Applying Expert System Technology in a Distributed Monitoring System: An Efficient Response Mechanism for Unknown Threats

by

Hamid Bin Muhammad Cooke, BComp

A dissertation submitted to the

School of Computing

in partial fulfilment of the requirements for the degree of

Bachelor of Computing with Honours

University of Tasmania

November, 2005

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma in any tertiary institution, and that, to my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis.

…….……

Hamid Cooke

Abstract

Detecting unknown threats is a paradox; how do you detect a threat if it is not known to exist? The answer is that unknown threat detection is the process of making a previously unknown threat identifiable in the shortest possible time frame. This thesis examines the possibility of creating an unknown threat detection mechanism that security experts can use for developing a flexible protection system for networks. A system that allows the detection of unknown threats through distributed host based monitoring and the incorporation of dynamic and flexible logics with situational knowledge is described as well as the mechanisms used to develop such a system is illustrated. The system not only allows the detection of new threats but does so in a fast and efficient manner to increase the available time for responding to these threats.

Acknowledgments

The research could not be achieved without numerous people helping along the way. I would like to take this opportunity to thank these people appropriately.

I firstly would like to thank my supervisor Dr Byeong Ho Kang for offering the thesis. I also wish to thank Dr Kang for his courtesy and tolerance of my constant bombarding of seemingly identical questions. In addition to this providing the guidance and observations that expanded my research to the produce the final working system; thank you.

I also wish to acknowledge my secondary supervisor Professor Gil Chul Park for providing advice and the Worm simulator; thank you.

Nothing of this magnitude can be achieved without the support of loved ones so I wish to acknowledge Elizah for giving me the strength to continue the research as well as the timely proof readings; thank you.

While on the subject of proofings I wish to acknowledge David Benda for his one-day turn around time in proof reading my thesis. The comments you provided were exactly what I needed to 'trim the fat' as it were; thank you.

I also want to thank the battlers of Honours Room 3 namely: Stephen, Bruno, Simon (aka Manny) and Duncan. I acknowledge the fact that these guys helped out in moments of stress and for providing good, helpful insights and ideas. How we survived the airless, cramped office space I do not know!

Honourable mentions go to Ivan for insight into his system and coding advice along the way; thank you

Table of Contents

Listing of Figures and Tables

1 Introduction

The digital age has revolutionised the way we do business, gather information, and communicate. People can instantly communicate with friends and relatives around the world through instant messaging and e-mail applications while checking the latest stock prices. The massive interconnection that is the Internet encourages the fast dissemination of information and data which is vital for research and knowledge expansion. The explosive growth of the Internet, however, has come at a cost; people with malicious intentions now have a window into every computer and viruses plague corporations and home users alike. The number of threats that Internet users face is growing exponentially due mainly to the anonymity of the Internet and ease of malicious code mutations and distribution. The safeguards that we use to counter these threats do well to stop a number of the threats however the process by which they operate have a major flaw; they can only protect against threats that are known. These threats have been previously unleashed upon an unsuspecting network of systems and caused destruction and havoc. After the carnage is finished the makers of threat defences determine how to protect systems against the threat and release updates and patches for their software. This situation forms a veritable arms race that is always in the favour of threat creators due to the reactive nature of the security mechanisms. To keep ahead of the competition the threat creators are always developing new techniques by which to unleash debilitating attacks. To compound the problem the threats are ever changing, deviating from the original form so as to avoid detection by the security systems. To shift the balance back in the favour of the defenders what is needed is a way of detecting and stopping any threat; whether it be known from previous experience or not. The main focus at the current point in time is the unknown threats as these have yet to be seen and will cause significant damage.

Unknown threat detection has become a highly investigated area of the Internet security community. Unknown threat detection is the process of identifying a previously unknown attack by finding the factors that will make it identifiable for every occasion it appears. This process seems paradoxical; how can unknown threats

be identified if they are unknown? How can these unknown threats be detected if they are not known to exist? The answers to these questions are both negative and positive. If an unknown threat is created perfectly then it can spread without detection until it is time to strike. However, just as flawless software cannot be created; there is no perfect unknown threat. The unknown threat will have some weakness within its operation that will indicate that something suspicious is occurring. The activities of the threat appear, analogous to a human illness, as symptoms on the host machine. After careful examination these symptoms will identify the existence of a problem and hence the unknown threat.

An example of the unknown threat detection process in action is the investigations during and after the attack of the first Internet worm.

On the evening of 2nd of November, 1988 the Internet came under attack from a mysterious new threat (Spafford 1991, p. 1). Machines at the Cornell University and MIT were exhibiting strange. Symptoms of a new type of attack began to emerge; machines would get loaded with running processes until they would crash. During these initial stages there was a fear that *"…the program was somehow tampering with system resources in a way that could not be readily detected—that while a cure was being sought, system files were being altered or information destroyed."* (Spafford 1991, p. 1) After three days of damage limitation the threat was identified as what is now known as the Morris worm, and corrective measures were successfully created. After the infection dissipated a conference to classify the threat and after much debate the term 'Worm' spawned into existence within computing circles.

The Morris worm illustrated to the security personnel of the time the importance of the ability to detect new unknown threats before they can cause significant damage. The focuses of this work turned towards expecting new threats and creating mechanisms to better lockdown and protect hosts. It soon became apparent to the computer security field that it would be impossible to conceive what every new threat would be; there are just too many possible avenues of attack. After much research it was determined that the key for successful protection against an unknown threat is through reduction of the time it takes for the identification and response to an unknown threat after manifestation.

The research of thesis focuses on reducing response times and decreasing the time it takes to classify an unknown attack. In addition to the decreasing identification time, the unknown threat detection system will ensure that while it identifies unknown threats, normal conditions are not identified as threats (a false positive) and that a threat is not classified as normal traffic (a false negative).

This thesis will investigate the background and issues that are relevant to the network security domain as well as the techniques proposed in the development of a distributed monitoring system. Then the aims of the research system are described with reference to the background and the issues pertaining to the domain. Following this the methods used to implement and test the research system will be explored along with the analysis and discussion of the results produced. Finally conclusions will be drawn along with the proposition of possible future extensions to the research system.

2 Background

2.1 Networks

A network is a series of devices that interact through a common communication scheme. The Open Systems Interconnection (OSI) is standard reference model used to describe the processes involved in the communication between two nodes in a network (Mueller 2002). This reference model breaks the functions of network communication into seven, self-contained, abstract layers. Each layer in the OSI reference model relies on the functions and services provided by the layer/s below it.

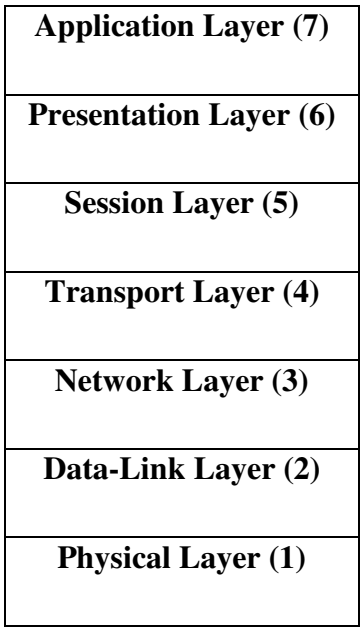

Figure 2-1: The OSI Reference Model (Mueller 2002)

2.1.1 Data Communication

In order for data to be transmitted on a network it must use an encapsulation process. Firstly data is broken up into small pieces called datagrams. Then it is passed through the equivalent OSI reference model layers, with a number of layers adding headers that contain useful processing information. When the process is complete the final encapsulated data, called a packet, has all of the necessary information for transmission over networks.

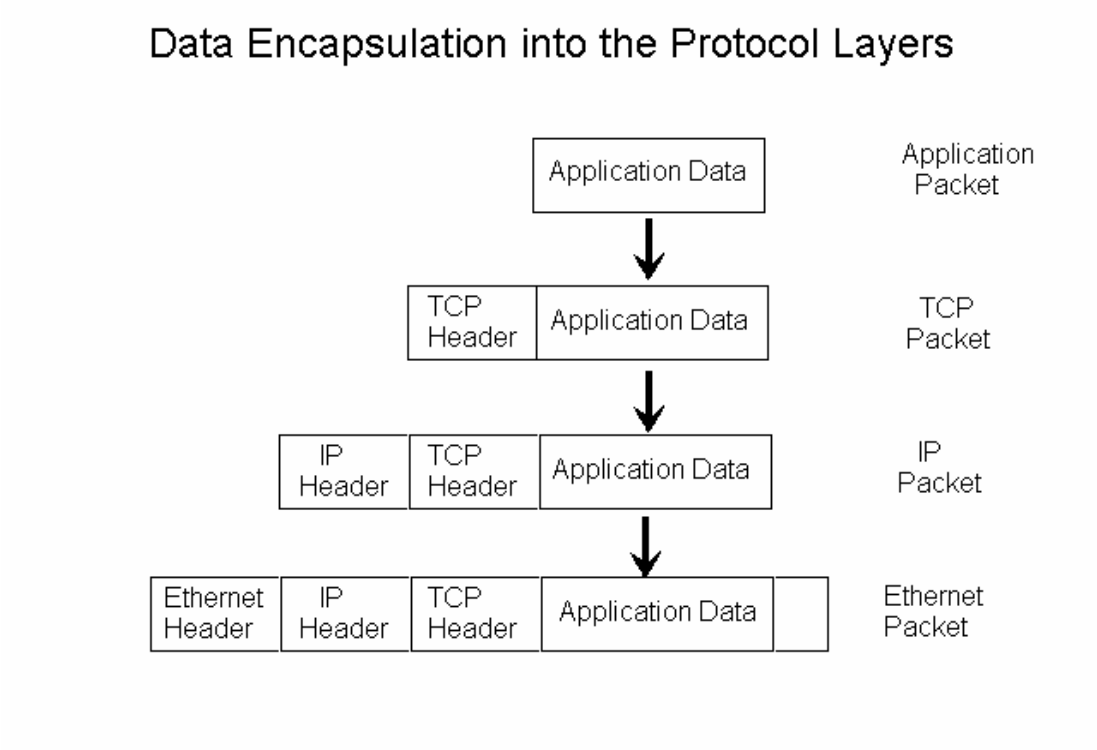

Figure 2-2: The Data Encapsulation Process (Computer Technology Documents)

2.1.2 Connection-Oriented and Connection-Less

Communication between two hosts on a network occurs in two modes; with established connections and without. In connection-less communication the source host just addresses the packet to the intended recipient without prior notification. When the packet arrives at the destination the recipient host understands, through protocol identification, that no logical connection is required and processes the packet immediately. An example of a protocol that uses connection-less operation is user datagram protocol (UDP). In connection-oriented communication the source host must first establish a logical connection, or session, with the destination host before transmission of data occurs. The source host and the destination host undergo a handshaking procedure after which, if successful, a session is established and data transmission proceeds. A protocol that uses connection-oriented communication is the transport control protocol (TCP). Figure 2-3 shows the handshake procedure defined for TCP.

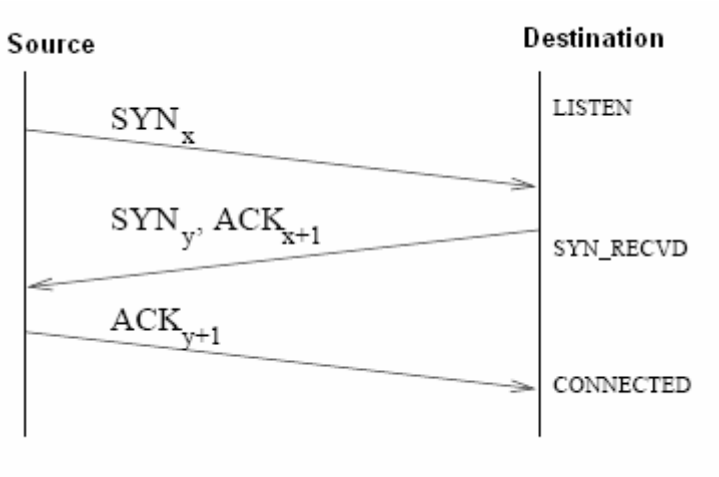

Figure 2-3: Three-Way Handshake Process (Schuba et al. 1997)

2.2 Network Security

2.2.1 What is Network Security?

Security, in a general context, is defined as *"…a state of freedom from danger or risk"* (Ciampa 2005). When applied to networks, security becomes the ability for data transmission and access to system resources free from danger or risk. The resources requiring protection within a network are dubbed its assets; this includes resources such as a printers, computers, or data. The properties of the assets that need protection are the confidentiality, the integrity and the availability of the assets (Pfleeger & Pfleeger 2003).

The confidentiality of the information is the assurance that the information will only be obtainable by parties deemed authorised to access it. This ensures that information will not be disclosed to people who have no rights to access it. An example of confidentiality protection would be to ensure that competing companies does not get access to an organisation's research plans. The term access covers not only viewing the material, but printing, copying, or even the knowledge of the existence of the protected material.

The integrity of an asset is the assurance that the asset has only been modified by people authorised to do so. In this context, the term modification refers to the writing, changing, altering status, deleting and creation of false information.

The availability of an asset is the timely access by authorised parties to the asset. If a legitimate party attempts to access an asset then that party should have access as soon as physically possible. If an authorised person is denied access to certain time-critical information for a certain period of time then it can affect the usefulness of the information. An example of time-critical information is the current stock prices.

If the network assets have these three critical properties protected then it can be deemed secure. However security is not the assurance that all attacks on assets will be prevented rather it is a defence that attempts to ward off attacks and prevent the collapse of a system in the event of an attack (Ciampa 2005).

2.2.2 What makes attacks possible?

Network and Internet, attacks are successful because of software bugs (errors) and vulnerabilities that exist within the various applications and protocols in use on the network. Each attack will exploit one or more known vulnerabilities in order achieve its malicious goal. The solution to this problem seems obvious; create software that has no software bugs or vulnerabilities. Many companies have tried to create bug free software through expensive testing and software validation procedures however this has proven to be impossible. As Schneier (Schneier 2004) states *"Squashing software bugs that affect performance is hard; finding software flaws that affect security is even harder."* (Schneier 2004, p. 204)

There are a number of examples over the years of multi-million dollar software developments failing catastrophically due to software failures. The following is just one of the many software error induced disasters.

On June 4, 1996 the unmanned rocket named Ariane 5 exploded forty seconds after lift-off. The rocket was developed over a decade by the European Space Agency at a cost of $7 billion. The resulting enquiry into the disaster found that the cause of the explosion was due to a software error. During lift-off a 64 bit floating point number

was converted to a 16 bit signed integer. A number, that was larger than the largest possible integer, failed in the conversion process which resulted in the rocket veering off course and exploding (Arnold 2000). If the European Space Agency, even after so much funding and development time, still has software bugs at launch time then how is a normal organisation with a restricted software development budget supposed to create error free software?

The CERT Coordination Center (CERT/CC) has compiled statistics on vulnerabilities discovered from various organisations that report to it. Figure 2-4 shows the reported vulnerabilities over the previous decade. The figures displayed are best case due to the fact that not all companies report vulnerabilities to CERT/CC. The reason that many companies do not wish to report and publish all of the vulnerabilities of their systems as it may impact on consumer confidence and hence profit margins.

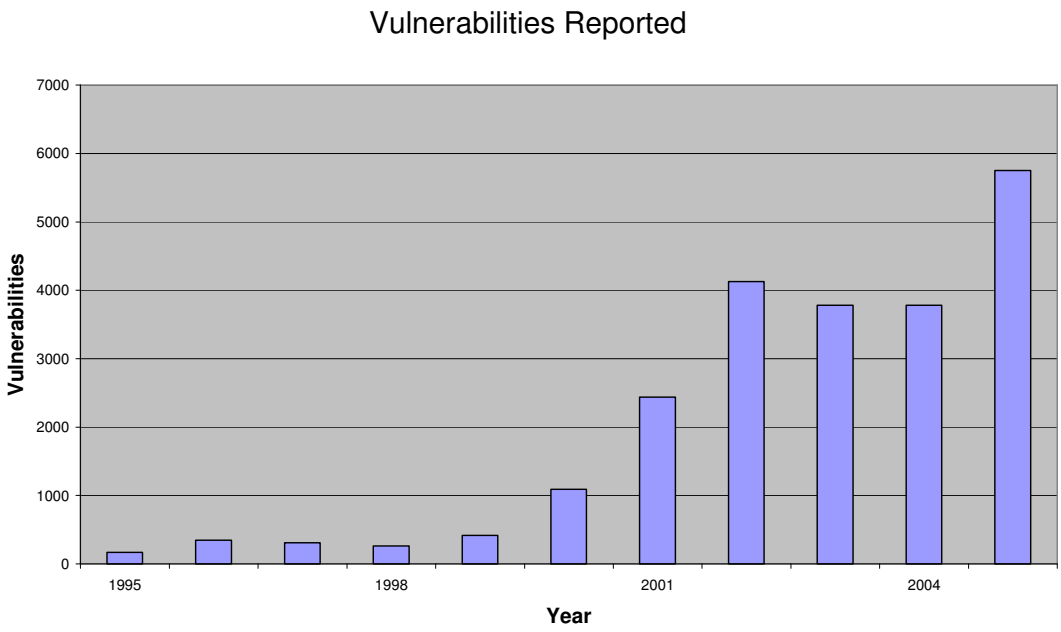

Figure 2-4: CERT/CC Reported Vulnerabilities (CERT Coordination Center 2005)

Figure 2-4 shows that the number of software vulnerabilities has been steadily increasing over the last decade. This is due to software and protocols becoming more complex; adding support for more new features and functionality as well as interactions with increasingly complex data structures. When vulnerability is discovered it is normal practice for the responsible company to produce a correcting

patch. The patch will fix the reported problem however it is possible that this patch may fix the original security hole while creating a new one.

Malicious people create attacks that exploit these vulnerabilities to penetrate unprotected hosts. CERT/CC has been keeping track of the number of security incidents reported to them by various corporations. Figure 2-5 shows the increasing trend in the number of incidents reported over a period of sixteen years.

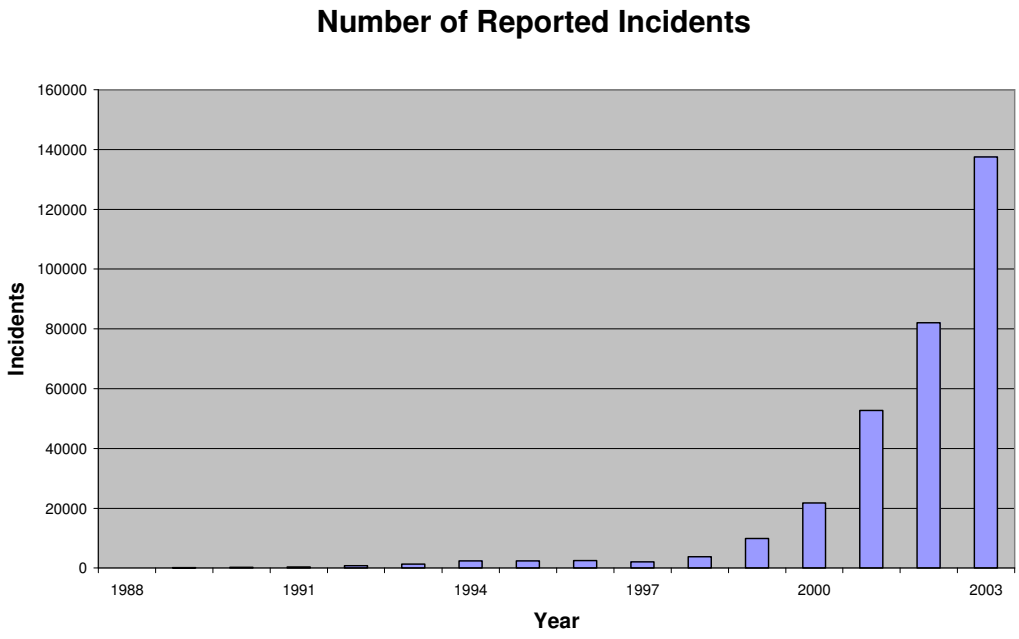

Figure 2-5: Reported Incidents (CERT Coordination Center 2005)

As can be seen there is a direct relationship between the increasing number of reported vulnerabilities and the number of reported security incidents. Other factors that increase the amount of security incidents are anonymity, ease of malicious code distribution through the Internet and ease of threat mutation.

2.3 The Threats

The term threat has a number of interpretations depending upon the context in which it is used. In computing the term threat is defined as any action, malicious or otherwise, which can have an undesirable effect on the assets and resources within a computer system (Amoroso 1994). There are many different types of threat that exist

within the computing field; the following sections will broadly investigate a subset of the computing threat matrix.

2.3.1 General Intrusion

An intrusion is where an unauthorised person/s attempts to gain access to system resources that they are not permitted to access. Typically this threat occurs from outside of the network and uses the organisations outside connection/s as the means of entry. The other form of this threat, which has been steadily increasing, is the internal intrusion. Internal intrusion occurs when disgruntled or curious employees attempt to cause damage by using their legitimate network access. The disgruntled employee wishes to disrupt the normal operations of the organisation while the curious employee wishes to investigate the possibility of disruption. Internal intrusions are typically successful as the employee will be trusted within the network.

The intruder, whether internal or external, has many possible tools in his/her arsenal to achieve maximum possibility of success. One of the simplest, and typically the most effective tools is the use of social engineering (Dolan 2004). This is where the attacker uses social skills in order to elicit information from a person for the purposes of compromising organisational assets. The attacker can call the helpdesk posing as a legitimate employee who has lost their password. The unsuspecting support staff member will divulge the information to the attacker assuming him/her to be legitimate. Another example of the successfulness of this technique is when an attacker poses as maintenance personnel in order to gain insider access. Typically no-one will challenge an appropriately dressed cleaning staff. With this inside access the would-be cleaner can then gain access to the machines through lax security such as username/password combinations written on sticky notes attached to monitors.

In addition to the elegant and simple process of social engineering the intruder can use many other techniques including brute forcing passwords, stack smashing, and using protocol and software vulnerabilities to name a few. The main principle of this threat is to gain access to the network assets using any means possible and from there achieving some set malicious goal.

2.3.2 Denial of Service

The way in which the class of Denial of Service (DoS) threats operate are through the abuse of a particular service in order to disrupt or prevent normal system operation (Cheswick, Bellovin & Rubin 2003). The abuse of the service involves extreme overuse of a service in an attempt to strain software, hardware or network links to beyond intended capacity. DoS attacks have the effect of degrading the quality of service that a system or network link can provide to a point where it can no longer function. SYN flooding is an example of a specific DoS attack designed to attack Transmission Control Protocol/Internet Protocol (TCP/IP) networks (Schuba et al. 1997). It operates by sending a flood of TCP connection requests to the victim machine from a number of different impersonated (spoofed) addresses. The victim machine will then reserve system resources for each connection request until it runs out of resources. Thus denying any further connection requests from legitimate hosts. A recent mutation of the DoS threat is the Distributed Denial of Service (DDoS) attack whereby a number of zombies (attacker conscripted hosts) all DoS attack at once.

A well orchestrated DoS attack, such as the DDoS attack, can bring a target organisation network to its knees. The risks from this threat are very real as there are numerous variants of this type of threat and very little that can be done to stop them. As Cheswick (Cheswick, Bellovin & Rubin 2003) states *"Any public service can be abused by the public."* (Cheswick, Bellovin & Rubin 2003, p. 111)

2.3.3 Viruses

A virus is a program that passes on malicious code to other programs by modifying them and attaching themselves to the host program. The term 'virus' came from the malicious programs behaving similarly to biological viruses: it infects other systems by attaching itself to a host program and incapacitating it or coexisting with it (Pfleeger & Pfleeger 2003). In order to be highly successful at propagation the virus needs to spread before destroying an infected host, otherwise this instance of the virus dies with the host; thus reducing the chances of mass infection. The first ever computer virus strain, Brain, appeared in the middle of the 1980s (Peyton 2003).

These original viruses were more mischievous rather than malicious; they would play sounds or create pop-up windows. From these rather innocent beginnings the computer virus has evolved to become, in terms of damage and likelihood of attack, one of the most significant threats that exist to computers to date.

The major factor that makes viruses so virulent, just like the biological counterpart, is the ability of the virus to mutate. When it is first unleashed a virus will have certain characteristics and traits that will enable anti-virus detection and subsequent inoculation. However at any point during its lifetime the virus can mutate and exhibit new characteristics that will appear to an anti-virus system as a completely new unknown virus (if it is able to detect the mutated threat at all).

Dangerous as they are viruses tend to propagate slowly due to their reliance on human interaction. Users unwittingly spread viruses by running or copying infected files to the computer. A number of viruses use e-mail as a propagation method due to the significant user base of e-mail. An infected program is attached to an innocently titled e-mail and sent to numerous unsuspecting recipients. The recipients of the 'infected' e-mail are tricked into accessing the attachment; the computer becomes infected with the virus and the cycle perpetuates. However there is a new class of viruses called Worms that have the ability for mass infection through self propagation.

2.3.4 Worms

A Worm is a newer type of virus that spreads utilising its own means. That is, unlike a normal virus, a Worm requires no human intervention in order to propagate. This distinction between viruses and Worms is not sharp as there is a class of Worms called stealth or contagion Worms that hide signs of infection in otherwise innocuous user activity (Staniford, Paxson & Weaver 2002).

To achieve propagation a Worm must discover new active hosts to infect. A Worm can discover new victims through a variety of mechanisms (Weaver et al. 2003). The author of the Worm can create a pre-generated list of target hosts which he/she has identified in advance. Host discovery can also occur through the Worm extracting lists of hosts from a matchmaking service's (such as the Gamespy service)

metaserver. This metaserver contains a list of hosts and servers that are currently active. The active host lists significant aid the Worm in its propagation efforts. Another host discovery mechanism the Worm can use is a topological search of the current victim machine for local information. The Worm can examine the machine for local topology information that is contained in protocols, such as Network Yellow pages, and files, such as the hosts contained within /etc/hosts (Linux operating system file). This type of examination is quick and effective as these active hosts are directly accessible through the local network medium. The last method that a Worm seeking active host can use is through scanning. Scanning is where the Worm probes a set of address, sequentially or random, to identify active hosts. Due to the simplicity and effectiveness of such a strategy a number of Worms use this technique for active host discovery. Once an active host is discovered the Worm identifies the open ports on this host. The Worm will then mobilise and infect this system by using any of the software vulnerabilities contained on the victim machine. From there the entire process repeats until the Worm is finally discovered and neutralised.

The damage that a Worm can cause through automated self-propagation is significant. On July 19th, 2001 the Code Red Worm (version 2) began its vicious debut on the Internet. It was designed to scan for and infect active hosts until 00:00 UTC on July 20th, 2001. In just one day of infection the success rate of this Worm was incredible. Figure 2-6 shows just how effective Code Red's infection rate was.

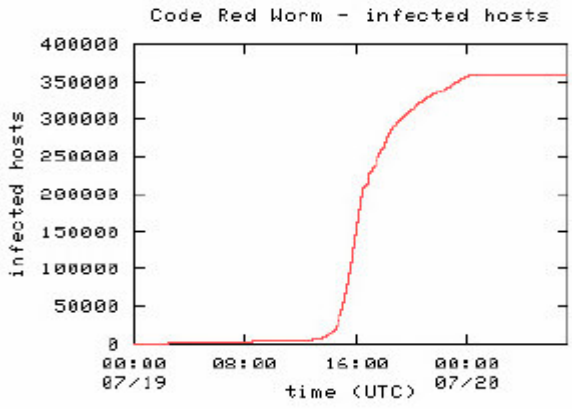

Figure 2-6: Code Red V2 Worm Infection Rate (Zou, Gong & Towsley 2002)

Other significant Worm infections have followed Code Red's success such as the SQL Slammer Worm which infected 90% of all vulnerable machines on the Internet

in ten minutes (Moore et al. 2003). The rate of infection was so fast that human intervention to stop it was impossible. Due to an implementation error the payload of this Worm was benign; however the number of scans that the infected hosts created was enough to disrupt functioning of the entire Internet. If the payload of this Worm was malicious what would the Internet landscape be like today?

From the observations of just two of the many Worm variants it can be seen that the risks associated with type of threat are significant. If there was a way to detect and stop any unknown Worm at the first few infected host the dangers would be greatly minimised.

2.4 The Threat Counter Measures

A number of different types of security mechanisms have been created to counter the numerous threats that users are exposed to when communicating with the 'outside' world. Each of these mechanisms varies in terms of protection strength, ease of implementation, maintenance and complexity however each the goal of providing a measure of computer security. The security mechanisms are most effective when deployed together, working in unison in an attempt to provide an all encompassing blanket of security. This 'defence in depth' provides better security because there are no single points of failure. If one system fails then the network's defence falls to the next protection mechanism that has been deployed.

Another consideration required when deploying threat counter measures is the deployment strategy used. There are two strategies used when deploying the security structures are perimeter security and host based security.

2.4.1 Perimeter Based Security

In perimeter based security the strategy is to defend the entire network by protecting the access to the network through guarding the main entrance from the 'outside'. The definition of the outside is any external network connected to the network that is not under administrative control of the organisation. This definition of 'outside' covers other organisational network connections, say to partner companies, and the Internet

connection/s. In this protection scheme the main focus is to strengthen the walls and the doorway to the organisation's network. The principle is that if the perimeter is tightly secured and regulated then any attempted outside access will be screened and only legitimate connections will be made. Figure 2-7 demonstrates this principle.

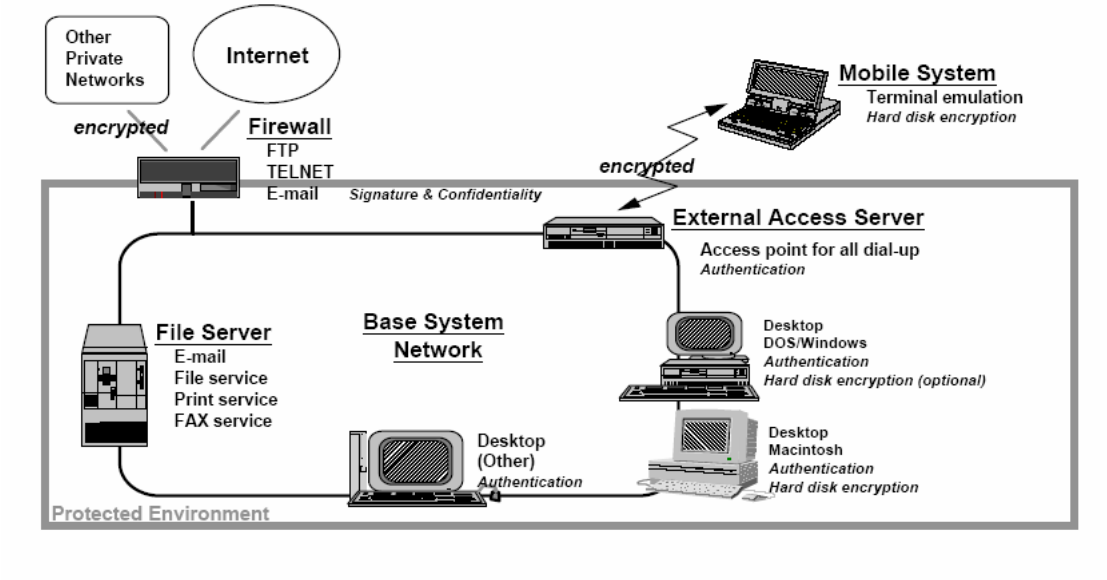

Figure 2-7: A Network using Perimeter Security (Avolio, F. M. & Ranum 1994)

As can be seen all outside access will pass through the 'guards' of the network, in this case a main firewall and a dial-up external access server (Khoussainov & Patel 2000). The main benefit of using this strategy to secure the network is the ease of administration of the security system. The monitoring logs and security configuration can be accessed from a few locations on the network.

In the early days of network security this protection scheme worked well; the external traffic is filtered and internal network is protected from unauthorised outside access. However over the years this scheme has proven less effective due to a number of reasons.

Firstly all data passes through only a few systems setup to analyse all of the network traffic. This creates a significant burden on the guard systems as the link to the outside networks are of considerable size to allow fast access from any internal host. This means that the amount of external access will be limited to the speed at which the guard devices can process network traffic. Also, any addition of extra filtering

logics will compound the issue as new rules will now have to be applied to all of the traffic.

If any of the network guards fail then the organisation will have to stop external access while this system is repaired or allow unprotected external access. Both options are not attractive to an organisation as either way it will lose productivity through non-access or when an internal host gets compromised.

In perimeter based security each of the internal network hosts is implicitly trusted. The idea is that the threats to the network only come from the outside. Although this may be the case in many networks it allows the creation of a dangerous scenario. A situation may arise where an intruder has gained access to the network and penetrated a host. This intruder may now use the transitive trust that exists within this network and jump to each host without challenge.

In addition to this the scheme purports to provide network security by creating a stronger doorway and a larger wall. However as Cheswick (Cheswick et al) points out *"…the attacker only has to win once. It does not matter how thick are your walls, nor how lofty your battlements; if an attacker finds one weakness…your system will be penetrated"* (Cheswick, P11). If an attacker does succeed in getting through the defences then, with a bit of skill and knowledge, it will be possible for the attacker to gain repeated access to the penetrated network without alerting anyone that this situation has occurred.

The thesis work acknowledges the significant issues with this network protection scheme and looks to further develop network security by utilising the features and functionality of the next security strategy; host based security.

2.4.2 Host based Security

The host based security strategy uses the principle that a secure network is one where every machine manages its own protection. This takes the concept of the perimeter (see 2.4.1 Perimeter Based Security) and reduces its scope down so that it surrounds each individual host. The host based strategy ensures that the operating system activities within each host are taken into account when making security decisions

(Yeung & Yuxin 2003). Where perimeter based security attempts to protect the network through securing the external connections and trusting each internal host implicitly; host based security employs a more paranoid view point to trust. Each host, within a host based security strategy, is analogous an island in an archipelago; they are all within the same cluster but each host (island) is isolated from the others. The isolation of each host alters the concept of 'outside', or non-trusted areas, to anything outside of each individual host. Using the host based security strategy can account for the insider threat and attacker penetration because of the reduced usage of transitive trust within the network.

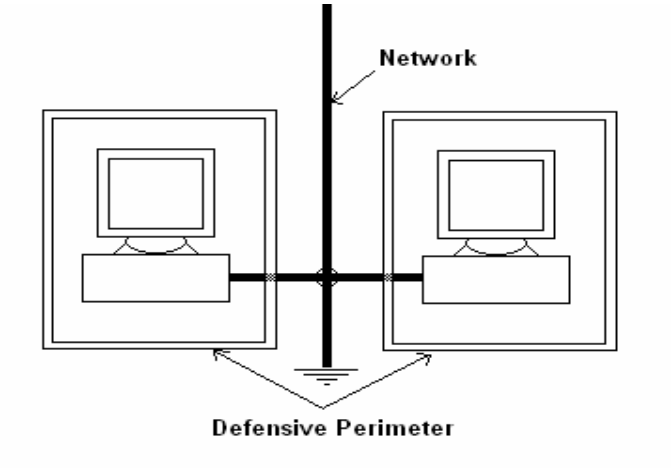

Figure 2-8: Host Based Security Strategy

The host based security strategy is not without drawbacks; there is a greater amount of administration due to each host being an individual security system. Another issue with the host based approach is that the security system/s used need to be scalable to cater for a significant number of hosts. This being the case the security benefits it provides is significant. Firstly the security configuration within each host cluster can be modified to better fit the host groups that are running the security system/s. Also more information can be taken into account with the security system/s being able to access all the pertinent host data directly. The host based security strategy is appealing for this research because no implicit trust is used and that is more fitting to an individual data profiling approach.

The security strategies mentioned previously are just that; strategies. The strategies require mechanisms for actually implementing the security functions. The following

is an investigation into the main components that provide computer security within either strategy.

2.4.3 Firewalls

A firewall is a hardware or software system that filters all traffic between multiple connected areas. The term area in this context can be used to denote networks, if applied to perimeter security, or the area internal to the host when applied to host based security. One of the connected areas is deemed the 'inside' region; the area that requires protection and can, to a high degree, be trusted. The other area is defined as the 'outside' region and has restrictions on the information that can arrive from this source. The main goal of a firewall is to ensure that only authorised traffic is allowed to enter the 'inside' area from the 'outside' area; the rest is filtered out.

In order for a firewall to be effective it must possess the following three qualities: always be invoked, be tamper-proof, and to be small and simple enough for thorough testing. Firewalls are an enforcement of a security policy and as such will not enforce the policy correctly if the users can bypass it, can modify the firewall code, or if the firewall code is complex; making it susceptible to hidden software bugs. (Pfleeger & Pfleeger 2003)

There are three types of firewalls in existence: packet filtering, application gateways and circuit-level gateways. Depending upon the security strategy that the organisation applies will determine the types of firewalls in service on the network.

2.4.3.1 Packet Filtering

These types of firewalls act at the network layer of the OSI reference model (see 2.1.1 Data Communication) and look at the headers of each packet to determine whether it requires filtering. Packet filters do not look at the data that a packet contains, rather it bases all filtering decisions using the addresses and various header options of each packet. This type of firewall comprises of a number of rules that are designed to best represent the decisions of an organisation's network policy (Avolio, F. 1999). The simple design of this firewall type means that it can be applied to both the perimeter based and host based security strategies. In perimeter based security this type of firewall would be typically built into the organisation's routers. This is

due to efficiency as all routers are required to look within each packet's header to make routing decisions. So the packet filtering firewall module does not create much overhead to the router's operation.

2.4.3.2 Application Gateways

Application gateways, as the name implies, act at the application layer of the OSI reference model and examine the application data within a packet stream. The application gateways operate by reassembling and analysing the application streams as opposed to observing individual packets. Application gateways base filtering decisions on application data such as a command set from a specific program, for example FTP commands (Ciampa 2005). Due to the processing required in rebuilding application data streams the application gateway is mainly used in perimeter based security on a separate, dedication machine. However as desktop power within an organisation is ever increasing it is becoming more feasible for application gateways to reside on each host.

2.4.3.3 Circuit Level Gateways

Circuit level gateways or proxy servers operate at the session layer of the OSI model. A circuit (or session) is a logical connection that is established between two machines for the duration of the communication process (see 2.1.2 Connection-Oriented and Connection-Less). This type of firewall verifies that the circuit is authorised upon initialisation and then establishes the connection on behalf of the communicating host (communication by proxy), thereby filtering the connections that can be made through the gateway. Circuit gateways operate on the principle that packet header forging is easy and prevalent and so shift the analysis and filtering up to a higher abstraction level; the session layer (Al-Tawil & Al-Kaltham 1999). The circuit-level gateway is typically only effective if all internal hosts use it as the proxy to the outside world. As such this firewall is typically implemented on a separate machine in the perimeter based security strategy. If each host contained a circuit-level gateway then it would defeat the purpose of this security mechanism; the host would be acting as a proxy for itself.

2.4.4 Intrusion Detection Systems

Amoroso (Amoroso 1999) defines the term intrusion detection as *"…the process of identifying and responding to malicious activity targeted at computing and networking resources."* (Amoroso 1999) This definition covers not only intruders who are attacking from the outside of a network but also the abusive actions of internal users (see 2.3.1 General Intrusion).

An intrusion detection system (IDS) is a device that monitors the activity of a network in an attempt to identify malicious activity. An analogy of the operation of an IDS is that of a burglar alarm; to alert in the event of an intrusion to a secure area. Once an IDS has signalled an alert there must be an entity to respond to the alert for the system to be effective. The responding entity can be a security expert who manually intervenes in the event of an alert, a passive IDS, or it can be an automated response module contained within the IDS itself, an active IDS (Janakiraman, Waldvogel & Zhang 2003).

Originally intrusion detection was achieved via a system administrator observing a console that displayed user activities. The next evolution of intrusion detection was through the manual sifting of printed log files. When the storage of electronic information became cheaper, the natural progression of the log file to computer memory occurred. Prior to this the scanning of the information was not done in real-time and was only a means to detect an attack after it had occurred. To improve the level of security provided by the IDS mechanism, researchers in the early 1990's developed real-time log analysis, which enabled attack detection as it happened. Current development into IDS technology has been directed towards the distribution of the intrusion detection entities to provide a larger network-wide context (Kemmerer & Vigna 2002). In terms of security strategy the IDS can be used in both host based and perimeter based security. The IDS originally only appeared in the perimeter based security strategy however as the desktop has increased in power it has begun integration into host based security strategies.

Generally an IDS can be classified into two types: signature based and anomaly based. The signature based IDS is less complex than the anomaly based IDS and so it can be implemented successfully in both perimeter based and host based security

strategies. The anomaly-based IDS is complex and processor intensive and is typically seen in the perimeter based security strategy.

2.4.5 Signature-Based Intrusion Detection

Whenever malicious activity occurs on a network it uses a specific process or method to execute its attack. This activity exhibits a pattern, or fingerprint, that is unique to each individual form of attack. The pattern shows itself in various ways, such as a port scan to a range of hosts (Lorimer 2003) or modified TCP packets that are distributed throughout a victim network. A signature-based intrusion detection system is designed to detect intrusions through the comparison of current network data against known, documented malware patterns. When a threat pattern match occurs the IDS will alert the system administrator that malicious activity is occurring along with information describing the nature of the attack.

The main issue when deploying a signature-based intrusion detection system as part of perimeter based or host based security strategy is with the use of signatures. An attacker can and will modify an existing attack so that its new signature will differ from the original to such a point that the IDS will no longer identify it as malicious activity (Pfleeger & Pfleeger 2003).

Another problem with signature-based intrusion detection is that it can only match malicious activity with known forms of attack. It will not be able to detect new, unknown attacks with undocumented attack patterns. In effect the attack has to be successful on at least one victim network before a solution is produced. To minimise this problem constant maintenance of the signature database is required; a potentially costly exercise in dollars as well as the network administrator time. Also how does the network administrator react when he/she finds out about a new threat and the subsequent patch/update has not been created? Patch creation is a complex process, which can take a long time.

Ideally this system, to have maximum effectiveness, should match known attacks, activity that exhibits slight variations on existing patterns, as well as be able to

discriminate attacks from normal traffic on the network. Due to the rigid nature of the signature-based IDS organisations are tending towards using anomaly-based intrusion detection approaches

2.4.6 Anomaly-Based Intrusion Detection

In anomaly detection the IDS creates a profile of acceptable usage for the users. The user profiles that are developed are used for comparison with current network data to detect abnormal, potentially malicious, traffic. The construction of this type of detection mechanism begins by determining what is normal for the observed user. The system then needs to determine the thresholds of each activity to flag as abnormal, and which activities to prioritise security decisions on. The detection principle for such a system is that it flags behaviour that is unlikely to originate from normal user activity, without any regard to actual intrusion situational data (Axelsson 2000).

The issue with using the anomaly-based intrusion detection approach is the problem of identifying correct, normal behaviour. The profiling requires that the user complete all the tasks that they would normally undertake over a protracted period. This being the case it typically takes a significant amount of time to develop a profile for each user as well as to extract the key features from the data. For the profiling to be truly successful in identifying the normal behaviour the user has to know in advanced all of the applications that they will be using as well as any other task that they would undertake, such as file transfers and database access.

2.4.7 Anti-Virus Solutions

An anti-virus program is a process that attempts to eradicate viruses by identifying any viral behaviours occurring on the system and then extracting the malicious code attachments. Anti-virus software operates in two ways: signature based and heuristic based.

When a computer virus is reported the virus is examined by anti-virus software developers and a signature file is created to be used for detection and eradication purposes. The virus signature file is added to the anti-virus software database and is

used for viral identification when a computer is later scanned. This method of viral detection is always reactive; someone must be infected and report the infection before a signature file of the virus is created. This is a significant problem because if the virus has a particularly effective means of propagation then by the time the corrective patch and signature is created the organisation may have already been infected. In addition, with signature-based anti-virus, if the signature database is not up to date then it will be, in practical terms, useless because it will not provide protection for recently identified viruses.

The other technique employed by anti-virus software for viral identification is to use heuristic based mechanism. Heuristics are rules of thumb gained from experiences that act as an intelligent guide (Giarratano & Riley 1998) for the anti-virus software that is monitoring the host. The anti-virus software uses the heuristics to determine whether a program is exhibiting viral activity and if so attempts to eradicate the virus. An issue with this heuristic technique is the determining that virus activity occurring mainly due to the fact that heuristics act as guides rather than provide a guarantee of virus detection. Another issue is that applications will have features that are similar to the activities of a virus (eg. replication). This can cause problems with numerous false alarms which can train the user to ignore false alarms as well as positive virus results (Surfer Beware 2004).

After investigation into networks, security, threats and their counter measures it becomes obvious how dynamic the entire domain is. Dynamic network traffic, security strategies implemented, the constant creation of new threats, and mutations of existing threats ensure that any inflexible security monitoring system will be of little use. The underlying technique used within a monitoring system must be flexible and adaptable to match the highly dynamic nature of the domain. Expert system techniques have been developed to solve solutions that are dynamic in nature. Expert systems can solve problems in domains where conventional techniques fail.

2.5 *Expert Systems*

Expert Systems are field within of artificial intelligence (AI) that uses specialised knowledge in a particular domain in an attempt to solve problems at the level of a human expert. The knowledge in expert systems may be either expertise or knowledge that is generally available from knowledgeable people as well as from other sources, such as academic reference in the domain. An expert system typically consists of the knowledge base, the working memory, the inference engine and some type of user interface (Luger 2002).

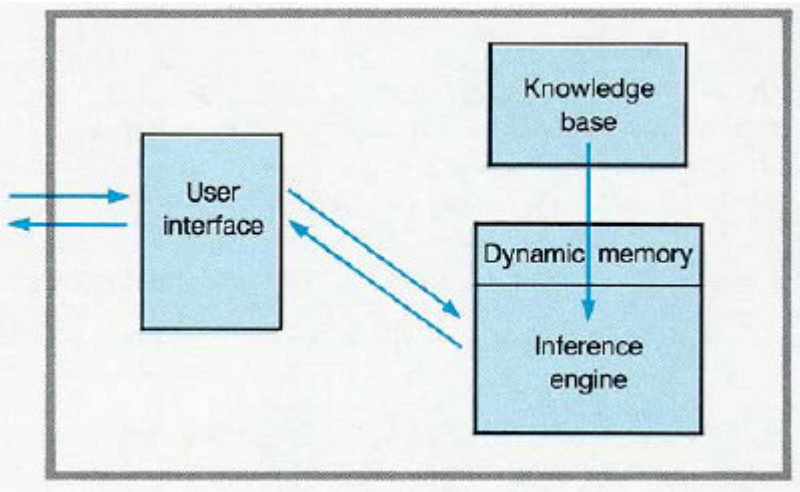

Figure 2-9: The Expert System Structure (Gallant 1988)

The knowledge base is the expert knowledge of a particular domain that is semantically structured in a form for the specific expert system. The working memory is used for storage of the artefacts derived from the reasoning process. These artefacts are asserted facts and temporary conclusions that have been deduced by the inference engine. The inference engine applies the situational knowledge to the solution of actual problems. The actions of the inference engine are analogous to that of an interpreter for the knowledge base (Luger 2002).

2.6 Rule Based Expert System

One of the more popular types of expert systems in use today is the rule-based expert system. The rule-based system represents its expert knowledge in the form of a structured sequence of rules. A rule is data that generally conforms to highly specialised grammars capable of using symbolic representations to define the conditions and actions (Hayes-Roth 1985).

These rules are expressed in the form:

if <conditions> then <actions>

Where if the conditions are true then the actions are executed (Griffin & Lewis 1998). The rules are used by the inference engine to determine the output or conclusions that the system should produce given the situational data.

The rule-based approach is popular due to a number of advantages that this form of expert system presents. The modular nature of the rule-based approach means that it is easy to encapsulate knowledge and expand the expert system via incremental development. Each new conclusion that the rule based system produces is added as an extension of the structured sequence of rules. Another benefit of the rule-based approach is that it is easy to build in explanation facilities with the rules because the antecedents of a rule specify what the necessary elements are to satisfy the rule. By keeping track of the rules that have been activated, an explanation facility can present the chain of reasoning undertaken to produce a certain conclusion. In addition the rules are structured, as a listing of IF..THEN statements, which makes it easier to explain to the expert the structure of the knowledge that is required of them (Giarratano & Riley 1998).

2.7 Classical Knowledge Acquisition

As discussed previously (2.5 Expert Systems) expert systems utilise specialised knowledge in a specific domain in order to solve problems. As such, the knowledge from an expert source, whether it is a human of literature, needs to be extracted to provide the basis for the problem solver. There are a number of possible techniques

that are used to extract relevant information for analysis; a few of these are detailed below.

2.7.1 Decision Trees

Decision trees are a way to represent rules by underlying data with hierarchical, sequential structures that recursively separate the data. A decision tree can be used for data analysis to provide descriptions and classification of the data that is being analysed. The description of data involves the reduction of a volume of data by transforming it into a more compact form that preserves the essential characteristics and provides an accurate summary. The classification of data is achieved through discovering whether the data contains any well defined, separate classes of objects, such that the classes can be meaningfully interpreted in the context of a substantive theory (Murthy 1998). Decision tree classifiers have been successfully used in many diverse fields such as radar signal classification, character recognition, remote sensing, expert systems and speech recognition. A benefit of using decision trees is their capability to break down a complex decision making process into a collection of simpler decisions, providing a solution that is often easier to interpret. Advantages of the use of decision tree based classification have been demonstrated (Murthy 1998). Knowledge acquisition from pre-classified examples can alleviate the bottleneck of knowledge acquisition from a domain expert. In addition to this the decision trees perform classification by a sequence of simple, easy to understand tests whose semantics are intuitively clear to domain experts. The main drawback of decision tree classification is that within decision tree classification there is a trade off between optimising efficiency and accuracy; for any given accuracy there is a bound on the efficiency (Safavian & Landgrebe 1991).

2.7.2 Neural Networks

Neural networks, also known as connectionist systems, steer away from the use of symbols in problem solving. Instead the intelligence aspect of neural networks arises from the simple interactions of the components through a process of learning by which the connections between these components are adjusted. The implementer of a neural network must create an encoding scheme that translates real world

information into numerical quantities within the network. The choice of encoding scheme will determine the eventual success or failure of the network to learn the solution to the problem. Neural networks have been applied to a number of tasks such as classification, pattern recognition, memory recall, prediction, optimisation, and noise filtering (Luger 2002).

The use of neural networks in the field of pattern recognition has been promising. Wiener (Wiener, Pederson & Weigend 1995) successfully demonstrated a neural network that has been used for topic spotting within a large data set. It demonstrated a number of advantages for pattern recognition that other techniques would not have provided. The system that was developed, from a practical standpoint, could be put in place relatively quickly when compared to hand-crafted systems. In addition this system was flexible and would adapt to the data environment that it was placed in.

However a drawback to using neural networks systems is that once they present a solution to the problem there is no discernable rule linked with the data that is extracted. The extracted data is held in a machine readable, or neural network, format that is of no use to a human expert (Wiener, Pederson & Weigend 1995).

2.7.3 Data Mining and Machine Learning

Data mining is defined as the process of discovering patterns within a data set. In data mining, the data is electronically stored and the searching is automated by a computer process. The concept is to build computer programs that sift through data archives automatically, seeking regularities or patterns. The patterns that the data mining discovers are meaningful and provide some advantage, normally an economical advantage (Witten & Frank 2000). The patterns that are found are used to make predictions about any new data obtained.

Learning has many philosophical descriptions, such as to gain knowledge or an understanding of a topic of interest. The terms ‘machine learning’ describe the process of a machine changing its underlying structures or data so that it’s expected future performance improves (Nilsson 1996). Machine learning and data mining are intrinsically tied together by the common goal of increased performance. In data mining the performance goal is the ability to make better predictions based on the

patterns that have been determined. Applications of data mining and machine learning have included selecting embryos for fertilisation, selecting cows to cull from a milking stock and analysing stock market figures.

Applying data mining and machine learning to any domain of interest has problems that need to be addressed. Data mining techniques work well at finding any patterns within a given dataset. Data mining finds patterns without regard to what the patterns are actually showing. There is a requirement by an expert to sift through the patterns and classify the relevant ones and discard the rest. Machine learning before they can 'learn' have a pre-requisite that the training data is pre-defined and pre-classified. This requirement limits the real-time possibilities of machine learning applications. An expert will have had to have analysed the data and classified it before it is inputted into the system.

2.8 Expert System Issues

Although expert systems have been proven to work effectively in the field, like MYCIN (Shortliffe 1976), these systems have exhibited various problems.

The process of formalising the knowledge of the expert, in a particular domain, into rules is not a simple, especially when the expert's knowledge has never been systematically explored. There may be inconsistencies, ambiguities, duplications, or other problems with the expert's knowledge that are not apparent until attempts are made to formally represent this knowledge in an expert system.

Another issue is that expert systems are explicitly designed to deal with uncertainty; they will make recommendations with the same confidence regardless of the accuracy or completeness of the data that it used to come to a conclusion. An expert system's advice, like that of a human expert, should degrade gracefully as it reaches the boundaries of ignorance; the limits of knowledge the expert system contains.

In addition many expert systems do not have facilities to represent causal knowledge. The expert systems do not have an understanding of the causes and effects in the underlying system. It is easier to program expert systems with shallow knowledge

based on empirical and heuristic knowledge than with deep knowledge as to the structures and objects within the problem domain (Giarratano & Riley 1998).

Expert system performance within their specialised domains is excellent, however hardly any of them have common sense knowledge; an ability possessed by most humans. This lack of common sense makes a lot of these systems difficult to extend beyond their original scope as the system does not know the extent of their limitations; their boundaries of ignorance (McCarthy 1984).

2.9 The Knowledge Paradigm Shift

The techniques described in 2.7 Classical Knowledge Acquisition are defined as the classic expert system approaches. Thorough applications of these systems have identified a number of issues in their use within particular domains. Each of these approaches do have significant benefits however only when the domains are restricted to a subset of all domains. For example Neural Networks have proven to be excellent in the games domain (Moriarty & Miikkulainen 1995) (Richards, Moriarty & Miikkulainen 1998), and data mining and machine learning in the domain of statistical analysis (Hastie, Tibshirani & Friedman 2001) (Glymour et al. 1997).

The concerns that arose from using expert systems were that there would be a need to use a specific expert system for any given domain. There was no general problem solver that could be applied to many diverse domains (Kang, Compton & Preston 1995). This lead to further investigation into the underlying representation of knowledge and understanding in the hope that if this is better understood then it could be applied to an expert system that covered many domains, no matter the structures of knowledge that the domain uses.

It was discovered that in previous expert systems the expert was required to model a domain using his/her experiences and adapting them to fit a common logic. This process was difficult because it was found that experts have trouble trying to model the domain using this rigid process. A number of key issues were found with this domain modelling process.

If the expert was interviewed by the expert system designer then the expert would adapt their explanation to fit the level of understanding of the designer. This is a significant problem because the true meaning of the domain information would be lost through this simplification process. It was found that the expert would do this unknowingly even though the interviewer explicitly asked for the correct knowledge in unadulterated form (Kang 2005).

Another issue was that the expert would typically have gaps in his/her knowledge and not know everything about the domain. When a situation arose where the expert did not have the required understanding then he/she would consult other experts or literature in order to fill these gaps. So in the end the expert would be able to provide a solution to all problems however the expert was typically unaware of these gaps until the specific unknown situation occurred. It is impossible to know everything that is unknown, the gaps, in any domain of interest (Compton et al. 1991).

Observations of the experts working in a domain revealed that the reason that the expert came to a certain conclusion was that he/she had reasoned off case information and had flexible representational logic in their mind that provided the conclusion. An example is a doctor diagnosing a patient, when the doctor hears the patient's symptoms the doctor will extract the key elements, use his/her experience to make a educated prediction of what the problem is, then extract case data that had similar matching symptoms to confirm his conclusion. The previous case data may not be identical to this current situation but the doctor is able to use experience to adapt the interpretations of the data or discard and select another case if it is not pertinent to the current situation (Compton et al. 1991).

The issues with expert systems, knowledge acquisition, and knowledge representation were becoming a limiting factor of expert system expansion in numerous domains. A new approach to expert systems and knowledge needed to be investigated to produce a system that worked over multiple domains (Compton & Jansen 1989). This research lead to the development of Case Based Reasoning and later on the Ripple Down Rule based approaches.

2.10 Case Based Reasoning

Case based reasoning is the process of reasoning using previous experience as a guide. Case based reasoning solves problems by using the experiences of the previous situation and retrieving the data that relates to that situation. The previous data is then used to identify possible problems with the solution being computed, interpret the new situation and create conclusions based off the past. The case data, once extracted, can also be used to find arguments that justify a conclusion, whether it is past or present (Kolodner 1991). The interpretations of the new situations are based off comparison and contrasting the new situation with the old ones and adapting them to fit. When a new conclusion has been made the situational data is added back into the memory to be used for future reasoning processes (Aamodt & Plaza 1994). The case based reasoning technique has been further developed through the creation of Ripple Down Rules and its successor Multiple Classification Ripple Down Rules.

2.11 Ripple Down Rules

Ripple Down Rules (RDR) is a knowledge acquisition method, based off case based reasoning, that attempts to constrain the interactions between the expert and the expert system shell to acquire only correct knowledge. RDR has come from the analysis of the way that experts attempt to provide domain knowledge for an expert system. Shaw (Shaw 1988) has demonstrated that experts may have quite different and inconsistent knowledge of a domain but are able to freely communicate knowledge with each other. Previous observation of experts during the maintenance phase of an expert system suggests that experts do not provide information on how they reached a specific judgement. Rather the expert tends to provide justifications that their judgement is correct. The problem with this is that the justification varies with the context and has to be engineered to fit in with the other knowledge that exists in the knowledge base (Compton et al. 1991).

The context of RDR is defined as the sequence of rules that are evaluated and have lead to a wrong conclusion, or no conclusion at all. When a rule with a new conclusion is appended to a RDR knowledge base, this rule is only evaluated after

the same sequence of rules is evaluated. The resulting structure is a set of sequence rules (IF…ELSEIF rules) containing exceptions which can themselves be ordered rules (Kang, Compton & Preston 1995).

The benefits of using of RDR include (Kang, Compton & Preston 1995):

The expert does not need to have knowledge about the structure of the knowledge and how the system attaches the rules to it. From the expert's viewpoint the rule is composed of whatever generality is preferred and the system handles the location of this rule. This is because all rule addition is prompted by the system misclassifying or failing to classify a case.

The use of RDR shifts the development emphasis of the expert system to the maintenance cycle by blurring the distinction between the initial development and maintenance stages. The difficulty of adding a rule to an RDR system is the same regardless of how long a knowledge base has been under development and how large it is. This feature allows the development of the knowledge base to evolve along with development of domain expertise.

A limitation of RDR is that the knowledge base may be ill structured due the experts initially adding rules that are too specific. Initially adding rules that are too specific result in numerous corrections rules which create an unbalanced RDR tree. In addition the RDR rule adding process is conducive to repetition of knowledge that already exists in the knowledge base.

Finally another problem with RDR is that there may be situations where multiple classifications may be required, for example if a patient has multiple independent diseases. The ripple down rule approach provides only a single interpretation, although this conclusion may contain a number of parts. A proposed solution to this is to produce separate rule trees. However this can lead to rule repetition because the information that is present in one tree may be required in the other domain tree (Compton et al. 1991). This problem was the driving force behind the creation of an extended RDR technique called Multiple Classification Ripple Down Rules (MCRDR).

2.12 Multiple Classification Ripple Down Rules

The aim of MCRDR is to preserve the advantages and strategy that has been developed by RDR and extend it to deal with multiple independent classifications. MCRDR, like RDR, is based on the assumption that the knowledge an expert provides is essentially a justification for a conclusion in a particular context.

The main benefit of using MCRDR as a knowledge acquisition strategy is that MCRDR can be the basis for the development of a general problem solver in used in many different problem domains, with multiple classification requirements. MCRDR allows the development of expert systems in a domain without having to pick the expert system that works best in that domain (see 2.9 The Knowledge Paradigm Shift). Another benefit discovered in the use of MCRDR is that it shows much greater complexity over that of its predecessor RDR without increasing the knowledge acquisition required for a single classification domain. The flexibility and usability of MCRDR means that it is the technique of choice for this research. The underlying processes and operations of MCRDR are further explored in the methodology section detailed later.

3 Methodology

3.1 Summation of Background Issues

The background provided through research of network security and expert system areas have laid the foundations for understanding the techniques that are needed and used with implementing a distributed security monitoring system. The following is a discussion of the issues that are the motivation for the investigation into a more efficient and flexible distributed monitoring system.

Network security is typically achieved through reactive measures such as applying patches to vulnerable machines and security software. When security is achieved in this manner it means that there always has to be a victim network that has been devastated before any rectification efforts are made. Meanwhile the threat will still be seeking out and attacking other victim networks. If an organisation is the first network to be hit by a new threat then there will be no counter-measures that a security expert can use to stop the threat.

This being the case there are security mechanisms, such as anomalous based threat detection, that can potentially identify this new threat and allow some sort of response; typically through isolation. These anomalous based threat detection mechanisms work by providing some sort of abstraction of the activity that is occurring on a system. The abstraction is provided so that the security expert will not have to observe every packet in order to find new threat patterns.

When looking at the abstracted activity the security expert will generally set, or have a mental image of, the activity thresholds that indicate normal activity and that activity which indicates an attack. When the security expert determines the threshold activity he/she will have to be careful to not set it too low or too high. If the threshold is set too low then the anomalous detection system will produce numerous false positives. The security expert will have to respond to every alarm in case it is a threat. Too many false positives will typically condition a dismissive attitude within the security expert; assuming that a majority of alarms are false and possibly missing a real alarm. If the threshold is set too high it will mean that the anomalous detection

system will only alarm on specific threat patterns. In this situation the false positives are greatly reduced however this comes at the expense of a potentially unidentified attack. Missing any attack defeats the purpose of having an anomalous threat detection system installed on the network. What is required of such a system is the ability to have a threshold indicator that can be flexibly and dynamically adjusted by the security expert.

To compound the threshold issue the anomalous detection system will typically use a generalised configuration that is applied to all machines. If it is not a network-wide common configuration then it is a configuration used for very generic classes of machines, such as all workstations and all servers. The generality of this configuration is a limiting factor in the anomalous detection system's ability to detect new threats. The main reason for the limitation is that a single general configuration that will identify all threats for every single host on a network does not exist. The reason for this is that each system may have completely individual roles. The workstation/server roles are the classic example of this. However on every network there are actually roles within the roles rather than just the general abstract categories. For example instead of just the overarching concept of the server there are DNS servers, E-Mail Servers, File Servers, SQL Servers, etc. The same principle applies to workstations on a network. There are graphics-oriented workstations, general office workstations, financial-oriented workstations, etc. Each individual role will create dynamic behaviours and traffic patterns for the differing systems.

In addition to the differing roles of the hosts another important factor that increases the dynamic environment of the network is that even though two systems may have the same roles there will be two different users working on those machines. The users will achieve the same tasks on those workstations but in a different manner. Hypothetically one user may work diligently on a task until completion and then use the Internet in the slack time while the other may work on the task while intermittently using E-mail facilities. From this hypothetical it can be seen that these two user's computers will display different traffic patterns and characteristics even though the computers have identical roles. Provision of an effective security monitoring scheme will require the factoring in of the individual roles that each host plays as well as the differing user activity on those hosts.

Another consideration of security monitoring systems is the choice of architecture that they employ; centralised or distributed. Centralised systems use a central controller that determines the actions of the subordinate components. In terms of security monitor system this means that each host will report abstract activity data to the central monitor which will then make a determination of the action (considering in-action as a form of action) that the subordinate host will complete (Banatre 1991). Distributed systems push the processing operations out to the individual hosts so that each host can determine for itself the correct course of action. Each host within the distributed framework will contain agent software that is self-contained and has all of the necessary functionality to achieve the intended goal, in term of this research, security monitoring (Wulf, Wang & Kienzle 1996). The selection of security system architecture will affect what overall operations and information that is available for the system. Within a centralised architecture each monitoring host functions as a reporter for the governing central analyser. This means that the only information that the analyser has to work with is the abstracted traffic data that each host reports. This abstract data is typically to general for proper analysis. Another drawback of the centralised architecture is the amount of reported traffic that flows on the monitored network. This reporting traffic consumes bandwidth as well as having to be catered for in the analysis process. One last obvious drawback of the centralised approach is that if the central controller is taken down then the system does not function. Centralised architectures were the deployment of choice in the early years of computing. One reason for this was that the hosts of the period had limited processing capabilities so a distributed architecture, with agent processing on each host, would have been prohibitive. However after extensive use of centralised monitoring systems in the field and observations of its short-comings there has been a deviation, in the security domain, toward the use of distributed architectures (Massie, Chun & Culler 2004). The main benefit of distribution that has caused the architectural shift is that each agent does all the analysis, processing, and action determination on each host independent of the other hosts. This means that the agent can focus on the monitoring of the host, thereby providing an in-depth analysis of this host. The benefits that distribution provides are attractive to the creation of a monitoring system mainly due to possibility of each agent being individualistic. The use of distribution means that the research system uses the host based security strategy (described in 2.4.2 Host based Security)

3.2 The Aims of the System

The aim of the system is to address the issues that have been previously discussed. To be successful in detecting unknown attacks the system will need to possess the following features.

The system is able adapt to the dynamic environment of networks. This caters for the fact that traffic on networks is always changing due to factors such as new network protocols and software. In addition to this the attacks that are occurring are changing dynamically as old attacks mutate and new attacks are formulated. To facilitate this feature the system uses an expert system in order to develop a constantly evolving knowledge base. The evolving knowledge base requires a system to provide easy maintenance of the knowledge it contains. To provide this feature for the knowledge base the system will use the multiple classification ripple down rules artificial intelligence technique. This technique allows flexible logic structure and the easy maintenance and addition of new logics within the knowledge base. It also provides constraints and validation to ensure the any new entered logics or refinement of the existing logics do not corrupt the existing knowledge base.

In addition to the network being dynamic a number of differing types of host exist within the network. The role that each host plays needs to be factored into the operation of the security monitoring system. To factor this in the entire system uses the distribution architecture. Each monitored host will have the system installed which can be tailored individually for that specific host. In this way each host will be monitored and each system can be adapted for the varying network conditions.

A monitoring system that creates too many false positives will induce a dismissive attitude in the security expert. As this monitoring system produces more false alarms the security expert will be more likely to ignore each alarm as another false positive. This repeated generation of false alarms within a monitoring system occurs because of over generalised logics that are used for identifying an attack. In addition the thresholds for the network activities could be set too low. The system solves this problem by using a flexible logic engine where the thresholds can be dynamically adjusted and the logics adapted to better identify the threats. The flexibility of operation stems from the use of the underlying MCRDR processes.

To produce a system that will be effective at identifying unknown threats it must allow the expert to quickly create some sort of response to the attack. The response may be to isolate the machines and/or subnet that are showing signs of infection or attack. Such a drastic response may not be possible because it will limit organisational functionality and the infection or attack may have already moved on to other machines and subnets. A better solution would be to analyse and identify the patterns exhibited on the infected system and then take that information and apply it to the other untouched machines. Then when infection appears on the other machine the patterns are already identified and the responsive measures are already in place. The system developed allows this through a knowledge importing mechanism as well as the underlying MCRDR processes that allow the knowledge to be adapted to better suit the recipient system. The response time in the system is improved because of the easy interface to and management of the underlying logics contained within each monitoring system. The usage of MCRDR within each monitoring system ensures quick and correct logic refinement or creation. This logic also includes the situational data at the time of the logic adaptation or creation for future reference. The expert can look at the situation data for any logic developed and use this to reason about a new situation or check that the original conclusion was correct.

The final aim of the research is to prove that the MCRDR technique can be successfully applied in the network security domain. The proof of this concept is demonstrated by the ability of the security monitoring system to achieve the goals aforementioned.

3.3 An Unknown Threat Scenario

To better demonstrate the principles of the monitoring system investigation into a hypothetical attack scenario is necessary. Consider the following unknown attack scenario:

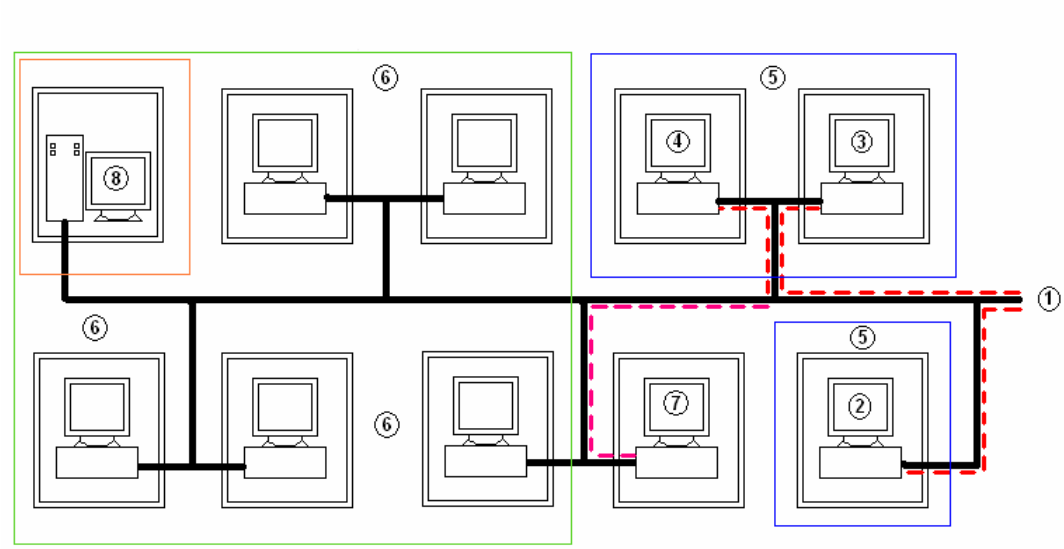

Figure 3-1: Unknown Attack Scenario

This diagram shows a small organisational sub-network with nine hosts and a server. In the figure the following is occurring:

1. An unknown attack has breached the organisation and is beginning to run rampart on the organisation sub-network.

2. The unknown attack compromises its first host and symptoms are beginning to be exhibited on the host. The monitoring system on the host detects the change in activity and sends an alert to the security expert (typically the network administrator).

3. The unknown attack seeks out a new victim and compromises another host. This host shows symptoms of infection as it differs from normal activities. The monitoring system sends another alert to the security expert. The expert at this stage will be getting wary that something suspicious is occurring on the sub-network.

4. The unknown attack has found and compromised a third host. Once again this host identifies the new activity and sends an alert to the security expert. Typically this would be enough to confirm the expert's suspicion and he/she will proceed to begin investigation as to what is occurring on the sub-network.

5. The expert checks the infected machines through interfacing with the monitoring system. The expert will see that a new attack is taking place and will use the monitoring system interface to identify the patterns that this new, previously unknown attack is exhibiting. The expert will create identifying logics, based on the attack patterns, within the monitoring system. These logics will also include the situational data which were used for analysis and logic creation.

6. The security expert will then takes the created logics and situational data and import it to other uncompromised hosts. These logics that identify the new, now known, attack and are linked to some sort of proactive action that will prevent the uncompromised machines from getting infected.

7. While this security expert is applying the logics to the other uncompromised machines the attack is still occurring and has compromised another host which did not have the logics and hence no proactive measures. Once again this compromised host will detect the new symptoms and send an alert to the security expert. However the expert is already fully aware of the situation and is already doing something about it.

8. The security expert arrives at the server and imports the logics and situational data into its monitoring system. However these logics may not be completely correct for the operations of this server. So the expert will refine and adapt the logics for this monitoring system so that it incorporates the operations of this server. This process ensures that the server will be protected without creating numerous false alarms, which as aforementioned will breed a dismissive attitude into the security expert.

The final step in this scenario is to do something with the already compromised machines. This research does not focus on what occurs at this point however the

security expert has many options. For example the expert can isolate each of the compromised machines and do a reformat and reinstall of the original configuration. The main issue for this scenario is the time it takes the expert to analysis, identify patterns and create logics for the new unknown attacks. It is proposed that using MCRDR processes within the monitoring system will decrease response times to the attack scenario demonstrated above when compared to other forms of logic creation that could have been used.

3.4 Rule Tree

MCRDR uses a tree-like structure for containing, analysing and extracting the logics or rules for any system. In the implemented system a tree data structure was created in addition to the functions for interacting with the rule structure. Each rule within the tree contains the pointer/s to the cornerstone case data associated with them for later extraction and analysis.

The expert does not have to manipulate the rule tree directly. The interactions with the rule tree were through the monitoring system interface. The process of rule addition, refinement and manipulation within the system are transparent to ensure that the knowledge base does not get corrupted and to not burden the expert with undue operations.

3.5 MCRDR Knowledge Acquisition

The process of acquiring knowledge in a generalised MCRDR system is demonstrated by the flowchart in Figure 3-2.

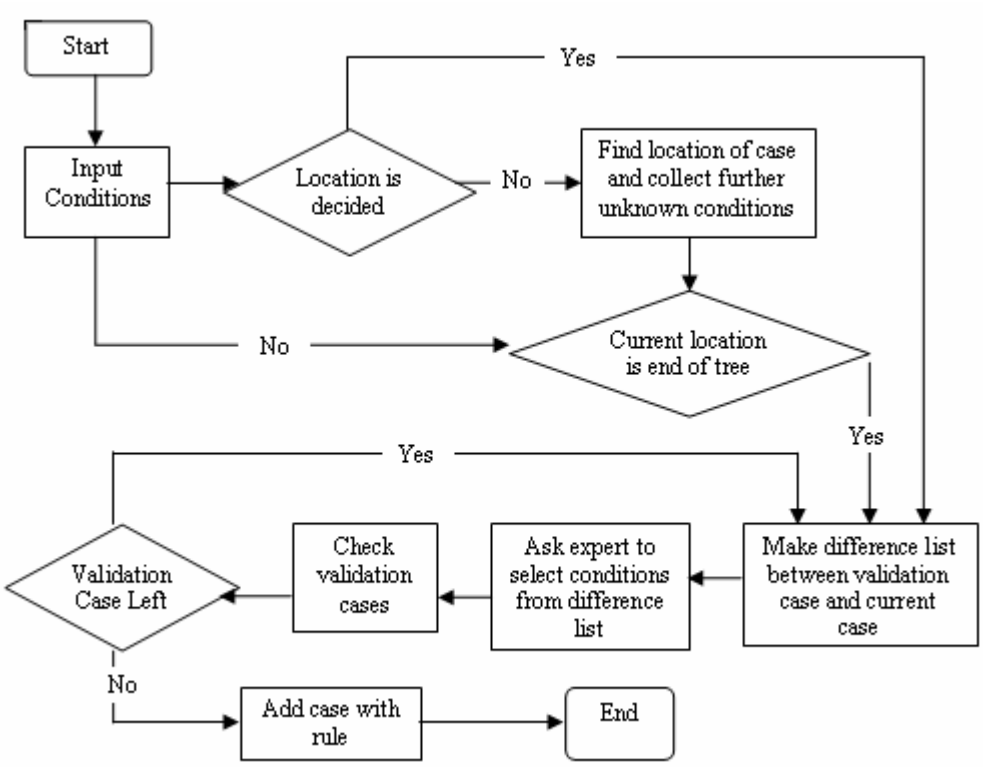

Figure 3-2: System Knowledge Acquisition Process (Kang et al. 1997)

The process of knowledge acquisition in terms of the security monitoring system is as follows:

1. The packet capturing module is started and it begins acquiring network traffic data.

2. The case time period elapses and the case data is created.

3. The case data is fed through to the inference engine and the MCRDR process begins. Firstly each rule on the first level is compared with the case data to see whether the conditions are satisfied. If the rule is satisfied it is fired and the children of this rule are considered. This process repeats with the children and so forth until none of the fired rule's children has fired. The last rule node in the tree to have fired along any branch becomes the conclusion for the rule. This means that there is the possibility of multiple conclusions for this case.

4. The current case data and the conclusion information, such as which rule concluded with what, is displayed to the expert via the system interface. If

there is no conclusion then a new classification needs to be made and the interface displays this fact.

5. The expert determines whether the conclusion was correct or not by observing the case data and what conclusions where produced with which rule. If it was correct then repeat process from step 2. Otherwise the conclusion was not correct and so the expert selects which rules concluded incorrectly or to adds a completely new classification.

6. If the expert has determined that a rule has concluded incorrectly then the expert is shown the different conditions between one of the cornerstone cases that exist for the incorrect rule and the current case data. The cornerstone list that exists for the incorrect rule is also displayed. The expert selects various conditions from the difference list and then validates the conditions for the new rule. Any of the cornerstone cases that do not satisfy the conditions are removed from the displayed list leaving only the ones that are satisfied by the selected conditions. The expert can select the cornerstone cases that he/she wishes to let match this rule's conditions. If the expert selects all of the remaining cornerstones to let match then the new rule is valid and is added as a refinement (child) of the incorrect rule with the selected conditions and the current case data as the cornerstone. Otherwise the expert is presented with the differences between the current case and one of the remaining cornerstones. This process continues until there are no cornerstones in the list or the expert select the all of the remaining cornerstone case data that will be allowed to match. Then the new rule is added as a refinement of the incorrect rule with the conditions selected and the current case data as the cornerstone. When the new rule is created go back to step 2 of this process.

7. If the expert has selected to create a new classification then the same process occurs as in step 7 however the cornerstone case list contains every cornerstone case currently contained within the monitoring system. In this situation the condition list from which the expert can select from is all of the conditions of the current case data. The new rule, with its selected conditions and cornerstone case, is added as a top level rule (child of the root) so that it

will become a new branch of reasoning for the MCRDR process to follow. When the new classification is created go back to step 2 of this process.

8. While the expert is interacting with the rule tree structure the packet capturing module is still capturing packets and creating cases but the inference module will wait until the expert has dealt with the current working case. The created cases are buffered (put in the database) until the expert has finished working and created a new conclusion or indicated that the current case is correct. This ensures that real time operation can continue without interrupting the expert and also it ensures that any new rules (logics) that the expert creates on this current case get applied to the cases following.

3.5.1 Validation Process

In addition to the validation process ensuring that the new logics do not incorrectly fire an existing logic the system also checks to see whether the conditions are valid. This is because the conditions are displayed but they are able to be edited. If the edited condition does not satisfy the existing case data then it is deemed invalid and must be corrected. The invalidation continues until the edited condition satisfies the existing case data

3.5.2 Rules Types

There are three types of rules that exist within the system. These types of rules determine what occurs within the inference module as well as the location of the newly created rule type.

Refinement rules are the typical rules that are created to provide a new conclusion for a wrongly classifying rule. The conclusion of this rule will replace the conclusion of the parent rule if its conditions are satisfied. The location of a new refinement rule is as a child of the parent rule (the rule that gave the wrong classification).

Stopping rules are rules that have conditions, which when satisfied, will cause the parent rule of this rule to not fire, in other words fire a NULL conclusion. The location of the newly created stopping rule is at one of the child positions of the

parent rule (the rule that gave the wrong classification). Stopping rules play a major role in MCRDR in preventing wrong classifications being given for a case.

New classification rules are rules that provide a completely new branch of reasoning and are not associated with any other rule (other than root). When the rule and conditions have been validated the newly created rule is added as a child of the root.

3.6 System Design

The following is the abstract design of the system. The thing to observe is that the network traffic thread, and hence packet capturing, is executed in parallel with the main system thread. Parallel execution allows the real-time execution of the monitoring system without packet loss.

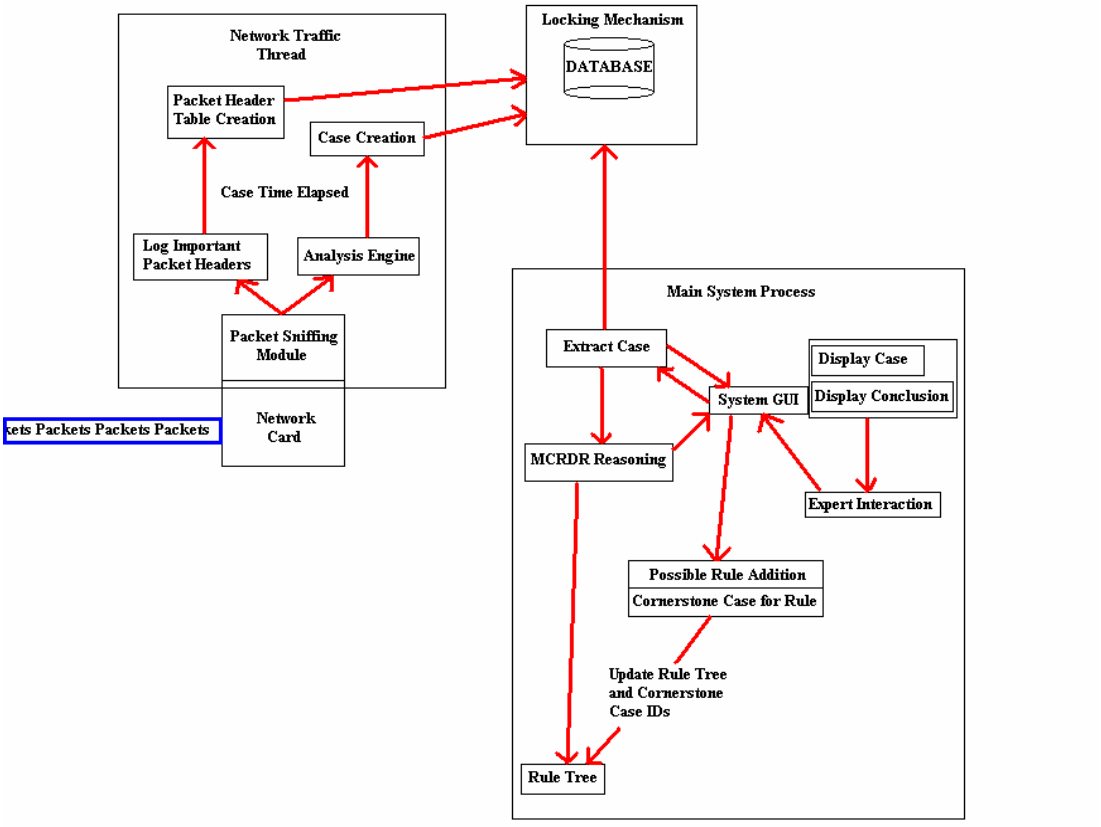

Figure 3-3: Overall System Design

3.7 Real-time Issues

3.7.1 Packet Loss

In order to detect attacks in progress the system must collect every packet that is received or sent on the host network card. If any packets are not captured then it will impede the knowledge acquisition process. The lost packets may contain information that is vital to an attack pattern being detected.

With this issue in mind the system was designed such that the packet capturing engine is run in parallel with the other system processes. To ensure that the packet capturing engine was not losing packets (by not processing them) it needed to be tested against another packet capturing tool. A freely available packet capturing software called Ethereal was used for comparison purposes. Ethereal is not a commercial product however the underlying engine has been used in a number of commercially used products. From this, it is concluded that the Ethereal software is sound and operates correctly.

Tests were conducted where the packet capturing engine and Ethereal were run concurrently over differing periods of time. Over the time period logs were created by the packet capturing engine and Ethereal. These logs were then compared to identify any packets that the packet capturing engine may have missed and that Ethereal found. After many such experiments it was deemed that packet capturing engine operated successfully and all packets were correctly identified.

3.7.2 Case Window Size:

The time interval between cases being created is the window size. The window size determines the amount of network traffic is inputted into the analysis engine before the case is created. The selection of the window period will affect the overall system in a number of ways.

Having the window size set to a small value will mean that there won't be much data to build a case from. The patterns for detecting anomalous behaviour will be very specific. A small window size may also mean that anomalous patterns may be

misclassified as normal traffic as the expert will not see all of the information relevant to an attack

Having the window size set to a large value will mean that there will be a significant amount of data processed before a case is created. A large window size forces the system to deal with a significant amount data and the resultant memory requirements. For example a ping flood attack was analysed which produce over ten thousand packets in ten seconds. This overflowed the buffer, the IP array, designated to hold the address information and the system began writing into operating system memory.

The large window size may also mean that data representing an attack may be hidden within a significant amount of normal traffic. This issue is reduced by the system creating a case from summary information; however an expert may miss an attack pattern from the amount of information from which to make a classification.

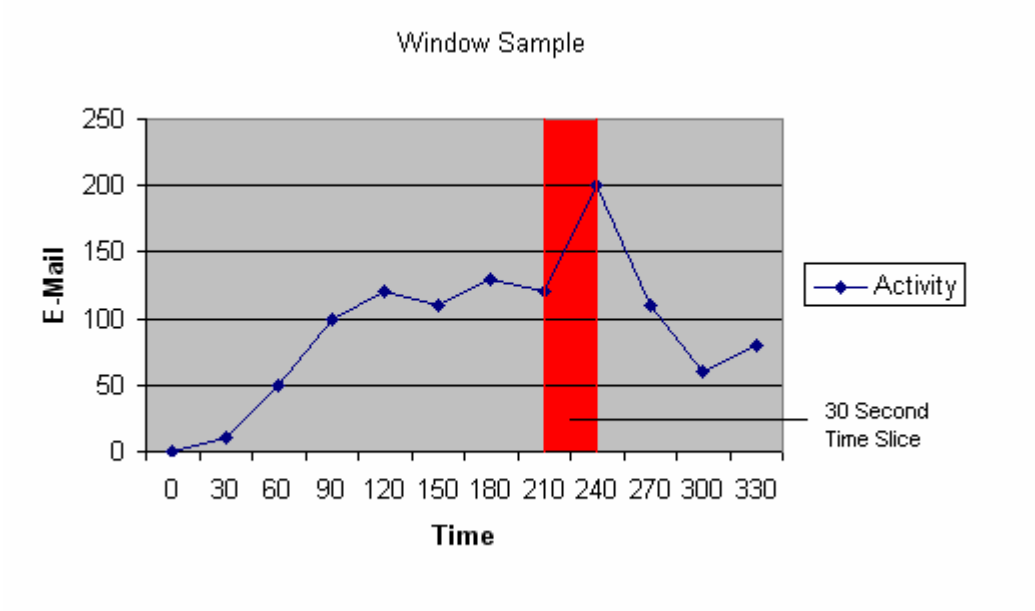

Figure 3-4: A 30 Second Window

As can be seen from the above figure a 30 second window size will correctly see the sharp rise and fall of the e-mail activity. This activity may be a potential indicator of anomalous behaviour. Figure 3-5 shows the same data but using a minute time slice.

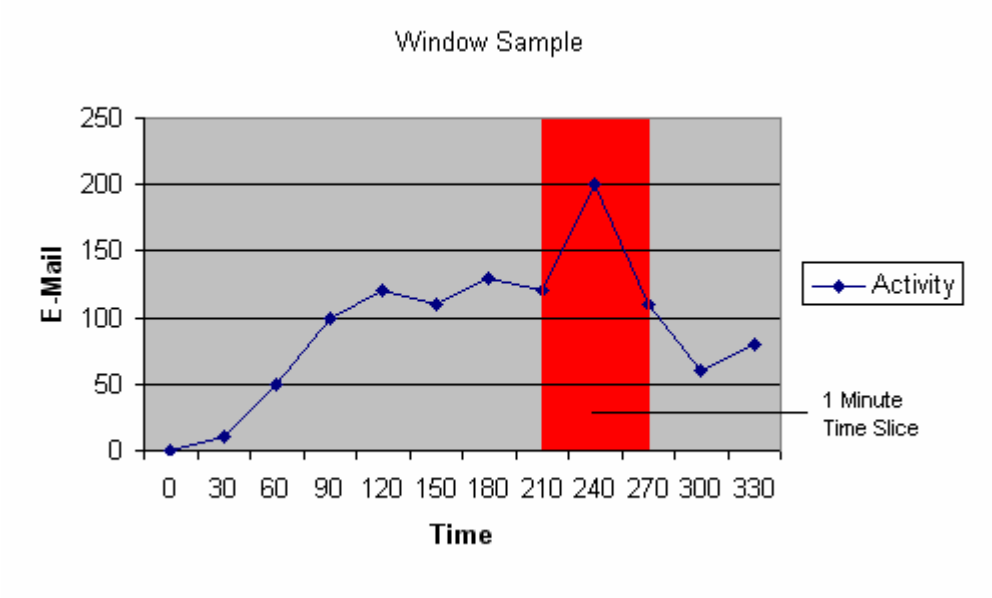

Figure 3-5: A 60 Second Window

As can be seen the data over the minute interval will be summarised and the sharp rise and fall of the activity will be generalised to a large amount of e-mail activity in that time slice.

Another issue that arises from having a large window size is that within a given windowing period multiple attacks may occur. An expert may identify the combined attack pattern as a single attack that has elements of both individual attacks. However this issue isn't significant as over the course of time the expert will classify each of these attacks with more specific rules.

In the experiments the window size was set to twenty seconds. The packet capturing module obtains information from the network for twenty seconds and then it analyses the data, builds summary information of that data, and then creates a case. While the analysis engine computes the summary information the packet capturing module continues to gather network information for the next case.

3.7.3 Storage of the Captured Data

Every packet has headers that contain important information necessary for transmission through a network medium. The headers identify a packets source address, destination address, source port, destination port, and destination protocol. It

also contains other information that is necessary for routing the packets over networks and the Internet.

Once collected the network data that the packet capturing module gathers is stored in a database. This information is stored in the database in two parts, the important packet information and the case data created from the network data.

In addition to the storage of packet information in the database every single packet and most of the header information is stored in a separate text file. This is so that the expert can get an extremely detailed analysis of the traffic (see 7.2 Appendix B: Detailed Packet Analysis).

3.7.3.1 Packet Information Storage

When a packet is detected by the packet capturing module some key characteristics of the packet are recorded in a database.

Column Name	Description
IP Source Address	This identifies the system that sent the packet. This address is not unique to each machine and this address is easily 'spoofed'. Spoofing is where a machine sends packets with a source address that is not their own in an attempt to fool the recipient that this sender is a trusted system.
Source Port	The source port identifies the protocol or application that created the packet on the source machine.

IP Destination Address	This identifies the recipient system of the packet. In the normal processes of a TCP/IP network only the system identified by the destination address will process and extract the information from this packet. There are exceptions to this rule for example systems called proxies will extract information from all packets in order to make filtering decisions for network traffic. Typically the destination address will identify a single host however there are a category of destination addresses called broadcast addresses. These broadcast addresses are used to send a single packet that all hosts within a network or sub-network will process. If these broadcast addresses didn't exist then a host wanting to send packets to all hosts, within a network or sub-network, would have to send a packet address specifically to each recipient host.
Destination Port	The destination port is used by the recipient machine to identify the protocol or application that the packet should be processed by.

Table 3-1: IP Database Table

Column Name	Description
Parameter	This is used to identify the name of the condition that is being stored for the case
Expression	This is used to store the expression sign of the condition
Value	This is used to store the actual value of the condition

Table 3-2: Case Database Table

3.8 Not All Users are Security Experts

In attempting to situate this system in a real environment a number of issues become obvious: The system is only as good as the security knowledge of the user. If the user is not a security expert then it makes it difficult for the user to identify the patterns for various attacks. As well as the user will have other work to complete rather than interface with an expert system. This being the case a feature that allows the administrator to dynamically import acquired knowledge, rules and situational data, had been created within the monitoring system.

An expert will typically not want the system to keep asking for prompting for rules that should be automated. This is easily rectified by the addition a simple checkbox for automating a rule that has been deemed definitely correct. The rule will still be used in further MCRDR validations otherwise the knowledge base could become invalid. This process will allow for easy integration with the action modules that is discussed in future work. The action modules would not prompt the expert to carry out their assigned tasks rather it would be automated.

3.9 Existence of Generalised Threat Patterns

The unknown threat detection system works by observing the network activity exhibited on the monitored host. This limitation allows the system to be developed within the time allocated to an honours project. However this limitation means that patterns must exist for the various threats at the network level for the monitoring system to perform successfully at the intended goals. To test the overall goals of the distributed security monitoring system it was found through the experiments and results, detailed later, that generalised threat patterns do exist at the network level.

3.10 Evaluation of Experiments

To determine that an unknown threat detection system is operationally effective the system setup to be monitoring on a victim machine while it was bombarded with the various threats. Meanwhile the expert interfaced with the system in order to classify

the threats as they occurred. The threats used in the experimentation are not unknown; they have been identified and are well documented in literature. However in order to test a system that identifies unknown threats it would be impossible to have an unknown threat attack situation. The only possible way would be to attach a monitored host to the Internet and hope that it looks inviting enough for an attacker to attack it in the first place and secondly that he/she will use a previously unknown attack in which to do so. As can be reasoned this may take a long time, if at all. The assumption used for testing is that at some point in computing history the attacks were not known and had to be identified through some means. So testing of the system was carried out by identifying candidate threats and creating or downloading the source code for an attack. Meanwhile the expert will identify the patterns and subsequent conditions that best represent and identify the attack from the norm.

The candidate threats were selected from a varying mix of complexity and what type of systems the threat would typically attack, such as mainly server oriented. The threats that were used in the experiments are detailed below.

3.10.1 Denial of Service

Denial of Service is where an attacker will abuse a publicly accessible service to the point where legitimate service activity can no longer occur (see 2.4.2 Denial of Servce). Two different forms of this threat were selected for testing. First a basic denial of service attack, ping flooding, which can affect both normal type hosts and servers was selected. Next a server oriented denial of service attack that is directed at E-Mail servers, mail bombing, was selected.

3.10.1.1 ICMP DoS – Ping Flooding

The terms ‘ping’ and ‘pinging’ refers to a system sending a test packet to another host to see whether it has connectivity to that host. Normally the sending host will send out four small Internet Control Message Protocol (ICMP) packets to the target host. When the target host receives these packets it will reply back using the same ICMP protocol. Normally this type of operation is legitimate and is even the basis for some of the older monitoring systems that would poll hosts to ensure network-wide connectivity. However there are software utilities and built-in ping functionality within operating systems that allow a host to flood out a significant amount of ICMP

packets to a destination or range of destinations. The destination host/s will have to process each packet and send a reply back to each packet as part of the design of the ping protocol. This significantly reduces the amount of normal network traffic that the victim host can process. The denial of service effect can be exponentially increased by increasing the number of flooding hosts to a point where the pipeline to the victim host becomes completely saturated and no normal network activity can be achieved.

The testing of such a denial of service attack was done using multiple hosts using ping flooding software called UtilityPing that is freely available from the Internet. Running a denial of service experiment on a live network will degrade the network's performance. So to reduce the likelihood of performance degradation on the University network, and to not give the Information Services staff a headache, the attacking hosts only flooded in short bursts.

3.10.1.2 Server DoS - E-Mail Bombing

There are many different versions of the sendmail protocol that exist for e-mail delivery over the Internet. The basic principle to sendmail is that no e-mail should ever be lost. The developers of sendmail made the protocol extremely robust in order to cater for the lossless principle. This factor makes it extremely difficult to defend against e-mail denial of service attacks. This form of attack has been dubbed e-mail bombs or e-mail bombing. The primary objective of e-mail bombs is to flood the e-mail server so that it becomes unavailable or unserviceable. The secondary objectives from the e-mail bomb are to degrade the availability of communication systems, undermine the integrity of organisations, or distribute illicit material (Bass et al. 1998).

During the first half of 1997, Langley Air Force Base came under repeated e-mail bombing attacks via the Internet. The mail system administrators, unaware that they were coming under attack with an e-mail bomb, simply rebooted the servers when they stopped responding (Bass et al. 1998).

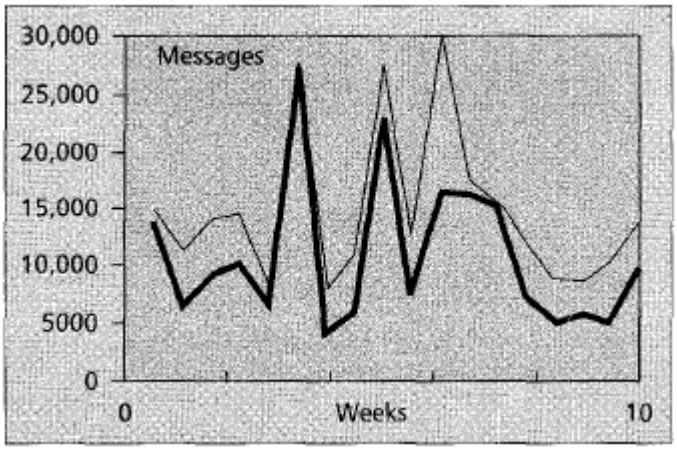

Figure 3-6: Langley AFB 1997 Mail Activity (Bass et al. 1998)

In the above figure the total amount of mail over the period is shown by the light line while the dark line indicates the mail due to the e-mail bomb.

To test this form of denial of service attack an e-mail server was setup to be a victim while another host repeatedly bombed the victim server. The e-mail bomb used for testing was a simple PHP script that was run on a web server (see Appendix). This test is to check that the monitoring system can be used for various, differing host roles; in this test it was an e-mail server.

3.10.2 Information Gathering

Information gathering is where the attacker, whether it is a Worm or a human, probes the target machine/s to determine important information about the victim. The gathered information includes: operating system type and version, protocols and services in use and open ports to exploit. This process dubbed “fingerprinting” shortcuts the time it takes the attacker to compromise the machine. With an awareness of the configuration of the target the attacker will know exactly which vulnerability to exploit.

The following experiments investigate some of the techniques used in the fingerprinting process; namely various port scanning techniques. Port scanning describes the process of identifying open ports that exist on the open machine. These ports are typically reserved for certain protocols and services and once identified can be exploited. Within TCP/IP network there are typically two protocols used for

transporting data; Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

3.10.2.1 UDP Port Scan

UDP communication is connection-less so no handshaking procedure is required. The UDP port scan experiment was conducted using freely available port scanning software called SuperScan 4. In this application it was possible to isolate the protocol and ranges of ports to scan. The experiment involved using a target victim machine connected to an attacker who would scan the UDP ports of the victim with the default 'vulnerable' port ranges within SuperScan 4. The expert interfaced with the monitoring system on the victim machine to create logics identifying the attack pattern.

3.10.2.2 TCP Port Scan

TCP communication is achieved through establishing a logical connection between sender and recipient. The logical communications channel is established through a handshaking procedure (described in 2.1.2 Connection-Oriented and Connection-Less). When the handshake has been successfully completed normal data communications can commence. The software used in this experiment was once again SuperScan 4. The application was set to scan the TCP ports that were open on the victim machine. The expert interacted with the interface of the monitoring system on the victim machine to create the logics that identified the attack pattern.

3.10.3 Worm Attack

Worms are a form of virus that can propagate using their own means (described in 2.3.4 Worms). A Worm will scan for active vulnerable machines and then mobilise to infect it.

In this experiment a worm simulation was created and used by a monitored host to attack another monitored machine. The worm simulation provided was not identified ahead of time and the expert had to identify discriminating patterns from analysis. The worm displayed curious properties that are detailed in the results section.

3.10.4 Base-lining

In order to gain a perspective of the activity on a network an analyst will observe the traffic flows during normal usage periods. This establishes a base-line for the network from which any exceptional behaviour can be identified.

A static analysis of an individual point on the network will not divulge behaviours over a subnet and so will not be comprehensive enough to apply the patterns as general trends for the entire network. Therefore an analyst will observe multiple areas of the network to create a profile of its behaviour. The areas that the analyst samples data from will depend on the scope and perspective of the network monitoring (Shimeall, Dunlevy & Pesante 2001).

Once the initial base-lining and analysis has been done, a network analyst can use this information to identify anomalous activity that could have been otherwise masked by normal traffic.

In this experiment the expert observed and created logics for the normal activity flows and network chatter that appeared on a host during normal operation. The aim was to create logics that would identify normal traffic conditions. If the host is base-lined then there are logics that will clearly distinguish between normal and anomalous activities.

4 Results

The experiments using the developed system have produced some interesting results. These results are discussed in the following sections.

4.1 Denial of ServiceAttack Type

As described previously in the background (2.3.2 Denial of Service) the denial of service class of attacks attempt to disrupt system and network activity by overloading a publicly accessible service. The following is the generalised pattern discovered for a ping flooding attack. This shows the activity exhibit on the victim host.

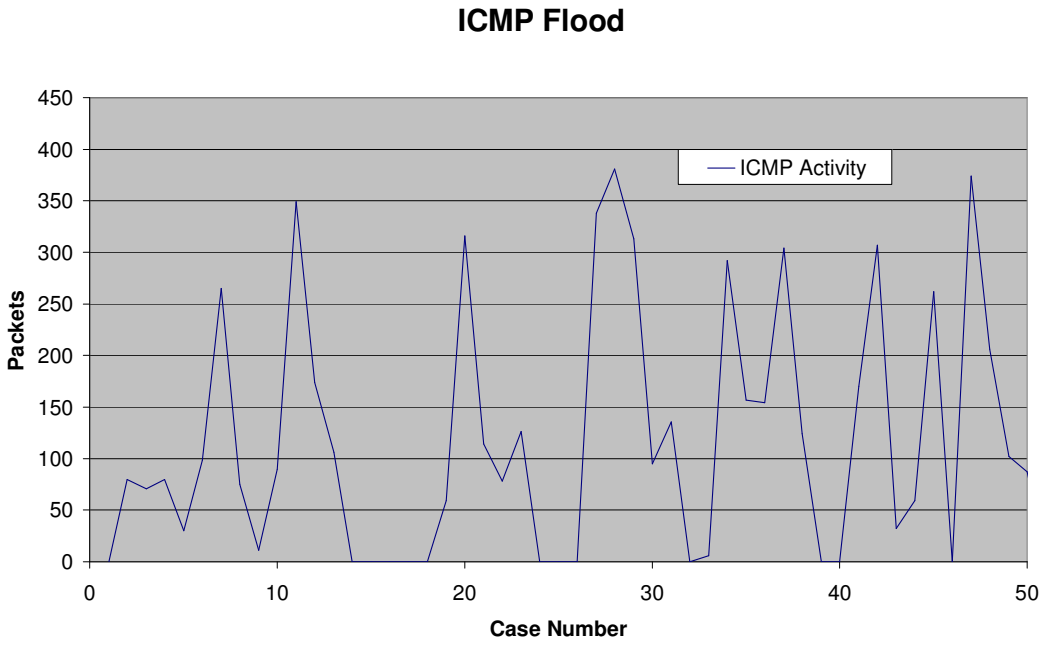

Figure 4-1: Ping Flood Victim's Activity

As can be seen from the above figure the testing started with a small amount of pinging. At around case eight multiple machines began flooding the victim and the ICMP activity spikes. At various points with the experiment multiple machines flooded the victim which presented itself as large ICMP spikes. In the normal form of the ping flooding attack the ICMP activity would peak to maximum and then flat-

line for the period of time the attack occurs. However, as mentioned previously, the flooding was limited to reduce live network degradation.

The case condition that is selected to identify the attack is Incoming_ICMP. Figure 4-2 is the rule tree that was created by the expert in the process of monitoring this threat

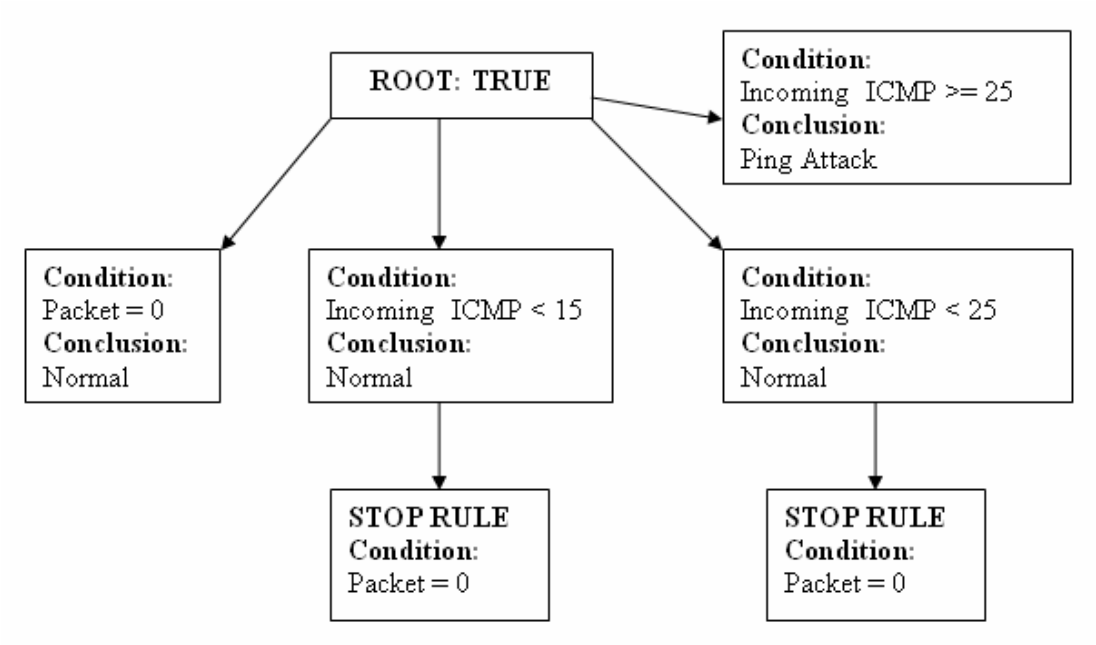

Figure 4-2: Rule Tree created in the Ping DoS experiment. To save space only the core rules are displayed

From the observations of Figure 4-2 it can be seen how the rule tree in the monitoring system using MCRDR can be easily refined and expanded upon. Each of the first layer rules (children of the root) is a separate branch of reasoning (or line of inquiry). Each of the first layer rules will be checked to see whether the conditions are satisfied and if so that branch is further investigated. Figure 4-2 demonstrates how a stop rule within the monitoring system works. For instance the rule with condition “Incoming ICMP < 15” has conclusion Normal, that is unless the “Packet = 0” is not met which will then create a NULL (non-text) conclusion. This flexibility allows the expert to stop certain rules firing when the expert realises the conclusions under certain conditions.

To test that the expert monitoring system can be used to monitor different types of hosts, name a server type, a different more selective threat was used in the next

experiment. The attack that was used in the next experiment is a technique called E-Mail bombing.

The objective of an e-mail bomb is to overload an e-mail server so that it becomes unavailable or is unserviceable. Repeated e-mail bombing will degrade the quality of service of the organisational communications infrastructure. This is due to the fact that the numerous bombed e-mails need to be processed and tends to overflow the mail spool (mail storage directory) (Bass et al. 1998). The ramifications are that the organisation will have difficulty in electronic communication with partners and consumers.

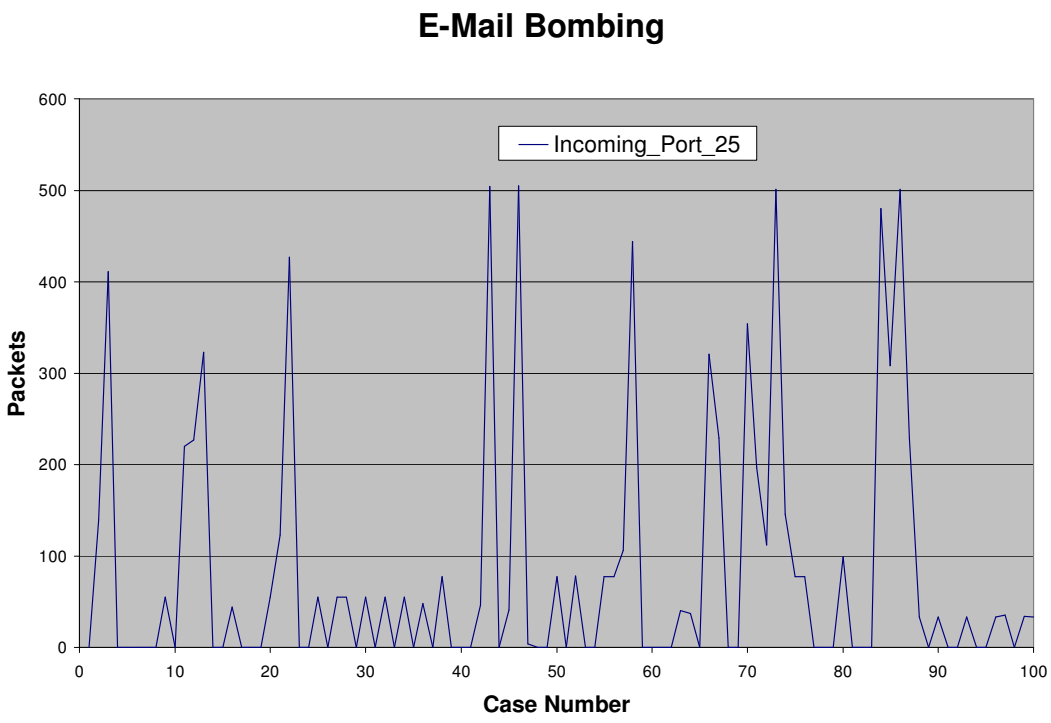

Figure 4-3: SMTP Pattern during E-mail Bombing

The activity shown in Figure 4-3 demonstrates how the port 25, or Simple Mail Transfer Protocol (SMTP), rates peak at the points where the e-mail server is being bombed. Normal e-mail activity is clearly distinguished by the much smaller peaks. From this graph it is clear that a generalised pattern on port 25 does exist for e-mail bombing. The main problem with identifying this threat is at level is should the threshold be set. The expert will need to ascertain the levels of e-mail traffic that the organisation receives over various time periods, such as over the weekend. The expert identifies the peaks of the normal e-mail activities and at what times these

peaks occur. Then the expert can include time information as well as the port 25 activity in order to determine threshold levels for identifying e-mail bombing. In addition if the threshold levels are set too low or high they can be dynamically adjusted later on by the expert using the interface, and hence the underlying MCRDR processes. An interesting item to note is the comparison of this graph to the graph of the e-mail attack at Langley AFB (see Figure 3-6: Langley AFB 1997 Mail Activity (Bass et al. 1998)). Although the scale is significantly different and the activity show in the Langley AFB figure goes off actual e-mails rather than incoming port 25 activity, the similarities are noteworthy.

The case condition that is selected to identify this attack is Incoming_Port_25 (see 7.1 Appendix A: The Case Conditions). The following diagram is the rule tree that was created by the expert in the process of monitoring this threat

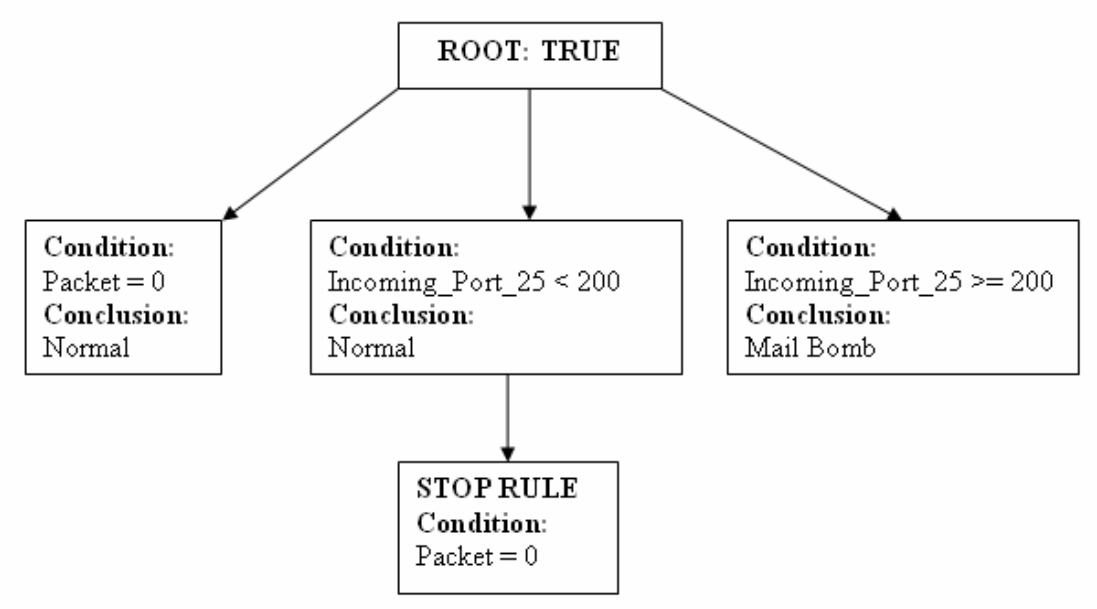

Figure 4-4: Rule Tree created in the E-Mail DoS experiment. To save space only the core rules are displayed

To create the rule to protect against this threat the expert simply selects the Incoming_Port_25 condition and edits its value to reflect the threshold margin that will indicate the threat. If the initial rules within the system have incorrect thresholds then the flexibility of the implementation will allow dynamic refinement.

Another condition that can be factored into better identifying the e-mail bombing threat is the Time condition (see 7.1 Appendix A: The Case Conditions). The Time

condition can be used to develop logics based around the peak user e-mail periods in combination with setting a generalised conditional threshold. Using Time as a rule condition allows the expert to create better and more flexible logics for threat identification.

4.2 Information Gathering Threat

Before an attacker launches an attack he/she will attempt to determine that a target host is active, determine which operating system is being used, and the open ports. This information gathering process has been dubbed fingerprinting. The information that the attacker gathers from this process allows him/her to make a determination of the available vulnerabilities to exploit. An attacker can launch attacks upon the target without determining this predetermined information but this takes a significant amount of time. In addition if the target host has any sort of monitoring installed then it is highly likely that these failed attacks will trip an alarm so that the target gets locked down or isolated. Figure 4-5 shows experimental results from the port scanning threat. The two different types of scanning that are used within the experiments are UDP and TCP port scans. The two protocol scanning techniques are tested separately due to the differing behaviours of each technique. UDP and TCP port scanning will differ because of the connection-less and connection-oriented nature of the protocols (see 2.1.2 Connection-Oriented and Connection-Less).

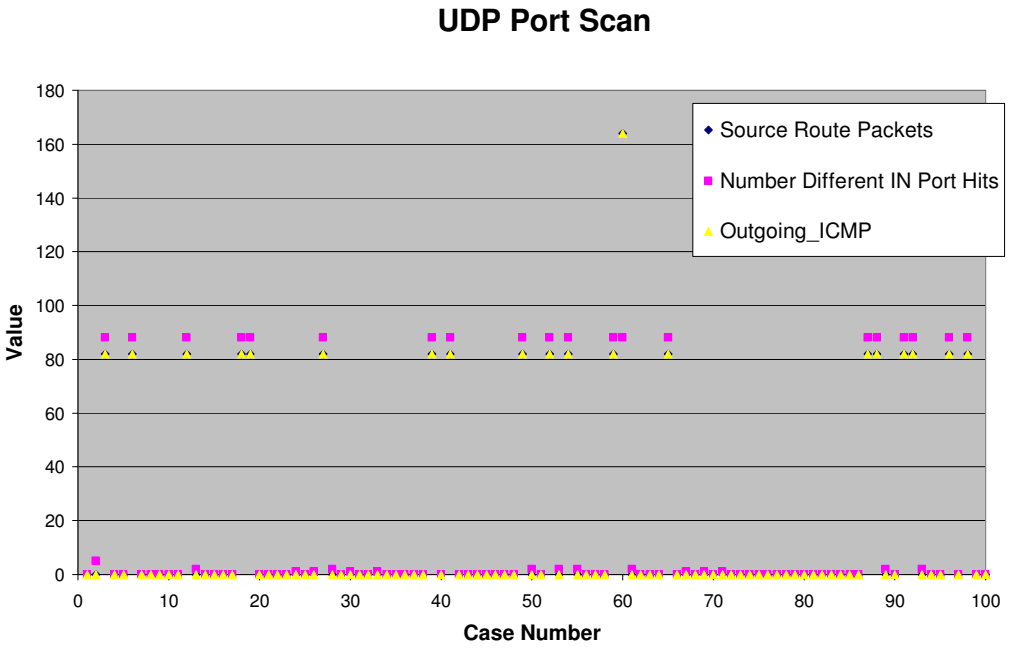

Figure 4-5: UDP Port Scan significant conditions

Figure 4-5 shows the conditions that were significantly affected after a UDP port scan on the monitored victim machine. Each of the conditions within the graph peaked whenever a UDP scan was targeted at the victim. The Number_Different_IN_Port_Hits indicates the number of different incoming ports that were hit on the monitored victim. Values of around eighty different ports in twenty seconds are abnormally high for a host undertaking normal tasks. In order for UDP port scanning identification to be less generalised, to reduce the possibility of false alarms, it is desirable to have multiple conditional thresholds to work from. Outgoing_ICMP is another condition that was significantly affected during the UDP scanning process. However, it is desirable for ICMP type conditions to be reserved for use in identifying the ping flooding threat. This is because pinging uses the ICMP protocol. Another noteworthy condition was the number of source route packets created during the UDP scanning process. Source routing within packets is typically used by routers for testing and congestion management (see 7.1 Appendix A: The Case Conditions for a further description). Within any organisational network it is quite uncommon to find packets with the source routing option set. This is a good condition to use as part of the logics for UDP scan identification. Further analysis of the results produced another viable condition.

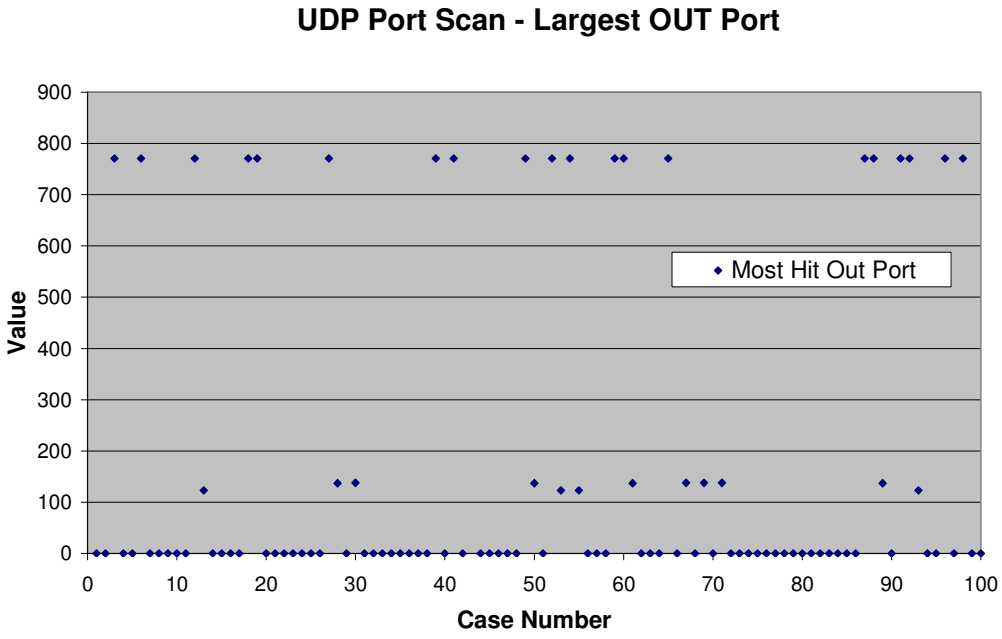

Figure 4-6: Most Hit Outgoing UDP Port

Note that a value of 0 actually means no activity during the case time rather than Port 0 being the most hit port

Figure 4-6 shows the outgoing UDP port that was hit the most during each of the cases. If this graph and the previous graph are compared then a direct relation can be drawn; during the periods where the previous conditions, such as Source Route packets, are spiking the most hit out port is 771. There is a direct one-to-one relationship between the conditional spikes and port 771 being the most hit out port. After multiple experiments on different systems, using different operating systems (Mandrake) and different scanning utilities (NMap) this relationship remained. From this it has determined that the best way to identify this threat is to use the Outgoing_Port_771 condition (see 7.1 Appendix A: The Case Conditions) in addition to a combination of the previous conditions mentioned above.

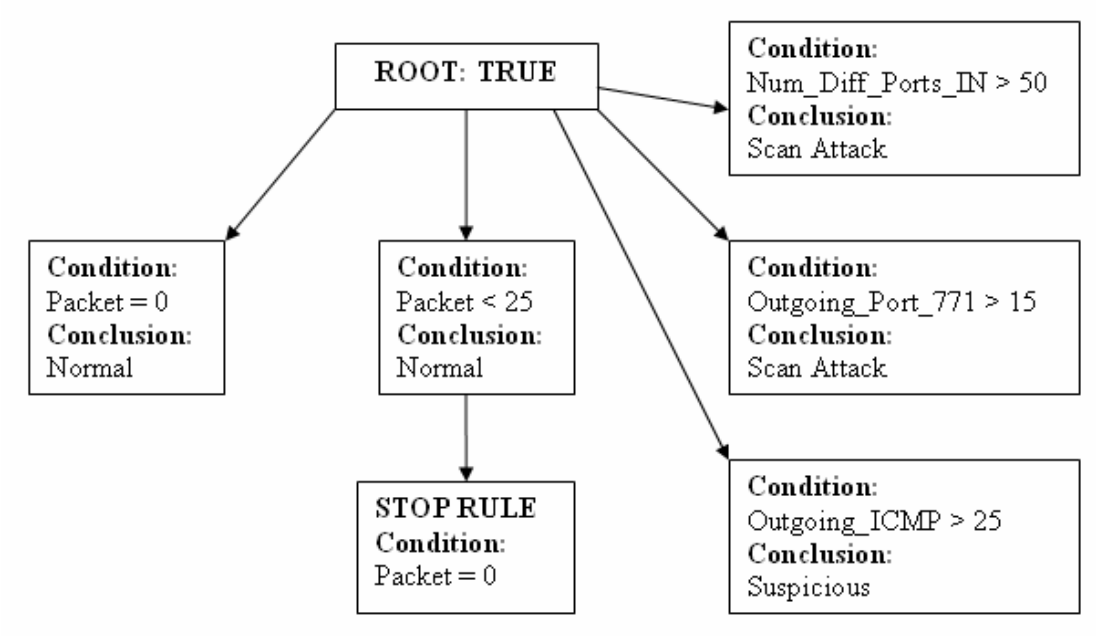

Figure 4-7: Rule Tree created in the UDP scanning experiment

To save space only the core rules are displayed

Figure 4-7 shows how each of the conditions occupies its own branch within the rule tree. Each of the conditional logics is compared on each pass of the reasoning process. This flexibility allows extra rules with differing condition logics to be easily incorporated within the rule tree.

With TCP port scanning the behaviour is different due to the three-way handshake process undertaken before a TCP communication session are established.

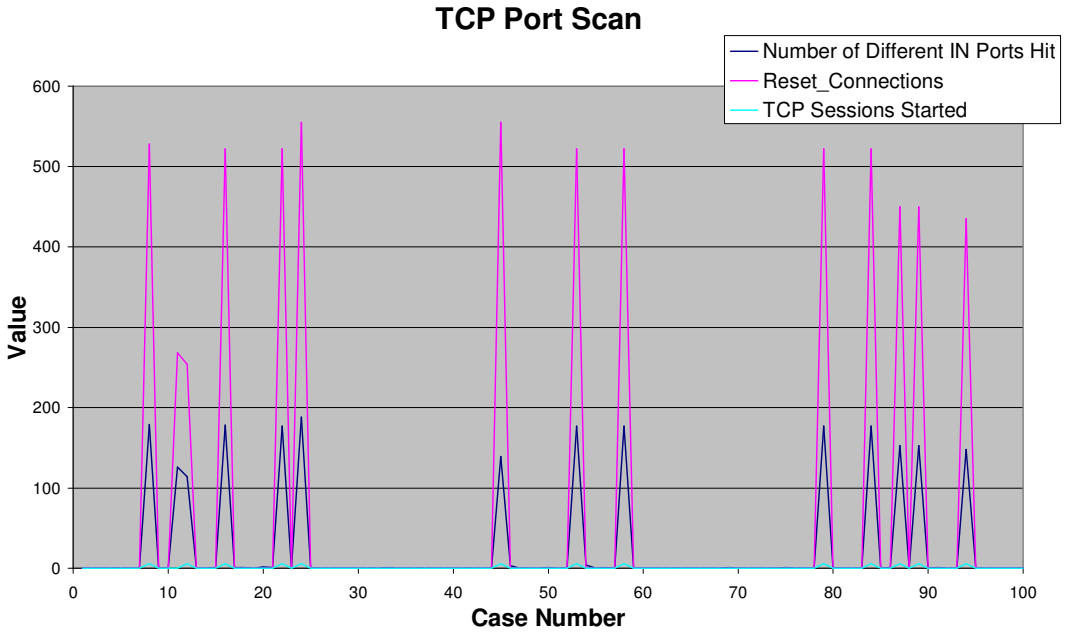

Figure 4-8: TCP Port Scan significant conditions

As can be observed, the key conditions spike dramatically when a TCP port scan occurs on the victim machine. The first condition that could be used identification purposes is the number of different incoming ports hit. This, like UDP before it, is quite a significant condition as the number of active protocols and services during routine operation will typically be low. The most significant condition that is used to identify TCP port scanning is obviously the Reset_Connections condition. Where each of the activity spikes occur the Reset_Connections condition has a major upsurge. Reset connections within the TCP protocol occur as a result of a protocol violation in the handshake procedure (see 2.1.2 Connection-Oriented and Connection-Less and 7.1 Appendix A: The Case Conditions). The TCP port scan software will send a specially crafted TCP packet that will get the target machine to respond if it happens to hit an open TCP port. Any closed ports will respond to this crafted packet with a protocol violation and reset the attempted connection. TCP protocol violations happen very rarely on any network so this condition becomes the chief identifier for the TCP port scanning threat. One interesting point to note from the above graph is the TCP Sessions Started condition. It indicates how many of the

TCP port connections that the target machine responded to; telling the attacker which of the TCP ports is open on the victim.

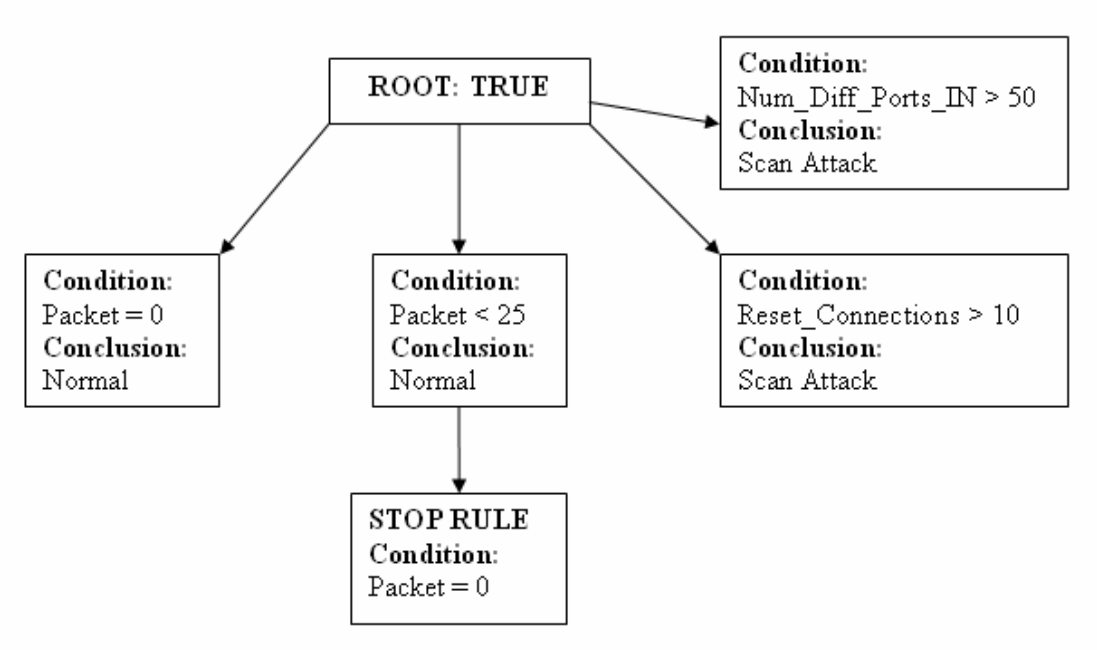

Figure 4-9: Rule Tree created in the TCP scanning experiment

To save space only the core rules are displayed

The rule tree within Figure 4-9 shows that in circumstances where there are only a few conditions needed to identify an attack the resulting rule tree created is minimal. Even though the rule tree is minimal, provided the thresholds have been set correctly, it is very efficient and effective at identifying the attack,

4.3 Worm Threat

The Worm threat describes the actions of a Worm in its attempts to mobilise and attack vulnerable hosts (see 2.3.4 Worms). In this experiment a worm simulation was used to attack another machine while the expert interfaced with the monitoring system and attempted to discover identifying patterns.

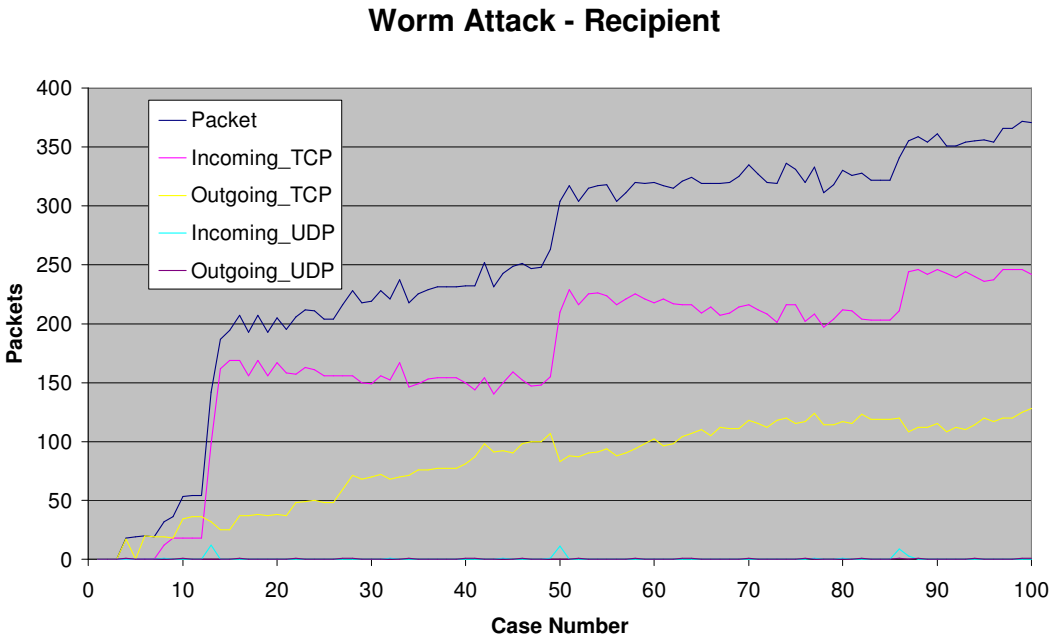

Figure 4-10: The Worm Attack Behaviour

Figure 4-10 shows the activity levels of various conditions observed during the Worm attack experiment. From the above graph the conditions do not show any obvious conditions that are suitable for identifying the Worm. The conditions are typical traffic patterns that are reasonable for any network. The only behaviour that is consistent is that the traffic tends to be constantly increasing. These conditions alone are not enough to conclusive identify the Worm. The activities in Figure 4-10 could be attributed to normal traffic operations on the monitored host. This is activity normality shows some of the complex behaviours exhibited by a Worm; self-preservation by attempting to obfuscate its activities as normal network traffic patterns. The longer a Worm can disguise its activities as benign, the better its chances of infecting more machines.

For the expert to be able to identify the Worm threat there needs to be a more definitive condition. Using the above conditions would create a large number of false alarms, no matter what threshold is set for the conditions. It is highly likely that these types of patterns would exist in normal network usage. Some other condition/s needed to be found which will identify the Worm threat.

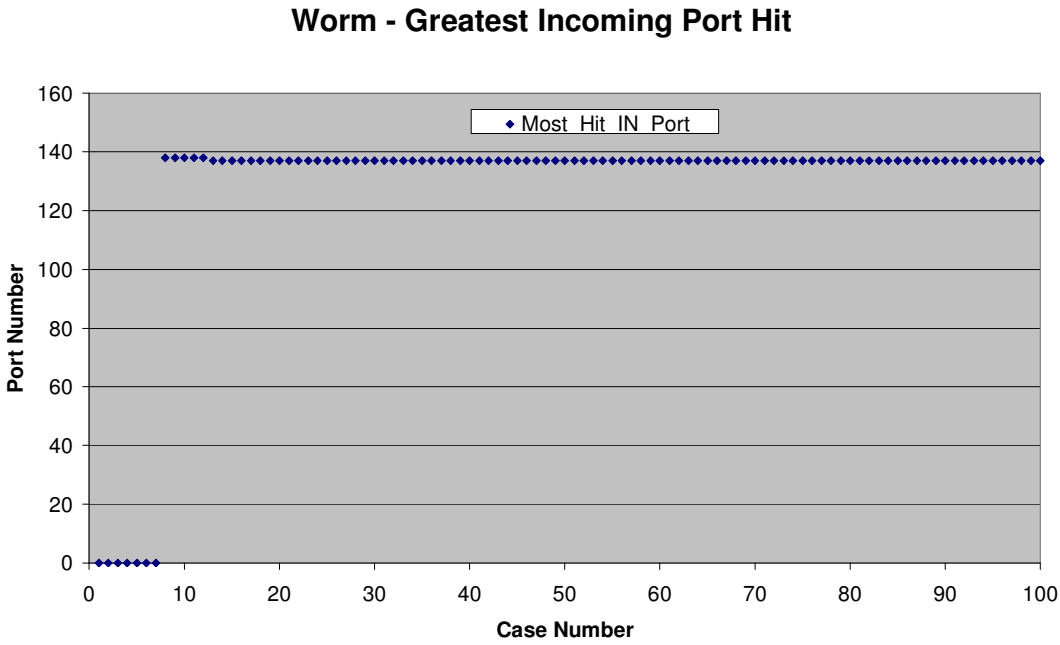

Figure 4-11: The Most Hit Incoming Ports by an Attacking Worm

Figure 4-11 shows that there exists a clear trend in the ports that the attacking Worm is attempting to exploit. The greatest incoming port typically hit is port 137 and port 138. These ports are used by the NetBIOS datagram service (Mueller 2002), for connectionless broadcast services to applications. This type of traffic is quite normal as applications such as the Messenger service use this port for communication. However the amount of traffic produced on these ports by the Worm is telling.

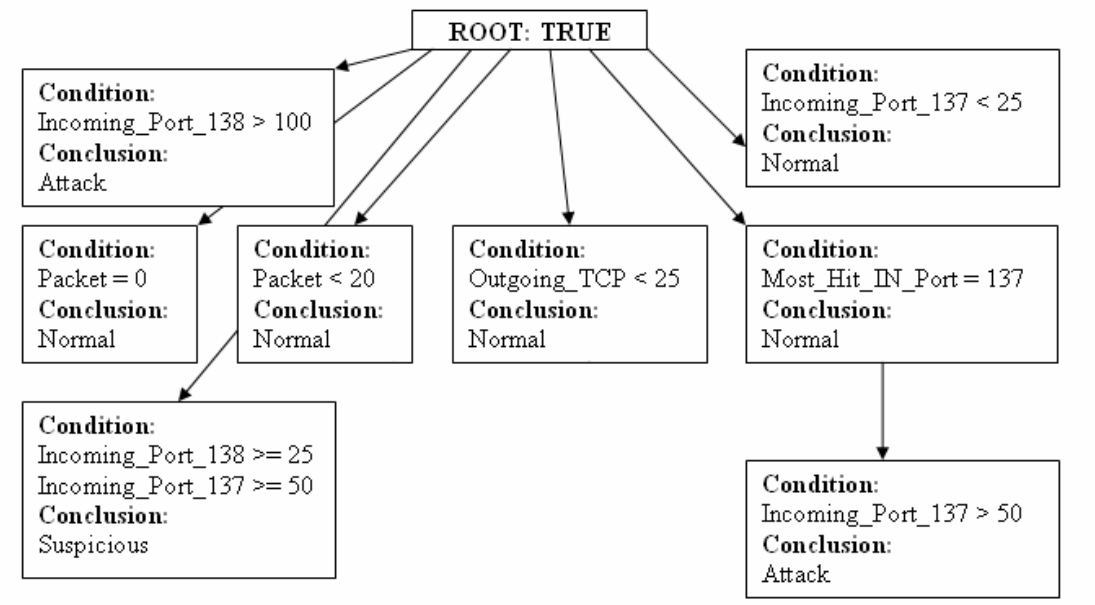

Figure 4-12: Rule Tree created in the Worm experiment

To save space only the core rules are displayed

The rule tree contained in Figure 4-12 demonstrates how the expert identifies a well disguised threat. First of all the expert will identify the conditions that show normality within the host by using very abstract and general rules. As soon as the expert realises that one of the normal rules concluded incorrectly, a refinement rule that adjusts the conclusion is added to the incorrect rule. This can be seen in Figure 4-12 where a rule with conditions "Most_Hit_IN_Port = 137" wrongly concluded Normal and is refined to conclude attack when the condition "Incoming_Port_137 > 50" is satisfied.

4.4 Base-lining

Base-lining is the process of identifying the normal traffic flows that occur on a host or network. This experiment was conducted on a host conducting normal activities while connected to the University network.

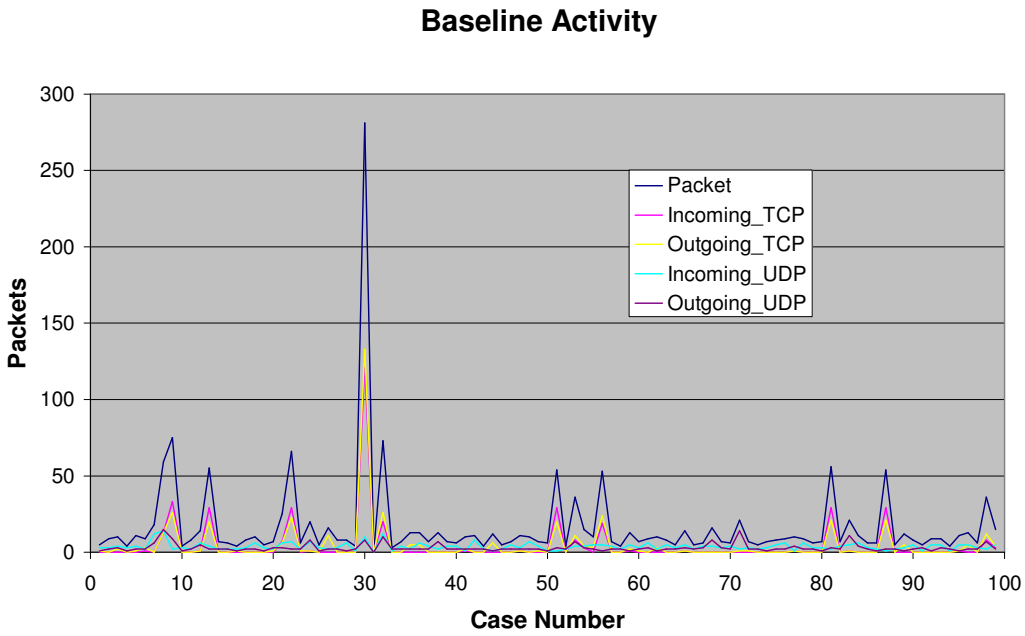

Figure 4-13: Observations of normal host activity

As can be observed from Figure 4-13 the network activities were typically minimal. There are periods, like around case thirty, when the traffic spikes. These spikes are attributed to network based services communicating their availability. The services that are in operation during this base-lining experiment are: an e-mail client, mapped

network drives within Windows Explorer, and network wide anti-virus software. At various points in time each of these services has polled or accessed the appropriate server, which is indicated by the spikes in traffic. The large spike in the traffic is due to all services accessing or advertising to their respective servers at the same time. From this graph it can be seen that the conditions typically have minimal values. The expert will select low type thresholds for each of the above conditions to identify normal traffic. Each of the spikes can be classified as normal using other conditions that clearly identify which service is creating the spike, for example an e-mail based spike can be classified with a separate e-mail logic branch in the rule tree using a POP3 (Post Office Protocol version 3) condition. If the above conditions are used for identifying the spiking traffic as normal then there will be a number of false positives as these conditions are quite generalised. In addition to this if the generalised conditions are used to identify spiking normal traffic then at what level are the thresholds going to be set that will cover legitimate significant traffic situations, such as copying a large file across the network?

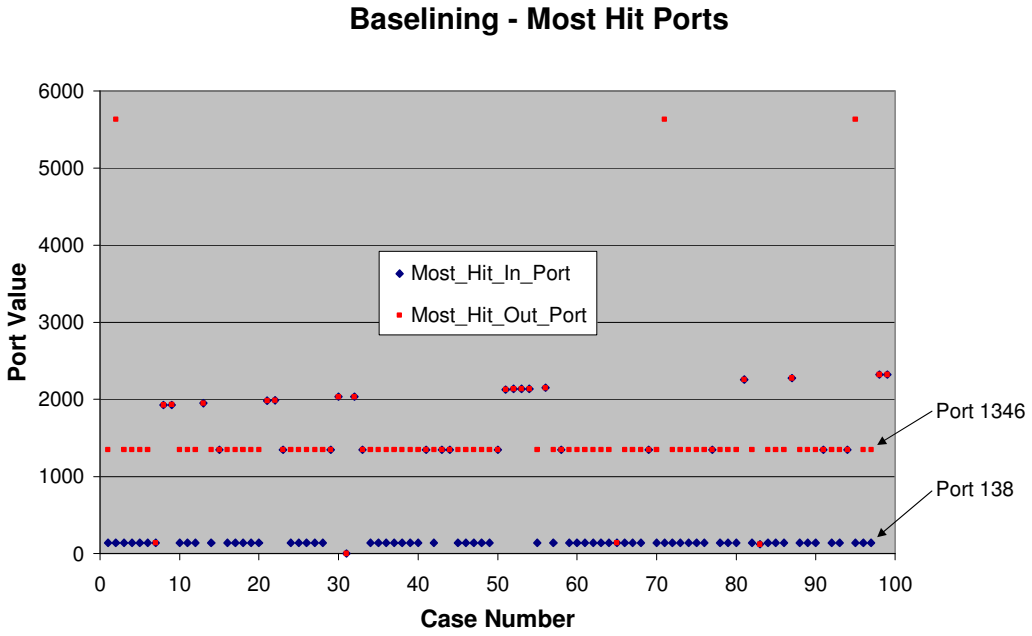

Figure 4-14: The most hit incoming/outgoing ports within any given case

In this graph port values of 0 do not indicate port 0 was the most hit port; rather it indicates that there was no port activity during this case.

Figure 4-14 shows the most hit incoming and outgoing ports during the base-lining experiment. The most hit outgoing port tends to cluster around port 1346 which is

registered as providing a licensing service. The most hit incoming port clusters around ports 137 and 138 which have been mentioned previously as being used for NetBIOS datagram service. These port values and the NetBIOS datagram service have been previously mentioned as being a main identifying condition within the Worm attack experiment. This is a clear indication of the Worm was trying to hide its actions in amongst normal traffic activity. The question that is raised from this is that will the system produce enormous amounts of false alarms trying to identify Worm activity? The answer to this question is that it will require some effort to ensure the threshold level set for the Incoming_Port_138 condition is high enough so that normal traffic is not identified and low enough to ensure Worm activity is detected. However this is not a significant issue as even if the thresholds are originally set incorrectly the monitoring system allows dynamic adjustment of these threshold values.

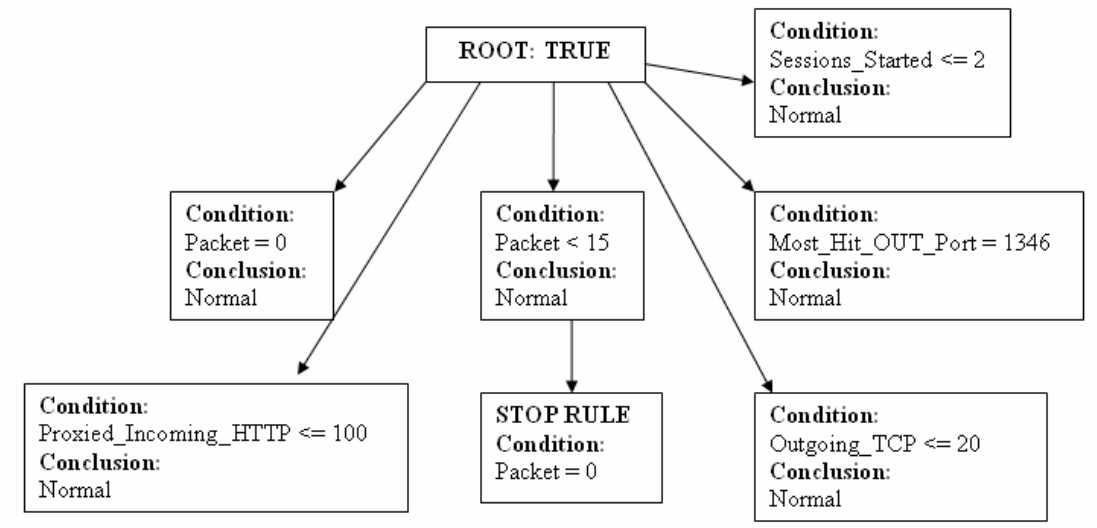

Figure 4-15: Rule Tree created in the Base-Lining experiment

To save space only the core rules are displayed

The expert will determine the conditions that identify the normal activity on the organisational network and edit their threshold values appropriately. If the Worm attack data (see Figure 4-10: The Worm Attack Behaviour) and the base-lining data (see Figure 4-13: Observations of normal host activity) is looked at individually they each have clear, distinct patterns. However when combined together into a single graph it immediately becomes apparent that the Worm activity dominates the graph and hence identifiable.

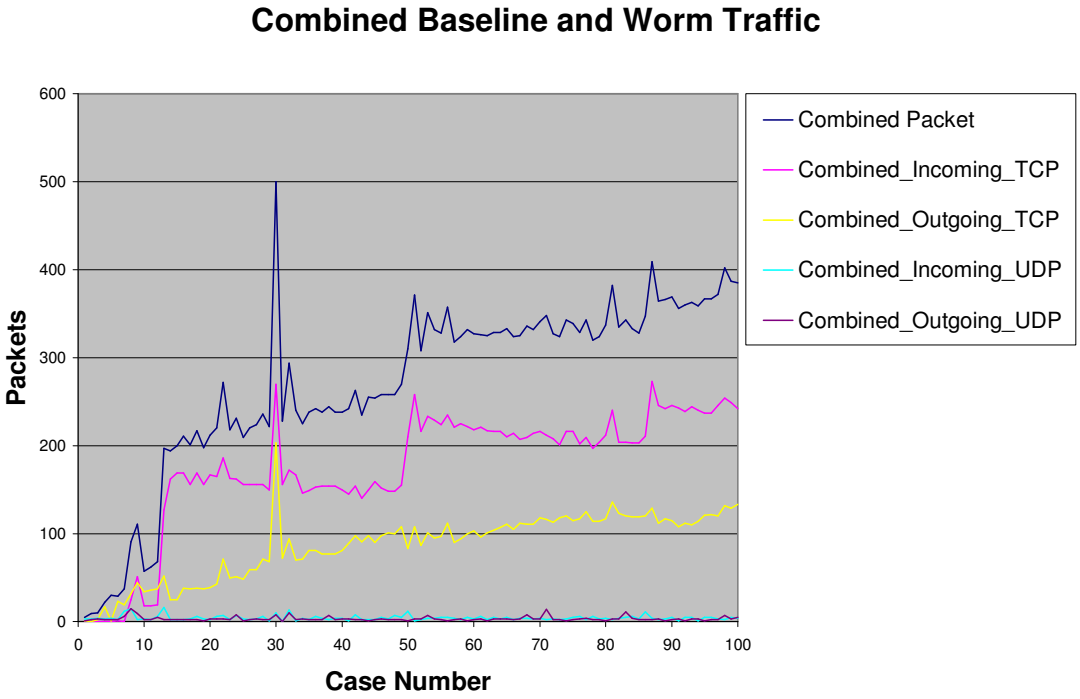

Figure 4-16: Combined Base-Line and Worm Attack Data

After the completion and analysis of the experiments it was discovered that there are two types of generalised threat patterns existed that could be used to group these patterns: peak patterns and incremental patterns. The peak patterns are the threats that exhibit spikes in the condition/s when the attack occurs on the host, such as the E-mail Bomb pattern. Whereas the incremental patterns are the threat patterns that have a generalised increasing trend within the condition/s when it strikes, such as the Worm attack. Further analysis of results will use these two generalised patterns types rather than investigate every individual attack.

4.5 Speed of rule addition

The monitoring system captures the network data in real-time and buffers the cases in a database so that the expert can concentrate on the case that he/she is analysing. What typically occurs is the expert will fall behind the current cases being collected. However when the monitoring system concludes correctly on a case, and the expert has confirmed this, then the expert will catch up to the current traffic analysis data. So the more correct conclusions the system produces, the closer the expert will be to

approaching the currently captured data. The following sections illustrate the time it took the expert to create rules, in the peak threat and the incremental threat pattern types.

4.5.1 Peak Threat Patterns

The peak threat patterns are easier to discover and classify as these patterns contain conditions that exhibit obvious activity spikes when the host is under attack. The expert simply observes the spiking condition/s and creates rules based on these conditions. Figure 4-17 shows the times taken to add rules in the Ping Flood experiment.

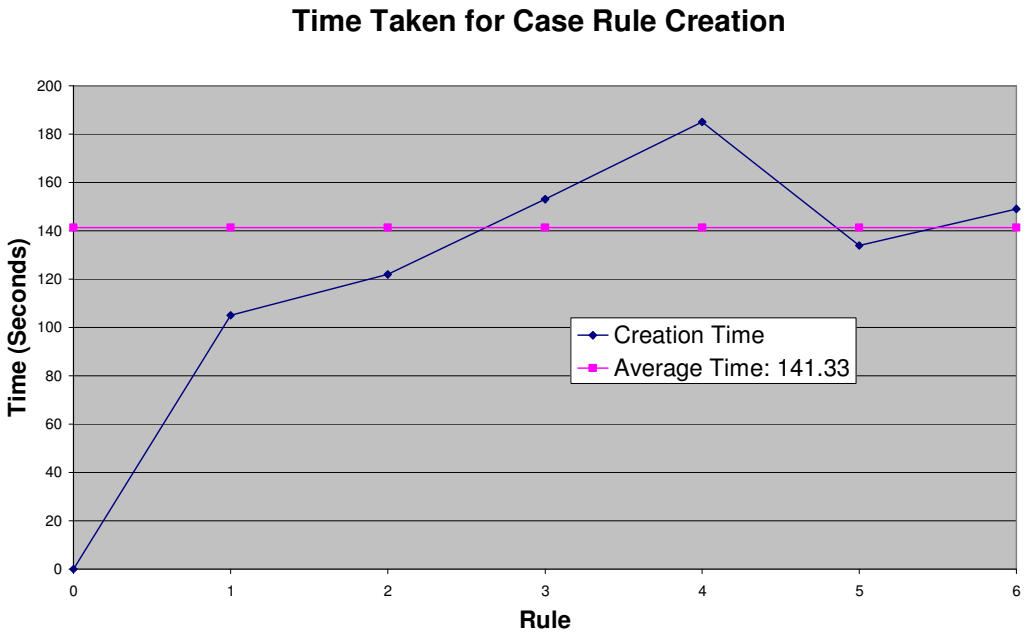

Figure 4-17: Rule Creation Time in the Ping Flood Experiment

As can be observed the time taken to add rules to identify this Denial of Service attack averaged around 2 minutes 30 seconds.

4.5.2 Incremental Threat Pattern

The incremental threat patterns are more difficult to create rules for due to the more intensive analysis required by the expert to identify the pattern. The in-depth analysis is required because the expert wanting to ensure that correct conditions are being

selected to identify the threat to reduce false positives. Also, due to the condition patterns being more difficult to identify, the expert will usually create more abstract rules. As more rules are created the average number of cornerstone cases from which to validate against increase. This has the effect of lengthening the validation/revalidation cycle. Figure 4-18 shows the times taken to add rules in the Worm experiment.

Figure 4-18: Rule Creation Time in the Worm Experiment

As can be observed from the results the average time to add a rule in the Worm experiment was around 3 minutes 15 seconds. Also there exists a slight trend for the rule creation time to increase as the number of rules in the system increased and hence the number of cornerstone cases that need to be validated against also increased.

Overall rule addition within the distributed monitoring system averaged around three minutes. The majority of this time was used by the expert to ensure analysis all of relevant data so that no threat patterns were left undiscovered and normal traffic conditions misclassified. The process of rule addition within the monitoring system is trivial and the time taken for rule creation is comparable to other existing systems. Observations of experts maintaining the GARVIN ES1 system showed that the time

taken to add rules for each case, whether refinements of previous rules or completely new classifications, averaged around ten per hour (Compton & Jansen 1989).

4.6 Accuracy of Threat Detection

The accuracy of the monitoring system in threat detection is a true measure of the performance and capabilities of such a system. If the monitoring system was inaccurate in identifying threats then the usability of such a system rapidly diminishes. The accuracy of threat detection in the system is dependent on the expert being able to establish the logics that will identify the threat now and in the future. The expert does not necessary have to know about every single threat, known and unknown, rather he/she has to apply an understanding of the situation aspects that will identify the threat symptoms.

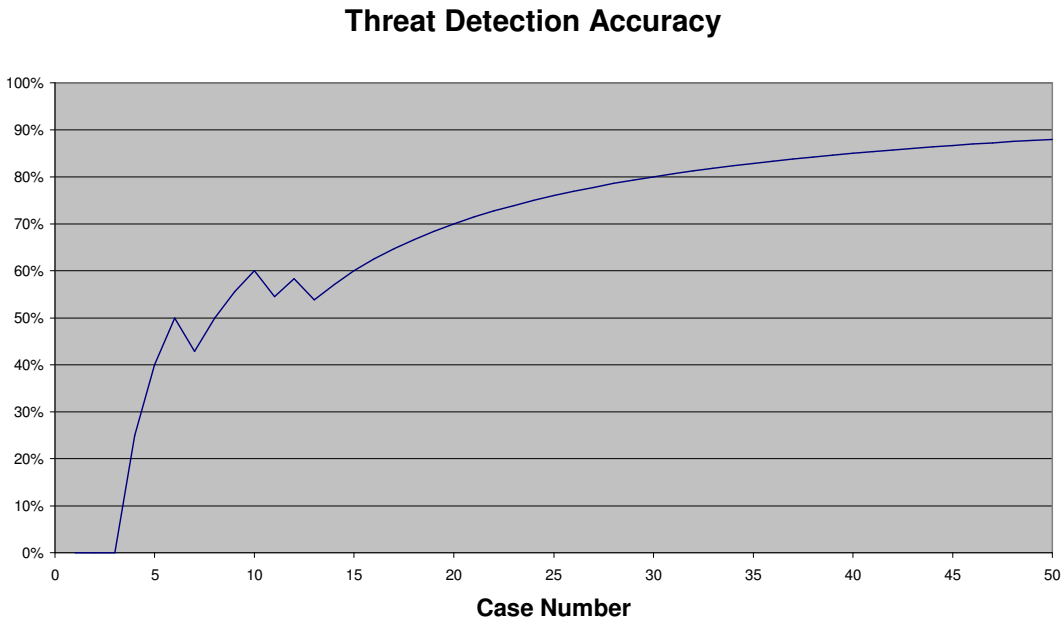

Figure 4-19: Detection Accuracy for the Ping Flooding Experiment

Figure 4-19 shows the accuracy of the monitoring system in identifying a Peak Threat Pattern; ping flooding. The results show that at first the monitoring system cannot identify the threat at all. This is due to there being no logics and situational data to identify the threat. However, as the expert begins adding the logics and knowledge for threat identification the accuracy of the monitoring system improves dramatically. This improvement progresses to a point where the monitoring system is

identifying the threat around 90% of the time. A similar scenario occurs for the threat detection accuracy for the incremental threat patterns.

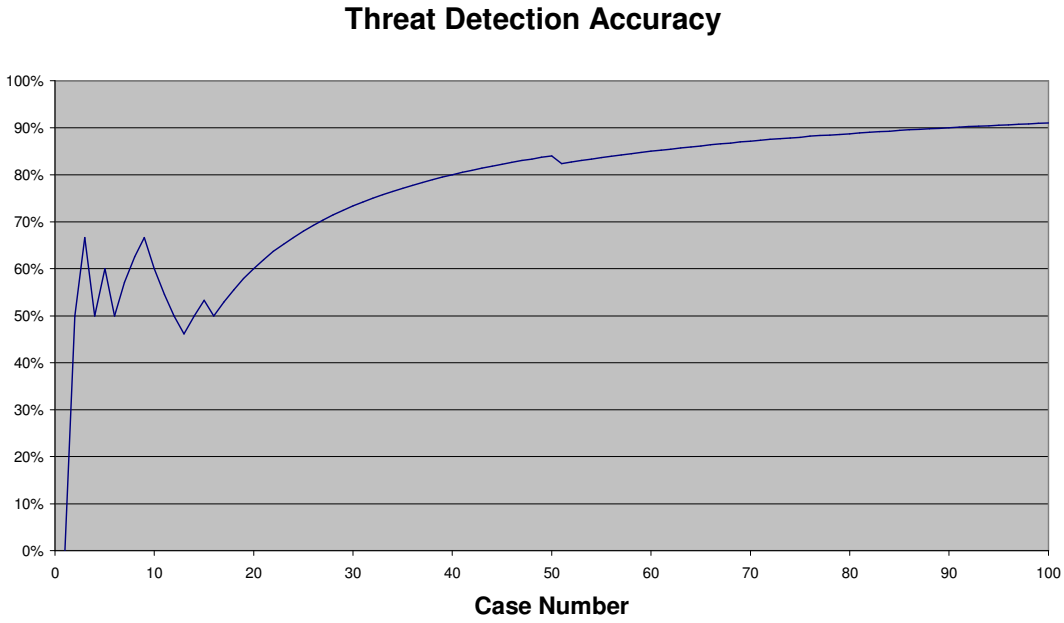

Figure 4-20: Detection Accuracy for the Worm Experiment

Figure 4-20 shows that, although it takes longer to achieve consistent accuracy above 50% due to the expert creating more abstract logics, the monitoring system eventually reaches 91% accuracy in identifying the Worm threat. Results of this nature show that, even though it may take the expert some time to develop the correct logics for identifying any threat, in the long term the monitoring system will be able to adapt and identify any threat with a minimum of false alarms.

The results from the experimentation show that generalised threat patterns at the network level of a host system do exist. The results also indicate that the monitoring system is flexible and can dynamically adapt its logics and thresholds to identify threats on the individual hosts. In addition to these factors the monitoring system reduces the burden upon an expert in the task of creating and refining the rules within the system, to a point where adding rules becomes a trivial task that can be achieved in a minimal amount of time. The monitoring system also achieves these features and functions while still maintaining threat detection accuracy and keeping false alarms to a minimum.

5 Conclusions and Future Work

The research in this study focussed on the production of a distributed monitoring system for use in the detection of unknown threats. The monitoring system developed is driven by logics which allow the expert to correctly identify network threat patterns as they manifest on a host. The logics that are developed over time will give the expert a clear understanding of the way in which the unknown threats behave on the various hosts. The addition of these logics within the system is dynamic and flexible; allowing the expert to create generalised rules that can be further refined or a series of quite specific threat defining rules. It can be seen that the flexible logics allow the monitoring system to adapt to the dynamic environment that exists within networks. The flexibility also allows the monitoring system to be tailored to the specific behaviours and operations of each individual monitored host.

The dynamic and flexible rule structure is only one half of the unknown threat detection equation; reducing the time it takes to create the rules, and hence respond to the threats is the other. The results show that the addition of rules and situational knowledge within the monitoring system is a trivial task for the expert. This gives the expert more time to extract, adapt and apply the knowledge throughout the monitored network. This reduces the response time to the previously unknown threat on the monitored network.

Having an effective response to an unknown threat requires that the system that alerts the expert must be accurate. The experimentation results show that the accuracy of the monitoring system, after the addition of some of the threat defining logics, is high. The accuracy of the system is always improving as the system is run and the logics are refined. This accuracy will keep the number of false positives (false alarms) to a minimum so that the expert is not conditioned into ignoring the system alerts.

Finally it can be concluded that the attainment of the previous aims has proven that the use of MCRDR within the network security domain was successfully applied and is valid within this domain.

Although the system has been proven to achieve its intended goals does not mean that it cannot be improved. The following is a discussion on the possible future extensions that will enhance the system.

Currently the system operates passively; it simply assists an expert to identify and create protection logics for possible threats. However the expert, to further enhance the protection aspect of the monitoring system, requires are a set of modules that can respond to the newly identified threats. If a plug-in action module interface were created then this will allow the creation of response modules that carry out specific tasks. The plug-in interface will allow new threats to be countered with new action modules that are created by the expert.

If a proxy module was created and incorporated within the monitoring system, then it would allow the rebuilding of packet streams to identify the application commands that were being executed on the host. This would allow the possible creation of more, all encompassing logics that identify an unknown threat. For example a FTP stream could be rebuilt and any commands that should not be executed on the server would be identified in real-time.

Finally a server system could be created that interacted and observed each of the monitoring modules on the hosts. This server would have a more abstracted view of the behaviours being exhibited on the network and be able to take actions based on networks and sub-networks rather than individual hosts. Tying the server system in with the distributed monitoring system will mean greater security coverage for the network.

6 References

Aamodt, A & Plaza, E 1994, 'Case-Based Reasoning: Foundation Issues, Methodological Variations, and System Approaches', *AI Communications*, vol. 7, no. 1, pp. 39 - 59.

Al-Tawil, K & Al-Kaltham, IA 1999 'Evaluation and testing of internet firewalls ', *Int. J. Netw. Manag.* , vol. 9 no. 3 pp. 135-49

Amoroso, EG 1994, *Fundamentals of Computer Security Technology*, Prentice Hall, New Jersey.

---- 1999, *Intrusion Detection - An Introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response*, AT&T Laboratories, New Jersey.

Arnold, DN 2000, *The Explosion of the Ariane 5*, viewed 20\10\2005 2005.

Avolio, F 1999, 'Firewalls and Internet Security, the Second Hundred Years', *The Internet Protocol Journal (IPJ)*, vol. 2, no. 2, p. 8.

Avolio, FM & Ranum, MJ 1994, *A Network Perimeter with Secure External Access*, Trusted Information Systems, Incorporated, Glenwood.

Axelsson, S 2000, *Intrusion Detection Systems: A Survey and Taxonomy*, Chalmers University of Technology, Goteborg, 14 March 2000, <http://www.cs.uvic.ca/~wkui/Courses/topic/IDSSurvey.pdf>.

Banatre, M 1991, 'Hiding distribution in distributed systems', in *Proceedings of the 13th international conference on Software engineering*, IEEE Computer Society Press, Austin, Texas, United States, pp. 189-96.

Bass, T, Freyre, A, Gruber, D & Watt, G 1998, 'E-Mail Bombs and Countermeasures: Cyber Attacks on Availability and Brand Integrity', *IEEE Network*, pp. 10 - 7.

CERT Coordination Center 2005, *CERT/CC Statistics 1988 - 2005*, <http://www.cert.org/stats/>.

Chappell, LA 2001, *Packet Filtering: Catching the Cool Packets!*, podbooks.com.

Cheswick, WR, Bellovin, SM & Rubin, AD 2003, *Firewalls and Internet Security - Repelling the Wily Hacker*, 2nd edn, Addison Wesley.

Ciampa, M 2005, *Security+ Guide to Network Security Fundamentals - Second Edition*, Thomson Course Technology.

Compton, P & Jansen, R 1989, 'A philosophical basis for knowledge acquisition', paper presented to 3rd european knowledge acquisition for knowledge based systems workshop.

Compton, P, Edwards, G, Kang, B, Lazarus, L, Malor, R, Menzies, T, Preston, P, Srinivasan, A & Sammut, C 1991, 'Ripple Down Rules: Possibilities and Limitations', paper presented to 6th Banff AAAI Knowledge Acquisition for Knowledge Based Systems Workshop, Banff.

Computer Technology Documents *Network Protocol Levels*, viewed 22nd October 2005, <http://www.comptechdoc.org/independent/networking/guide/netstandards.html>.

Dolan, A 2004, *Social Engineering*, SANS.

Gallant, SI 1988, 'Connectionist Expert Systems', *Communications of the ACM*, vol. 31, no. 2, pp. 152 - 69.

Giarratano, J & Riley, G 1998, *Expert Systems - Principles and Programming*, Third edn, PWS Publishing Company, Boston.

Glymour, C, Madigan, D, Pregibon, D & Smyth, P 1997, 'Statistical Themes and Lessons for Data Mining', *Data Mining and Knowledge Discovery*, pp. 11 - 28.

Griffin, NL & Lewis, FD 1998, *A Rule-Based Inference Engine which is Optimal and VLSI Implementable*.

Hastie, T, Tibshirani, R & Friedman, J 2001, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer-Verlag, New York.

Hayes-Roth, F 1985 'Rule-based systems ', *Commun. ACM* vol. 28 no. 9 pp. 921-32

Janakiraman, R, Waldvogel, M & Zhang, Q 2003 'Indra: A peer-to-peer approach to network intrusion detection and prevention ', in *Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises* IEEE Computer Society, p. 226

Kang, BH 2005, Research Discussion to H Cooke.

Kang, BH, Compton, P & Preston, P 1995, 'Multiple Classification Ripple Down Rules: Evaluation and Possibilities', paper presented to 9th Bannf AAAI Knowledge Acquisition for Knowledge Based Systems Workshop, Bannf.

Kang, BH, Yoshida, K, Motoda, H & Compton, P 1997, 'Help Desk System with Intelligent Interface', *Applied Artificial Intelligence*, no. 11, pp. 611 - 31.

Kemmerer, RA & Vigna, G 2002, 'Intrusion Detection: A Brief History and Overview', *Security & Privacy*, p. 4.

Khoussainov, R & Patel, A 2000, 'LAN security: problems and solutions for Ethernet networks', paper presented to Computer Standards & Interfaces, Belfield, Dublin, Ireland.

Kolodner, JL 1991, 'Improving Human Decision Making through Case-Based Decision Aiding', *AI Magazine*, vol. 12, no. 2, pp. 52 - 68.

Lorimer, S 2003, 'A Real-Time IDS Monitoring Multiple Gateways', University of Tasmania.

Luger, GF 2002, *Artificial Intelligence - Structures and Strategies for Complex Problem Solving*, Fourth edn, Pearson Education Limited.

Massie, ML, Chun, BN & Culler, DE 2004, 'The ganglia distributed monitoring system: design, implementation, and experience', paper presented to Parallel Computing.

McCarthy, J 1984, *Some Expert System Need Common Sense*, Stanford University, California, <http://www-formal.stanford.edu/jmc/>.

Moore, D, Paxson, V, Savage, S, Shannon, C, Staniford, S & Weaver, N 2003, 'Inside the Slammer Worm', *IEEE Security and Privacy*, vol. 1, no. 4, pp. 33-9.

Moriarty, DE & Miikkulainen, R 1995, 'Discovering Complex Othello Strategies through Evolutionary Neural Networks', *Connection Science*, vol. 7, no. 3 & 4, pp. 195 - 209.

Mueller, S 2002, *Upgrading and Repairing Networks*, 3rd edn, Que Corporation.

Murthy, SK 1998, *Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey*, Siemens Corporate Research, Princeton.

Nilsson, NJ 1996, *Introduction to Machine Learning*, Stanford.

Peyton, E 2003, *Corporate Anti-Virus Protection - A Layered Approach*, SANS.

Pfleeger, CP & Pfleeger, SL 2003, *Security in computing*, 3rd Int edn, Prentice Hall PTR, Upper Saddle River, N.J.

Richards, N, Moriarty, DE & Miikkulainen, R 1998, 'Evolving Neural Networks to Play Go', *Applied Intelligence*, vol. 8, no. 1, pp. 85 - 96.

Safavian, S & Landgrebe, D 1991, *A Survey of Decision Tree Classifier Methodology*, School of Electrical Engineering.

Schneier, B 2004, *Secrets & Lies*, Wiley Computer Publishing, Indi.

Schuba, CL, Krsul, IV, Kuhn, MG, Spafford, EH, Sundaram, A & Zamboni, D 1997, 'Analysis of a Denial of Service Attack on TCP', paper presented to IEEE Symposium on Security and Privacy.

Shaw, M 1988, 'Validation in a knowledge acquisition system with multiple experts', paper presented to Proceedings of the International Conference on Fifth Generation Computer Systems, Tokyo.

Shimeall, TJ, Dunlevy, CJ & Pesante, L 2001, *Challenges of Predictive Analysis for Networks*, Carnegie Mellon University.

Shortliffe, E 1976, *Computer-Based Medical Consultations: MYCIN*, American Elsevier, New York.

Spafford, EH 1991, *The Internet Worm Incident - Technical Report CSD-TR-933*, Department of Computer Sciences, Purdue University, West Lafavette.

Staniford, S, Paxson, V & Weaver, N 2002, 'How to Own the Internet in Your Spare Time', paper presented to Proceedings of the 11th USENIX Security Symposium.

Surfer Beware 2004, *Anti Virus Software*, Surfer Beware, viewed 11/05/05 <http://www.surferbeware.com/computer-viruses-information.htm#How%20does%20anti-virus%20software%20work?>.

Weaver, N, Paxson, V, Staniford, S & Cunningham, R 2003, 'A Taxonomy of Computer Worms', paper presented to Proceedings of the 2003 ACM workshop on Rapid Malcode, Washington DC, USA.

Wiener, E, Pederson, JO & Weigend, AS 1995, *A Neural Network Approach to Topic Spotting*.

Witten, IH & Frank, E 2000, *Data Mining - Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann Publishers.

Wulf, WA, Wang, C & Kienzle, D 1996 'A new model of security for distributed systems ', in *Proceedings of the 1996 workshop on New security paradigms* ACM Press, Lake Arrowhead, California, United States pp. 34-43

Yeung, D-Y & Yuxin, D 2003, 'Host-based intrusion detection using dynamic and static behavioural models', *Pattern Recognition - The Journal of the Pattern Recognition Society*, vol. 36, pp. 229-43.

Zou, CC, Gong, W & Towsley, D 2002, 'Code Red Worm Propagation Modeling and Analysis', paper presented to ACM Conference on Computer and Communications Security.

7 Appendices

7.1 Appendix A: The Case Conditions

The following table shows all the conditions used within the system and the description of what each represents.

Condition Name	Description
Packet	This condition identifies the number of packets that were received in the case time. The packets counted include packets not specifically addressed to this host but still received by the network card. This traffic is typically network 'chatter'; that is traffic sent by routers and other network devices for specific protocol purposes.
Time	This condition is the time when the case is created. This is so that time can be used as a condition for a conclusion
Sessions Started	This condition indicates the number of TCP sessions that were started within the case time period. The variable that counts the sessions started is incremented after every successful TCP session no matter whether they are eventually terminated or not.
Reset Connect	This condition indicates the number of reset TCP connections within the case time period. A TCP session is reset when the host detects that there has been a TCP protocol violation.
Active Sessions	This condition indicates the number of TCP sessions that are currently active on the monitored host at the end of the case time.

Source Route Packets	Source route packets are those packets that have been told which route to follow in order to reach its intended destination. This is used by routers for congestion management and by network experts for experimentation. This feature however is typically not used and seen by the host and can be used as a sign that suspicious activity is occurring on the monitored host. When this operation is in effect on a packet flags are set which the system can detect. This condition indicates the number of these packets that were sent and received in the case time period.
Most Hit IN Port	This condition shows the inbound port, UDP or TCP, that has received the most hits. In order to determine whether the packet was destined for an inbound port on the monitored host the packet's destination address is compared to this host's source IP address for a match. In addition to this if the packet's destination address is a directed broadcast to this host's subnet then this will increment this condition as well
Most Hit OUT Port	This condition shows the outbound port, UDP or TCP, that has received the most hits. In order to determine whether the packet originated from this host the packet's source address is compared to the host's IP address for a match.
Num Diff Ports IN	This condition shows the number of different inbound ports that were hit within the case time. The packet is checked to see whether it was destined for this host to ensure that network chatter is not included in this conditions, which would throw off the readings
Num Diff Ports OUT	This condition shows the number of different outbound ports that were hit within the case time. The packet is

	checked to see whether it originated from this host to ensure that network chatter is not included in this conditions, which would throw off the readings
Incoming TCP	This condition shows the amount of inbound TCP or connection oriented activity that took place within the case time period. The system checks to ensure that the packet was destined for this host via direct or broadcast addressing.
Outgoing TCP	This condition shows the amount of outgoing TCP or connection oriented activity that took place within the case time period.
Incoming UDP	This condition shows the amount of incoming UDP or connection-less activity that took place within the case time period. The system checks to ensure that the packet was destined for this host whether it be through direct or broadcast addressing.
Outgoing UDP	This condition shows the amount of outgoing UDP or connection-less activity that took place within the case time period. The system checks to confirm that the originator of this packet was this host. Otherwise network chatter will be included which will throw out the accuracy of this condition
Incoming Port 25	This condition is used to see how much of the packet activity was due to incoming E-mails. Port 25 is used for receiving E-mail while typically port 110 (Post Office Protocol 3 port) is used to check and download E-mail. This condition will be very active on a host that acts as an E-Mail server.
Outgoing Port 25	This condition is used to indicate how much of the

	network activity was due to the sending of E-mails
Incoming Port 771	This condition indicates the amount of activity due to incoming port 771 packets. The port that is used to respond to refused UDP communications.
Outgoing Port 771	This condition indicates the amount of activity due to responses for refused UDP communications.
Incoming Port 68	This condition indicates the amount of traffic destined for this host on port 68
Incoming Port 21	This condition indicates the amount of FTP traffic that is inbound to this host. Typically this host will have FTP server functionality for this condition to increment
Outgoing Port 21	This condition indicates the amount of outgoing FTP traffic from the monitored host. The host will request FTP data from a server which will have a source of port 21.
Incoming Port 138	This condition indicates the amount of traffic destined for the NetBIOS datagram service. This port is used for applications such as messenger services.
Incoming Port 137	This condition indicates the amount of traffic destined for the NetBIOS datagram service. This port is used for applications such as messenger services.
Proxied Incoming HTTP	Due to the configuration of the University network all incoming HTTP traffic will come from the proxy server on port 8080. This condition monitors this as well as any internal University network HTTP traffic destined for this host.
Outgoing HTTP	This condition logs the number of HTTP requests that the monitored host issues.

Incoming ICMP	When another machine pings this monitored host this condition will increment to show this activity. In addition to this other incoming ICMP traffic (such as Destination Unreachable packets)will be logged by this condition.
Outgoing ICMP	When the monitored host sends out ICMP data, such as a ping, it will be detected and logged by this condition.
IGMP	Internet Group Management Protocol allows hosts to participate in multicasting groups. Multicast groups are identified by a single address. IGMP allows a router to determine which hosts are within which groups on a given network segment. This condition monitors all IGMP traffic.
PIM Count	PIM or Protocol Independent Multicast is multicasting technique that is not tied to any routing protocol. This condition logs the traffic that uses this protocol.

Table 7-1: The Case Conditions used in the Monitoring System

7.2 *Appendix B: Detailed Packet Analysis*

The following is all of the logged header information for a single packet. The packet header information is all saved for further detailed analysis by the expert if required. Each packet header and its values are written to a text file separate from the monitoring system.

For reference the following are the header structures contained within a packet.

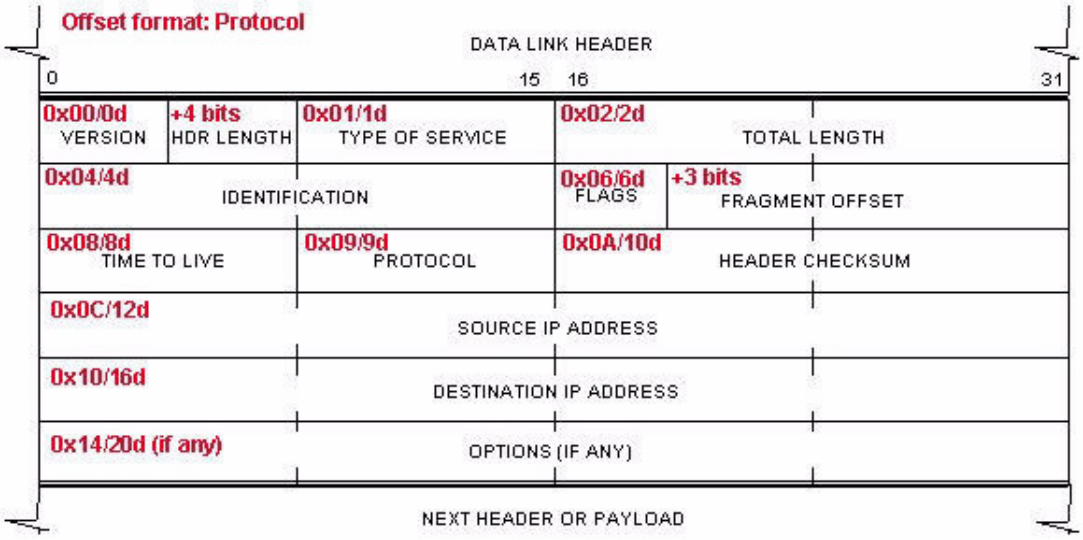

Figure 7-1: The IP Header and its various fields (Chappell 2001)

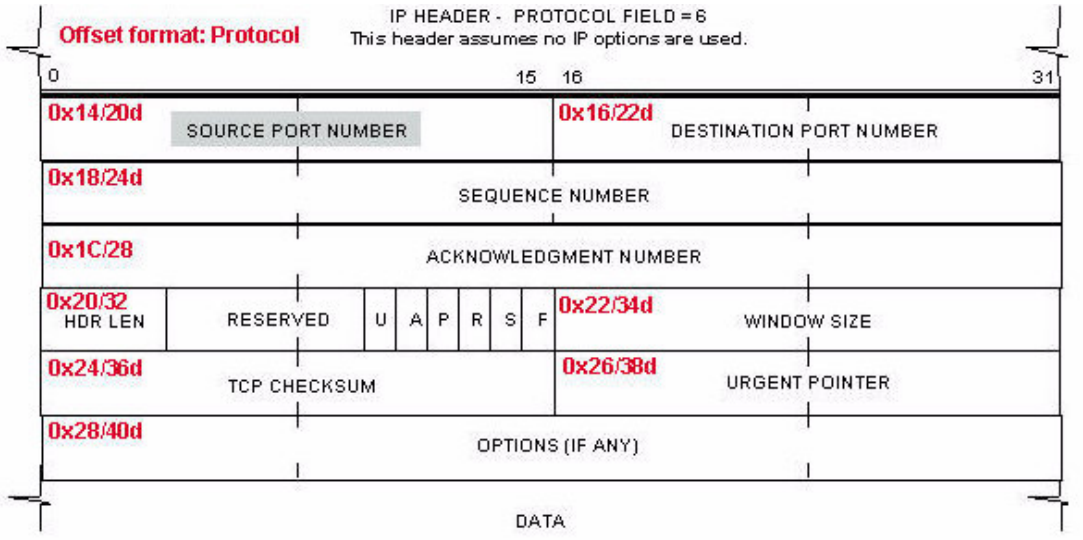

Figure 7-2: The TCP Header and its various fields (Chappell 2001)

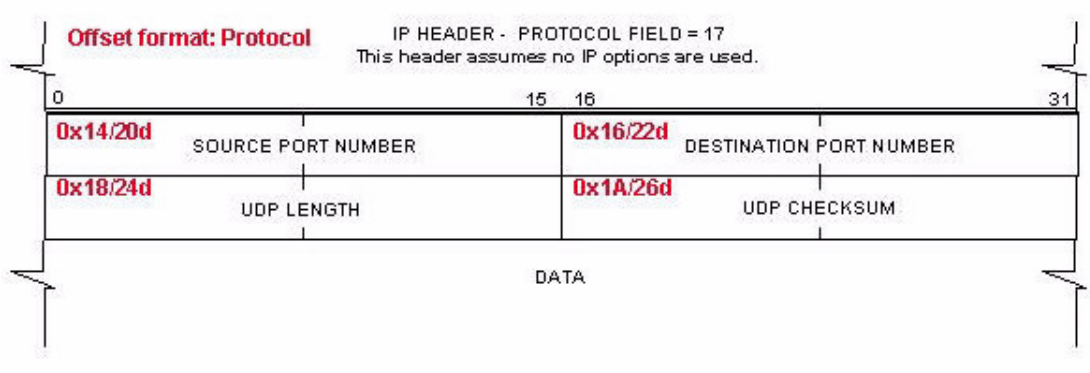

Figure 7-3: The UDP Header and its various fields (Chappell 2001)

The following is an IP packet extracted from the detailed log file. This packet is using the UDP protocol for transmission. The IP addresses have been altered for confidentiality reasons.

IP HEADER:
01:41:45,377929 len:285 XXX.XXX.36.238:1346 -> XXX.XXX.34.25:1347
Version: 4
Internet Header Length: 5
Service Type: 0
Precedence: 0
Delay: 0
Throughput: 0
Reliability: 0

Total Length: 271
Identification: 4e25
Flags & Fragment Offset: 0
Flags: 0
Reserved: 0
Do Not Fragment: 0
Set Fragment: 0
Fragment Offset: 0
TTL: 128
Protocol: 17
CRC: 40191

Option Previous: 5420543
Option Code: 5
Copy: 0
Class: 0
Number: 5

UDP HEADER

Source Port: 1346
Destination Port: 1347
Message Length: 64256
Checksum: 14576

The following is another IP packet extracted from the detailed log file. This packet is using the TCP protocol for transmission. The IP addresses have been altered for confidentiality reasons.

IP HEADER:
01:41:44,766360 len:62 XXX.XXX.34.25:80 -> XXX.XXX.36.82:3294
Version: 4
Internet Header Length: 5
Service Type: 0
    Precedence: 0
    Delay: 0
    Throughput: 0
    Reliability: 0

Total Length: 48
Identification: 4fa3
Flags & Fragment Offset: 0
Flags: 0
    Reserved: 0
    Do Not Fragment: 0
    Set Fragment: 0
Fragment Offset: 0
TTL: 127
Protocol: 6
CRC: 40455

Option Previous: 500cde
Option Code: 0
    Copy: 0
    Class: 0
    Number: 0

TCP HEADER:
Source Port: 80
Destination Port: 3294
Sequence Number: 5e9fc3b5
Header length: 7
Reserved: 0
Code bits: 18
    URGENT: 0
    ACKNOWLEDGE: 1
    PUSH: 0

RESET: 0
SYNCHRONISATION: 1
FINISH: 0

INFORMATION: TCP SESSION ESTABLISHING FROM 131.217.34.25:80 **TO** 131.217.36.82:3294