online]. Available: http://www.snort.org/dl/

[75] ——, "Sourcefire vrt certified rules 2.4," 2008. [Online]. Available: http://www.snort.org/pub-bin/downloads.cgi/Download/vrt_pr/snortrules-pr-2.4.tar.gz

[76] K. Liston, "Can you explain traffic analysis and anomaly detection," SANS Institute Intrusion Detection FAQ, February 2008.

[77] J. Farshchi, "Statistical based approach to intrusion detection," SANS Institute Intrusion Detection FAQ, February 2008.

[78] Snort, "Snort forums archive," 2007. [Online]. Available: http://www.snort.org/archive-7-4696.html

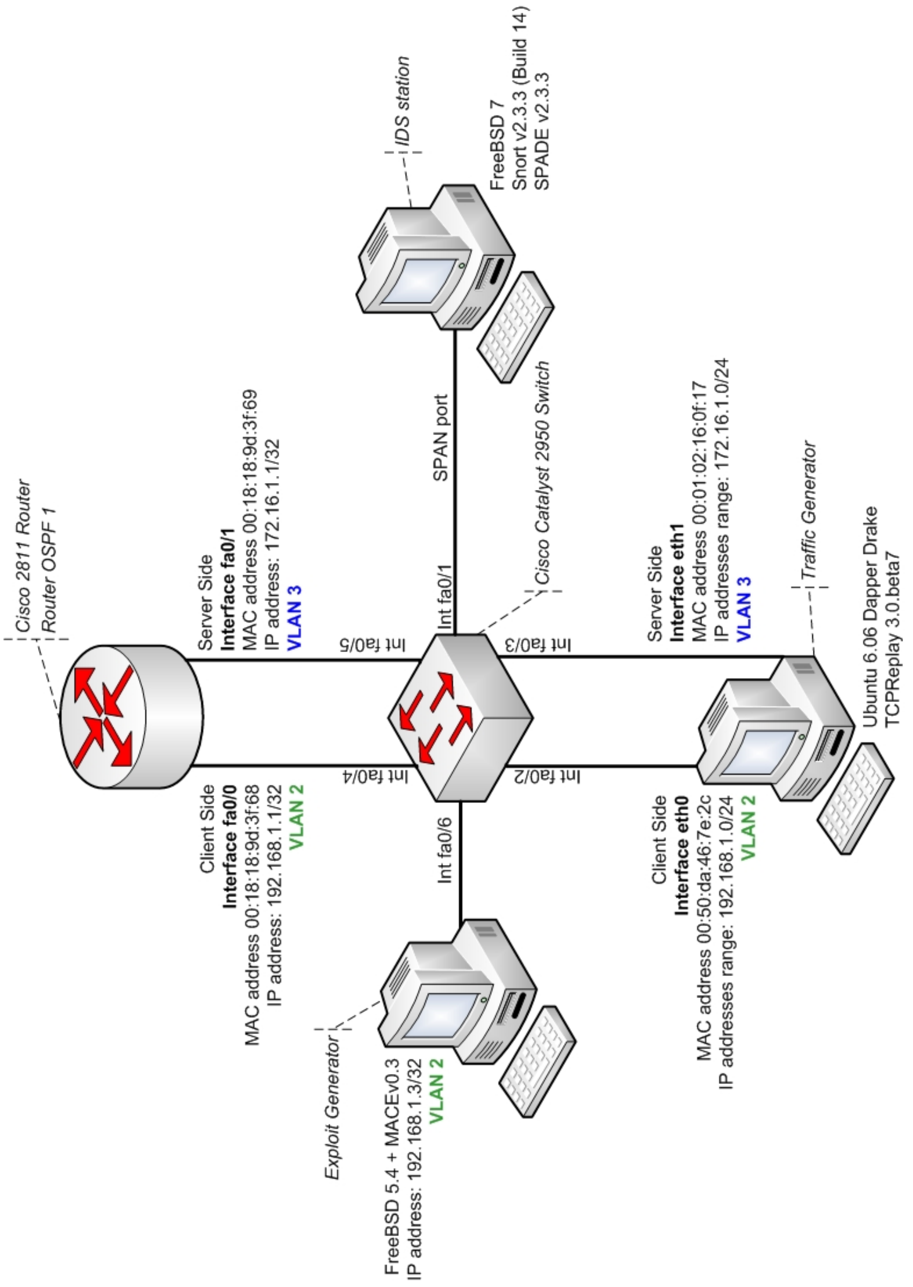
[79] Insecure.org, "Nmap security scanner," 2008. [Online]. Available: http://nmap.org/download.html

[80] THC, "Hydra - fast and flexible network login hacker," 2006. [Online]. Available: http://freeworld.thc.org/thc-hydra/hydra-5.4-win.zip

[81] B. Spink, "Crushftp," 2008. [Online]. Available: http://www.crushftp.com/download.html

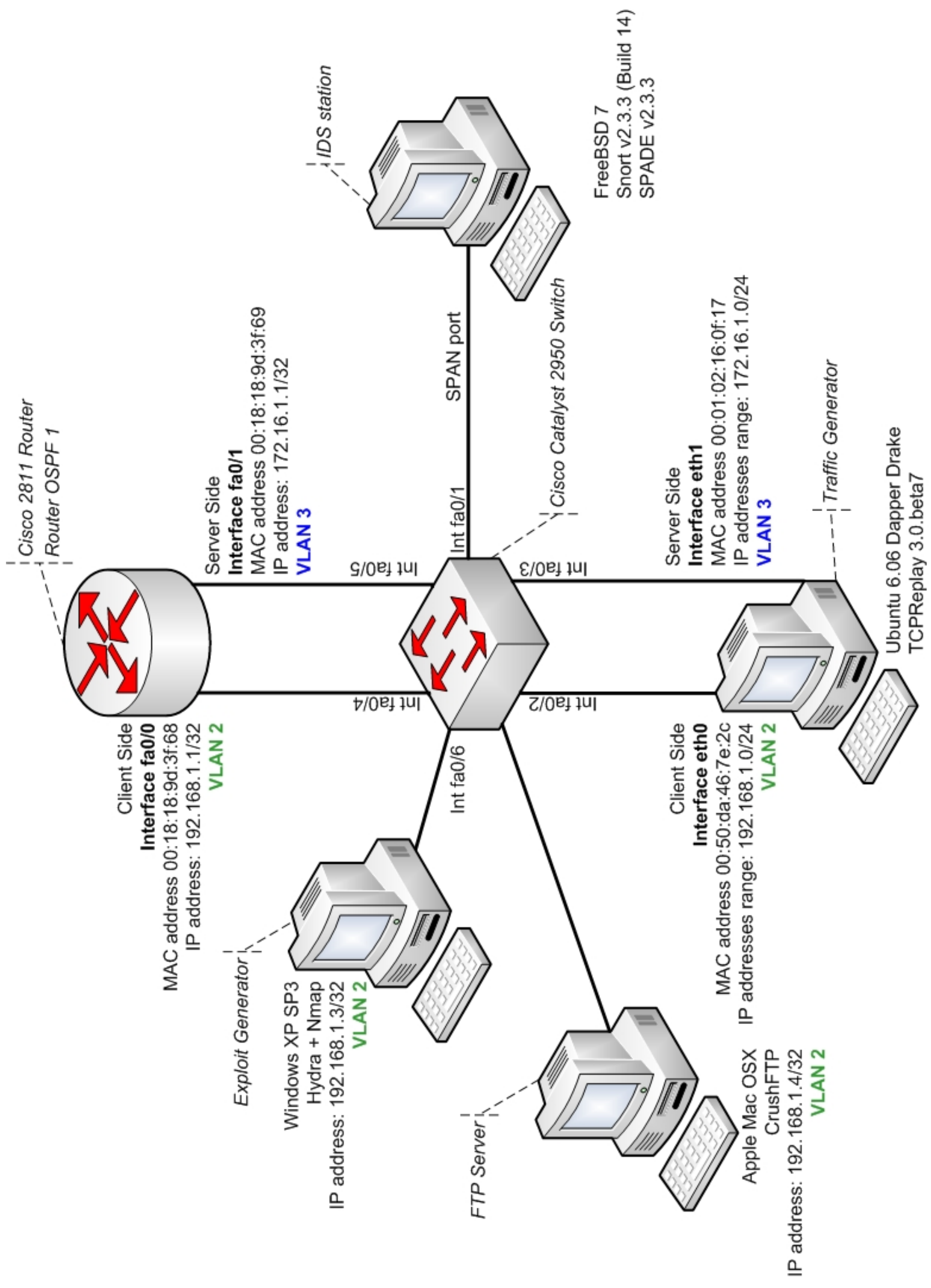
Appendix A

Training Window Diagram



Appendix B

Scenario Diagram



Appendix C

Experiment Systems Characteristics

C.1 Intrusion Detection Systems

Both IDSes run on an Intel Pentium 4, CPU 2.6GHz and 512MB of **R**andom **A**ccess **M**emory (RAM). The OS is FreeBSD 7.

C.2 Training Window exploit generator

MACE v0.3 runs on an Intel Pentium 4, CPU 2GHz and 512MB of RAM. The OS on this machine is FreeBSD 5.4.

C.3 Scenario exploit generator

The machine running the attack on the FTP server is an Intel Pentium 4, CPU 2.6GHz and 512MB of RAM. The OS is Microsoft Windows XP Professional version 2002 SP3.

C.4 FTP server

The FTP server runs on an Apple Mac OSX v10.4.11 laptop with the following characteristics: PowerPC G4, CPU 1GHz and 512MB of RAM.

Appendix D

Using TCPReplay- A practical tutorial

Disclaimer

The following text is as provided by Graves and Saliou [72].

D.1 Introduction

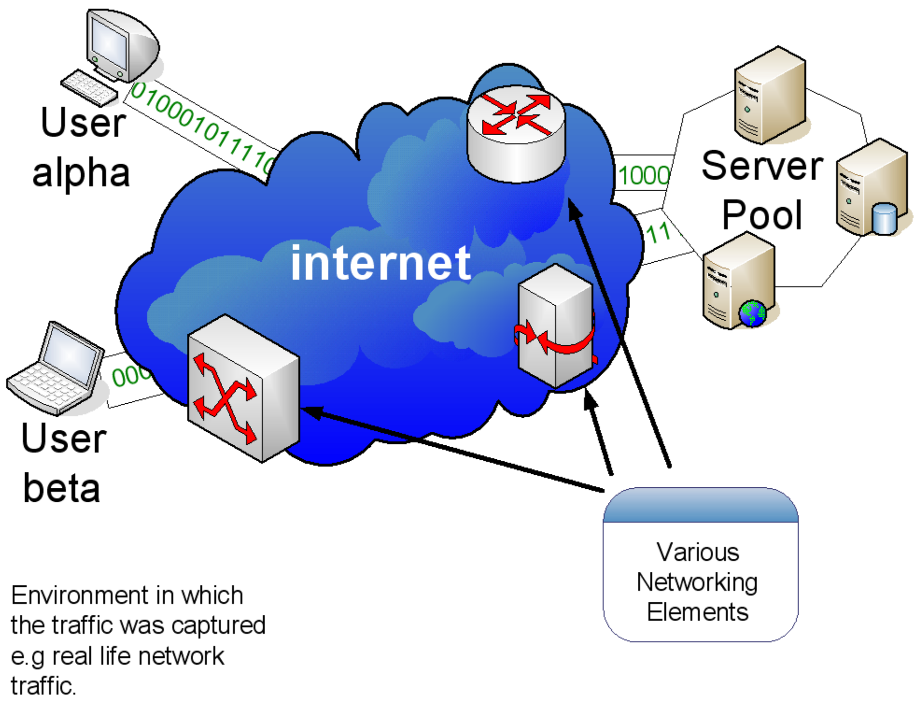
One of the most common questions we get asked, relates to how network traffic can be generated. Although there are a number of tools available, we've found the tcpreplay suite of tools to be one of the best suited for the task. Tcpreplay allows the replay of a captured `pcap` file across a network. This can be useful for a whole range of purposes, such as IDS rule verification, security testing, firewall performance, and network performance testing. What makes this tool excellent is its ability to edit layer 2-3 information within the packet. The other traffic generation tools are based on logic, can reproduce spikes, as the traffic is generated on the fly, and not from an original pcap file. Such abilities enable one to use traffic from a different network, and craft it in a way to fit in with a particular network configuration. For manipulation of layer 4 information and above, please refer to the excellent tool called Netdude.

This tutorial has been written in response to the lack of proper, documented examples on how to use tcpreplay. There is documentation available, but it is often hard to decipher such resources, especially if one is new to a specific field, or is unsure of the fundamental technologies utilised. This tutorial provides the steps we went through in order to get this tool functioning in an effective manner. Most importantly, this guide presents a 'how to' within the practical context of our test and development network.

The initial steps, such as installation of the tool, will not be covered here. They are covered by resources such as <BLAH>. We have been using this tool since version 2.x, and it has installed easily, and without any hassle on a number of different systems. Primarily, we used OpenBSD <blah> and FreeBSD <blah>, and latterly, for version 3.x onwards, we have been using a Dapper install of Ubuntu. This guide specifically relates to TCPReplay version 3.0.beta7. If you're reading this and you're using a later version of 3.x, then the fundamentals will be the same. Ensure that the version being used is not 2.x, as a lot of things have changed. In version 3.x, all of the layer information rewriting takes place a-priori, and not on the fly.

D.2 Technical Considerations

Tcpreplay is part of a suite of tools developed to enable an individual to have maximum control over the playback of network traffic. This suite of tools includes tcpprep and tcprewrite. All of these are installed as default when tcpreplay is installed on a machine. A full description of each of these tools can be found at the website <BLAH>.

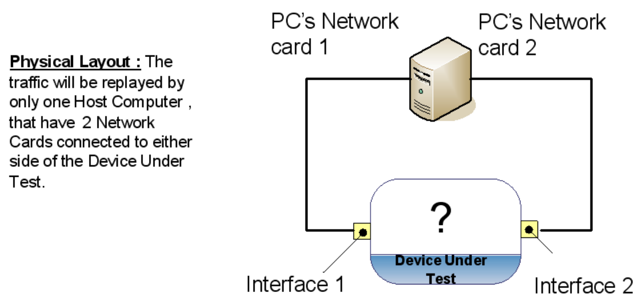

User alpha
0100010111110
internet
1000
11
Server Pool
000
User beta
Various Networking Elements
Environment in which the traffic was captured e.g real life network traffic.

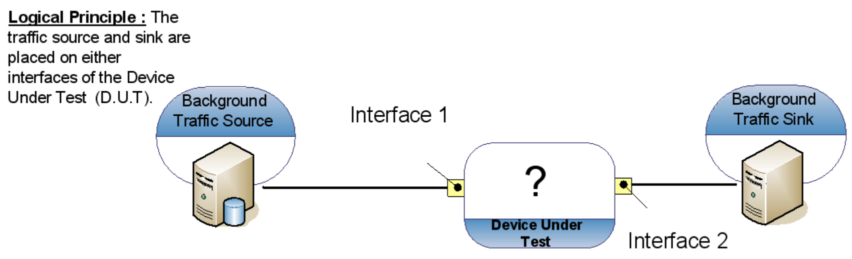
In this example, were going to be using some of the training traces from the DARPA defence evaluation tests. These can be found here <BLAH> and are a good source of ready to use pcap files. You can, of course capture your own network traces, but that's becoming increasingly difficult to do, mainly due to legal reasons. It is not advisable to use traces that contain potentially private information. When data is captured in the wild, it will contain a multitude of differing types of packet. From the ubiquitous http traffic, to ftp traffic, routing information, such as OSPF information can be found, to more exotic packets that float around. Therefore, your trace will contain a mixture of packets from a wide range of devices, protocols and so on. If your goal is to merely play network traffic over a LAN, without the needs for routing, or addressing, then it's possible to play this traffic without having to manipulate it. If you need to route the traffic in order to simulate a network boundary, or are intending on triggering firewall rules, then you need to be able to manipulate the layer 2 & 3 information that governs these processes. The tool tcprewrite is used to modify this information. Specifically, when rewriting layer 2 information in anticipation of the need to route data between subnets, we must consider how this information is utilised by the routing infrastructure.

Tcpreplay is capable of replaying a pcap file out of one or two interfaces. When playing the traffic out of two interfaces, tcpreplay simulates the client/server relationship normally seen in network traffic. In this manner, although the file resides on one machine, the traffic can be played *across* a device, such as a router. Figure 2 highlights this principle. Therefore, each interface on the device running tcpreplay can be thought of as either the client or the server. As a result, tcpreplay needs to be told through which interface to play the client stream, and the server stream. This can be achieved by using the tcpprep utility. This scans the pcap file, and produces a small cache file. These files are not large. The trace file we use is 250MB and the cache file produced is about 155KB.

D.3 Implementation

The hardest part of figuring out how to use these tools relates to the theory above. Each tool has a specific function, and relates to the manner in which the test framework is configured, and needs to run. This section provides a description of the traffic used in the rest of this tutorial, along with a requirements analysis of the traffic to be played on the network.


Physical Layout : The traffic will be replayed by only one Host Computer , that have 2 Network Cards connected to either side of the Device Under Test.
PC's Network card 1
PC's Network card 2
?
Device Under Test
Interface 1
Interface 2


Logical Principle : The traffic source and sink are placed on either interfaces of the Device Under Test (D.U.T).
Background Traffic Source
Interface 1
?
Device Under Test
Interface 2
Background Traffic Sink

Our test network has been configured for the purpose of testing a router. The primary requirement determines that the device be capable of routing the traffic played from one network to another. In order to do this, layer 2 information must be rewritten, and a client/server relationship must be defined. This will be taken care of by tcprewrite and tcpprep. A secondary requirement is that of testing the router's ability to check traffic against Access Control Lists (ACLS). We therefore need to rewrite the layer 3 information. This will also be taken care of by tcprewrite. Figure BLAH outlines the physical network layout.

There are three distinct steps that must be followed in order for these requirements to be fulfilled.

1. Divide the pcap file into the client server relationship with tcprep

```
tcpprep -a bridge  -o <OUT.prep> -i <ORIGINAL.trace>
```

The switch –a (auto) has a number of different modes. Dependant on which is selected, it divides the trace file into a client server relationship. In this instance, we use bridge mode. The other available modes are; Router, Client and Server. For more information on the logic involved in each of these options, please refer to <BLAH>.

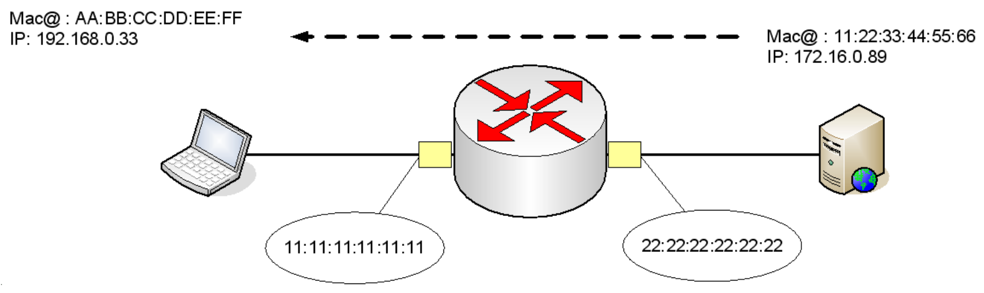
The switches –o and –i merely relate to the files being used. –o should be used to specify the name of the small cache file, and –i to specify the input file.

2. Rewrite the layer 3 and 4 information with tcprewrite

```
sudo tcprewrite
--dmac 00:0c:ce:85:ab:61,00:0c:ce:85:ab:60
--smac 00:0c:ce:85:ab:60,00:0c:ce:85:ab:61
-e 10.0.10.64/26:10.0.20.64/26
-C
-c <OUT.prep>
-o <NEWFILE.trace> -i <ORIGINAL.trace>
```

The use of tcprewrite incorporates some fundamental routing knowledge. In order to achieve a proper routing operation, we must establish which interfaces on the router face the particular network we are using. Figure BLAH highlights this. Each of the machines possess a MAC address, but they are of no interest to the router. The client is on a network which has, as its router interface, the MAC address 11:11:11:11:11:11. If the packets from the client were to be rewritten with this MAC address, when they reach the router, it will discard them because as far as the router is concerned, they have already reached their destination. In order to get the packet to traverse the device, the *opposite* MAC address must be specified from that of the originating network.

The –dmac and –smac switches allow us to specify which MAC addresses should be forged for the specific network. In this example, –dmac will take two comma delimitated MAC addresses. The first will replace the destination MAC address of the outbound packets for the server traffic. The second will be used for the client traffic. The –smac switch



3. Play the newly rewritten file out of the appropriate interfaces

```
sudo tcpreplay
-i eth2 -I eth1
-l 0  -M 2.0 -c
OUT.prep NEWFILE.trace
```

Interfaces must be up without an IP address to work properly.

tcpreplay 3.x notes

Unlike previous versions, 3.x onwards appears to have gone back to its roots. It appears that tcpprep, and tcprewrite needs to be run before the tcpreplay command in order to rewrite all the necessary layer 2 information. If the process is not performed in this manner, all kinds of weird gnarly stuff happens. Such as the MAC addresses of the packets being all set to the same, thus making it look like there's only one side of a conversation.

1) RUN tcpprep FIRST!!!

tcpprep -a bridge -o <OUT.prep> -i <ORIGINAL.trace>

2) Now run tcprewrite. Whenever one rewrites any of the layer information, a tcprep file is needed in order to tell tcprewrite which conversation is which.

sudo tcprewrite
–dmac 00:0c:ce:85:ab:61,00:0c:ce:85:ab:60
–smac 00:0c:ce:85:ab:60,00:0c:ce:85:ab:61
-e 10.0.10.64/26:10.0.20.64/26
-C
-c <OUT.prep>
-o <NEWFILE.trace> -i <ORIGINAL.trace>

3) Finally, play the two files using tcpreplay

sudo tcpreplay -i eth2 -I eth1 -l 0 -M 2.0 -c OUT.trace NEWFILE.trace

IMPORTANT 1: once again, very important, whichever MAC address is the destination, needs to be the MAC address that the opposite interface is connected to on the router. Very important. So, in this case:
eth1 is attached to router MAC 000cce85ab60
eth2 is attached to router MAC 000cce85ab61

So the destination of eth is """"61, which therefore needs to be FIRST in the tcprewrite dmac field. Conversely, the smac field needs the opposite MAC address to be specified as the destination hop. If you only get one of the right, only half the traffic will be routed properly - if you get neither right, it won't work at all.

IMPORTANT 2: the first interface defined after the -i option will always be the server - this must be factored into consideration about the design of the network.

IMPORTANT 3: there appears to be a problem when rewriting the IP address information with tcprewrite. If the IP address range is the same as that of the network addressing scheme used, the ARP cache within the router gets corrupted - thus stopping any meaningful communications between the poisoned machines on the network.

Appendix E

Router Configuration

Cisco 2811 Router

```
en
!
conf t
!
int fa0/0
ip address 192.168.1.1 255.255.255.0
no shut
!
int fa0/1
ip address 172.16.1.1 255.255.255.0
no shut
exit
!
router ospf 1
!
end
```

Appendix F

Switch Configuration

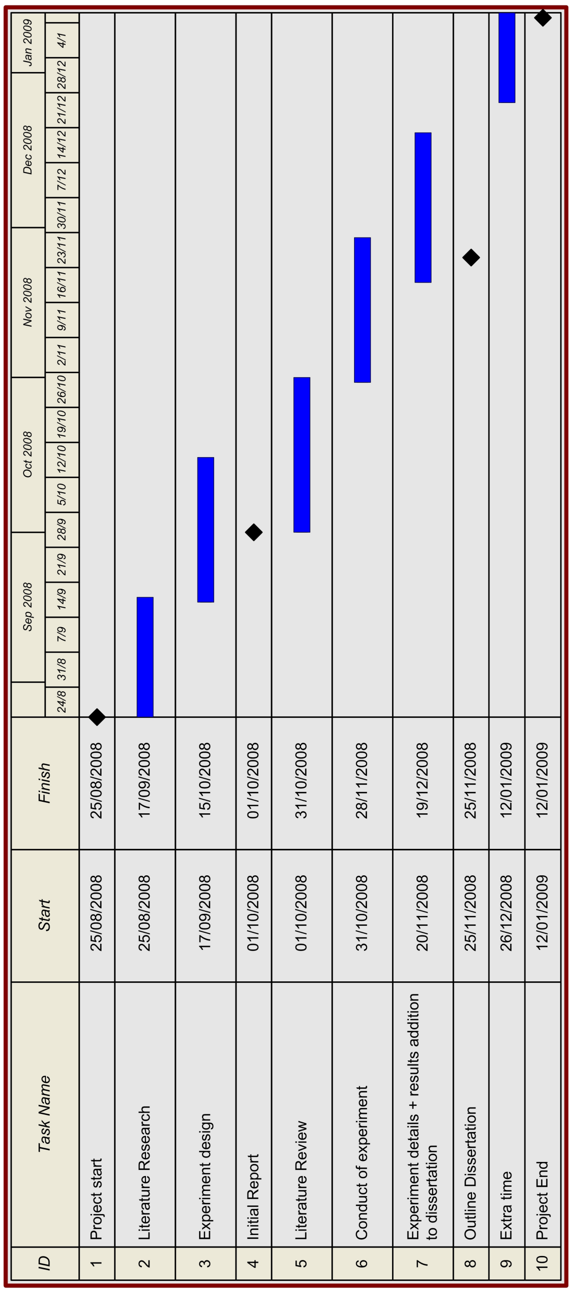
Cisco Catalyst 3560 Switch

```
en
!
vlan database
vlan 2
vlan 3
exit
!
conf t
!
int fa0/2
switchport mode access
switchport access vlan 2
no shut
!
int fa0/4
switchport mode access
switchport access vlan 2
no shut
!
int fa0/3
switchport mode access
switchport access vlan 3
no shut
!
int fa0/5
switchport mode access
switchport access vlan 3
no shut
!
int fa0/6
switchport mode access
switchport access vlan 2
```

```
no shut
!
int fa0/7
switchport mode access
switchport access vlan 2
no shut
!
monitor session 1 source int fa0/2 rx
monitor session 1 source int fa0/3 rx
monitor session 1 source int fa0/6 rx
monitor session 1 source int fa0/7 rx
monitor session 1 destination int fa0/1
!
end
```

Appendix G

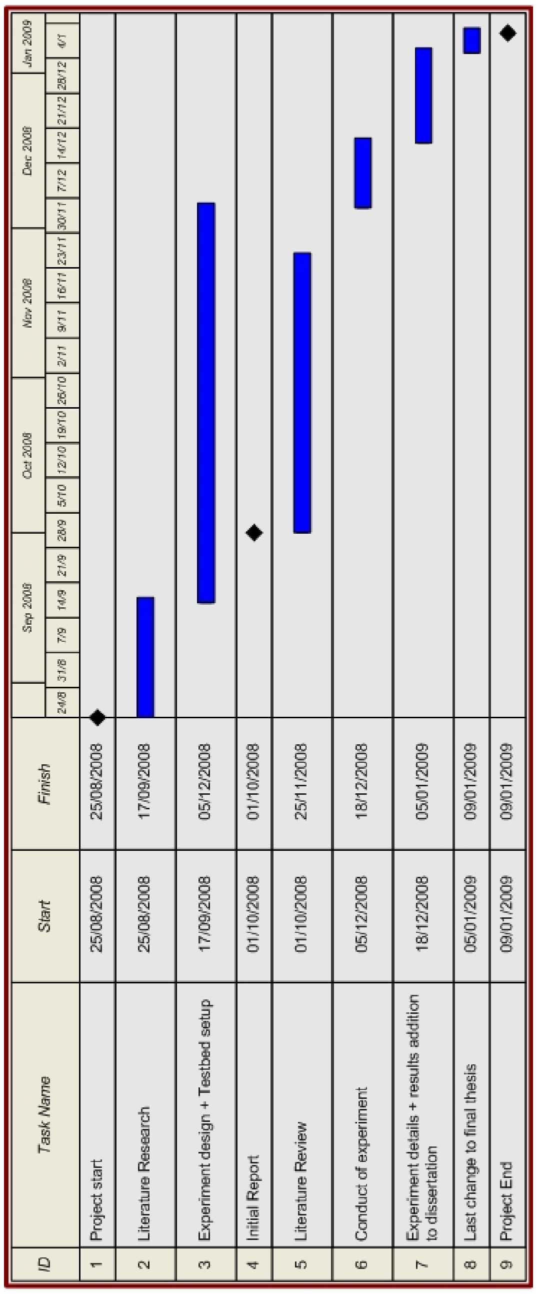
Initial Gantt Chart



ID	Task Name	Start	Finish
1	Project start	25/08/2008	25/08/2008
2	Literature Research	25/08/2008	17/09/2008
3	Experiment design	17/09/2008	15/10/2008
4	Initial Report	01/10/2008	01/10/2008
5	Literature Review	01/10/2008	31/10/2008
6	Conduct of experiment	31/10/2008	28/11/2008
7	Experiment details + results addition to dissertation	20/11/2008	19/12/2008
8	Outline Dissertation	25/11/2008	25/11/2008
9	Extra time	26/12/2008	12/01/2009
10	Project End	12/01/2009	12/01/2009

Appendix H

Final Gantt Chart



ID	Task Name	Start	Finish
1	Project start	25/08/2008	25/08/2008
2	Literature Research	25/08/2008	17/09/2008
3	Experiment design + Testbed setup	17/09/2008	05/12/2008
4	Initial Report	01/10/2008	01/10/2008
5	Literature Review	01/10/2008	25/11/2008
6	Conduct of experiment	05/12/2008	18/12/2008
7	Experiment details + results addition to dissertation	18/12/2008	05/01/2009
8	Last change to final thesis	05/01/2009	09/01/2009
9	Project End	09/01/2009	09/01/2009

Appendix I

Project Management

NAPIER UNIVERSITY
SCHOOL OF COMPUTING
PROJECT DIARY 1

Student: Julien Corsini **Supervisor:** Lionel Saliou
Date: 19/08/2008 **Last diary date:** /

Objectives:

- Discuss plan ideas.
- Setup Turnitin and different hand-in dates.
- Gather more research papers for the literature review.

Progress:

- Material papers read, giving a better idea about the general topic and a possible plan.
- Literature review plan ideas found.

Supervisor's Comments:

- Will discuss with Hall Hazel the Turnit-in facilities and whether this applies to MSc projects

NAPIER UNIVERSITY
SCHOOL OF COMPUTING
PROJECT DIARY 2

Student: Julien Corsini **Supervisor:** Lionel Saliou
Date: 05/09/2008 **Last diary date:** 19/08/2008

Objectives:

- Material research for literature review completed

Progress:

- 28 papers read so far but material research has not been completed on time.
- Still 8 papers to read, and possibly another 12 to check.

Supervisor's Comments:

NAPIER UNIVERSITY
SCHOOL OF COMPUTING
PROJECT DIARY 3

Student: Julien Corsini **Supervisor:** Lionel Saliou
Date: 17/09/2008 **Last diary date:** 05/09/2008

Objectives:

- Finish reading remaining papers
- Design experiment
- Identify elements required to build the experimental environment / testbed
- Design evaluation methodology
- Establish availability of software applications
- Learn how to configure and deploy anomaly-detection IDS
- Deadlines for initial report and outline dissertation

Progress:

- Remaining + 10 papers read
- Experiment designed and elements identified
- Software applications availability established (THOR / MACE)
- Deadlines set
- Evaluation methodology designed
- The main issue will be to learn how to create an anomaly IDS

Supervisor's Comments:

- Excellent idea to contact researchers who developed anomaly-based IDSes
- Email sent to MACE authors
- Authors replied to Prof. Bill Buchanan - hopefully, the required software will be provided soon

NAPIER UNIVERSITY
SCHOOL OF COMPUTING
PROJECT DIARY 4

Student: Julien Corsini **Supervisor:** Lionel Saliou
Date: 01/10/2008 **Last diary date:** 17/09/2008

Objectives:

- Find anomaly-detection IDS
- Definition of different types of IDSes and what they respectively detect
- Write initial report
- Check GnuPlot tool

Progress:

- Open source anomaly-detection IDS found (NETAD/PHAD/SPADE)
- Definitions established
- Initial report written and reviewed
- GnuPlot installed only - Tutorials to do

Supervisor's Comments:

- Feedback on Initial Report provided
- Update to be completed

NAPIER UNIVERSITY
SCHOOL OF COMPUTING
PROJECT DIARY 5

Student: Julien Corsini **Supervisor:** Lionel Saliou
Date: 16/10/2008 **Last diary date:** 01/10/2008

Objectives:

- Establish list of hardware/software requirements for MACE, for the anomaly detection IDS and to replay DARPA evaluation set
- Update initial report and meet with Professor Bill Buchanan
- Define experiment in detail
- Write a section of literature review

Progress:

- List of requirements + experiment defined in experimentation paper
- Meeting with Professor Bill Buchanan done
- Section of literature still to be written: rough draft of threats section established

Supervisor's Comments:

- Create skeleton of the literature review with bullet points
- Use review of each paper as starting point of the core of the text
- Alternatively, start to write on a more favorable topic such as experiment design and so on

NAPIER UNIVERSITY
SCHOOL OF COMPUTING
PROJECT DIARY 6

Student: Julien Corsini **Supervisor:** Lionel Saliou
Date: 31/10/2008 **Last diary date:** 16/10/2008

Objectives:

- Write "Threats" section + one other section from literature review
- Arrange hardware availability for experiment
- Start setting up the experiment testbed

Progress:

- All four sections of literature review written up (first draft, 20 pages, introduction and conclusion left to do)
- Private access to 2 computers secured, thanks to Ricardo Lazzarini
- Testbed set up not started

Supervisor's Comments:

Add topics:

- How to deal with IDS detection output
- Evaluation methods for IDSes
- Consider moving definition/theory to another chapter

NAPIER UNIVERSITY
SCHOOL OF COMPUTING
PROJECT DIARY 7

Student: Julien Corsini **Supervisor:** Lionel Saliou
Date: 13/11/2008 **Last diary date:** 31/10/2008

Objectives:

- Setup experiment testbed
- Refine experiment
- Address literature review comments

Progress:

- MACE setup successfully (Exploits generator)
- Snort setup successfully (Signature-detection IDS)
- SPADE setup successfully (Anomaly-detection IDS)

Supervisor's Comments:

- Difficulties in setting up the test environment. This is mainly due to the compatibility between the operating system (FreeBSD) and the required software components, such as Harpoon.
- Anomaly based IDS SPADE has been successfully installed
- Student to investigate installing Harpoon using the package manager instead of tar-balls.
- Student to revisit literature review and address comments.
- Student to refine the scenario of his evaluation.

NAPIER UNIVERSITY
SCHOOL OF COMPUTING
PROJECT DIARY 8

Student: Julien Corsini **Supervisor:** Lionel Saliou
Date: 10/12/2008 **Last diary date:** 13/11/2008

Objectives:

- Setup experiment traffic generator
- Setup tools for scenario
- Carry out experiment part 1 and scenario
- Address literature review comments

Progress:

- DARPA + Tcpreplay set
- Hydra + Nmap + FTP server set
- Experiment part 1 done
- Scenario test-run done but results are not satisfying
- Still a few comments to address in literature review

Supervisor's Comments:

- Consider creating your own FTP rules
- Watch out for active/passive FTP server
- Port 20 might be used for data transfer
- Consider building the configuration of both IDSes from the ground up, such as focusing only on protecting the FTP server

Acronyms

MACE	**M**alicious tr**A**ffic **C**omposition **E**nvironment
DoS	**D**enial-**o**f-**S**ervice
DDoS	**D**istributed **D**enial-**o**f-**S**ervice
IDS	**I**ntrusion **D**etection **S**ystem
LARIAT	**L**incoln **A**daptable **R**eal-time **I**nformation **A**ssurance **T**estbed
NIST	**N**ational **I**nstitute of **S**tandards and **T**echnology
IT	**I**nformation **T**echnologies
TCP	**T**ransmission **C**ontrol **P**rotocol
IP	**I**nternet **P**rotocol
ACL	**A**ccess **C**ontrol **L**ist
OSI	**O**pen **S**ystems **I**nterconnection
ICMP	**I**nternet **C**ontrol **M**essage **P**rotocol
UDP	**U**ser **D**atagram **P**rotocol
DMZ	**D**e**M**ilitarized **Z**one
HIDS	**H**ost-based **I**ntrusion **D**etection **S**ystem
NIDS	**N**etwork-based **I**ntrusion **D**etection **S**ystem
SPADE	**S**tatistical **P**acket **A**nomaly **D**etection **E**ngine
EMERALD	**E**vent **M**onitoring **E**nabling **R**esponses to **A**nomalous **L**ive **D**isturbances

CPU	**C**entral **P**rocessing **U**nit
DARPA	**D**efense **A**dvanced **R**esearch **P**rojects **A**gency
MIT	**M**assachusetts **I**nstitute of **T**echnology
IDEVAL	**I**ntrusion **D**etection **Eval**uation
KDD	**K**nowledge **D**iscovery and **D**ata mining
FTP	**F**ile **T**ransfer **P**rotocol
HTTP	**H**yper**T**ext **T**ransfer **P**rotocol
SSL	**S**ecure **S**ockets **L**ayer
PBA	**P**olymorphic **B**lending **A**ttacks
DUT	**D**evice **U**nder **T**est
RAM	**R**andom **A**ccess **M**emory
SSH	**S**ecure **SH**ell
OS	**O**perating **S**ystem
MAC	**M**edia **A**ccess **C**ontrol
PHAD	**P**acket **H**eader **A**nomaly **D**etector
NETAD	**NET**work **A**nomaly **D**etector
NIC	**N**etwork **I**nterface **C**ard
OSSIM	**O**pen **S**ource **S**ecurity **I**nformation **M**anagement
VLAN	**V**irtual **L**ocal **A**rea **N**etwork
SPAN	**S**witched **P**ort **AN**alyser
ARP	**A**ddress **R**esolution **P**rotocol
IPS	**I**ntrusion **P**revention **S**ystem
IETF	**I**nternet **E**ngineering **T**ask **F**orce

ISO	**I**nternational **O**rganization for **S**tandardization
CCITT	**I**nternational **T**elegraph and **T**elephone **C**onsultative **C**ommittee
ITU-T	**I**nternational **T**elecommunication **U**nion - **T**elecommunication