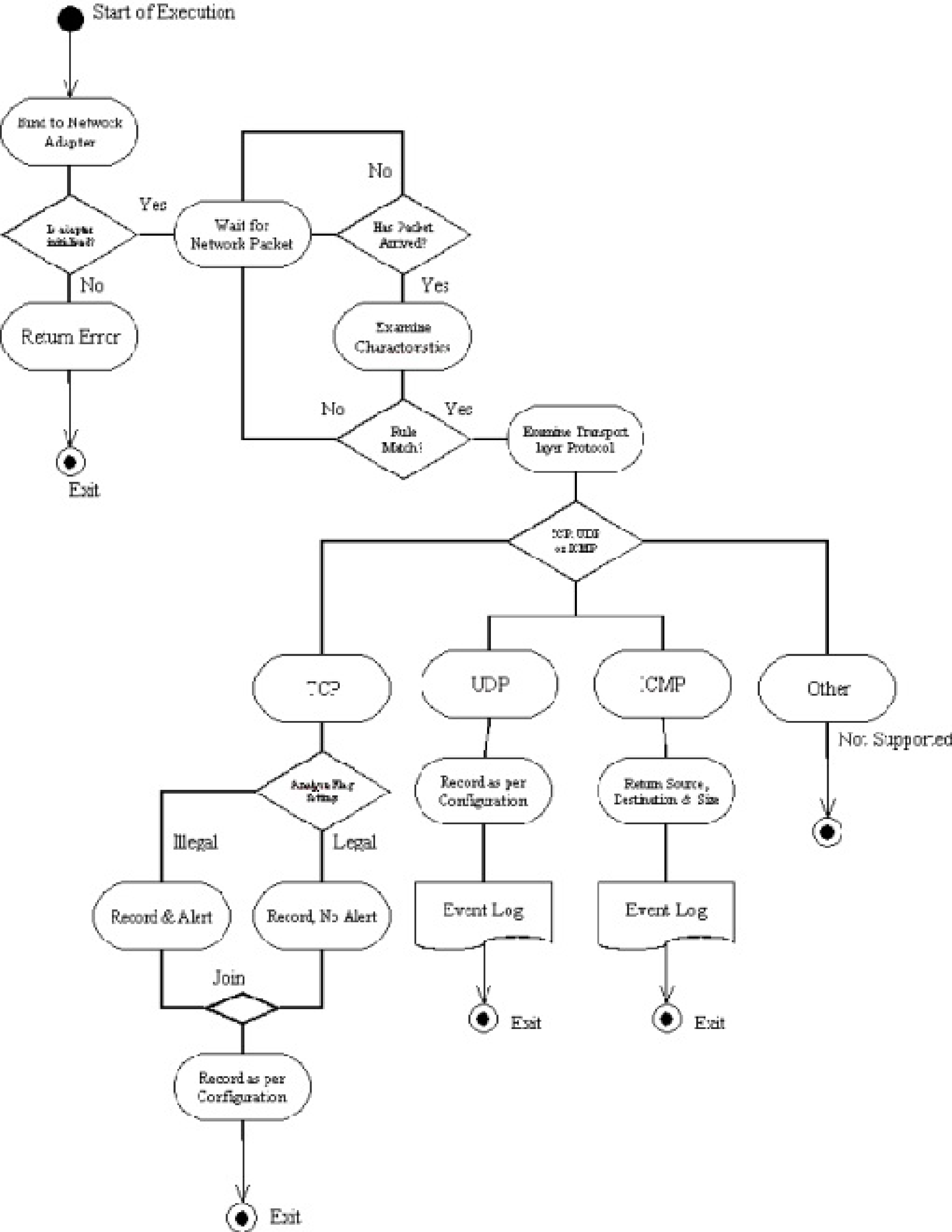

Start of Execution
Bind to Network Adapter
Is adapter initialised?
Yes
No
Return Error
Exit
Wait for Network Packet
Has Packet Arrived?
No
Yes
Examine Characteristics
Rule Match?
No
Yes
Examine Transport layer Protocol
TCP, UDP or ICMP
TCP
UDP
ICMP
Other
Not Supported
Analyse Flag Settings
Illegal
Legal
Record & Alert
Record, No Alert
Join
Record as per Configuration
Exit
Record as per Configuration
Event Log
Exit
Return Source, Destination & Size
Event Log
Exit

ANGLIA POLYTECHNIC UNIVERSITY

Design, Implementation & Evaluation of a Router Based or Stand-Alone Software Intrusion Detection System

David Norris

A Project/Dissertation in partial fulfillment of the
Requirements of Anglia Polytechnic University
For the degree of Master of Science

Submitted: April 2004

Table of Contents

Acknowledgements
Abstract
Abbreviations
I. Introduction
II. Literature Review
III. Legal Aspects
IV. Aims & Objectives
V. Initial Software Design
VI. OSI Model Design Considerations
VII. Program Configuration System Design

VIII. Configuration Based Vs Adaptive Detection
IX. Detection Thresholds
X. Local Connection Status
XI. Testing & Evaluation
XII. Conclusion

Glossary

References

Bibliography

Appendices

I. Existing Intrusion Detection Methodologies
II. Definitions of Attack Types
III. Desirable Characteristics of a Software Component
IV. Development & Testing Environment
V. Security Considerations
VI. Intrusion Detection Systems as opposed to Firewalls
VII. List of Illustrations

Acknowledgements

I wish to thank all those people and organisations, who collectively provided the background information, 'third party' source code, and resources, which were essential to the success of this difficult (and often frustrating!), project.
I would also like to take this opportunity to thank Paul Querelle for his guidance and support during the research, design, and drafting of the project.

Abstract

The aim of the project is to perform a quantitive analysis of the theory, and practical implementation difficulties, inherent in intrusion detection methodologies. This is to be accomplished by analysis of both normal, and anomalous, network traffic, and of the difficulties inherent in the methodologies, whether due to intruder attempts to evade detection, or to practical limitations in terms of resource requirements. An IDS implementation in the 'C' programming language has been attempted.

Although not an 'object oriented' program, [1] *I have used UML to explain and demonstrate the reasoning behind the design.*

I. Introduction

This dissertation is concerned with the development of a 'standalone' or router-based intrusion detection system.
As both networks, and computers themselves, are not designed to protect against every misdemeanor or every possible form of misuse, they are intrinsically insecure. If this were not the case, there would be no requirement for either firewalls or intrusion detection systems.

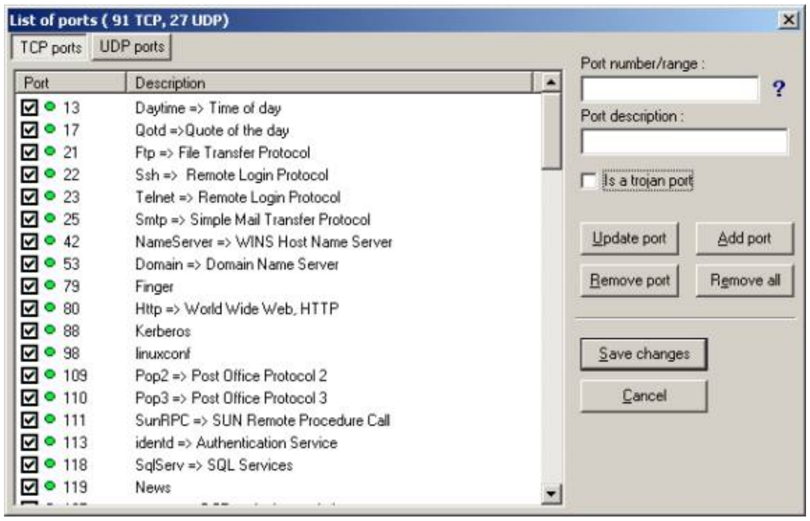
An IDS is intended to be as transparent as possible, in the sense that it monitors network traffic without actually altering its characteristics. The IDS must operate in 'real time', in order to genuinely monitor all of the traffic. In order to accomplish this task, resource usage must be kept to a reasonable minimum. In addition, event reporting must be accomplished to the highest possible degree of accuracy – an IDS which fails to detect 90% of anomalous traffic is actually worse than useless, since it will instill a false sense of security! However, it has to be admitted that the actual implementation will be somewhat of a compromise between the ideal and the practical, as there are inevitably some conflicting requirements to address.

II. Literature Review

At a time when the Internet connected relatively few organisations, securing a network was relatively simple; it implied blocking the few troublesome networks or addresses. However, Verwoerd (2001), states that manual response to attacks is no longer feasible. As a result of a proliferation of malicious program code and a variety of types of attack, many methodologies have been devised to detect them. Northcutt (2002), points out that as frequently attacked sites may see hostile activity from 15 to 60 addresses per day, commercial systems are even expected to highlight addresses which have already been reported!
However, it is evident that the development of viable intrusion detection methodologies is very much impeded by conflicting requirements. Essentially, it is a compromise between functionality, and a need to minimise resource usage. This is particularly difficult where 'stateful' network protocols are concerned; there is little point in designing a system able to remember the precise details of every network event recorded during the previous several weeks of operation, which uses an intrinsically complex detection algorithm, involving hundreds of separate binary operations, when the resource requirements are so great that your IDS drops considerable numbers of packets, possibly missing the remainder of an event which it is attempting to analyse. This is summarised by Ranum, (2001); "Detection of hostile actions becomes a matter of probability, and the likelihood of detecting a given hostile action depends on the number of sensors arranged to detect it." Therefore, it is a requirement to employ more than a single method of detection.
Thompson, H.H, (2002), states that a survey conducted in the same year by the FBI demonstrates that not only do malicious internal actions account for 80% of all network security incidents, they also, on average, cost an organization almost 50 times the financial loss inflicted by external incidents. However, such incidents also have indirect consequences. The process of recovering from network-based attacks can be extremely prodigal of employee hours; this can alone result in significant financial loss. In part, such lapses result from inexperience or lack of awareness on the part of the user. "Logically, one must bound what one is protecting before one can analyze how well one is protecting it." Schumacher, (1997). Hinde, (2003) reports that in a survey of UK businesses, 12% of participants had no virus protection, whereas 43% had no firewall protection. Zona Research (1998) found in their own survey in the US that although many organisations did take security seriously, they often 'were addressing the wrong issues'. Failure to keep their systems and policies up to date was the primary cause. The main implication for IDS is that it must be *readily adaptable*. The model on which IPGateKeeper's configuration is based is the "Attack Taxonomy", Barrus, (1998).

Another design consideration for an IDS is that it should be versatile; as information becomes less centralised, due to factors such as telecommuting, the use of portable computing devices, and ‘wireless’ networking, an IDS implemented in software rather than hardware becomes attractive. “Although appealing in terms of system design and extensibility, mobile programs are a security risk and require strong access control. Further, the mobile code environment is fluid, i.e. the programs and resources located on a host may change rapidly, necessitating an extensible security model.” Hashii, (2000). Wood, (N.d), points out that whereas once mission critical data was stored centrally on a mainframe, security is now no longer centralised. Third-generation cellular telephones are another example of decentralization. “Because a user may be able to log into a computer from anyplace in the world (e.g., using telnet or a dial-up line), there is no way of identifying the geographic location of a user even when the location of the computer where the account is held is known. With mobile phones and computing, the location of the user becomes even more difficult to determine.” Adkins, B.N. (2001). This is not the only problem facing security countermeasures. Since network development rarely remains stagnant, Hunter (N.d), asserts: “The main logistical concern is that a centralised approach does not scale well with network growth”. Therefore, it is reasonable to expect the trend away from centralised IDS to continue.
Intrusion detection is essentially the ‘science’ of the detection of network traffic, which does not, in any regard, obey the normal ‘rules’ governing network traffic. According to Bierman, (2001), “One of the main approaches of IDS, namely anomaly detection, is based on the assumption that an attack on a computer system will be noticeably different to normal system activity”. However, this assumption is seldom straightforward. Proctor (2001), specifies that any IDS is faced with much the same difficulty as that faced by Antivirus vendors; since the battle between those trying to protect their assets, and those developing new methods by which to breach system security is ongoing, adequacy essentially implies ‘being up to date’. As Worstell (2000), explains, ” Every new technology, product, or system brings with it a new generation of bugs and unintended conflicts or flaws.”
Software design is a science very much in its infancy. Northcutt (2001), explains that “although designing functional software is the key objective of software developers, few place such emphasis on designing software which functions securely.” This is one key problem; another is the attempt by designers to advertise ‘feature richness’ by including superfluous services as part of a default installation. As quoted in a white paper, Feingold (1994), attempting to determine whether a system has already been compromised can prove difficult. *“Modern operating systems are large, complex, and imperfect dynamic systems, with many places for attackers to hide and many opportunities for them to cover their tracks.”*
One common dilemma involves accessing the potential severity of even known threats. According to Northcutt (2003), analysts use the GIAC formula by which to calculate severity: (Criticality + Lethality) – (System + Network Countermeasures).[2] Additionally, as Beckers, (2003) explains, an individual attack of any type can vary in severity; depending on the nature of the target. “When attacks are directed at targets with the wrong operating system, they cannot be successful.” For this reason, I made a decision to make all detection configurable, allowing all activity, or only ‘significant’ events to be monitored.[3] “Detection is important, but there is such a thing as too much of a good thing.“ Proctor, (2001).

For ‘commonly’ encountered services, I have also assigned a rough probability and risk factor measure. [4] A brief description of common session layer protocols is provided in the ‘event specific’ logs; to facilitate this scheme. Although this may seem somewhat superfluous, many products provide such details; this screenshot is from the Languard Network Security Scanner.



Port	Description
13	Daytime => Time of day
17	Qotd =>Quote of the day
21	Ftp => File Transfer Protocol
22	Ssh => Remote Login Protocol
23	Telnet => Remote Login Protocol
25	Smtp => Simple Mail Transfer Protocol
42	NameServer => WINS Host Name Server
53	Domain => Domain Name Server
79	Finger
80	Http => World Wide Web, HTTP
88	Kerberos
98	linuxconf
109	Pop2 => Post Office Protocol 2
110	Pop3 => Post Office Protocol 3
111	SunRPC => SUN Remote Procedure Call
113	identd => Authentication Service
118	SqlServ => SQL Services
119	News

Fig. 1: Languard Security Scanner, by GFI Software LTD

There are two main reasons for the provision of both summaries of activity, (History.htm, statistics.htm) and 'packet specific' records, as determined by the 'Record=' command in the configuration. Firstly, not all details can be accommodated in the summaries, due to cosmetic constraints. An overly complex 'summary' would be somewhat difficult to interpret at a glance, and is therefore 'self defeating'. However, the packet level data is also vitally important, as it provides this critical evidence and detail – it is the background to events appearing in the summaries. "If your intrusion detection tool only stores alerts that were generated from analysis of raw data (event logs or TCP/IP data), without the raw data to back up the conclusion, the veracity of the detection may immediately be called into question." Proctor, (2001).

Bace (2000), points out that it is safest not to rely on a single security mechanism, as there is safety in redundancy. Therefore, it is best to use multiple detection schemes in parallel, as attempted by IPGateKeeper. For example, the monitoring of connection attempts to disparate ports provides no detection of suspect packet payloads, IP header lengths or illegal combinations of TCP flags.

A particular difficulty faced by an IDS concerns insertation/evasion attacks. As pointed out by Newsham and Ptacek (1998), an IDS may 'see' traffic differently to the target host. In the Appendix 'Security Considerations' I have explained the nature of the difficulty.

In order to remain anonymous, the perpetrators of certain attacks, which do not require a response to the initiating IP address, falsify the source addresses in the IP header, a practice known as 'spoofing'. "Unfortunately, the addresses were not designed to provide authentication, and an adversary can take advantage of this by forging an artificial request." [5] Heberlein, (N.d). One of the most common types of attack using false source addresses is the DDOS attack. These attacks are possible due to 'Raw Sockets", allowing packets to be written directly to a network interface, bypassing the protocol stack. "With little or no advance warning, a DDoS attack can easily exhaust the computing and communication resources of its victim within a short period of time." Douligeris, (2003).

Even in lieu of forged traffic, the IDS designer has to face the very real difficulty of outputting the data in a format admissible as evidence. "Legal proof is very different from "scientific proof". Scientific proof is based on scientific investigation, while legal proof is more about the rules of admissibility and what can be convincingly presented in court." Proctor, (2001).

However, since anomalous traffic can occur for a variety of reasons unrelated to malicious activity, any IDS is liable to generate both 'false positive' and negative alerts. A false positive causes needless anxiety – but a false negative is the worse scenario. According to Beckers (2003), "Most companies would rather have false positives than false negatives, but false positives are a common way to hide a real attack".

Finally, one particular area relevant to IDS, concerns the growing trend for less reputable software vendors to bundle commercial 'spyware' with otherwise legitimate products, without either customers knowledge or consent. "Spyware and 'adware' programs are being deliberately included with other legitimate software packages to deliver functionality that users do not want or need." Schultz, (2003). Here, I make a prediction that the detection of such 'Spyware' will become an increasingly desirable feature for IDS.

III. Legal Aspects

One difficulty faced by legislators is that, whereas technology continues to advance rapidly, legislation is, by definition, updated only occasionally. Many countries have yet to address the problem of Internet misuse. This is particularly pertinent to developing countries, where Internet access is currently rarified, but likely to become commonplace given time. The primary enforcement difficulty, however, is political. "The web is a global phenomenon, so law enforcement and the legal system are going to deal with more than one country. Each country is going to have its own rules and procedures for handling computer forensic evidence." Proctor, (2001).

However, even in many developed countries, legislation is in need of updating. In the United Kingdom, there is existing legislation concerning the use of computing facilities:

- The Computer Misuse Act, 1990 (c. 18): "An Act to make provision for securing computer material against unauthorised access or modification; and for connected purposes." [6]
- The Data Protection Act, 1998 [7]
- Privacy and Electronic Communications (EC Directive) Regulations 2003: "protection of fundamental rights and freedoms, and in particular the right to privacy, with respect to the processing of personal data in the telecommunications sector."
- The Interception of Communications Act 1985 (*IoCA*).
- The Regulation of Investigatory Powers' Act 2000 [8]
- The Communications' Act 2003

The above directives all apply to the usage of the Internet, within the United Kingdom. In general, the Computer Misuse Act 1990 makes it an offence to 'knowingly secure unauthorised access to any program or data held within any computer'. This offence applies whether or not the computer in question is located within the United Kingdom. The above offence, although fairly serious, is likely to be viewed still more seriously if it is committed with intent to commit, or facilitate, further criminal activity.

The main shortcoming of the Computer misuse Act is that, as it came into effect before the Internet was in widespread public use; it does not apparently address certain types of Internet based attack, for example DOS. This is due to the fact that it addresses unauthorised access to a system: "To establish guilt under the Act, there must be
an intention to secure unauthorised access and the perpetrator must know at the time of seeking access that there is no authorization to secure such access." Dautlich, (2003). The act also addresses the intent to "modify

the contents of any computer and by so doing to impair its operation, or to prevent or hinder access to any program or data held in it, or to impair the operation of any program or the reliability of any data." Computer Misuse Act, 1990, Section 3(1). Since DOS does not imply intent to "secure access to a system", or "modify data", it would appear that the legislation is in need of updating, so as to address DOS attacks.

As the Data Protection Act is concerned with unlawfully obtaining personal data, an individual or organization in breach of the Computer Misuse Act may also be in breach of the Data Protection Act.

The Privacy and Electronic Communications Regulations Directive, and the Regulation of Investigatory Powers Act, have potential implications for the design and deployment of an IDS, since it is effectively a type of wiretap.

With certain exceptions, it is an offence under Section 1 of the Regulation of Investigatory Powers Act (2000) for a person *"intentionally and without lawful authority to intercept, at any place in the United Kingdom, any communication in the course of its transmission by means of:* a public postal service, or a public telecommunication system." It is similarly an offence to intercept any communication in the course of its transmission by means of a private telecommunication system. However, it is permitted for an individual or organization to do so within *their own* communications system. This does, however, raise a number of interesting moral considerations, since in many organisations, managers are more interested in using firewalls and intrusion detection systems to spy on their employees, than to ensure the security of their networks.

The testing of an IDS can also potentially raise legal problems, due to the need to 'simulate' attack scenarios, and produce result sets to support your argument. In particular, port scanning (the act of scanning another machine to discover what services are running), is normally carried out by intruders attempting to discover insecure systems; therefore it will be regarded as a hostile act. Although there have not as yet been any prosecutions in the United Kingdom relating to scanning alone, it is possible that this activity may be interpreted as being contrary to the Computer Misuse Act, on the grounds of 'intent'. An analogy would be that although there is no actual offence in carrying out a survey to 'discover the percentage of the population who forgot to lock their doors', in practice no official is likely to accept this as an explanation! [9] Even where the act of reconnaissance scanning alone did not contravene legislation, this would certainly fall short of best practice if carried out on a public network.

Therefore, any testing must be carried out within a *private* network. [10] However, records generated by the IDS as the result of *incoming* probing activity can legitimately be used to demonstrate the operation of the IDS.

Where a given test has involved the use of falsified addresses (for example, to demonstrate the appearance of 'third party' traffic), or where a tool utilises random addresses, this has been clearly stated. The testing was conducted within the confines of my private LAN.

IV. Aims & Objectives

i. Design Objectives

The purpose of the software is to accomplish the following:

1 To monitor Internet Protocol traffic on arrival at the network interface adapter;

2 To distinguish between the transport layer protocols we wish to monitor; [11]

3 To identify those characteristics which define ‘abnormal’ network traffic events, such as those identified above;

4 To identify patterns inherent in particular types of ‘attack;

5 To either accurately record, or generate an alert, in the event of cases 3 and 4.

6 Ideally, the output should be in an easily recognizable form, as this facilitates Human Computer Interaction (HCI). This may be accomplished by formatting data according to the nature of the event. The system clock must be synchronised using NTP in order to ensure accuracy.

ii. Scope

Network Security is a never-ending struggle to defend an individual or organization's assets and reputation against constantly changing threats. Every release of a network facing software component has a vulnerability waiting to be found; [12] once discovered, automated 'probing' activity quickly follows. Although an organization can reduce the risk posed to its assets by using a firewall in order to exclude 'inappropriate' traffic, this cannot be guaranteed to be entirely effective.

"Despite the effort devoted to carefully designing such filtering, network security is very difficult to guarantee, since attacks exploit unknown weaknesses or bugs, which are always contained in system and application software".

Proctor, P.E, (2001), Practical intrusion detection handbook, London, Prentice Hall.

Lemonnier, (2001), defines IDSs as "monitoring programs aiming at detecting 'intruders' who are acting 'illegally' in a computer system."

The basic purpose of an IDS, is to detect, and/or either immediately generate an alert, or record the details of, an intrusion. Unfortunately, not everyone agrees on what precisely *constitutes* an intrusion. For the purpose of this project, a 'Suspicious Incident' implies an occurrence of network traffic, which is non-compliant with expected characteristics. *"Intrusions correspond to anomalous network activity, i.e., to traffic whose statistical profile deviates significantly from the normal"* McHugh, J. (2000).

One of the main difficulties inherent in the detection process is due to the sheer number of hardware and software components, which together determine the precise characteristics of the network traffic itself. An IDS must be partially modeled on the network topology:

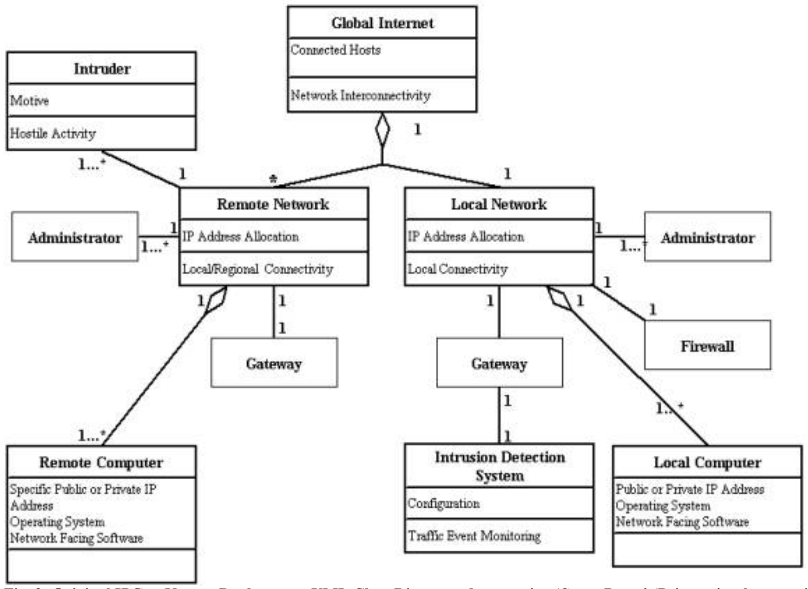


Fig. 2: Original IPGateKeeper Deployment UML Class Diagram; drawn using 'SmartDraw' (Prior to implementation).

However, the same analogy cannot readily be made within the network traffic environment. Any particular type of traffic, which is common on a given network, may be absent from another network during normal operation. Furthermore, although in theory network protocols are defined in the relevant RFC (request for consultation) documents, we face the problem that minorities of software applications do not adhere strictly to the specifications. Additionally, new features may be added to a protocol, which were not included in the original specification. As an example, the first two TCP flags, which were once reserved, have since been allocated.

It follows that therefore, any intrusion detection system is liable to generate false alerts [13]. Although it may be possible to minimise these, there is no means by which they may be completely eliminated.

iv. Method of Development

At an early stage, I took a decision to design the software 'From the Ground up'. Firstly, to develop a means by which to interface to the network, Secondly, to process the traffic at packet level, detecting known attack signatures, thirdly, performing sequence analysis, in order to identify known suspicious sequences in network traffic. "A system is a purposeful collection of interrelated components that work together to achieve some objective". Sommerville, (2000).

This 'evolutionary' design method has a distinct advantage, since system testing can be performed at *any* stage in the development process. This provides early opportunities to remedy possible malfunctions; deviance from the required operation can be remedied as soon as it becomes apparent, rather than later in the development process, when the cause may not be as obvious. Additionally, regular testing throughout the development process also provides opportunity for easier identification of erroneous code. [14]

"As few as 20% of the modules in a program can be responsible for as many as 80% of the defects".

B.W. Boehm, (1987).

V. Initial Software Design

The operating requirements are best expressed using UML. This allows the reader to easily visualise the sequence of events, which take place during the operation of the software, and to see at a glance how the various components interact with each other in order to accomplish the desired end result. At low level, an IDS needs to:

- Search for anomalous features of packets;
- Search for anomalous sequences of packets;
- Search for packets, which simply should not be present.

Figure 3 [15] illustrates the basic sequence of operations required:

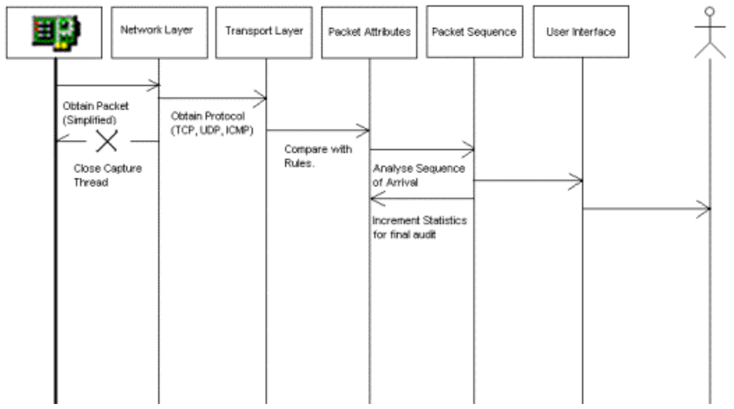

Network Layer
Transport Layer
Packet Attributes
Packet Sequence
User Interface
Obtain Packet (Simplified)
Close Capture Thread
Obtain Protocol (TCP, UDP, ICMP)
Compare with Rules.
Analyse Sequence of Arrival
Increment Statistics for final audit

Fig. 3: IPGateKeeper UML Operating Sequence Diagram

The definitions are as follows:

- The network interface is the 'source' of packets, insofar as the IDS concerned;
- The network layer protocol in question is the Internet Protocol, the header of each packet provides required parameters which determine the handling of each packet;
- The transport layer protocols of interest are TCP, UDP and ICMP; [16]
- The packet attributes are then compared with the configuration in order to determine whether a match has occurred (this in fact utilises both IP and TCP/UDP/ICMP attributes); for TCP alone, a flag examination is performed;
- The incoming traffic and responses may be taken into consideration; for example UDP scanning generates ICMP 'port unreachable' responses;
- Finally, where required, the relevant details are presented to the user – this may involve an immediate alert, or the saving of the relevant details for later analysis, or possibly a combination of the two. However, as the process of outputting data to the display is very time and resource intensive, it is essential to minimise the usage of the display.

Fig. 4 is a (simplistic) general interaction diagram representing the operation of IPGateKeeper.

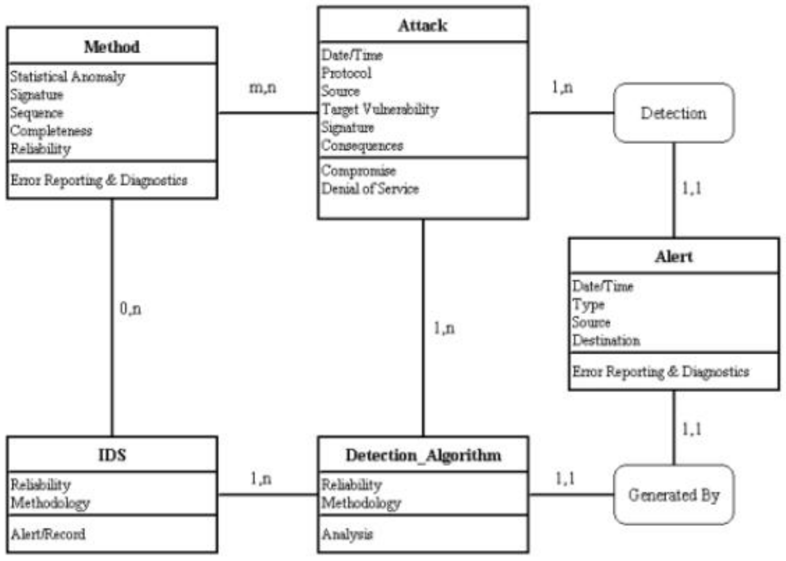
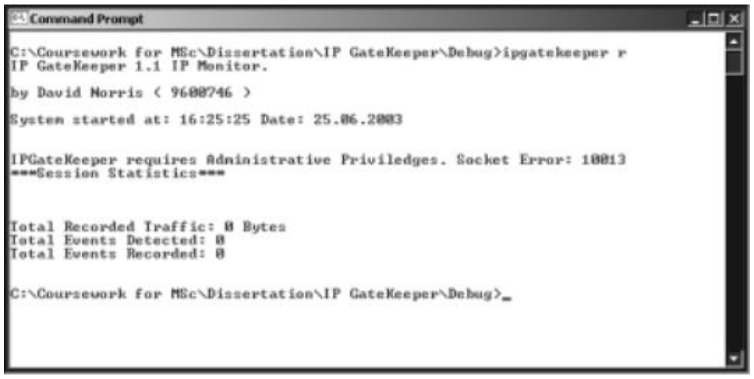


Fig. 4: Simplified IPGateKeeper UML Interaction Diagram

One particular problem to be overcome concerns the nature of the network traffic itself. Traffic does not flow evenly; there may be quiescent periods during which there may be relatively little activity; at other times, congestion may occur with many thousand packets arriving per second. In order to avoid either slowing down the system on which the IDS itself is running, and possibly missing events, or generating numerous false alerts, it must be possible to configure it with great precision and accurately. [17] However, it is also useful to relax the configuration during testing, in order to assist the test procedure itself. I have therefore decided to allow configuration of certain parameters via an .ini file.

Since IPGateKeeper requires access to 'Raw Sockets', it can only be run with administrative privilege. This can be accomplished using the 'Runas' command, in order to avoid having to run the whole system under an administrator account. [18] It is inadvisable to run as administrator during normal usage, as this makes the system vulnerable to a large number of dangers. In particular, it is much easier for an intruder to exploit security holes in processes, which are being run with administrative access. Unfortunately, this cannot be avoided for the IDS itself. Running the program as a normal user results in the WSAAccess error condition 10013. [19]



```
C:\Coursework for MSc\Dissertation\IP GateKeeper\Debug>ipgatekeeper r
IP GateKeeper 1.1 IP Monitor.

by David Norris ( 9600746 )

System started at: 16:25:25 Date: 25.06.2003

IPGateKeeper requires Administrative Priviledges. Socket Error: 10013
***Session Statistics***


Total Recorded Traffic: 0 Bytes
Total Events Detected: 0
Total Events Recorded: 0

C:\Coursework for MSc\Dissertation\IP GateKeeper\Debug>_
```

Fig. 5: WSA Access Error 10013

A particular difficulty, which became increasingly apparent at this stage, was that of timing. In order to sample network traffic, it is vital to analyse packets in ‘real time’, without excessive system resource overheads. As the analysis of packets does in itself require a certain time interval, δT, there is the problem that, on a high bandwidth network, that, during the time interval δT, more packets will arrive. As analysis of the original packet has not yet run to completion, these packets will not be detected. Therefore, only the first packet in this ‘sequence’ will be analysed! This would allow the vast majority of anomalous activity to bypass the IDS. The number of packets in a ‘sequence’ will depend firstly on the bandwidth of the network itself, secondly on the current state of the network, and thirdly, on the throughput of the local CPU.
Therefore, in order for ‘real time’ handling of the network traffic to be possible, the following requirements must be implemented:

- Careful selection of traffic for analysis – there is no point in attempting to analyse packets which are not of interest, for example, packets which are not of the TCP, UDP or ICMP protocols;
- The need for a ‘default ignore’ configuration; although this may sound counterproductive for an intrusion detection system, an experienced administrator will know which protocols and events they wish to monitor;
- The need to implement a multithreaded detection system, such that the detection and analysis operations run independently of each other. [20]
- The ability to accurately interpret ‘signatures’ which characterise traffic associated with a known attack. This has been implemented using the ‘WinPosix’ regular expression library, which is based on the Unix Posix standard.[21] This is released under the terms of the GNU General Public License, and can be obtained from http://www.tburke.net/info/reskittools/topics/winposix.

‘Winposix’ is a ‘regular expression library’, which allows a high-efficiency analysis of data (in this case IP packet payloads) to be performed in ‘real time’, and greatly simplifies the analysis procedure. “A good tool with a poor process is not going to be any more effective than a poor tool.” Proctor, (2001).
Even where the IDS can run in ‘real-time’, other factors must be considered; the detection of some attack types relies heavily on the detection of specific sequences between packets; these must be detected in the correct sequence. Factors such as network latency, and the accuracy of time references must be taken into

consideration. This is particularly true in the case of distributed systems. [22] “Even if the network latency is negligible, the timestamps may be different if the workloads of the computers are different.” (Ning, 2001).

Whenever an event matches a configuration rule, the details must be saved. The saving of data to disk is potentially able to cause even greater time delay than the analysis procedure. Worse, the hard disk may in itself be busy, due to disk writes by other applications or the operating system itself. Fortunately, the operating system itself caches data in preparation for a disk write operation, both to address the problem of time delay, and because it is only possible to write a whole disk cluster (for example, 512 bytes, or a multiple thereof), in a single write operation.

However, here there is a significant problem. Although it is desirable for an IDS to alert or record in ‘real time’, there are occasions on which the IDS itself must be patient – a ‘slow scan’ may be impossible to confirm for some hours until all the components have been detected.

“Analysis schemes appear to the status engine as partially ordered sequences of states or state transitions. Therefore, 'to recognise suspicious activity, the analysis engine must consider the event stream as a function of time. This requirement is not usually an issue when monitoring for events driven by an attack script or intrusion tool because the progression of events is rapid.” Bace, (2000)

Figure 6 depicts the sequence of required events during analysis.

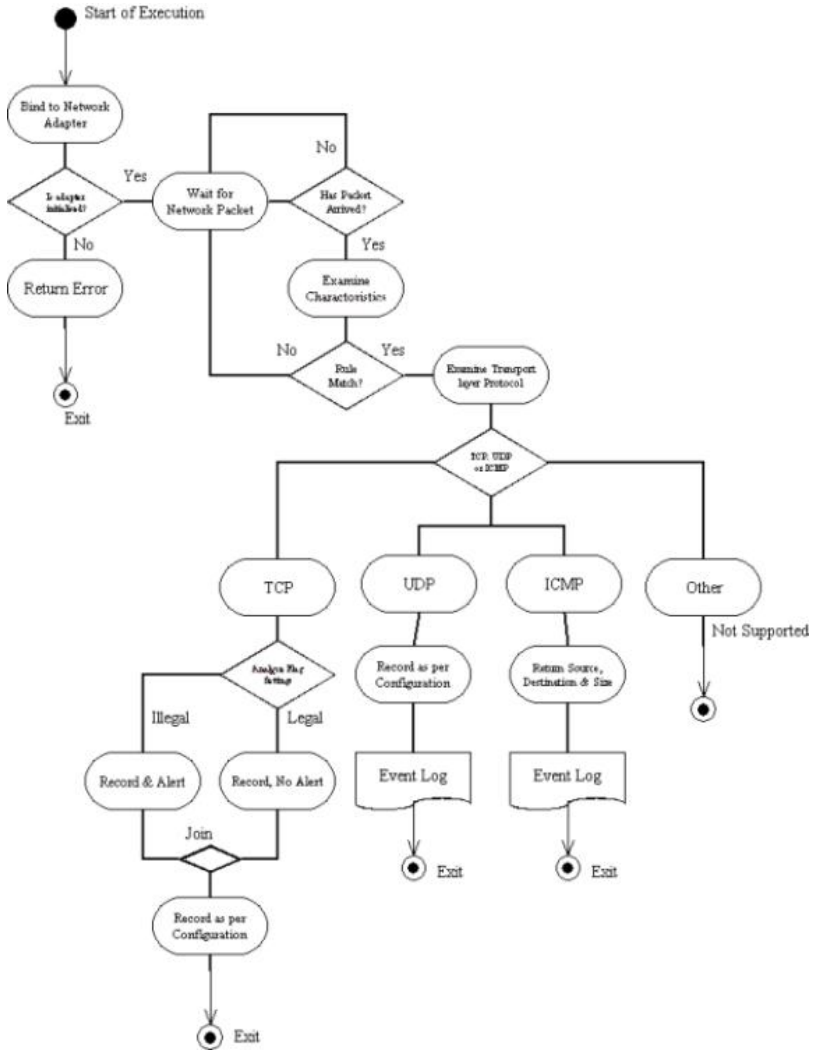


Fig. 6: IPGateKeeper UML Activity Diagram

Fig. 7 illustrates how the various sub-components of the program will interact. The class object names are in pseudo-code form.

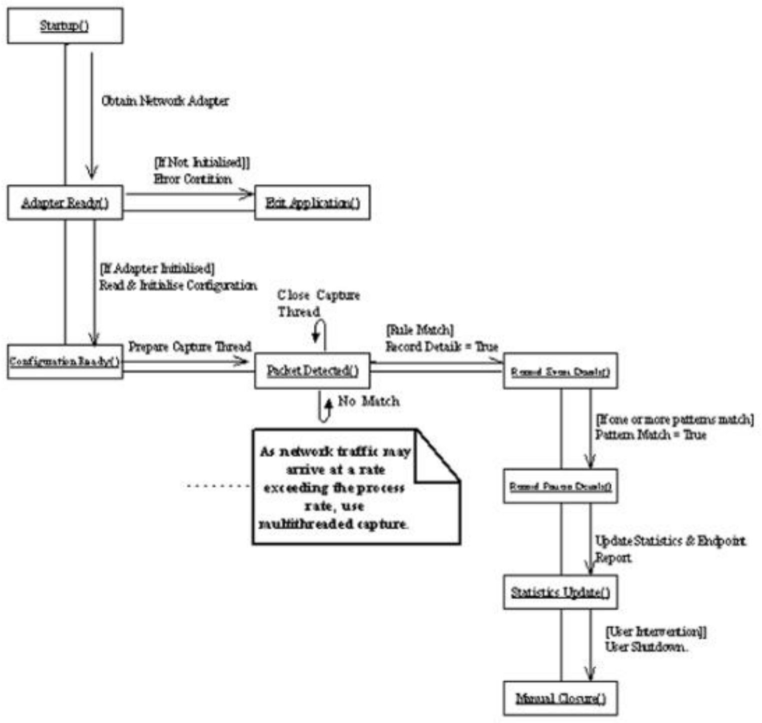


Fig. 7: IPGateKeeper UML Operating Sequence Diagram

VI. OSI Model Design Considerations

In order to explain the design of an IDS, it is necessary to explain how an IDS fits in with the OSI (Open System Interconnection) model, Fig. 8:

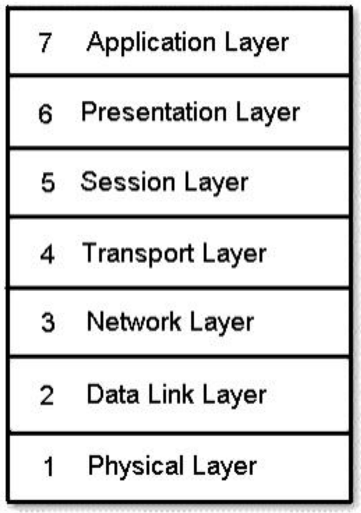
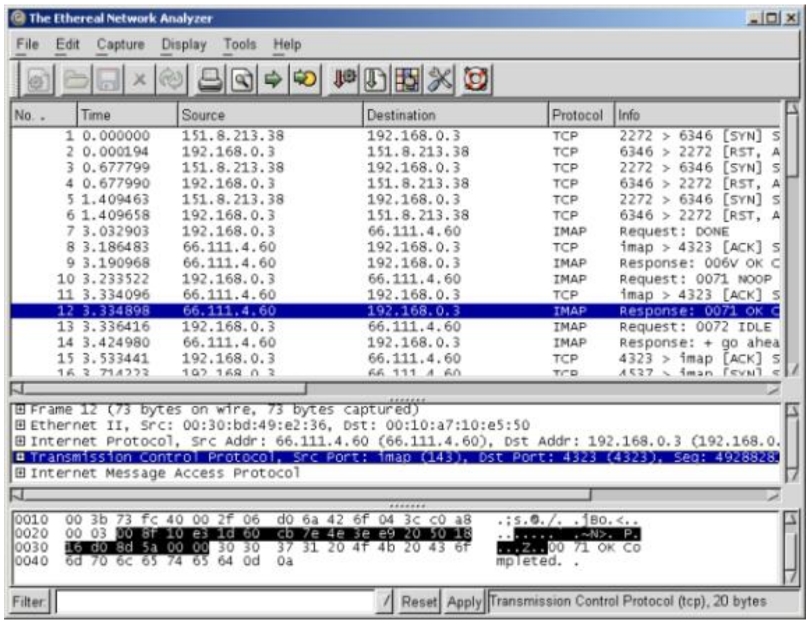


Fig 8: The seven layer OSI Model

The Internet Protocol itself resides at the Network layer, whereas the Transport Layer protocols include those protocols which are of direct interest; Internet Control Message Protocol, User Datagram Protocol, and Internet Control Message Protocol.
The fact that ICMP, UDP and TCP packets are 'wrapped' inside an IP packet implies that it is necessary to separate the IP packet data from the Transport layer data of interest. This involves accurately calculating the location of the last IP byte (the first TCP, UDP or ICMP byte follows the IP header.

Fig. 9 [23] below demonstrates how IP fits into the OSI Model. For the purposes of this project, I have decided only to support Internet Protocol Version 4 (IPV4) as I do not have access to IPV6.

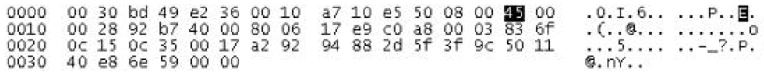

The Ethereal Network Analyzer
File Edit Capture Display Tools Help
Filter: Reset Apply

```
No. .  Time      Source          Destination     Protocol Info
     1 0.000000  151.8.213.38    192.168.0.3     TCP      2272 > 6346 [SYN] S
     2 0.000194  192.168.0.3     151.8.213.38    TCP      6346 > 2272 [RST, A
     3 0.677799  151.8.213.38    192.168.0.3     TCP      2272 > 6346 [SYN] S
     4 0.677990  192.168.0.3     151.8.213.38    TCP      6346 > 2272 [RST, A
     5 1.409463  151.8.213.38    192.168.0.3     TCP      2272 > 6346 [SYN] S
     6 1.409658  192.168.0.3     151.8.213.38    TCP      6346 > 2272 [RST, A
     7 3.032903  192.168.0.3     66.111.4.60     IMAP     Request: DONE
     8 3.186483  66.111.4.60     192.168.0.3     TCP      imap > 4323 [ACK] S
     9 3.190968  66.111.4.60     192.168.0.3     IMAP     Response: 006V OK C
    10 3.233522  192.168.0.3     66.111.4.60     IMAP     Request: 0071 NOOP
    11 3.334096  66.111.4.60     192.168.0.3     TCP      imap > 4323 [ACK] S
    12 3.334898  66.111.4.60     192.168.0.3     IMAP     Response: 0071 OK C
    13 3.336416  192.168.0.3     66.111.4.60     IMAP     Request: 0072 IDLE
    14 3.424980  66.111.4.60     192.168.0.3     IMAP     Response: + go ahea
    15 3.533441  192.168.0.3     66.111.4.60     TCP      4323 > imap [ACK] S
    16 3.714223  192.168.0.3     66.111.4.60     TCP      4537 > imap [SYN] S

⊞ Frame 12 (73 bytes on wire, 73 bytes captured)
⊞ Ethernet II, Src: 00:30:bd:49:e2:36, Dst: 00:10:a7:10:e5:50
⊞ Internet Protocol, Src Addr: 66.111.4.60 (66.111.4.60), Dst Addr: 192.168.0.3 (192.168.0.
⊞ Transmission Control Protocol, Src Port: imap (143), Dst Port: 4323 (4323), Seq: 4928828
⊞ Internet Message Access Protocol

0010  00 3b 73 fc 40 00 2f 06  d0 6a 42 6f 04 3c c0 a8   .;s.@./. .jBo.<..
0020  00 03 00 8f 10 e3 1d 60  cb 7e 4e 3e e9 20 50 18   ........ .~N>. P.
0030  16 d0 8d 5a 00 00 30 30  37 31 20 4f 4b 20 43 6f   ...Z..00 71 OK Co
0040  6d 70 6c 65 74 65 64 0d  0a                        mpleted. .

Transmission Control Protocol (tcp), 20 bytes
```

Fig. 9: An IP packet contained within an Ethernet Frame.

The TCP packet data is highlighted in both ASCII and Hexadecimal Byte Formats (which in 'C' would correspond to "%c" and "%02x" in a 'printf' statement). This illustrates how each TCP packet is 'wrapped' inside an Internet Protocol Packet (above the highlighted area), which is in turn 'wrapped' inside an Ethernet

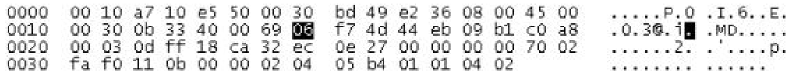
Frame. The TCP packet itself ‘wraps’ a session layer packet (below the highlighted area), in this example, a packet, which is part of an IMAP transaction. This illustrates how the OSI model operates.
The numbers on the left indicate the byte number in hexadecimal. As a byte is, in fact, a binary number between 0 and 255, it is convenient to display values in hexadecimal, since each hexadecimal digit represents four bits in binary.
Since the Internet Protocol (and the majority of transport and session layer protocols) do not encrypt data before sending it across the network, it is possible to actually view the contents of each packet directly. This is one reason for many security problems inherent in the OSI model. Secure HTTP (HTTPS) and Secure Shell (SSH) are examples of exceptions.
Since the data for each layer is identical (all are in the form of ASCII characters or two digit hexadecimal values, depending on how they are displayed), it is impossible to tell by simple observation where the data for each layer begins! It is, therefore, necessary to calculate the location of the start of each layer. The first byte of the transport layer data can be located by subtracting the IP header length from the remaining packet length. The header length is provided in the IP header, in the last four bits (the second hexadecimal digit) of the first IP byte. This value provides the IP header length in 32 bit data words. In order to obtain meaningful transport layer data, it is vitally important to correctly calculate the position of the first byte.



```
0000   00 30 bd 49 e2 36 00 10   a7 10 e5 50 08 00 45 00   .0.I.6.. ...P..E.
0010   00 28 92 b7 40 00 80 06   17 e9 c0 a8 00 03 83 6f   .(..@... .......o
0020   0c 15 0c 35 00 17 a2 92   94 88 2d 5f 3f 9c 50 11   ...5.... ..-_?.P.
0030   40 e8 6e 59 00 00                                   @.nY..
```

Fig. 10: The first IP byte

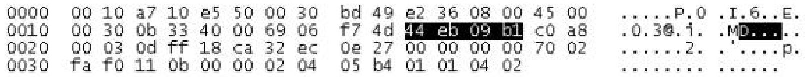
In this example packet, the byte highlighted provides the Internet protocol version (4) and the header length (20 bytes, five four bit data words). This is the first byte of the IP packet header; the bytes to the left comprise the Ethernet header.
It is possible to determine that this particular IP packet encloses a TCP packet by means of the protocol number byte:



```
0000   00 10 a7 10 e5 50 00 30   bd 49 e2 36 08 00 45 00   .....P.0 .I.6..E.
0010   00 30 0b 33 40 00 69 06   f7 4d 44 eb 09 b1 c0 a8   .0.3@.i. .MD.....
0020   00 03 0d ff 18 ca 32 ec   0e 27 00 00 00 00 70 02   ......2. .'....p.
0030   fa f0 11 0b 00 00 02 04   05 b4 01 01 04 02         ........ ......
```

Fig. 11: The transport layer protocol number

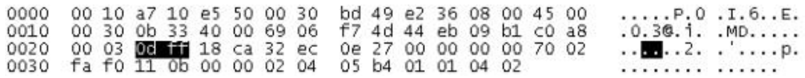
This is because TCP, when carried over IP, is protocol number 06h. The protocol numbers for ICMP and UDP are 01h and 11h (1 and 17 in decimal) respectively.
The source IP address requires four bytes of the IP header (a byte for each octet). In this example, we can see that the source address is 44.EB.09.B1 in hexadecimal, or
68.235.9.177 in decimal. The following four bytes contain the destination address (192.168.0.3).



```
0000   00 10 a7 10 e5 50 00 30   bd 49 e2 36 08 00 45 00   .....P.0 .I.6..E.
0010   00 30 0b 33 40 00 69 06   f7 4d 44 eb 09 b1 c0 a8   .0.3@.i. .MD.....
0020   00 03 0d ff 18 ca 32 ec   0e 27 00 00 00 00 70 02   ......2. .'....p.
0030   fa f0 11 0b 00 00 02 04   05 b4 01 01 04 02         ........ ......
```

Fig. 12: Destination address – 32-bit word

Fig. 13 shows the source port, [24] which is port 3583 in decimal. The next two bytes (18 CA) contain the destination port (6346).

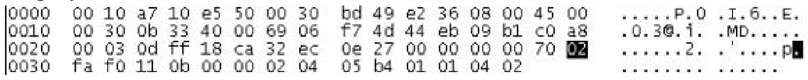


```
0000   00 10 a7 10 e5 50 00 30   bd 49 e2 36 08 00 45 00   .....P.0 .I.6..E.
0010   00 30 0b 33 40 00 69 06   f7 4d 44 eb 09 b1 c0 a8   .0.3@.i. .MD.....
0020   00 03 0d ff 18 ca 32 ec   0e 27 00 00 00 00 70 02   ......2. .'....p.
0030   fa f0 11 0b 00 00 02 04   05 b4 01 01 04 02         ........ ......
```

Fig. 13: The source port – 16-bit word

The remaining TCP data of interest to an intrusion detection system is the TCP flag setting, which allows it to determine the state of a connection. [25] This is only applicable to TCP, since ICMP and UDP are ‘connectionless’ protocols.

In this case, the packet is attempting to initiate a connection, since it has only the Syn flag set. This gives the flag byte a value of 02x.

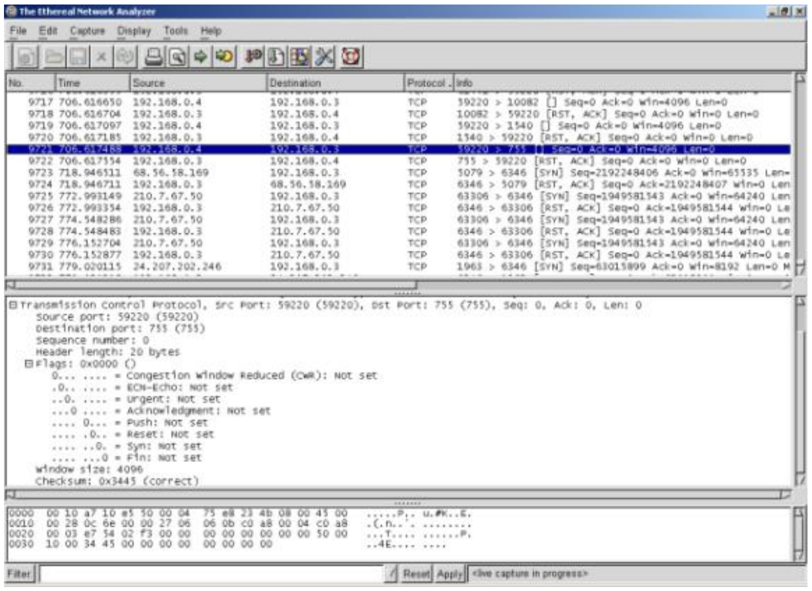


```
0000   00 10 a7 10 e5 50 00 30  bd 49 e2 36 08 00 45 00   .....P.0 .I.6..E.
0010   00 30 0b 33 40 00 69 06  f7 4d 44 eb 09 b1 c0 a8   .0.3@.i. .MD.....
0020   00 03 0d ff 18 ca 32 ec  0e 27 00 00 00 00 70 02   ......2. .'....p.
0030   fa f0 11 0b 00 00 02 04  05 b4 01 01 04 02         ........ ......
```

Fig. 14: The TCP Flag Byte

i. TCP Specific Considerations

Part of the role of an IDS is to detect anomalies in the packet data itself, such as illegal TCP flag settings, which are not found in normal traffic, as defined in RFC793. This is an example of an ‘out of specification’ TCP packet, which has been captured using Ethereal:


The Ethereal Network Analyzer
File Edit Capture Display Tools Help
Filter
Reset
Apply
<live capture in progress>

```
No.  Time        Source          Destination     Protocol Info
9717 706.616650  192.168.0.4     192.168.0.3     TCP      59220 > 10082 [] Seq=0 Ack=0 Win=4096 Len=0
9718 706.616704  192.168.0.3     192.168.0.4     TCP      10082 > 59220 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0
9719 706.617097  192.168.0.4     192.168.0.3     TCP      59220 > 1540 [] Seq=0 Ack=0 Win=4096 Len=0
9720 706.617185  192.168.0.3     192.168.0.4     TCP      1540 > 59220 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0
9721 706.617488  192.168.0.4     192.168.0.3     TCP      59220 > 755 [] Seq=0 Ack=0 Win=4096 Len=0
9722 706.617554  192.168.0.3     192.168.0.4     TCP      755 > 59220 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0
9723 718.946511  68.56.58.169    192.168.0.3     TCP      5079 > 6346 [SYN] Seq=2192248406 Ack=0 Win=65535 Len=
9724 718.946711  192.168.0.3     68.56.58.169    TCP      6346 > 5079 [RST, ACK] Seq=0 Ack=2192248407 Win=0 Len
9725 772.993149  210.7.67.50     192.168.0.3     TCP      63306 > 6346 [SYN] Seq=1949581543 Ack=0 Win=64240 Len
9726 772.993354  192.168.0.3     210.7.67.50     TCP      6346 > 63306 [RST, ACK] Seq=0 Ack=1949581544 Win=0 Le
9727 774.548286  210.7.67.50     192.168.0.3     TCP      63306 > 6346 [SYN] Seq=1949581543 Ack=0 Win=64240 Len
9728 774.548483  192.168.0.3     210.7.67.50     TCP      6346 > 63306 [RST, ACK] Seq=0 Ack=1949581544 Win=0 Le
9729 776.152704  210.7.67.50     192.168.0.3     TCP      63306 > 6346 [SYN] Seq=1949581543 Ack=0 Win=64240 Len
9730 776.152877  192.168.0.3     210.7.67.50     TCP      6346 > 63306 [RST, ACK] Seq=0 Ack=1949581544 Win=0 Le
9731 779.020115  24.207.202.246  192.168.0.3     TCP      1963 > 6346 [SYN] Seq=63015899 Ack=0 Win=8192 Len=0 M
```

```
⊟ Transmission Control Protocol, Src Port: 59220 (59220), Dst Port: 755 (755), Seq: 0, Ack: 0, Len: 0
    Source port: 59220 (59220)
    Destination port: 755 (755)
    Sequence number: 0
    Header length: 20 bytes
  ⊟ Flags: 0x0000 ()
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...0 .... = Acknowledgment: Not set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...0 = Fin: Not set
    Window size: 4096
    Checksum: 0x3445 (correct)
```

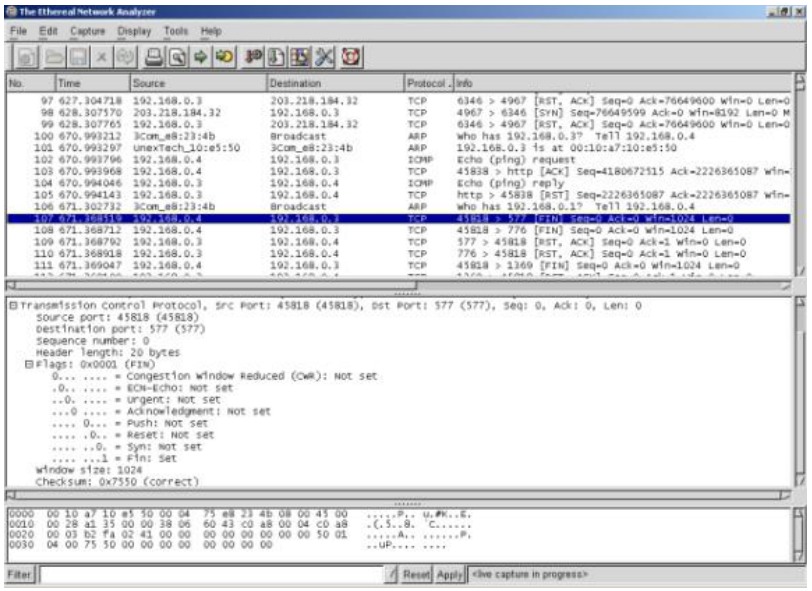
```
0000   00 10 a7 10 e5 50 00 04  75 e8 23 4b 08 00 45 00   .....P.. u.#K..E.
0010   00 28 0c 6e 00 00 27 06  06 0b c0 a8 00 04 c0 a8   .(.n..'. ........
0020   00 03 e7 54 02 f3 00 00  00 00 00 00 00 00 50 00   ...T.... ......P.
0030   10 00 34 45 00 00 00 00  00 00 00 00               ..4E.... ....
```

Fig. 15: An ‘out of specification’ TCP packet

This particular TCP packet has no flags set at all. This is in violation of RFC793, as all TCP packets have at least one flag set, whichever state a connection is currently in. Therefore, this particular packet makes no sense whatsoever to any operating system.

Since different operating systems will respond differently on receipt of such a packet, this allows the intruder to determine which operating system a given machine is running.

Another example of an ‘illegal’ TCP packet is one which has only the ‘FIN’ flag set. This particular setting is illegal, since in a normal TCP connection, the ‘ACK’ (acknowledgement) flag is set in all packets except for the first ‘SYN’ packet sent. Therefore, the setting of the ‘FIN’ flag without the ‘ACK’ flag is in violation of RFC793.


The Ethereal Network Analyzer
File Edit Capture Display Tools Help
Filter
Reset
Apply
<live capture in progress>

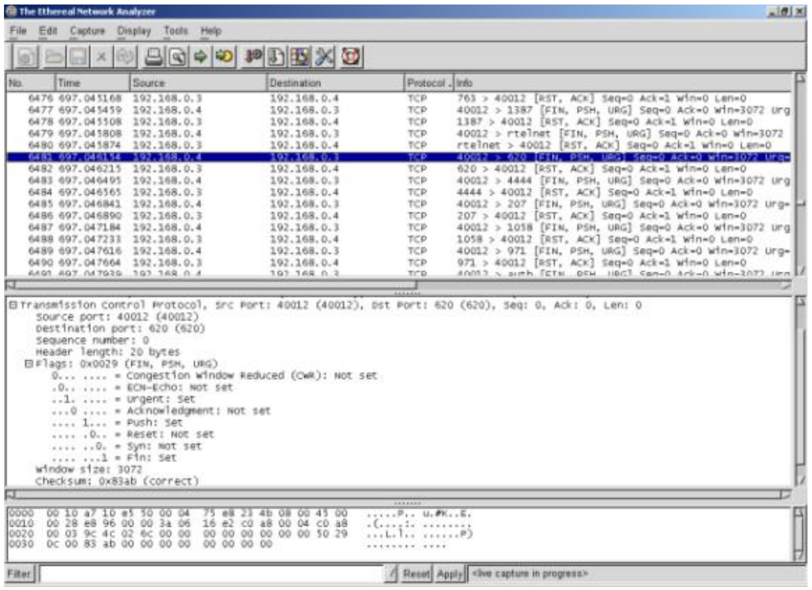
```
No.  Time          Source              Destination       Protocol  Info
  97 627.304718   192.168.0.3         203.218.184.32    TCP       6346 > 4967 [RST, ACK] Seq=0 Ack=76649600 Win=0 Len=0
  98 628.307570   203.218.184.32      192.168.0.3       TCP       4967 > 6346 [SYN] Seq=76649599 Ack=0 Win=8192 Len=0 M
  99 628.307765   192.168.0.3         203.218.184.32    TCP       6346 > 4967 [RST, ACK] Seq=0 Ack=76649600 Win=0 Len=0
 100 670.993212   3Com_e8:23:4b       Broadcast         ARP       Who has 192.168.0.3?  Tell 192.168.0.4
 101 670.993297   UnexTech_10:e5:50   3Com_e8:23:4b     ARP       192.168.0.3 is at 00:10:a7:10:e5:50
 102 670.993796   192.168.0.4         192.168.0.3       ICMP      Echo (ping) request
 103 670.993968   192.168.0.4         192.168.0.3       TCP       45838 > http [ACK] Seq=4180672515 Ack=2226365087 Win=
 104 670.994046   192.168.0.3         192.168.0.4       ICMP      Echo (ping) reply
 105 670.994143   192.168.0.3         192.168.0.4       TCP       http > 45838 [RST] Seq=2226365087 Ack=2226365087 Win=
 106 671.302732   3Com_e8:23:4b       Broadcast         ARP       Who has 192.168.0.1?  Tell 192.168.0.4
 107 671.368519   192.168.0.4         192.168.0.3       TCP       45818 > 577 [FIN] Seq=0 Ack=0 Win=1024 Len=0
 108 671.368712   192.168.0.4         192.168.0.3       TCP       45818 > 776 [FIN] Seq=0 Ack=0 Win=1024 Len=0
 109 671.368792   192.168.0.3         192.168.0.4       TCP       577 > 45818 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
 110 671.368918   192.168.0.3         192.168.0.4       TCP       776 > 45818 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
 111 671.369047   192.168.0.4         192.168.0.3       TCP       45818 > 1369 [FIN] Seq=0 Ack=0 Win=1024 Len=0
```

```
⊟ Transmission Control Protocol, Src Port: 45818 (45818), Dst Port: 577 (577), Seq: 0, Ack: 0, Len: 0
    Source port: 45818 (45818)
    Destination port: 577 (577)
    Sequence number: 0
    Header length: 20 bytes
  ⊟ Flags: 0x0001 (FIN)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...0 .... = Acknowledgment: Not set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...1 = Fin: Set
    Window size: 1024
    Checksum: 0x7550 (correct)
```

```
0000  00 10 a7 10 e5 50 00 04  75 e8 23 4b 08 00 45 00   .....P.. u.#K..E.
0010  00 28 a1 35 00 00 38 06  60 43 c0 a8 00 04 c0 a8   .(.5..8. `C......
0020  00 03 b2 fa 02 41 00 00  00 00 00 00 00 00 50 01   .....A.. ......P.
0030  04 00 75 50 00 00 00 00  00 00 00 00               ..uP.... ....
```

Fig. 16: Packet generated by a 'FIN scan'

A further example of an abnormal packet has the 'Urgent', 'Push' and 'Fin' flags set. Although not strictly in violation of RFC793, this combination of flag settings is unusual. This particular choice of flag settings is probably an attempt to bypass IDS and firewall rules which (by now) are able to detect the more 'obvious' anomalous combinations such as Syn/Fin.


The Ethereal Network Analyzer
File Edit Capture Display Tools Help
Filter
Reset
Apply
<live capture in progress>

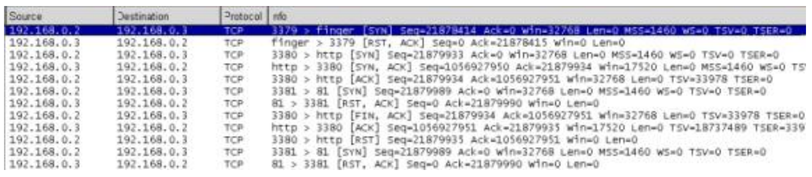
```
No.   Time          Source         Destination    Protocol  Info
6476 697.045168   192.168.0.3    192.168.0.4    TCP       763 > 40012 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
6477 697.045459   192.168.0.4    192.168.0.3    TCP       40012 > 1387 [FIN, PSH, URG] Seq=0 Ack=0 Win=3072 Urg
6478 697.045508   192.168.0.3    192.168.0.4    TCP       1387 > 40012 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
6479 697.045808   192.168.0.4    192.168.0.3    TCP       40012 > rtelnet [FIN, PSH, URG] Seq=0 Ack=0 Win=3072
6480 697.045874   192.168.0.3    192.168.0.4    TCP       rtelnet > 40012 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
6481 697.046134   192.168.0.4    192.168.0.3    TCP       40012 > 620 [FIN, PSH, URG] Seq=0 Ack=0 Win=3072 Urg=
6482 697.046215   192.168.0.3    192.168.0.4    TCP       620 > 40012 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
6483 697.046495   192.168.0.4    192.168.0.3    TCP       40012 > 4444 [FIN, PSH, URG] Seq=0 Ack=0 Win=3072 Urg
6484 697.046565   192.168.0.3    192.168.0.4    TCP       4444 > 40012 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
6485 697.046841   192.168.0.4    192.168.0.3    TCP       40012 > 207 [FIN, PSH, URG] Seq=0 Ack=0 Win=3072 Urg=
6486 697.046890   192.168.0.3    192.168.0.4    TCP       207 > 40012 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
6487 697.047184   192.168.0.4    192.168.0.3    TCP       40012 > 1058 [FIN, PSH, URG] Seq=0 Ack=0 Win=3072 Urg
6488 697.047233   192.168.0.3    192.168.0.4    TCP       1058 > 40012 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
6489 697.047616   192.168.0.4    192.168.0.3    TCP       40012 > 971 [FIN, PSH, URG] Seq=0 Ack=0 Win=3072 Urg=
6490 697.047664   192.168.0.3    192.168.0.4    TCP       971 > 40012 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
```

```
⊟ Transmission Control Protocol, Src Port: 40012 (40012), Dst Port: 620 (620), Seq: 0, Ack: 0, Len: 0
    Source port: 40012 (40012)
    Destination port: 620 (620)
    Sequence number: 0
    Header length: 20 bytes
  ⊟ Flags: 0x0029 (FIN, PSH, URG)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..1. .... = Urgent: Set
      ...0 .... = Acknowledgment: Not set
      .... 1... = Push: Set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...1 = Fin: Set
    Window size: 3072
    Checksum: 0x83ab (correct)
```

```
0000  00 10 a7 10 e5 50 00 04  75 e8 23 4b 08 00 45 00   .....P.. u.#K..E.
0010  00 28 e8 96 00 00 3a 06  16 e2 c0 a8 00 04 c0 a8   .(....:. ........
0020  00 03 9c 4c 02 6c 00 00  00 00 00 00 00 00 50 29   ...L.l.. ......P)
0030  0c 00 83 ab 00 00 00 00  00 00 00 00               ........ ....
```

Fig. 17: NMAP 'XMAS' scan packet

As the 'SYN' flag is used to request the initiation of a connection, and the 'FIN' flag indicates that a connection is to be terminated, these flags should never be set simultaneously. The aim of such packets is either to 'fingerprint' the operating system of the target system, as previously mentioned, or to detect the status of an IP port, without initiating a connection. Since a standard 'TCP connect' scan detects open ports by attempting to initiate a connection, it is very easy to detect. Fig. 18 demonstrates the effect of a TCP connect scan to open and closed ports:



```
Source         Destination    Protocol  Info
192.168.0.2    192.168.0.3    TCP       3379 > finger [SYN] Seq=21878414 Ack=0 Win=32768 Len=0 MSS=1460 WS=0 TSV=0 TSER=0
192.168.0.3    192.168.0.2    TCP       finger > 3379 [RST, ACK] Seq=0 Ack=21878415 Win=0 Len=0
192.168.0.2    192.168.0.3    TCP       3380 > http [SYN] Seq=21879933 Ack=0 Win=32768 Len=0 MSS=1460 WS=0 TSV=0 TSER=0
192.168.0.3    192.168.0.2    TCP       http > 3380 [SYN, ACK] Seq=1056927950 Ack=21879934 Win=17520 Len=0 MSS=1460 WS=0 TS
192.168.0.2    192.168.0.3    TCP       3380 > http [ACK] Seq=21879934 Ack=1056927951 Win=32768 Len=0 TSV=33978 TSER=0
192.168.0.2    192.168.0.3    TCP       3381 > 81 [SYN] Seq=21879989 Ack=0 Win=32768 Len=0 MSS=1460 WS=0 TSV=0 TSER=0
192.168.0.3    192.168.0.2    TCP       81 > 3381 [RST, ACK] Seq=0 Ack=21879990 Win=0 Len=0
192.168.0.2    192.168.0.3    TCP       3380 > http [FIN, ACK] Seq=21879934 Ack=1056927951 Win=32768 Len=0 TSV=33978 TSER=0
192.168.0.3    192.168.0.2    TCP       http > 3380 [ACK] Seq=1056927951 Ack=21879935 Win=17520 Len=0 TSV=18737489 TSER=339
192.168.0.2    192.168.0.3    TCP       3380 > http [RST] Seq=21879935 Ack=1056927951 Win=0 Len=0
192.168.0.2    192.168.0.3    TCP       3381 > 81 [SYN] Seq=21879989 Ack=0 Win=32768 Len=0 MSS=1460 WS=0 TSV=0 TSER=0
192.168.0.3    192.168.0.2    TCP       81 > 3381 [RST, ACK] Seq=0 Ack=21879990 Win=0 Len=0
```

Fig. 18: Example of a TCP connect scan

A connection attempt to port 79 (Finger) demonstrates that the initial Syn packet is received. As the target is not running a Finger server, it responds by sending an Ack/Rst packet. However, an attempt to connect to port 80 (HTTP) generates a Syn/Ack response, indicating that a server is running. Once the connection is established, the connection is closed using a Fin/Ack packet. However, an actual connection was temporarily established.

A subsequent connection attempt to port 81 reveals that it is closed. [26]
This flowchart demonstrates the possible outcomes of each TCP connection attempt:

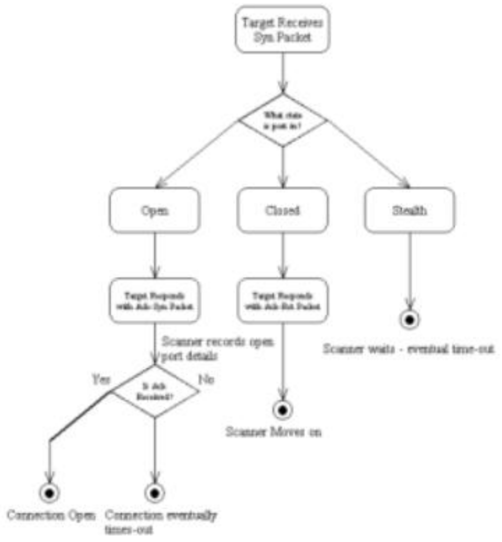

Target Receives Syn Packet
What state is port in?
Open
Closed
Stealth
Target Responds with Ack-Syn Packet
Target Responds with Ack-Rst Packet
Scanner records open port details
Scanner waits - eventual time-out
Yes
No
Is Ack Received?
Scanner Moves on
Connection Open
Connection eventually times-out

Fig. 19: Flowchart for a TCP Connect Scan. Many firewalls prevent any response to probing attempts '(stealth' mode).

However, 'Stealth' scans, which utilise 'illegal' TCP packets; do not ever actually attempt to establish a connection. This type of scan is much more difficult to detect. For example, a 'FIN scan' simply sends a packet with only the 'FIN' flag set. The remote system, on receiving this packet, typically ignores this packet if it is destined for a port, which is currently open (it makes no sense to the remote system). However in general a remote system will respond with an 'Ack/Rst' packet if the target port is closed. Since no connection was ever attempted, a connection attempt is not recorded.
In practice, it is possible to determine the *state* of a TCP connection only by viewing the flag settings. Note that the ACK flag may optionally be accompanied by either the PSH or URG flag; it is also possible for the ECN Echo or CWR flag to be set at intervals during the connection.

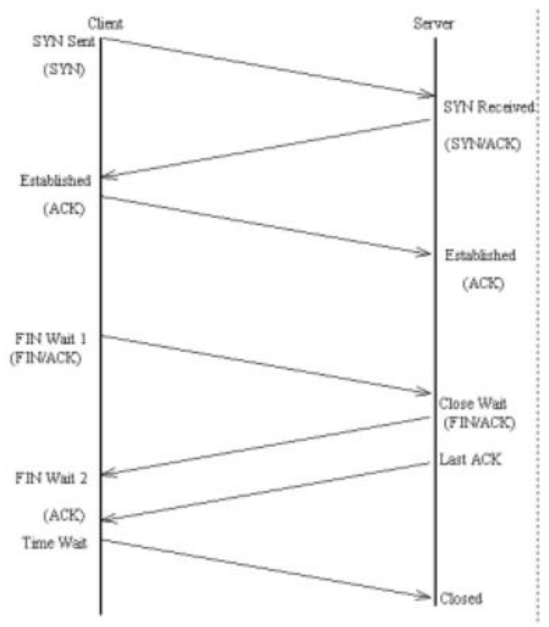


Fig. 20: TCP State Transition Diagram. Based on SecureNode Inc (2003).

Flag analysis is not as straightforward as expected. There are a number of applications, which do not adhere strictly to RFC793, both 'legitimate' and otherwise. This is because the RFC's describe how protocols *should* operate, as opposed to how they are implemented by the software developer. To quote an example, some 'Trojan horse' [27] programs use TCP for data transport, but do not use a standard 'TCP handshake' to initiate connections. Additionally, RFC's can in themselves go out of date. As a consequence, flag settings, which were once illegal, can become legal. Quoting an example, two of the eight flags (the two most significant [28] bits), which were designated as being reserved for future use in RFC793, have now been allocated to the 'Congestion Window Reduced' and 'ECN echo' functions, which have recently been added to the TCP specification. The significance of this is that, whereas the setting of either of these flags would have been illegal previously, their use is now legal.
It is obviously the relation of the flag setting to the TCP state transition diagram, which determines whether a given flag combination is legal.

UDP scanning is less reliable than TCP scanning, since UDP is 'unreliable'. [29] However, as UDP scanning is still widely performed, and other UDP attacks are possible, IPGateKeeper supports UDP. This is the flowchart demonstrating the possible outcomes of a UDP scan:

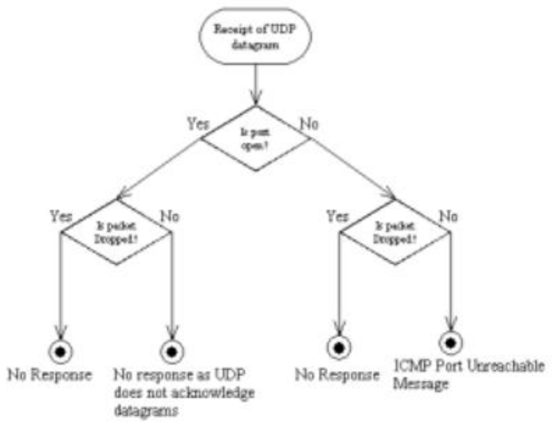


Fig. 21: UDP Scan Flowchart

ii. Performance Issues and Considerations

- IDS's have traditionally been classified as being either host based or network based. IPGateKeeper can run locally on a particular machine; however, it only realises its full potential whilst running on a gateway. This is because it can 'see' both internal traffic, and Internet traffic. IPGateKeeper is a passive IDS; this implies that, given that its operation does not appreciably restrict the throughput of the gateway, it is effectively 'transparent'.

"An advantage offered by network based ids is that it can be transparent to users. This makes it difficult to locate or to circumvent." Bace, (2000).

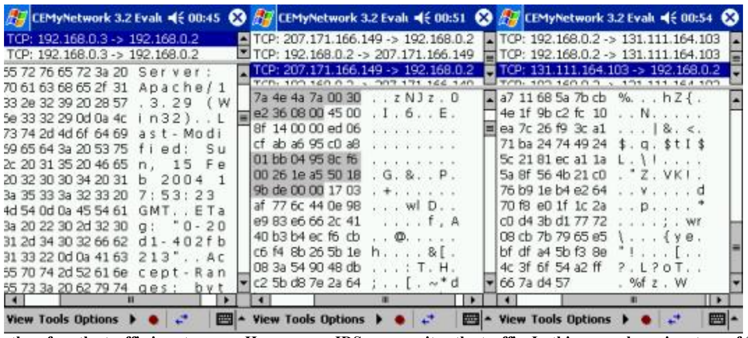
One of the first significant difficulties I experienced during the development process, concerns minimizing the loading on the system CPU. "IDSs must analyze large volumes of data while not placing a significant added load on the monitored systems and networks." Helmer, (2002). The difficulty is not so much the volume of data throughput in itself, but the 'per packet' handling of the traffic. At an early stage, it became obvious that, even on a relatively low bandwidth [30] Ethernet, it is possible for the IDS to 'see' as many as 100000 packets per second, from services which generate small packets. Interestingly, many vendors of hardware IDS products assume an average packet size of 1024 bytes for specification purposes; Edwards (2002), there are however some network protocols, which tend to generate large numbers of packets averaging below 100 bytes in size. The actual loading on an IDS can be considerably increased where a significant proportion of the traffic is of this nature, for example, on a subnet on which a domain name server for a large network resides. I very quickly deduced that, in order to process the traffic in 'real time', a multithreaded traffic handling technique would be required. [31]

- Another difficulty lies in the actual analysis of packets. Where analysis is generalised, the main overhead lies in the numbers of packets to be interpreted. For example, where we monitored all TCP traffic, this would involve tremendous performance overheads. In order to 'narrow down' the process, the configuration can be made specific, for example, where given attacks involve only one service run on a single port, it is possible to configure the program to ignore traffic destined for all other ports. Also, where a given attack has a well-known signature, the 'Text='

command can be used in conjunction with the 'Dest_Port' command, reducing the resource demands still further.

- In practice, the configuration is likely to become more complex over time, as new attacks are discovered.
- Yet another consideration involves storage capacity and performance. Transient data would need to be stored in arrays, as they offer the fast access times, which are essential in this application. As arrays offer only limited capacity, it became clear that usage must be reserved only for 'genuinely required' data. The use of dynamic memory allocation (for example, a 'linked list') was impractical due to the performance overheads inherent in the use of dynamic memory allocation.
- However, the use of arrays for storage does present difficulties, particularly pertaining to long-term storage. 'Slow scans' present a particular problem – scanning slowly is a 'stealth' technique for precisely this reason.
- It also became increasingly apparent that the resource overheads were considerably greater when processing packets, which matched configuration rules. Fortunately, in normal use, only a very small percentage of all packets would match configuration rules or adaptive 'conditions', therefore it is fortunate that the recording of large numbers of traffic types to multiple files (in order to facilitate testing), is a 'worst case' scenario.
- Network Heterogeneity presents all IDS's with an insurmountable problem. Due to the proliferation of protocols, many of which either do not adhere to RFC's, or are not defined in RFC's, it is often possible that the IDS is unable to interpret some packets correctly, thereby generating false alerts. There are particular problems with regard to encrypted traffic, such as HTTP over SSL (HTTPS). Although such traffic is intentionally secured against interception by eavesdroppers, it is also secured against interception by an IDS, as Fig. 22 illustrates.

"One thing to be aware of, however, is that a great many Internet-based attacks these days exist at the application layer. If these attacks are targeted against a server running SSL then all the legitimate traffic, as well as the attacks, will be encrypted – thus the IDS system will be unable to 'see into' the traffic to detect these." Midian, P, (N.d).


CEMyNetwork 3.2 Evalı 00:45
View Tools Options
CEMyNetwork 3.2 Evalı 00:51
View Tools Options
CEMyNetwork 3.2 Evalı 00:54
View Tools Options

```
TCP: 192.168.0.3 -> 192.168.0.2
TCP: 192.168.0.3 -> 192.168.0.2
55 72 76 65 72 3a 20  S e r v e r :
70 61 63 68 65 2f 31  A p a c h e / 1
33 2e 32 39 20 28 57  . 3 . 2 9   ( W
5e 33 32 29 0d 0a 4c  i n 3 2 ) . . L
73 74 2d 4d 6f 64 69  a s t - M o d i
59 65 64 3a 20 53 75  f i e d :   S u
2c 20 31 35 20 46 65  n ,   1 5   F e
20 32 30 30 34 20 31  b   2 0 0 4   1
3a 35 33 3a 32 33 20  7 : 5 3 : 2 3
4d 54 0d 0a 45 54 61  G M T . . E T a
3a 20 22 30 2d 32 30  g :   " 0 - 2 0
31 2d 34 30 32 66 62  d 1 - 4 0 2 f b
31 33 22 0d 0a 41 63  2 1 3 " . . A c
55 70 74 2d 52 61 6e  c e p t - R a n
55 73 3a 20 62 79 74  g e s :   b y t
```

```
TCP: 207.171.166.149 -> 192.168.0.2
TCP: 192.168.0.2 -> 207.171.166.149
TCP: 207.171.166.149 -> 192.168.0.2
7a 4e 4a 7a 00 30  . . z N J z . 0
e2 36 08 00 45 00  . I . 6 . . E .
8f 14 00 00 ed 06  . . . . . . . .
cf ab a6 95 c0 a8  . . . . . . . .
01 bb 04 95 8c f6  . . . . . . . .
00 26 1e a5 50 18  . G . & . . P .
9b de 00 00 17 03  . + . . . . . .
af 77 6c 44 0e 98  . . . wl  D . .
e9 83 e6 66 2c 41    . . . . f , A
40 b3 b4 ec f6 cb  . . @. . . . .
c6 f4 8b 26 5b 1e  h . . . . & [ .
08 3a 54 90 48 db  . . . : T . H .
c2 5b d8 7e 2a 64  ; . . . [ . ~ * d
```

```
TCP: 192.168.0.2 -> 131.111.164.103
TCP: 192.168.0.2 -> 131.111.164.103
TCP: 131.111.164.103 -> 192.168.0.2
a7 11 68 5a 7b cb  %. . . h Z { .
4e 1f 9b c2 fc 10  . . N . . . . .
ea 7c 26 f9 3c a1  . . . | & . < .
71 ba 24 74 49 24  $ . q . $ t I $
5c 21 81 ec a1 1a  L . \ ! . . . .
5a 8f 56 4b 21 c0  . " Z . V K ! .
76 b9 1e b4 e2 64  . . v . . . . d
70 f8 e0 1f 1c 2a  . . p . . . . *
c0 d4 3b d1 77 72  . . . . ; . wr
08 cb 7b 79 65 e5  \ . . . { y e .
bf df a4 5b f3 8e  " ! . . . [ . .
4c 3f 6f 54 a2 ff  ? . L ? o T . .
66 7a d4 57          . %f z . W
```

Fig. 22: HTTP (Left) is unencrypted; therefore the traffic is not secure. However, an IDS can monitor the traffic. In this example, a signature of 'Apache' could have been used in conjunction with Dest_Port=80 to detect connections to Apache servers. However, HTTPS (Centre) and SSH (Right) are strongly encrypted, preventing signature detection.

It follows that services using a combination of protocols are only as secure as the least secure protocol utilised.

The most important conclusions are firstly, that there is invariably a tradeoff between the additions of extra 'features'; for example, an improved mathematical procedure intended to improve accuracy, and the

requirement to keep resource usage to a reasonable minimum. It also follows that any software IDS is limited by the capability of the hardware on which it runs.
Another important limitation, for hardware and software IDS alike, is their inability to recognise attacks concealed within the payload of encrypted traffic. The use of SSL forces an IDS to rely exclusively on header information alone.

VII. Program Configuration System Design

i. 'Passive' Configuration

Without a means to configure the IDS, the user would have to depend absolutely upon 'hard coded' operations built into the program itself. An example of a hard-coded operation is the detection of traffic from a reserved IP address range. As the only possible sources of such traffic on the public Internet are illegitimate, this detection 'rule' does not need to be user configurable. As modification of this rule could only make the system less functional, it is best to hard code it – this is an example of the 'least privilege' principle. Only aspects of the rule base, which need to be configurable, should actually *be* user configurable. It is also inadvisable to make configurable, parameters, which would adversely affect some or all of the detection methods themselves. The configuration system, which is partially based on Barrus, (1998), is intended to accomplish the following:

- Allowing the program to concentrate only on traffic of interest. Once a packet has been identified as being 'irrelevant', it is ignored.
- As the majority of attack types, and many security vulnerabilities, tend to be specific to services which operate on 'well known' port numbers, which are generally adhered to, it is useful to specify only certain source or destination port numbers. This would filter out traffic destined to/from all other ports.
- In the same way, it may sometimes be desirable to specify given source or destination address ranges, or even individual hosts, which you wish to handle separately. For example, you may wish to monitor traffic from known 'rouge hosts', which have been known to cause difficulties in the past. [32]
- It is possible to monitor traffic either with or without a 'signature'. As many attack tools contain 'strings' of characters, which can be used to identify the tool, it is useful to record only traffic containing these strings.
- Certain services, such as Telnet and DNS, tend to produce small packets; unusually large packets directed to their assigned ports are, at best, regarded with suspicion, hence size analysis.
- Although 'signature' based analysis is of definite value, it is important to consider that a slight miss-match will cause an event to be missed. It is common practice for intrusion tools to be 'open source' so that anyone can obtain the source code and modify it such that the signatures will no longer match the rules already in use.[33]

"Their effectiveness is strictly related to the extent to which IDSs are updated with the signatures of the latest attacks developed." Giacinto, G, Roli, F, Didaci, L. (2003), [WWW] Fusion of multiple classifiers for intrusion detection in computer networks, Pattern Recognition Letters(24),

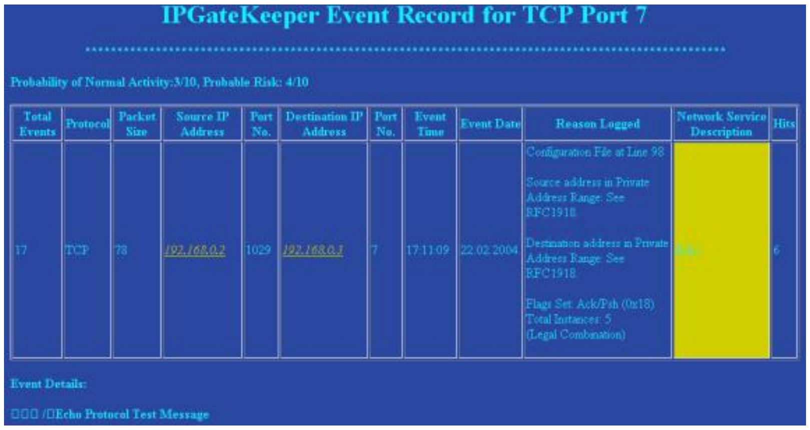
University of Cagliari, Cagliari, Italy. www.elsevier.com/locate/patrec

Another reason for minimizing the usage of the 'Text=' and 'Not_Found=" commands, concerns their tendency to produce false alarms. For example, if traffic related to a virus were to contain the string 'virus', and this string were used as a signature, it would produce very high rate of 'false positives' if the IDS were deployed in a medical research establishment.

"Unfortunately, general signatures designed by security experts usually generate high volumes of false alarms".

Proctor, P.E, (2001), Practical intrusion detection handbook, London, Prentice Hall.

The Chargen (Fig. 23) and Echo (Fig. 24) protocols [34] provide a useful method of verifying that the signature detection process operates as expected.



IPGateKeeper Event Record for TCP Port 7

****

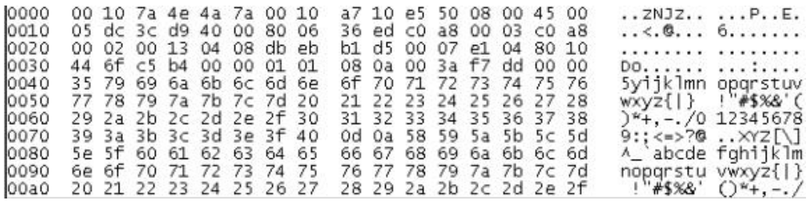
Probability of Normal Activity:3/10, Probable Risk: 4/10

Total Events	Protocol	Packet Size	Source IP Address	Port No.	Destination IP Address	Port No.	Event Time	Event Date	Reason Logged	Network Service Description	Hits
17	TCP	78	192.168.0.2	1029	192.168.0.3	7	17:11:09	22.02.2004	Configuration File at Line 98 Source address in Private Address Range: See RFC1918 Destination address in Private Address Range: See RFC1918 Flags Set: Ack/Psh (0x18) Total Instances: 5 (Legal Combination)	[illegible]	6

Event Details:

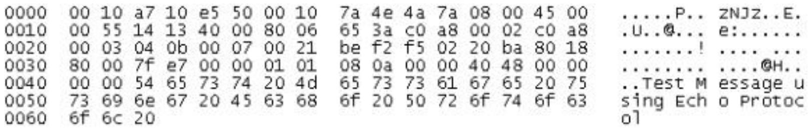
□□□ /□Echo Protocol Test Message

Fig. 23: IPGateKeeper record for an Echo protocol test



```
0000  00 10 7a 4e 4a 7a 00 10  a7 10 e5 50 08 00 45 00   ..zNJz.. ...P..E.
0010  05 dc 3c d9 40 00 80 06  36 ed c0 a8 00 03 c0 a8   ..<.@... 6.......
0020  00 02 00 13 04 08 db eb  b1 d5 00 07 e1 04 80 10   ........ ........
0030  44 6f c5 b4 00 00 01 01  08 0a 00 3a f7 dd 00 00   Do...... ...:....
0040  35 79 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   5yijklmn opqrstuv
0050  77 78 79 7a 7b 7c 7d 20  21 22 23 24 25 26 27 28   wxyz{|}  !"#$%&'(
0060  29 2a 2b 2c 2d 2e 2f 30  31 32 33 34 35 36 37 38   )*+,-./0 12345678
0070  39 3a 3b 3c 3d 3e 3f 40  0d 0a 58 59 5a 5b 5c 5d   9:;<=>?@ ..XYZ[\]
0080  5e 5f 60 61 62 63 64 65  66 67 68 69 6a 6b 6c 6d   ^_`abcde fghijklm
0090  6e 6f 70 71 72 73 74 75  76 77 78 79 7a 7b 7c 7d   nopqrstu vwxyz{|}
00a0  20 21 22 23 24 25 26 27  28 29 2a 2b 2c 2d 2e 2f    !"#$%&' ()*+,-./
```

Fig. 24: Chargen protocol packet

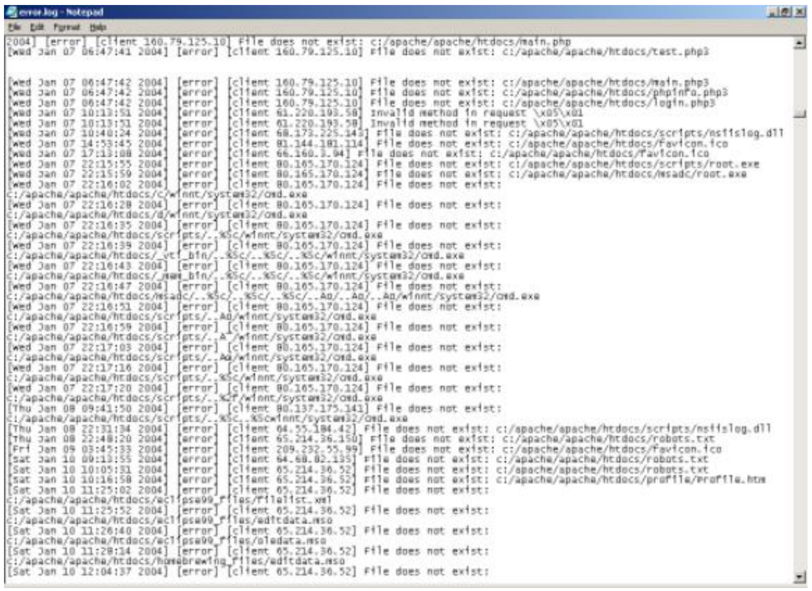


```
0000  00 10 a7 10 e5 50 00 10  7a 4e 4a 7a 08 00 45 00   .....P.. zNJz..E.
0010  00 55 14 13 40 00 80 06  65 3a c0 a8 00 02 c0 a8   .U..@... e:......
0020  00 03 04 0b 00 07 00 21  be f2 f5 02 20 ba 80 18   .......! .... ...
0030  80 00 7f e7 00 00 01 01  08 0a 00 00 40 48 00 00   ........ ....@H..
0040  00 00 54 65 73 74 20 4d  65 73 73 61 67 65 20 75   ..Test M essage u
0050  73 69 6e 67 20 45 63 68  6f 20 50 72 6f 74 6f 63   sing Ech o Protoc
0060  6f 6c 20                                           ol
```

Fig. 25: Echo Protocol packet

A particularly useful application of payload analysis pertains to the recognition of data at the application layer. For example, where incoming traffic is permitted to reach a public web server, a firewall must allow the connections to succeed. Therefore, it is unable to filter malicious traffic directed to the server. An IDS, however, can recognise requests for insecure CGI scripts, and many other exploits such as those, which involve the command prompt, simply by examining the packet payload. [35]

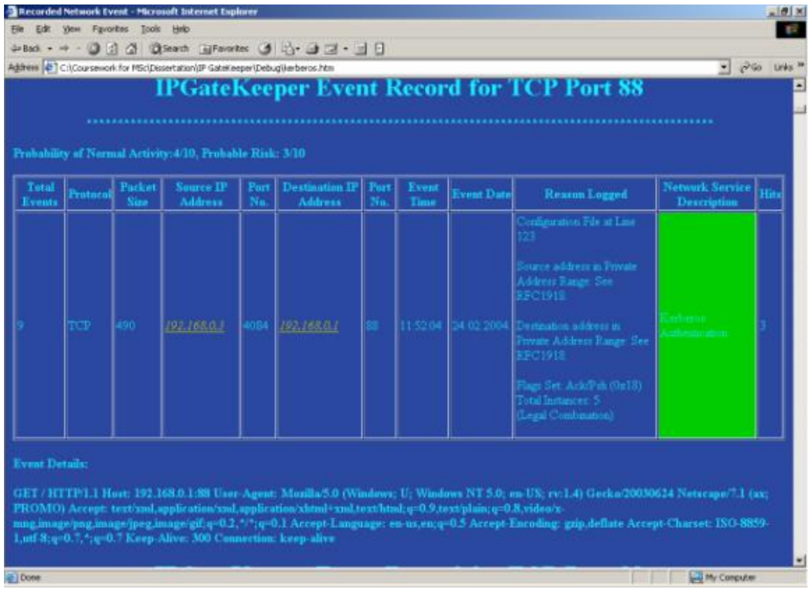
"An IDS, for instance, can determine whether the requested CGI program is one that is known to be insecure." Northcutt, (2001).



```
2004] [error] [client 160.79.125.10] File does not exist: c:/apache/apache/htdocs/main.php
[Wed Jan 07 06:47:41 2004] [error] [client 160.79.125.10] File does not exist: c:/apache/apache/htdocs/test.php3


[Wed Jan 07 06:47:42 2004] [error] [client 160.79.125.10] File does not exist: c:/apache/apache/htdocs/main.php3
[Wed Jan 07 06:47:42 2004] [error] [client 160.79.125.10] File does not exist: c:/apache/apache/htdocs/phpinfo.php3
[Wed Jan 07 06:47:42 2004] [error] [client 160.79.125.10] File does not exist: c:/apache/apache/htdocs/login.php3
[Wed Jan 07 10:13:51 2004] [error] [client 61.220.193.58] Invalid method in request \x05\x01
[Wed Jan 07 10:13:51 2004] [error] [client 61.220.193.58] Invalid method in request \x05\x01
[Wed Jan 07 10:40:24 2004] [error] [client 68.173.225.143] File does not exist: c:/apache/apache/htdocs/scripts/nsiislog.dll
[Wed Jan 07 14:53:45 2004] [error] [client 81.144.181.114] File does not exist: c:/apache/apache/htdocs/favicon.ico
[Wed Jan 07 17:13:08 2004] [error] [client 66.160.3.94] File does not exist: c:/apache/apache/htdocs/favicon.ico
[Wed Jan 07 22:15:55 2004] [error] [client 80.165.170.124] File does not exist: c:/apache/apache/htdocs/scripts/root.exe
[Wed Jan 07 22:15:59 2004] [error] [client 80.165.170.124] File does not exist: c:/apache/apache/htdocs/msadc/root.exe
[Wed Jan 07 22:16:02 2004] [error] [client 80.165.170.124] File does not exist:
c:/apache/apache/htdocs/c/winnt/system32/cmd.exe
[Wed Jan 07 22:16:28 2004] [error] [client 80.165.170.124] File does not exist:
c:/apache/apache/htdocs/d/winnt/system32/cmd.exe
[Wed Jan 07 22:16:35 2004] [error] [client 80.165.170.124] File does not exist:
c:/apache/apache/htdocs/scripts/..%5c/winnt/system32/cmd.exe
[Wed Jan 07 22:16:39 2004] [error] [client 80.165.170.124] File does not exist:
c:/apache/apache/htdocs/_vti_bin/..%5c/..%5c/..%5c/winnt/system32/cmd.exe
[Wed Jan 07 22:16:43 2004] [error] [client 80.165.170.124] File does not exist:
c:/apache/apache/htdocs/_mem_bin/..%5c/..%5c/..%5c/winnt/system32/cmd.exe
[Wed Jan 07 22:16:47 2004] [error] [client 80.165.170.124] File does not exist:
c:/apache/apache/htdocs/msadc/..%5c/..%5c/..%5c/..Ao/..Ao/..Ao/winnt/system32/cmd.exe
[Wed Jan 07 22:16:51 2004] [error] [client 80.165.170.124] File does not exist:
c:/apache/apache/htdocs/scripts/..Ao/winnt/system32/cmd.exe
[Wed Jan 07 22:16:59 2004] [error] [client 80.165.170.124] File does not exist:
c:/apache/apache/htdocs/scripts/..A /winnt/system32/cmd.exe
[Wed Jan 07 22:17:03 2004] [error] [client 80.165.170.124] File does not exist:
c:/apache/apache/htdocs/scripts/..Ao/winnt/system32/cmd.exe
[Wed Jan 07 22:17:16 2004] [error] [client 80.165.170.124] File does not exist:
c:/apache/apache/htdocs/scripts/..%5c/winnt/system32/cmd.exe
[Wed Jan 07 22:17:20 2004] [error] [client 80.165.170.124] File does not exist:
c:/apache/apache/htdocs/scripts/..%2f/winnt/system32/cmd.exe
[Thu Jan 08 09:41:50 2004] [error] [client 80.137.175.141] File does not exist:
c:/apache/apache/htdocs/scripts/..%5c..%5cwinnt/system32/cmd.exe
[Thu Jan 08 22:31:34 2004] [error] [client 64.55.184.42] File does not exist: c:/apache/apache/htdocs/scripts/nsiislog.dll
[Thu Jan 08 22:48:20 2004] [error] [client 65.214.36.150] File does not exist: c:/apache/apache/htdocs/robots.txt
[Fri Jan 09 03:45:33 2004] [error] [client 209.232.55.99] File does not exist: c:/apache/apache/htdocs/favicon.ico
[Sat Jan 10 09:13:55 2004] [error] [client 64.68.82.135] File does not exist: c:/apache/apache/htdocs/robots.txt
[Sat Jan 10 10:05:31 2004] [error] [client 65.214.36.52] File does not exist: c:/apache/apache/htdocs/robots.txt
[Sat Jan 10 10:16:58 2004] [error] [client 65.214.36.52] File does not exist: c:/apache/apache/htdocs/profile/profile.htm
[Sat Jan 10 11:25:02 2004] [error] [client 65.214.36.52] File does not exist:
c:/apache/apache/htdocs/eclipse99_files/filelist.xml
[Sat Jan 10 11:25:52 2004] [error] [client 65.214.36.52] File does not exist:
c:/apache/apache/htdocs/eclipse99_files/editdata.mso
[Sat Jan 10 11:26:40 2004] [error] [client 65.214.36.52] File does not exist:
c:/apache/apache/htdocs/eclipse99_files/oledata.mso
[Sat Jan 10 11:28:14 2004] [error] [client 65.214.36.52] File does not exist:
c:/apache/apache/htdocs/homebrewing_files/editdata.mso
[Sat Jan 10 12:04:37 2004] [error] [client 65.214.36.52] File does not exist:
```

Fig. 26: Examples of such scripting requests arrive at my own server with alarming regularity.

Yet another beneficial aspect of character analysis, however, pertains to 'service recognition'. Although the IANA [36] officially registers port assignments, it is possible to operate services on any port. [37] Possible incentives to do so include a perceived measure of security. It is therefore naive to assume that simply because a service listens on a given port, that it is the officially registered service! As an example, TCP port 88 is assigned to the Kerberos Authentication Protocol; however, my own gateway runs an HTTP service on this port for administration, as demonstrated by Fig. 27.



Recorded Network Event - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Coursework for MSc\Dissertation\IP GateKeeper\Debug\kerberos.htm
Go Links
Done
My Computer

IPGateKeeper Event Record for TCP Port 88

Probability of Normal Activity:4/10, Probable Risk: 3/10

Total Events	Protocol	Packet Size	Source IP Address	Port No.	Destination IP Address	Port No.	Event Time	Event Date	Reason Logged	Network Service Description	Hits
9	TCP	490	192.168.0.1	4084	192.168.0.1	88	11:52:04	24.02.2004	Configuration File at Line 123 Source address in Private Address Range: See RFC1918 Destination address in Private Address Range: See RFC1918 Flags Set: Ack/Psh (0x18) Total Instances: 5 (Legal Combination)	Kerberos Authentication	3

Event Details:

```
GET / HTTP/1.1 Host: 192.168.0.1:88 User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.4) Gecko/20030624 Netscape/7.1 (ax;
PROMO) Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,video/x-
mng,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1 Accept-Language: en-us,en;q=0.5 Accept-Encoding: gzip,deflate Accept-Charset: ISO-8859-
1,utf-8;q=0.7,*;q=0.7 Keep-Alive: 300 Connection: keep-alive
```

Fig. 27: An example of a 'non-standard' service

Since the majority of unencrypted services have identifiable 'banners', such as the "HTTP" string present in one of the first Ack packets generated, it is possible to identify a service with a great deal more certainty than can be accomplished on the basis of port assignment alone.

As packet data can be represented in hexadecimal or ASCII notation, the signature analysis procedure can utilise either. The primary reason I chose the ASCII notation (in the configuration) is that it interfaces better to human observers. This not only assists demonstration, but most importantly it resembles closely the entries found in the record files generated by applications such as web servers. Additionally, the use of ASCII as opposed to hexadecimal avoids the requirement to convert signature data between the two formats, byte for byte, when preparing the configuration file. However, the output can be in either or both formats – see the 'Format=' configuration command.

One major difficulty faced by all IDSs lies in detection of signatures spanned across multiple packets. For example, the signature 'etc/passwd' could be broken into 'et', '/pa', and 'sswd', in three packets. This obscures the signature. In order to decode the signature, it is necessary to queue the data, for later reassembly. However, the performance overhead is often unaffordable – there is little use in running the risk of tying up excessive system resources, and of missing subsequent packets (possibly containing the remainder of the signature). In practice, partially assembled signature data would have to 'time out' eventually, since there is a good chance that partial signatures may frequently occur in traffic unrelated with intrusive activity. Northcutt, (2002) states *"Ultimately, it is a tradeoff of functionality and speed, and speed is the current winner."*

This tradeoff has caused problems in the implementation of IPGateKeeper, for example, where I have become reliant on array storage for temporary data.

The 'Not_Found=" command can be used to identify traffic to/from port 80 which is unlikely to be HTTP traffic, to quote an example. Signature analysis can also assist in identification of which server, and version of that server, is in use. [38]

One possible motive to run a service on a non-standard port is the attempted 'concealment' of the service. However, this is inadvisable as it not only fails to genuinely conceal the service, but also has the potential to cause other problems, as an example of which, some clients do not offer the option to specify a particular connecting port. It is, therefore, best practice to adhere to standards. [39]

"Security through obscurity is no security at all". Curtin, Ranum, (2000).

One obvious question concerns my decision to make the IDS packet based as opposed to exclusively sequence-based. Although TCP is 'connection oriented', with the flag settings allowing determination of the state of the connection at any given point in time, UDP and ICMP are not connection based, and so each individual packet must be treated as a separate entity. Furthermore, sequence analysis may be performed after packet analysis. However if only Syn packets are processed, then no further analysis can be performed on those that were ignored. For example, if we only examine TCP packets which have only the 'Syn' flag set, then it is possible to record the fact that a connection attempt was made, however there is no way of knowing whether the connection attempt succeeded, or how long the connection lasted. [40] It is also not possible to identify any signatures, which may be present in packets other than the initial Syn packet – which should not carry any payload anyhow! [41]

Rather, it is best to implement the configuration around the attributes of IP packets, which are of interest to an IDS. The basic configuration, from an intrusion detection perspective, as modeled as follows:

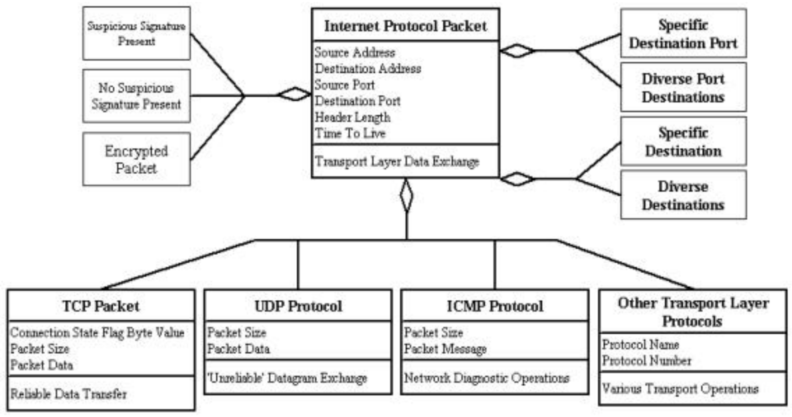


Fig. 28: IP Gatekeeper configuration design. Traffic attributes relevant to IDS design are selected from RFC's 768 (UDP), 791 (IP), 792 (ICMP) & 793 (TCP). Partially based on Barrus, (1998), "Intrusion Attack Class Hierarchy".

It is essential to consider that this configuration only pertains to packet-level analysis, as opposed to sequence-level analysis. This is a description of the configuration command set.

Command	Effect	Example Usage
$	Specifies that the line is a comment and is not a configuration rule. Equivalent to 'REM' in an MSDOS batch file.	$This is a comment.
!	Inverts a rule. Boolean Not.	Dest_Port!=80
Record=	Specifies the file to record to. Should have the .htm or .html extension. If omitted, output is displayed on the screen. See notes.	Record=udp.htm
Protocol=	Specifies TCP, UDP or ICMP. If omitted, all three are recorded.	Protocol=1 (For ICMP) Protocol=6 (For TCP) Protocol=17 (For UDP)
Host_Source=	Specifies a particular source IP Address or range. Useful when dealing with known 'rouge' hosts.	Host_Source=192.168.0.4 (Single)
Host_Dest=	Specifies a particular destination address or range.	Host_Dest=192.168 (Multiple)
Source_Port=		Source_Port=1024

Configuration Notes:

- Commands appearing on the same line in the configuration file are Boolean 'ANDed' together. As an example, the following rule would record the header details alone, if the IP address was 80.4.4.163, the destination address in the 194.83.45.xxx subnet, the destination port was 80, and the string contained the "GET /default.ida?" signature:

```
Format=0 Source_Address=80.4.4.163
Dest_Address= 194.83.45
Dest_Port=80 Text= GET /default.ida?
```

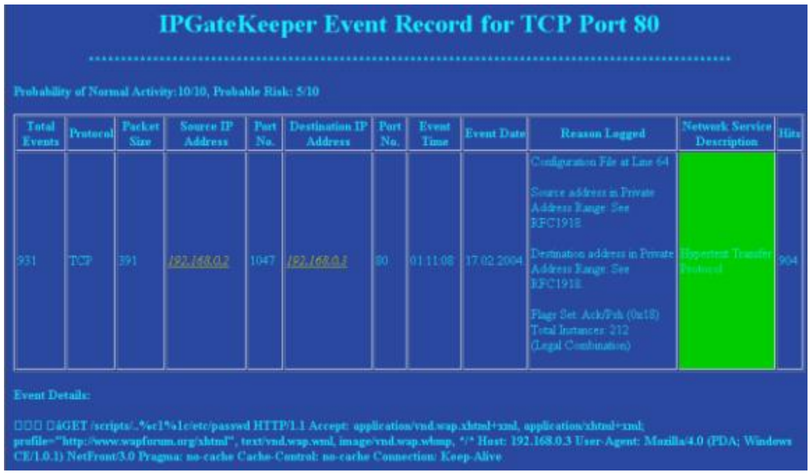
- Commands on separate lines are Boolean 'ORed' together – recording takes place if any one or more conditions are met.
- If the configuration file is missing, an error is generated on startup;
- If the configuration file is empty, no event will generate an alert!

Dest_Port=	Specifies the source port on the remote system. Needed only occasionally. Omit otherwise. Specifies the destination port on the target system(s).	Dest_Port=80	• One or more rule matches may be recorded into the same file. This is at the users discretion; • If no file is specified, output is sent directly to the display.
Lower_Size_Limit=	If omitted, all ports are monitored. This is usually undesirable.	Lower_Size_Limit=128	• Commands should be used in a compatible combination; for example the following combination is anomalous: Lower_Size_Limit=256 Upper_Size_Limit=128 – This may result from user error.
Upper_Size_Limit=	Packets below this size will not trigger an alert.	Upper_Size_Limit=1024	
Text=	Packets above this size will not trigger an alert. These commands are intended mainly to assist in reducing 'false positives'.	Text= etc\passwd	• The Lower_Size_Limit and Upper_Size_Limit commands apply to the individual packet size. This is true whether the packet is fragmented or not. It is therefore pointless in setting the values above the MTU for the network in question.
Not_Found=	Only packets containing the specified signature will generate an alert. Use with other commands. MUST be last command on line in configuration file.	Not_Found=default.ida?	• Both commands, and parameters, are case sensitive!
Format=	Generate an alert if a signature match is not detected, but all other conditions are met. Useful for detecting variants on signatures. Determines the 'line specific' recording method. See Below.	Format=0	• The Format= command determines the format in which the packet event details are logged. This command was extensively used during the testing and 'troubleshooting' of the initial packet interpretation. • The 'Protocol=' command must be used with care, particularly in combination with the 'Source_Port' and 'Dest_Port' commands; it is

best used only with reason. An example of such a reason is where the same session layer protocol (operating on the same port), utilises both TCP and UDP, and only one of the two is of interest. [42]

- Commands must not contain spaces. They must be a contiguous line, free of spaces.

The 'Text=' and 'Not_Found' commands are partially intended to provide a means by which to differentiate between malicious and non-malicious activity, which are impossible to identify by other means. This is an example of such a signature event:



IPGateKeeper Event Record for TCP Port 80

****

Probability of Normal Activity:10/10, Probable Risk: 5/10

Total Events	Protocol	Packet Size	Source IP Address	Port No.	Destination IP Address	Port No.	Event Time	Event Date	Reason Logged	Network Service Description	Hits
931	TCP	391	192.168.0.2	1047	192.168.0.3	80	01:11:08	17.02.2004	Configuration File at Line 64 Source address in Private Address Range: See RFC1918 Destination address in Private Address Range: See RFC1918 Flags Set: Ack/Psh (0x18) Total Instances: 212 (Legal Combination)	Hypertext Transfer Protocol	904

Event Details:

```
□□□ □âGET /scripts/..%c1%1c/etc/passwd HTTP/1.1 Accept: application/vnd.wap.xhtml+xml, application/xhtml+xml; profile="http://www.wapforum.org/xhtml", text/vnd.wap.wml, image/vnd.wap.wbmp, */* Host: 192.168.0.3 User-Agent: Mozilla/4.0 (PDA; Windows CE/1.0.1) NetFront/3.0 Pragma: no-cache Cache-Control: no-cache Connection: Keep-Alive
```

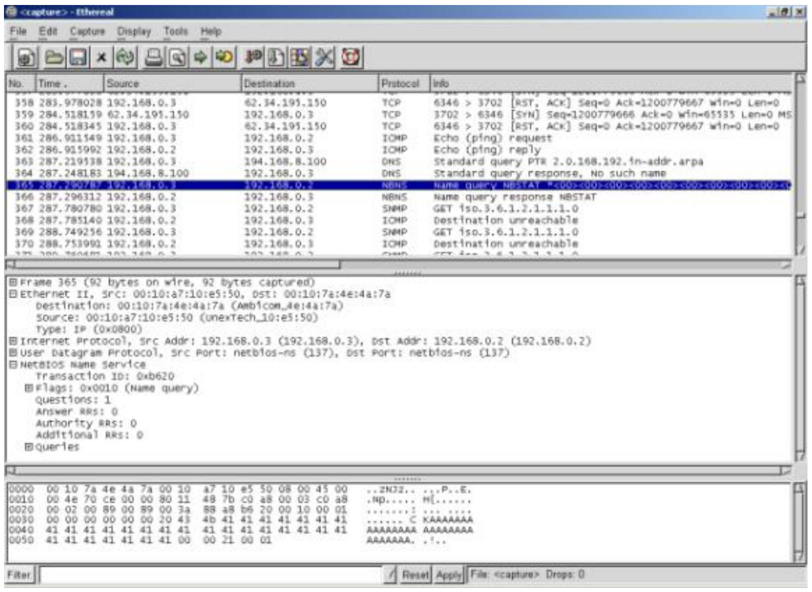
Fig. 29: Example signature detection record

This fictitious HTTP request clearly refers to the Unix password file (etc/passwd).

Many other IDS's utilise signatures also, for example 'Snort'. [43]

"Extraction of suitable features representing network connections is based on expert knowledge about the characteristics that distinguish attacks from normal connections. These features can be subdivided into two groups: features related to the data portion of packets (called payload) and features related to the network characteristics of the connection, extracted from the TCP/IP headers of packets" Northcutt, Novak,. (2001).

Netbios Name lookups provide an example of one type of traffic presenting difficulties. Many MS Windows software applications use the Netbios Name Service (UDP port 137) in lieu of DNS (for example, they may send a Netbios Name query to the address concerned where a DNS lookup times out, as illustrated in Fig. 30).



```
No.  Time .        Source          Destination      Protocol  Info
358 283.978028 192.168.0.3     62.34.195.150    TCP     6346 > 3702 [RST, ACK] Seq=0 Ack=1200779667 Win=0 Len=0
359 284.518159 62.34.195.150   192.168.0.3      TCP     3702 > 6346 [SYN] Seq=1200779666 Ack=0 Win=65535 Len=0 MS
360 284.518345 192.168.0.3     62.34.195.150    TCP     6346 > 3702 [RST, ACK] Seq=0 Ack=1200779667 Win=0 Len=0
361 286.911549 192.168.0.3     192.168.0.2      ICMP    Echo (ping) request
362 286.915992 192.168.0.2     192.168.0.3      ICMP    Echo (ping) reply
363 287.219538 192.168.0.3     194.168.8.100    DNS     Standard query PTR 2.0.168.192.in-addr.arpa
364 287.248183 194.168.8.100   192.168.0.3      DNS     Standard query response, No such name
365 287.290787 192.168.0.3     192.168.0.2      NBNS    Name query NBSTAT *<00><00><00><00><00><00><00><00><00><0
366 287.296312 192.168.0.2     192.168.0.3      NBNS    Name query response NBSTAT
367 287.780780 192.168.0.3     192.168.0.2      SNMP    GET iso.3.6.1.2.1.1.1.0
368 287.785140 192.168.0.2     192.168.0.3      ICMP    Destination unreachable
369 288.749256 192.168.0.3     192.168.0.2      SNMP    GET iso.3.6.1.2.1.1.1.0
370 288.753991 192.168.0.2     192.168.0.3      ICMP    Destination unreachable
```

```
⊞ Frame 365 (92 bytes on wire, 92 bytes captured)
⊟ Ethernet II, Src: 00:10:a7:10:e5:50, Dst: 00:10:7a:4e:4a:7a
    Destination: 00:10:7a:4e:4a:7a (Ambicom_4e:4a:7a)
    Source: 00:10:a7:10:e5:50 (UnexTech_10:e5:50)
    Type: IP (0x0800)
⊞ Internet Protocol, Src Addr: 192.168.0.3 (192.168.0.3), Dst Addr: 192.168.0.2 (192.168.0.2)
⊞ User Datagram Protocol, Src Port: netbios-ns (137), Dst Port: netbios-ns (137)
⊟ NetBIOS Name Service
    Transaction ID: 0xb620
  ⊞ Flags: 0x0010 (Name query)
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ⊞ Queries
```

```
0000  00 10 7a 4e 4a 7a 00 10  a7 10 e5 50 08 00 45 00   ..zNJz.. ...P..E.
0010  00 4e 70 ce 00 00 80 11  48 7b c0 a8 00 03 c0 a8   .Np..... H{......
0020  00 02 00 89 00 89 00 3a  88 a8 b6 20 00 10 00 01   .......: ... ....
0030  00 00 00 00 00 00 20 43  4b 41 41 41 41 41 41 41   ...... C KAAAAAAA
0040  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0050  41 41 41 41 41 41 41 00  00 21 00 01              AAAAAAA. .!..
```

Fig. 30: Netbios Name Service query

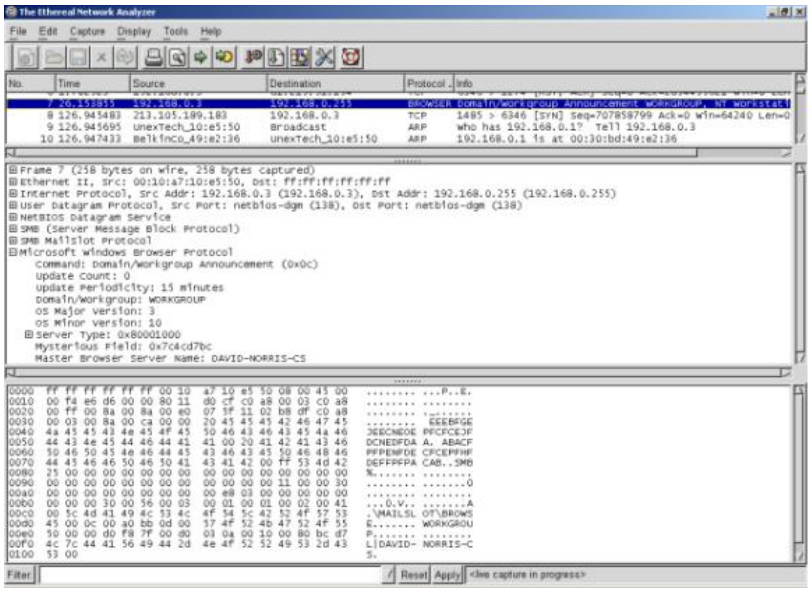
This can result in a large number of false alerts being generated. The use of the ‘Text=’ and ‘Not_Found’ commands provide a means by which it is possible to differentiate between a normal name query, and anomalous activity, such as a UDP port scan to single port. [44] (Although UDP scanning is able to identify services, which utilise UDP for transport, such as TFTP and NTP, UDP scanning is less reliable than TCP scanning, for the simple reason that some packets are dropped; additionally, some sites block ICMP destination unreachable’ messages.)

Whereas the Netbios ports certainly present one of the most easily exploited security exposures, [45] the generation of an alert whenever a normal Netbios name query is detected will result in a great deal of false alerting.

Another example of the use of signature recognition concerns DNS ‘Zone Transfers’. [46] The ‘gethostbyaddr’ function call is widely used by web browsers. Where the signature of a known web browser is absent, a zone transfer is the most probable cause.

A further use for the ‘Text=’ and ‘Not_Found’ commands is to distinguish between individual computers on a network which utilises DHCP. In a situation where addresses change frequently, the DNS or Netbios names can provide a means by which the identity of individual computers can be determined quickly. This can be accomplished by monitoring activity such as Windows Domain Announcements, which are broadcast using the Netbios Datagram service, (Fig. 31). Other possible sources of such information include DNS, WINS and SNMP.

It is, however, vital to realise that signature detection is of use only in combination with protocols, which do not encrypt traffic before its transmission.



```
No.  Time          Source                Destination           Protocol  Info
   7 26.153855     192.168.0.3           192.168.0.255         BROWSER   Domain/Workgroup Announcement WORKGROUP, NT Workstat
   8 126.945483    213.105.189.183       192.168.0.3           TCP       1485 > 6346 [SYN] Seq=707858799 Ack=0 Win=64240 Len=0
   9 126.945695    UnexTech_10:e5:50     Broadcast             ARP       Who has 192.168.0.1?  Tell 192.168.0.3
  10 126.947433    Belkinco_49:e2:36     UnexTech_10:e5:50     ARP       192.168.0.1 is at 00:30:bd:49:e2:36
```

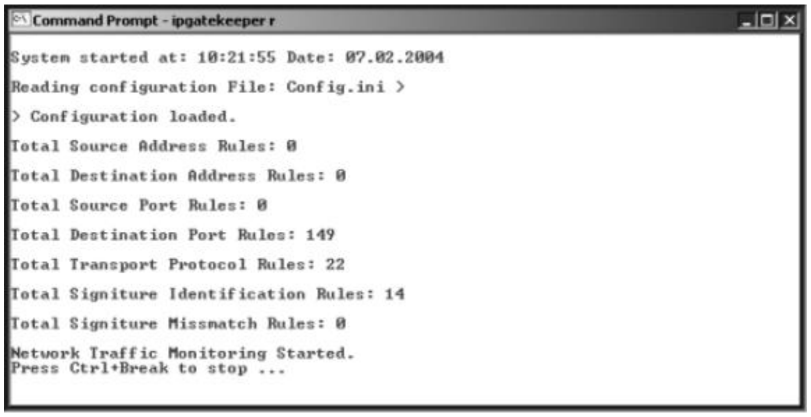
```
⊞Frame 7 (258 bytes on wire, 258 bytes captured)
⊞Ethernet II, Src: 00:10:a7:10:e5:50, Dst: ff:ff:ff:ff:ff:ff
⊞Internet Protocol, Src Addr: 192.168.0.3 (192.168.0.3), Dst Addr: 192.168.0.255 (192.168.0.255)
⊞User Datagram Protocol, Src Port: netbios-dgm (138), Dst Port: netbios-dgm (138)
⊞NetBIOS Datagram Service
⊞SMB (Server Message Block Protocol)
⊞SMB MailSlot Protocol
⊟Microsoft Windows Browser Protocol
    Command: Domain/Workgroup Announcement (0x0c)
    Update Count: 0
    Update Periodicity: 15 minutes
    Domain/Workgroup: WORKGROUP
    OS Major Version: 3
    OS Minor Version: 10
  ⊞Server Type: 0x80001000
    Mysterious Field: 0x7c4cd7bc
    Master Browser Server Name: DAVID-NORRIS-CS
```

```
0000  ff ff ff ff ff ff 00 10  a7 10 e5 50 08 00 45 00   ........ ...P..E.
0010  00 f4 e6 d6 00 00 80 11  d0 cf c0 a8 00 03 c0 a8   ........ ........
0020  00 ff 00 8a 00 8a 00 e0  07 5f 11 02 b8 df c0 a8   ........ ._......
0030  00 03 00 8a 00 ca 00 00  20 45 45 45 42 46 47 45   ........  EEEBFGE
0040  4a 45 45 43 4e 45 4f 45  50 46 43 46 43 45 4a 46   JEECNEOE PFCFCEJF
0050  44 43 4e 45 44 46 44 41  41 00 20 41 42 41 43 46   DCNEDFDA A. ABACF
0060  50 46 50 45 4e 46 44 45  43 46 43 45 50 46 48 46   PFPENFDE CFCEPFHF
0070  44 45 46 46 50 46 50 41  43 41 42 00 ff 53 4d 42   DEFFPFPA CAB..SMB
0080  25 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   %....... ........
0090  00 00 00 00 00 00 00 00  00 00 00 00 11 00 00 30   ........ .......0
00a0  00 00 00 00 00 00 00 00  00 e8 03 00 00 00 00 00   ........ ........
00b0  00 00 00 30 00 56 00 03  00 01 00 01 00 02 00 41   ...0.V.. .......A
00c0  00 5c 4d 41 49 4c 53 4c  4f 54 5c 42 52 4f 57 53   .\MAILSL OT\BROWS
00d0  45 00 0c 00 a0 bb 0d 00  57 4f 52 4b 47 52 4f 55   E....... WORKGROU
00e0  50 00 00 d0 f8 7f 00 d0  03 0a 00 10 00 80 bc d7   P....... ........
00f0  4c 7c 44 41 56 49 44 2d  4e 4f 52 52 49 53 2d 43   L|DAVID- NORRIS-C
0100  53 00                                              S.
```

The Ethereal Network Analyzer
File Edit Capture Display Tools Help
Filter
Reset Apply
<live capture in progress>

Fig. 31: Netbios Broadcast, using the Netbios Datagram Service (UDP port 138)

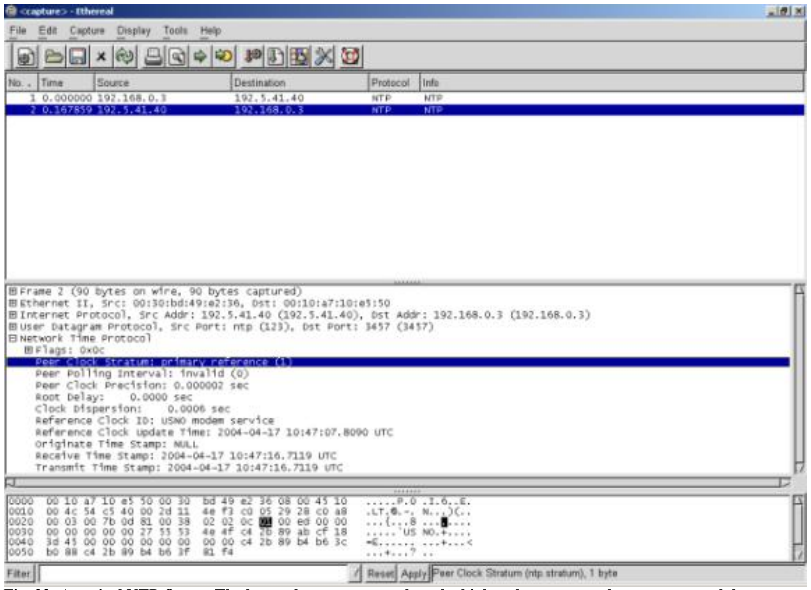
The “Format=” command is used to display payload data. This is only of use in combination with ‘signature’ detection.
This is a description of the record file formatting, using the signature “GET /default.ida?” as an example:
The ‘Format=’ command takes a value of 0 through 6 as it’s parameter. This should ideally be the first command on a line.
Format=0: Will record only the packet source and destination address and port number, the system timestamp, plus the reason the event was logged (the line number in the configuration file). This is the default setting. Also, for ‘specifically logged’ ports (which commonly offer network services on either MS Windows, Unix/Linux or Macintosh systems, whether intentionally or otherwise), or which are used by known malicious software, for example the ‘Netbus Trojan Horse’, a colour coding is used as per the calculated security risk. This risk factor is determined according to a number of factors, including whether or not the traffic is likely to be due to malicious software, which service(s) are known to use the port, and the likelihood of the service to be running with the knowledge of the user (many superfluous services, such as Domain Name Services, email, time, news, file transfer protocol and web services, are often included in a default installation, although the average installation will not need the majority of them!). Any service, which is running without the system owner’s knowledge, will not be maintained, and therefore there is a high probability that one or more of them will be discovered to have security loopholes. Unfortunately, until software developers place more emphasis on security, rather than simply functionality and ‘ease of use’, the situation is unlikely to change.
One further factor influencing the choice of ‘risk factor’ value is the previous security record of a given service; although unfortunately, there is no way of knowing which actual software component or version is responsible for the traffic without further investigation. On startup, confirmation of the system configuration is given, along with the system time and date.



```
System started at: 10:21:55 Date: 07.02.2004
Reading configuration File: Config.ini >
> Configuration loaded.
Total Source Address Rules: 0
Total Destination Address Rules: 0
Total Source Port Rules: 0
Total Destination Port Rules: 149
Total Transport Protocol Rules: 22
Total Signiture Identification Rules: 14
Total Signiture Missmatch Rules: 0
Network Traffic Monitoring Started.
Press Ctrl+Break to stop ...
```

Fig. 32: Startup Summery

NTP provides an accurate time reference, allowing a system's clock to be maintained to a very high level of accuracy. Fig. 33 shows a typical NTP query:



```
No. . Time     Source        Destination   Protocol Info
    1 0.000000 192.168.0.3   192.5.41.40   NTP      NTP
    2 0.167859 192.5.41.40   192.168.0.3   NTP      NTP
```

```
⊞ Frame 2 (90 bytes on wire, 90 bytes captured)
⊞ Ethernet II, Src: 00:30:bd:49:e2:36, Dst: 00:10:a7:10:e5:50
⊞ Internet Protocol, Src Addr: 192.5.41.40 (192.5.41.40), Dst Addr: 192.168.0.3 (192.168.0.3)
⊞ User Datagram Protocol, Src Port: ntp (123), Dst Port: 3457 (3457)
⊟ Network Time Protocol
   ⊞ Flags: 0x0c
     Peer Clock Stratum: primary reference (1)
     Peer Polling Interval: invalid (0)
     Peer Clock Precision: 0.000002 sec
     Root Delay:     0.0000 sec
     Clock Dispersion:     0.0006 sec
     Reference Clock ID: USNO modem service
     Reference Clock Update Time: 2004-04-17 10:47:07.8090 UTC
     Originate Time Stamp: NULL
     Receive Time Stamp: 2004-04-17 10:47:16.7119 UTC
     Transmit Time Stamp: 2004-04-17 10:47:16.7119 UTC
```

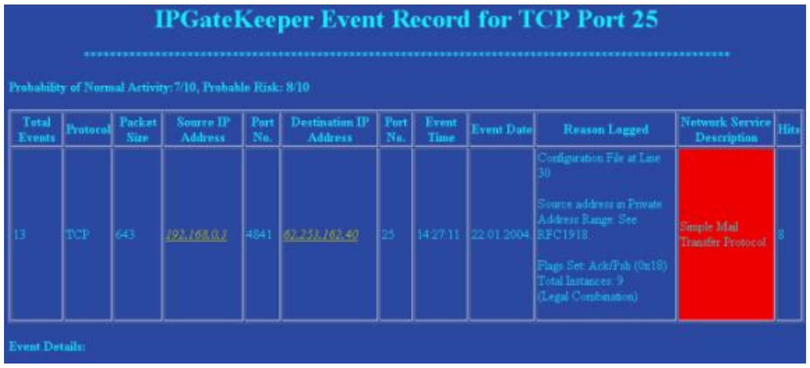
```
0000  00 10 a7 10 e5 50 00 30  bd 49 e2 36 08 00 45 10   .....P.0 .I.6..E.
0010  00 4c 54 c5 40 00 2d 11  4e f3 c0 05 29 28 c0 a8   .LT.@.-. N...)(..
0020  00 03 00 7b 0d 81 00 38  02 02 0c 01 00 ed 00 00   ...{...8 ........
0030  00 00 00 00 00 27 55 53  4e 4f c4 2b 89 ab cf 18   .....'US NO.+....
0040  3d 45 00 00 00 00 00 00  00 00 c4 2b 89 b4 b6 3c   =E...... ...+...<
0050  b0 88 c4 2b 89 b4 b6 3f  81 f4                     ...+...? ..
```

Peer Clock Stratum (ntp stratum), 1 byte

Fig. 33: A typical NTP Query. The lower the stratum number, the higher the accuracy; however network latency must be taken into account also.

The colour scheme used in the packet level output is only intended as a rough guide, particularly in the dynamically assigned port range, where a given port may carry other traffic. Note that no pattern analysis between packets is performed at this level, as this would make subsequent analysis 'conditionally dependent', for example, if only Syn packets were analysed at this stage, this would prevent analysis of patterns between packets at later stages. [47]

The use of Format=0 in the configuration will result in this output:



IPGateKeeper Event Record for TCP Port 25

****

Probability of Normal Activity:7/10, Probable Risk: 8/10

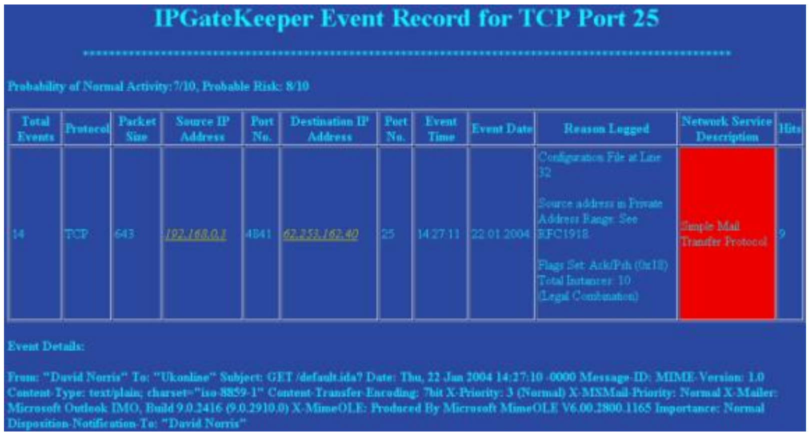
Total Events	Protocol	Packet Size	Source IP Address	Port No.	Destination IP Address	Port No.	Event Time	Event Date	Reason Logged	Network Service Description	Hits
13	TCP	643	192.168.0.1	4841	62.253.162.40	25	14:27:11	22.01.2004	Configuration File at Line 30 Source address in Private Address Range: See RFC1918 Flags Set: Ack/Psh (0x18) Total Instances: 9 (Legal Combination)	Simple Mail Transfer Protocol	8

Event Details:

Fig. 34: Output log generated by the ‘Format=0’ command

No character details are provided; only a summery.

Format=1: additionally records the ASCII content of the Transport Layer Packet:



IPGateKeeper Event Record for TCP Port 25

****

Probability of Normal Activity:7/10, Probable Risk: 8/10

Total Events	Protocol	Packet Size	Source IP Address	Port No.	Destination IP Address	Port No.	Event Time	Event Date	Reason Logged	Network Service Description	Hits
14	TCP	643	192.168.0.1	4841	62.253.162.40	25	14:27:11	22.01.2004	Configuration File at Line 32 Source address in Private Address Range: See RFC1918 Flags Set: Ack/Psh (0x18) Total Instances: 10 (Legal Combination)	Simple Mail Transfer Protocol	9

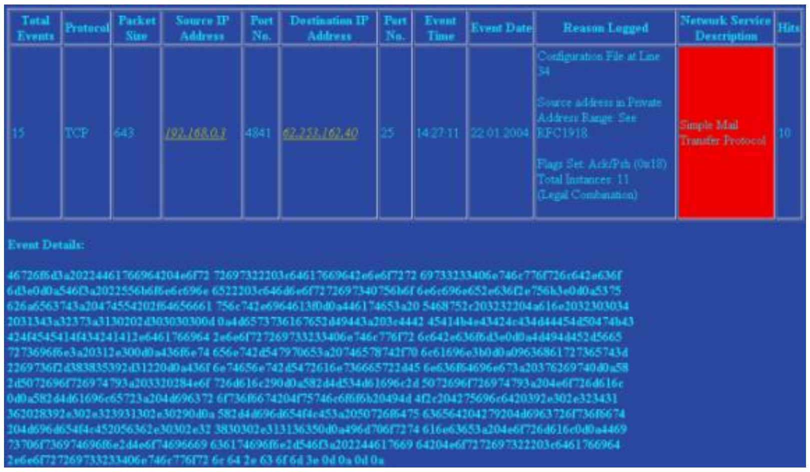
Event Details:

```
From: "David Norris" To: "Ukonline" Subject: GET /default.ida? Date: Thu, 22 Jan 2004 14:27:10 -0000 Message-ID: MIME-Version: 1.0
Content-Type: text/plain; charset="iso-8859-1" Content-Transfer-Encoding: 7bit X-Priority: 3 (Normal) X-MSMail-Priority: Normal X-Mailer:
Microsoft Outlook IMO, Build 9.0.2416 (9.0.2910.0) X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2800.1165 Importance: Normal
Disposition-Notification-To: "David Norris"
```

Fig. 35: Output log generated by the ‘Format=1’ command

Here, the whole ASCII content is recorded; to record only the line containing the signature, simply specify the destination port number and the signature following the ‘Text=’ command.

Format =2: Is identical to Format=1 with the exception that the details are given in hexadecimal:



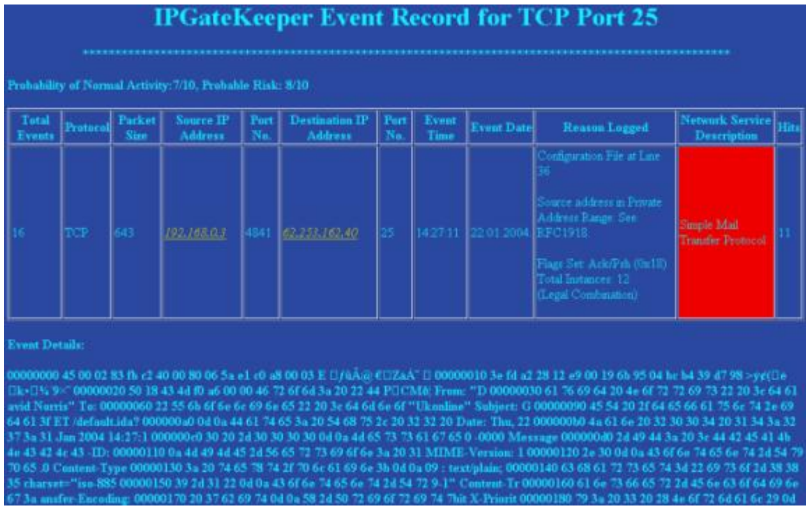
Total Events	Protocol	Packet Size	Source IP Address	Port No.	Destination IP Address	Port No.	Event Time	Event Date	Reason Logged	Network Service Description	Hits
15	TCP	643	192.168.0.1	4841	62.253.162.40	25	14:27:11	22.01.2004	Configuration File at Line 34 Source address in Private Address Range: See RFC1918 Flags Set: Ack/Psh (0x18) Total Instances: 11 (Legal Combination)	Simple Mail Transfer Protocol	10

Event Details:

```
46726f6d3a20224461766964204e6f72 72697322203c64617669642e6e6f7272 69733233406e746c776f726c642e636f
6d3e0d0a546f3a2022556b6f6e6c696e 6522203c646d6e6f7272697340756b6f 6e6c696e652e636f2e756b3e0d0a5375
626a6563743a20474554202f64656661 756c742e6964613f0d0a446174653a20 5468752c203232204a616e2032303034
2031343a32373a3130202d303030300d 0a4d6573736167652d49443a203c4442 45414b4e43424c434d44454d50474b43
424f4545414f434241412e6461766964 2e6e6f727269733233406e746c776f72 6c642e636f6d3e0d0a4d494d452d5665
7273696f6e3a20312e300d0a436f6e74 656e742d547970653a20746578742f70 6c61696e3b0d0a09636861727365743d
2269736f2d383835392d31220d0a436f 6e74656e742d5472616e736665722d45 6e636f64696e673a20376269740d0a58
2d5072696f726974793a203320284e6f 726d616c290d0a582d4d534d61696c2d 5072696f726974793a204e6f726d616c
0d0a582d4d61696c65723a204d696372 6f736f6674204f75746c6f6f6b20494d 4f2c204275696c6420392e302e323431
362028392e302e323931302e30290d0a 582d4d696d654f4c453a2050726f6475 636564204279204d6963726f736f6674
204d696d654f4c452056362e30302e32 3830302e313136350d0a496d706f7274 616e63653a204e6f726d616c0d0a4469
73706f736974696f6e2d4e6f74696669 636174696f6e2d546f3a202244617669 64204e6f7272697322203c6461766964
2e6e6f727269733233406e746c776f72 6c 64 2e 63 6f 6d 3e 0d 0a 0d 0a
```

Fig. 36: Output log generated by the ‘Format=2’ command

Format=3: Records the IP Header details in addition to the Transport layer packet details, in both Hexadecimal and ASCII formats:



IPGateKeeper Event Record for TCP Port 25

****

Probability of Normal Activity:7/10, Probable Risk: 8/10

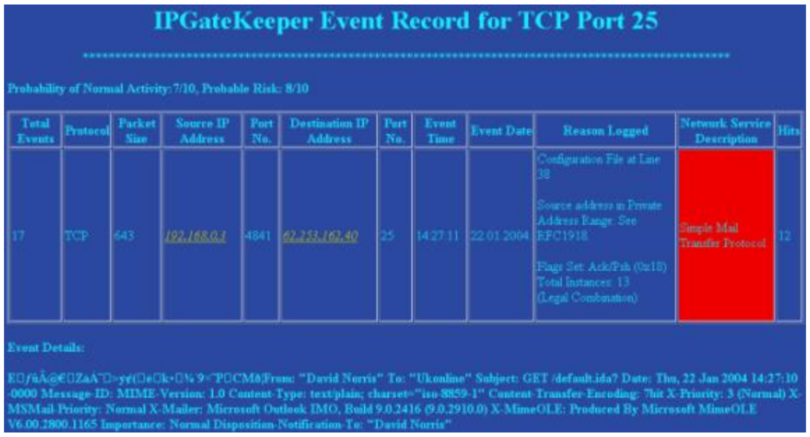
Total Events	Protocol	Packet Size	Source IP Address	Port No.	Destination IP Address	Port No.	Event Time	Event Date	Reason Logged	Network Service Description	Hits
16	TCP	643	192.168.0.3	4841	62.253.162.40	25	14:27:11	22.01.2004	Configuration File at Line 36 Source address in Private Address Range: See RFC1918 Flags Set: Ack/Psh (0x18) Total Instances: 12 (Legal Combination)	Simple Mail Transfer Protocol	11

Event Details:

```
00000000 45 00 02 83 fb c2 40 00 80 06 5a e1 c0 a8 00 03 E □/ûÂ@ €□ZaÀ¨ □ 00000010 3e fd a2 28 12 e9 00 19 6b 95 04 bc b4 39 d7 98 >ý¢(□é
□k•□¼ 9×˜ 00000020 50 18 43 4d f0 a6 00 00 46 72 6f 6d 3a 20 22 44 P□CMð¦ From: "D 00000030 61 76 69 64 20 4e 6f 72 72 69 73 22 20 3c 64 61
avid Norris" To: 00000060 22 55 6b 6f 6e 6c 69 6e 65 22 20 3c 64 6d 6e 6f "Ukonline" Subject: G 00000090 45 54 20 2f 64 65 66 61 75 6c 74 2e 69
64 61 3f ET /default.ida? 000000a0 0d 0a 44 61 74 65 3a 20 54 68 75 2c 20 32 32 20 Date: Thu, 22 000000b0 4a 61 6e 20 32 30 30 34 20 31 34 3a 32
37 3a 31 Jan 2004 14:27:1 000000c0 30 20 2d 30 30 30 30 0d 0a 4d 65 73 73 61 67 65 0 -0000 Message 000000d0 2d 49 44 3a 20 3c 44 42 45 41 4b
4e 43 42 4c 43 -ID: 00000110 0a 4d 49 4d 45 2d 56 65 72 73 69 6f 6e 3a 20 31 MIME-Version: 1 00000120 2e 30 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79
70 65 .0 Content-Type 00000130 3a 20 74 65 78 74 2f 70 6c 61 69 6e 3b 0d 0a 09 : text/plain; 00000140 63 68 61 72 73 65 74 3d 22 69 73 6f 2d 38 38
35 charset="iso-885 00000150 39 2d 31 22 0d 0a 43 6f 6e 74 65 6e 74 2d 54 72 9-1" Content-Tr 00000160 61 6e 73 66 65 72 2d 45 6e 63 6f 64 69 6e
67 3a ansfer-Encoding: 00000170 20 37 62 69 74 0d 0a 58 2d 50 72 69 6f 72 69 74 7bit X-Priorit 00000180 79 3a 20 33 20 28 4e 6f 72 6d 61 6c 29 0d
```

Fig. 37: Output log generated by the 'Format=3' command

Format=4: Records both the Network Layer and Transport layer data in ASCII only:



IPGateKeeper Event Record for TCP Port 25

****

Probability of Normal Activity:7/10, Probable Risk: 8/10

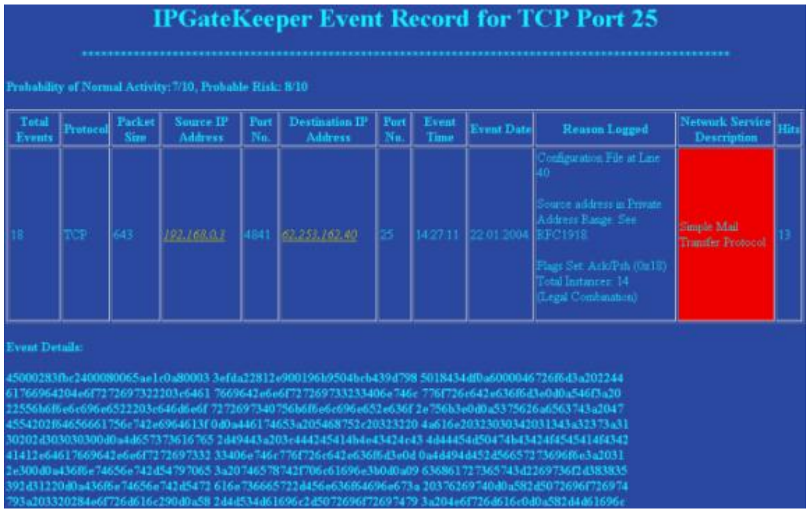
Total Events	Protocol	Packet Size	Source IP Address	Port No.	Destination IP Address	Port No.	Event Time	Event Date	Reason Logged	Network Service Description	Hits
17	TCP	643	192.168.0.3	4841	62.253.162.40	25	14:27:11	22.01.2004	Configuration File at Line 38 Source address in Private Address Range: See RFC1918 Flags Set: Ack/Psh (0x18) Total Instances: 13 (Legal Combination)	Simple Mail Transfer Protocol	12

Event Details:

```
E□/ûÂ@€□ZaÀ¨□>ý¢(□é□k•□¼ 9×˜P□CMð¦From: "David Norris" To: "Ukonline" Subject: GET /default.ida? Date: Thu, 22 Jan 2004 14:27:10
-0000 Message-ID: MIME-Version: 1.0 Content-Type: text/plain; charset="iso-8859-1" Content-Transfer-Encoding: 7bit X-Priority: 3 (Normal) X-
MSMail-Priority: Normal X-Mailer: Microsoft Outlook IMO, Build 9.0.2416 (9.0.2910.0) X-MimeOLE: Produced By Microsoft MimeOLE
V6.00.2800.1165 Importance: Normal Disposition-Notification-To: "David Norris"
```

Fig. 38: Output log generated by the 'Format=4' command

Format=5: Records both the Network Layer and Transport layer data in Hexadecimal only:



IPGateKeeper Event Record for TCP Port 25

****

Probability of Normal Activity:7/10, Probable Risk: 8/10

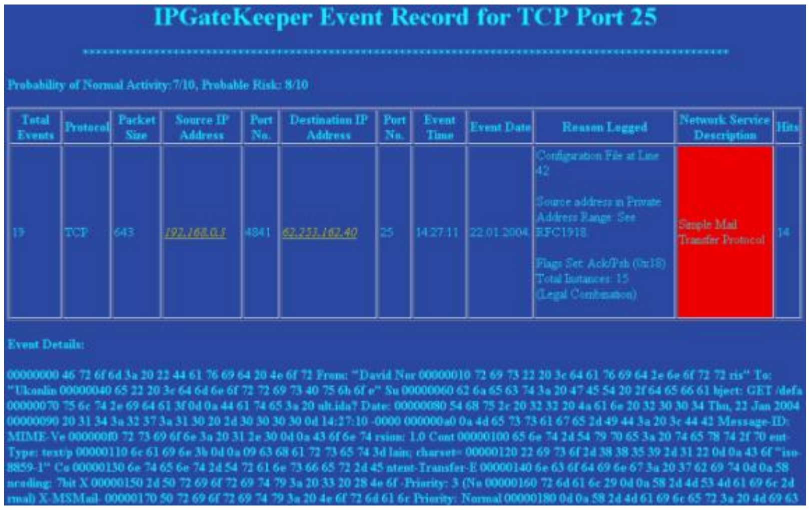
Total Events	Protocol	Packet Size	Source IP Address	Port No.	Destination IP Address	Port No.	Event Time	Event Date	Reason Logged	Network Service Description	Hits
18	TCP	643	192.168.0.3	4841	62.253.162.40	25	14:27:11	22.01.2004	Configuration File at Line 40 Source address in Private Address Range: See RFC1918 Flags Set: Ack/Psh (0x18) Total Instances: 14 (Legal Combination)	Simple Mail Transfer Protocol	13

Event Details:

```
45000283fbc24000800065ae1c0a80003 3efda22812e900196b9504bcb439d798 5018434df0a600004672 6f6d3a202244
617669642044e6f7272697322203c6461 7669642e6e6f7272697332333406e746c 776f726c642e636f6d3e0d0a546f3a20
22556b6f6e6c696e6522203c646d6e6f 7272697340756b6f6e6c696e652e636f 2e756b3e0d0a5375626a6563743a2047
4554202f64656661756c742e6964613f 0d0a4461746 53a205468752c20323220 4a616e2032303034203134 3a32373a31
30202d3030303030 0d0a4d657373616765 2d49443a203c444245414b4e43424c43 4d44454d50474b43424f4545414f4342
41412e64617669642e6e6f7272697332 33406e746c776f726c642e636f6d3e0d 0a4d494d452d56657273696f6e3a2031
2e300d0a436f6e74656e742d54797065 3a20746578742f706c61696e3b0d0a09 636861727365743d2269736f2d383835
392d31220d0a436f6e74656e742d5472 616e736665722d456e636f64696e673a 20376269740d0a582d5072696f726974
793a203320284e6f726d616c290d0a58 2d4d534d61696c2d5072696f72697479 3a204e6f726d616c0d0a582d4d61696c
```

Fig. 39: Output log generated by the 'Format=5' command

Format=6: Records only the Transport layer data as Hexadecimal and ASCII:



IPGateKeeper Event Record for TCP Port 25

****

Probability of Normal Activity:7/10, Probable Risk: 8/10

Total Events	Protocol	Packet Size	Source IP Address	Port No.	Destination IP Address	Port No.	Event Time	Event Date	Reason Logged	Network Service Description	Hits
19	TCP	643	192.168.0.1	4841	62.253.162.40	25	14:27:11	22.01.2004	Configuration File at Line 42 Source address in Private Address Range: See RFC1918 Flags Set: Ack/Psh (0x18) Total Instances: 15 (Legal Combination)	Simple Mail Transfer Protocol	14

Event Details:

```
00000000 46 72 6f 6d 3a 20 22 44 61 76 69 64 20 4e 6f 72 From: "David Nor 00000010 72 69 73 22 20 3c 64 61 76 69 64 2e 6e 6f 72 72 ris" To:
"Ukonlin 00000040 65 22 20 3c 64 6d 6e 6f 72 72 69 73 40 75 6b 6f e" Su 00000060 62 6a 65 63 74 3a 20 47 45 54 20 2f 64 65 66 61 bject: GET /defa
00000070 75 6c 74 2e 69 64 61 3f 0d 0a 44 61 74 65 3a 20 ult.ida? Date: 00000080 54 68 75 2c 20 32 32 20 4a 61 6e 20 32 30 30 34 Thu, 22 Jan 2004
00000090 20 31 34 3a 32 37 3a 31 30 20 2d 30 30 30 30 0d 14:27:10 -0000 000000a0 0a 4d 65 73 73 61 67 65 2d 49 44 3a 20 3c 44 42 Message-ID:
MIME-Ve 000000f0 72 73 69 6f 6e 3a 20 31 2e 30 0d 0a 43 6f 6e 74 rsion: 1.0 Cont 00000100 65 6e 74 2d 54 79 70 65 3a 20 74 65 78 74 2f 70 ent-
Type: text/p 00000110 6c 61 69 6e 3b 0d 0a 09 63 68 61 72 73 65 74 3d lain; charset= 00000120 22 69 73 6f 2d 38 38 35 39 2d 31 22 0d 0a 43 6f "iso-
8859-1" Co 00000130 6e 74 65 6e 74 2d 54 72 61 6e 73 66 65 72 2d 45 ntent-Transfer-E 00000140 6e 63 6f 64 69 6e 67 3a 20 37 62 69 74 0d 0a 58
ncoding: 7bit X 00000150 2d 50 72 69 6f 72 69 74 79 3a 20 33 20 28 4e 6f -Priority: 3 (No 00000160 72 6d 61 6c 29 0d 0a 58 2d 4d 53 4d 61 69 6c 2d
rmal) X-MSMail- 00000170 50 72 69 6f 72 69 74 79 3a 20 4e 6f 72 6d 61 6c Priority: Normal 00000180 0d 0a 58 2d 4d 61 69 6c 65 72 3a 20 4d 69 63
```

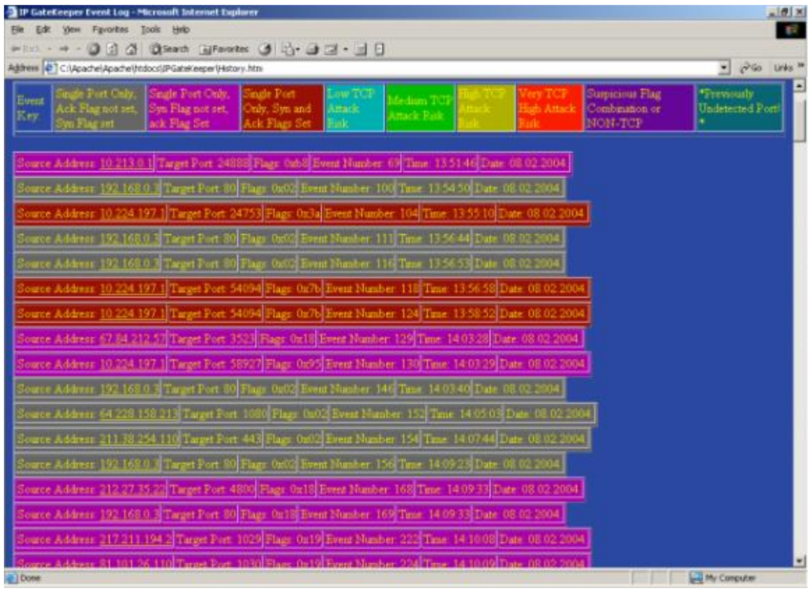
Fig. 40: Output log generated by the 'Format=6' command

Although I had originally planned to resolve the IP addresses used in the records to hostnames (eliminating the requirement to do this manually), I decided not to do so, for three main reasons:

- As addresses are numerical values, it is easier to perform analysis of addresses, for example, to determine whether they have been falsified;
- DNS entries can change over time; for example, due to change of ownership;
- It is not unknown for domain name servers to be compromised, in order to provide false information.

There are some cases, however, where the IP address may itself be incorrect! This is not due to a software 'bug', but deliberate falsification on the part of the intruder. Some tools, such as NMAP, disguise the source of a scan by also sending a number of packets with falsified addresses. If one of the false addresses happens to match your own, you will see the responses. Also, a variant of the 'Syn' scan sets the source address in the IP header to that of another machine within the same subnet. As a network adapter operating in promiscuous mode is able to capture packets destined for other addresses in the same subnet, the scanning tool is able to capture the responses, in order to determine whether ports on the target are open, irrespective of whether the machine having this particular address is up or not! In this case, it is not possible to determine the actual machine carrying out the scan, but it is possible to conclude which subnet the scan originated from. [48]

Furthermore, certain types of DOS attack, which (unlike scanning) do not require a response to the attacker, use one or more false source addresses. This makes the culprit virtually untraceable. In this example, the address 10.224.197.1 is obviously falsified: [49]



IP GateKeeper Event Log - Microsoft Internet Explorer

Address: C:\Apache\Apache\htdocs\IPGateKeeper\History.htm

Event Key	Single Port Only, Ack Flag not set, Syn Flag set	Single Port Only, Syn Flag not set, ack Flag Set	Single Port Only, Syn and Ack Flags Set	Low TCP Attack Risk	Medium TCP Attack Risk	High TCP Attack Risk	Very TCP High Attack Risk	Suspicious Flag Combination or NON-TCP	*Previously Undetected Port! *

Source Address: 10.213.0.1	Target Port: 24888	Flags: 0xb8	Event Number: 69	Time: 13:51:46	Date: 08.02.2004
Source Address: 192.168.0.3	Target Port: 80	Flags: 0x02	Event Number: 100	Time: 13:54:50	Date: 08.02.2004
Source Address: 10.224.197.1	Target Port: 24753	Flags: 0x3a	Event Number: 104	Time: 13:55:10	Date: 08.02.2004
Source Address: 192.168.0.3	Target Port: 80	Flags: 0x02	Event Number: 111	Time: 13:56:44	Date: 08.02.2004
Source Address: 192.168.0.3	Target Port: 80	Flags: 0x02	Event Number: 116	Time: 13:56:53	Date: 08.02.2004
Source Address: 10.224.197.1	Target Port: 54094	Flags: 0x7b	Event Number: 118	Time: 13:56:58	Date: 08.02.2004
Source Address: 10.224.197.1	Target Port: 54094	Flags: 0x7b	Event Number: 124	Time: 13:58:52	Date: 08.02.2004
Source Address: 67.84.212.57	Target Port: 3523	Flags: 0x18	Event Number: 129	Time: 14:03:28	Date: 08.02.2004
Source Address: 10.224.197.1	Target Port: 58927	Flags: 0x95	Event Number: 130	Time: 14:03:29	Date: 08.02.2004
Source Address: 192.168.0.3	Target Port: 80	Flags: 0x02	Event Number: 146	Time: 14:03:40	Date: 08.02.2004
Source Address: 64.228.158.213	Target Port: 1080	Flags: 0x02	Event Number: 152	Time: 14:05:03	Date: 08.02.2004
Source Address: 211.38.254.110	Target Port: 443	Flags: 0x02	Event Number: 154	Time: 14:07:44	Date: 08.02.2004
Source Address: 192.168.0.3	Target Port: 80	Flags: 0x02	Event Number: 156	Time: 14:09:23	Date: 08.02.2004
Source Address: 212.27.35.22	Target Port: 4800	Flags: 0x18	Event Number: 168	Time: 14:09:33	Date: 08.02.2004
Source Address: 192.168.0.3	Target Port: 80	Flags: 0x18	Event Number: 169	Time: 14:09:33	Date: 08.02.2004
Source Address: 217.211.194.2	Target Port: 1029	Flags: 0x19	Event Number: 222	Time: 14:10:08	Date: 08.02.2004
Source Address: 81.101.26.110	Target Port: 1030	Flags: 0x19	Event Number: 224	Time: 14:10:09	Date: 08.02.2004

Fig. 41: Falsified ('Spoofed') IP addresses

Here are some examples of general command usage. The actual command is shown in italics:

Format=0 Record=Audit.htm - will match all events. Packet details will appear in Audit.htm, and the traffic analysis will be performed on all network traffic, regardless of addresses, port numbers, and other parameters. Note that this is very resource intensive, and should never be carried out except for testing purposes under low traffic conditions.

Format=0 Lower_Size_Limit=128 Upper_Size_Limit=1024 Record=Audit.htm –
As the first example, only analysis of packets within the specified range will be analyzed. The size constraints can often be used to minimise resource usage or the generation of false positives (As an example, if an attack tool never generated packets outside a certain size range).

Protocol=6 Format=0 Record=tcp.htm – Performs analysis of TCP traffic alone. Other transport layer protocols are ignored.

Format=0 Dest_Port=513 Record=c:\Apache\Apache\htdocs\ipgatekeeper\Rlogin.htm
- Analyses traffic destined for port 513 (Remote Login), and records details to a file in another directory (It is essential to specify the path correctly!).

Format=0 Host_Dest=192.168.0.3 Dest_Port=80 Text=GET /default.ida? Record=CodeRed.htm – Will monitor traffic directed to the specific host 192.168.0.3, port 80, containing the 'Code Red' worm's signature. As commands appearing on the same configuration line are Boolean 'ANDed' together, ALL of the conditions must be met for events to be recorded.

As lines are 'ORed' together, it is possible to 'OR' two or more conditions together simply by placing them on two separate lines. For example, the following combination:

Format=1 Lower_Size_Limit=1500 File=Example.htm
Format=1 Dest_Port=69 File=Example.htm

Would match traffic if the packet size exceeded 1500 bytes, OR if the target port is port 69. The results can be saved either in the same file, or separate files, according to the File= command.

Config.ini is the configuration file, as this is the default setting.

This is an activity diagram, demonstrating how each configuration rule is implemented. All of the commands are included, except for Protocol=.

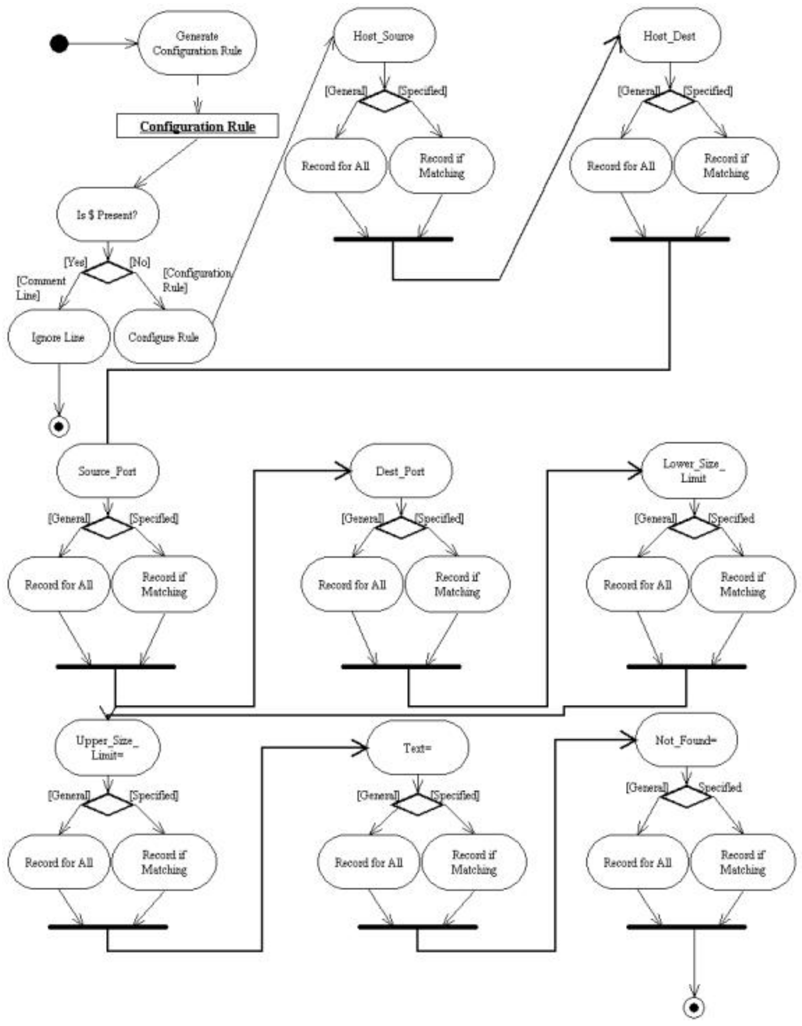


Fig. 42: Configuration Command Activity Diagram (Using 'Logic Trees')

As previously mentioned, commands on the same line are Boolean 'ANDed' to produce rules, whereas rules (separate lines) are Boolean 'ORed' together. Therefore:
Rules *must* be declared entirely on one single line.
Where a command is not present in a line, it is assumed to have a Boolean value of 1; (As A . 1 equates to A; this is one of the laws of Boolean algebra). This ensures that unused commands have no effect;
Nonsensical combinations of commands will result in nothing being recorded; for example Dest_Port=80 Dest_Port=113 in the same rule will not match any event! This is simply because both conditions can never be met simultaneously (A . NOT A is never true!).
This is the Boolean logic tree implementation for a single given rule:

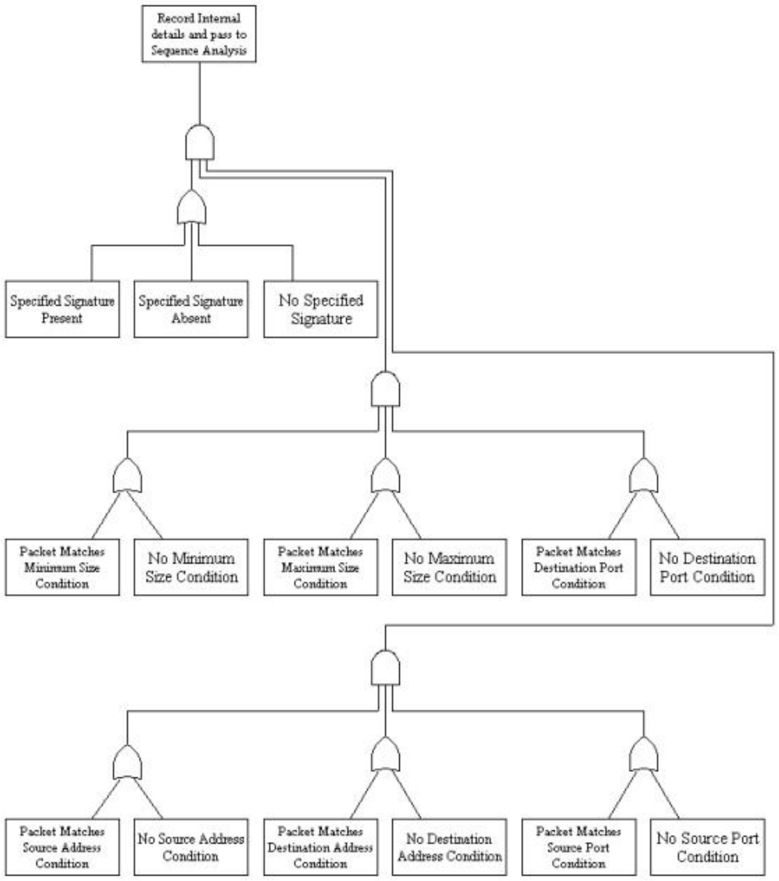

Record Internal details and pass to Sequence Analysis
Specified Signature Present
Specified Signature Absent
No Specified Signature
Packet Matches Minimum Size Condition
No Minimum Size Condition
Packet Matches Maximum Size Condition
No Maximum Size Condition
Packet Matches Destination Port Condition
No Destination Port Condition
Packet Matches Source Address Condition
No Source Address Condition
Packet Matches Destination Address Condition
No Destination Address Condition
Packet Matches Source Port Condition
No Source Port Condition

Fig. 43: Logic Tree for a single rule [50]

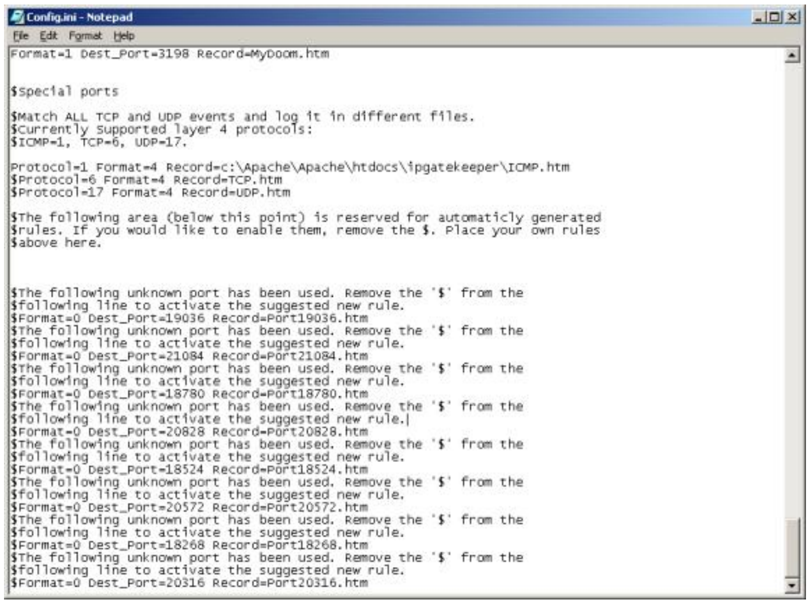
ii. Active Program Configuration

Since one problem to which IDS users are all too accustomed, is the need to manually configure the IDS, either in response to new known threats, or in response to false positive/negative results. Updating rule-based systems can be very prodigal of staff time, particularly where the IDS is monitoring a large network connecting a variety of systems. In order to address the HCI aspect of system design, it is desirable to minimise the amount of monitoring and administration required. For this reason, I decided to add a capability for the system to 'generate' rules in response to network events.

This, however, raises a potential problem: Will the IDS generate, and implement, new rules based on false alerting? If this behavior should occur, then in effect the IDS would present the user with a 'worst of both worlds' scenario! [51]

- In order to avoid this danger, I have decided to compromise. Whilst allowing IPGateKeeper to generate rules based on events encountered, I have decided to write them to the end configuration file such that they are commented out. The user can then decide the course of action.

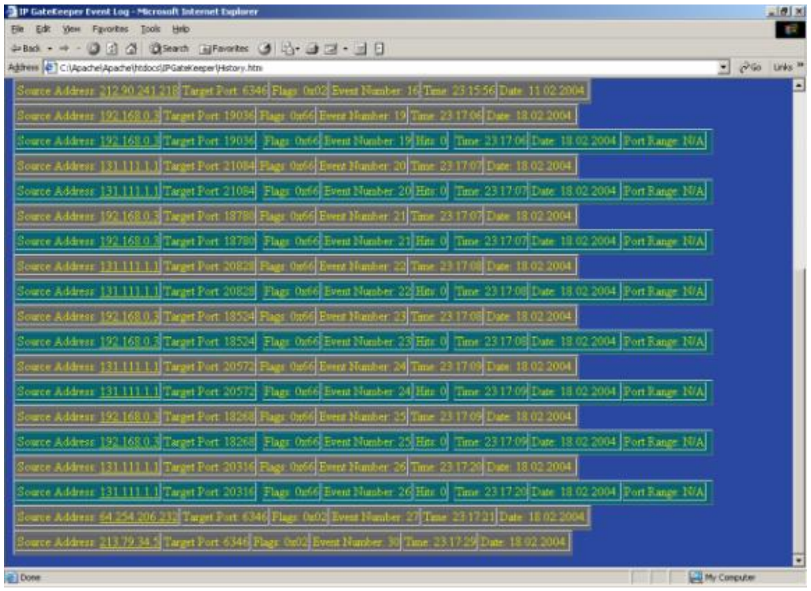
The automatic generation of rules must only take place once the system has been running for an adequate period of time.



```
Format=1 Dest_Port=3198 Record=MyDoom.htm


$Special ports

$Match ALL TCP and UDP events and log it in different files.
$Currently supported layer 4 protocols:
$ICMP=1, TCP=6, UDP=17.

Protocol=1 Format=4 Record=c:\Apache\Apache\htdocs\ipgatekeeper\ICMP.htm
$Protocol=6 Format=4 Record=TCP.htm
$Protocol=17 Format=4 Record=UDP.htm

$The following area (below this point) is reserved for automaticly generated
$rules. If you would like to enable them, remove the $. Place your own rules
$above here.



$The following unknown port has been used. Remove the '$' from the
$following line to activate the suggested new rule.
$Format=0 Dest_Port=19036 Record=Port19036.htm
$The following unknown port has been used. Remove the '$' from the
$following line to activate the suggested new rule.
$Format=0 Dest_Port=21084 Record=Port21084.htm
$The following unknown port has been used. Remove the '$' from the
$following line to activate the suggested new rule.
$Format=0 Dest_Port=18780 Record=Port18780.htm
$The following unknown port has been used. Remove the '$' from the
$following line to activate the suggested new rule.|
$Format=0 Dest_Port=20828 Record=Port20828.htm
$The following unknown port has been used. Remove the '$' from the
$following line to activate the suggested new rule.
$Format=0 Dest_Port=18524 Record=Port18524.htm
$The following unknown port has been used. Remove the '$' from the
$following line to activate the suggested new rule.
$Format=0 Dest_Port=20572 Record=Port20572.htm
$The following unknown port has been used. Remove the '$' from the
$following line to activate the suggested new rule.
$Format=0 Dest_Port=18268 Record=Port18268.htm
$The following unknown port has been used. Remove the '$' from the
$following line to activate the suggested new rule.
$Format=0 Dest_Port=20316 Record=Port20316.htm
```

Fig. 44: Sample automatically generated rules in response to events

The generation of a new rule is also accompanied by an alert, and an entry in the history report:



```
Source Address: 212.90.241.218 Target Port: 6346 Flags: 0x02 Event Number: 16 Time: 23:15:56 Date: 11.02.2004
Source Address: 192.168.0.3 Target Port: 19036 Flags: 0x66 Event Number: 19 Time: 23:17:06 Date: 18.02.2004
Source Address: 192.168.0.3 Target Port: 19036 Flags: 0x66 Event Number: 19 Hits: 0 Time: 23:17:06 Date: 18.02.2004 Port Range: N/A
Source Address: 131.111.1.1 Target Port: 21084 Flags: 0x66 Event Number: 20 Time: 23:17:07 Date: 18.02.2004
Source Address: 131.111.1.1 Target Port: 21084 Flags: 0x66 Event Number: 20 Hits: 0 Time: 23:17:07 Date: 18.02.2004 Port Range: N/A
Source Address: 192.168.0.3 Target Port: 18780 Flags: 0x66 Event Number: 21 Time: 23:17:07 Date: 18.02.2004
Source Address: 192.168.0.3 Target Port: 18780 Flags: 0x66 Event Number: 21 Hits: 0 Time: 23:17:07 Date: 18.02.2004 Port Range: N/A
Source Address: 131.111.1.1 Target Port: 20828 Flags: 0x66 Event Number: 22 Time: 23:17:08 Date: 18.02.2004
Source Address: 131.111.1.1 Target Port: 20828 Flags: 0x66 Event Number: 22 Hits: 0 Time: 23:17:08 Date: 18.02.2004 Port Range: N/A
Source Address: 192.168.0.3 Target Port: 18524 Flags: 0x66 Event Number: 23 Time: 23:17:08 Date: 18.02.2004
Source Address: 192.168.0.3 Target Port: 18524 Flags: 0x66 Event Number: 23 Hits: 0 Time: 23:17:08 Date: 18.02.2004 Port Range: N/A
Source Address: 131.111.1.1 Target Port: 20572 Flags: 0x66 Event Number: 24 Time: 23:17:09 Date: 18.02.2004
Source Address: 131.111.1.1 Target Port: 20572 Flags: 0x66 Event Number: 24 Hits: 0 Time: 23:17:09 Date: 18.02.2004 Port Range: N/A
Source Address: 192.168.0.3 Target Port: 18268 Flags: 0x66 Event Number: 25 Time: 23:17:09 Date: 18.02.2004
Source Address: 192.168.0.3 Target Port: 18268 Flags: 0x66 Event Number: 25 Hits: 0 Time: 23:17:09 Date: 18.02.2004 Port Range: N/A
Source Address: 131.111.1.1 Target Port: 20316 Flags: 0x66 Event Number: 26 Time: 23:17:20 Date: 18.02.2004
Source Address: 131.111.1.1 Target Port: 20316 Flags: 0x66 Event Number: 26 Hits: 0 Time: 23:17:20 Date: 18.02.2004 Port Range: N/A
Source Address: 64.254.206.232 Target Port: 6346 Flags: 0x02 Event Number: 27 Time: 23:17:21 Date: 18.02.2004
Source Address: 213.79.34.5 Target Port: 6346 Flags: 0x02 Event Number: 30 Time: 23:17:29 Date: 18.02.2004
```

IP GateKeeper Event Log - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Apache\Apache\htdocs\IPGateKeeper\History.htm
Done
My Computer

Fig. 45: Notification of 'anomalous' port usage

Although a lot of event anomalies could be used to trigger 'automatic reconfiguration', I decided to apply the procedure to the use of 'anomalous' IP ports only. The reasons are as follows:

- I took the decision not to make the detection of flag settings configurable, since in general legal settings do not change greatly over time; additionally flag anomalies are applicable only to TCP;
- IP addresses could also be used as a basis for 'automatic reconfiguration' to take place; however it is generally quite normal for previously unknown addresses to legitimately access public access servers; were addresses used, a large amount of 'false positives' would be generated. Also, the use of IP address falsification would allow for a DOS attack on the IDS itself. (The detection of reserved addresses is already 'hard coded' in any event.)

The inherent strength in ‘Active Program Configuration’ is its ability to notify the user of changes, which have taken place during runtime. Port usage is a prime candidate for this type of time-based configuration, since many popular applications can be manually (or in some cases, even automatically) configured to bypass simple port filtering firewall rules. Some popular ‘peer to peer’ file sharing applications are an example. Furthermore, as most such applications listen on ports in the dynamic range, it is possible to differentiate between ‘listening server related’ traffic, and secondary TCP connections, only by determining whether traffic is commenced by means of a Syn packet directed to a given host on a dynamic port (and whether that host then responds with an Ack/Syn packet).
This illustrates yet another point – many firewalls and IDS simply monitor for incoming connection attempts; it is also useful to be able to answer the question: did anything respond, and if so, was the port open? As an example, no ‘legitimate’ application is known to listen on TCP port 12345; the only known application to do so by default is the ‘Netbus’ Trojan horse. If the IDS records not only this incident, but also the fact that a system behind the gateway responded with an Ack/Syn packet, then it has also recorded the fact that a host is *probably* running this backdoor, provided that the system involved was running a Microsoft OS. [52]
Although beyond the scope of this project, ‘Active Program Configuration’ could also be modified to update firewall configuration files, given that the location and formatting of the log file was known. However, many vendors use binary files for this purpose, and keep the formatting of this file a closely guarded secret. Additionally, many hardware/firmware devices require authentication prior to configuration changes.

VIII. Configuration Based Vs Adaptive Detection

Although the configuration based detection method does accurately detect patterns, which exactly match specified conditions, we face obvious problems in the sense that small deviations from expected characteristics will cause events to be missed in their entirety. Worse still, it is not possible to write rules, which match unknown threats. Also, there is the difficulty that, given the combination of the multitude of potential intruders, and the rate at which new exploits are discovered, updating the configuration quickly becomes a never-ending ‘arms race’.
Adaptive detection, since it attempts to learn the characteristics of network traffic as it is actually ‘seen’, is free of this shortcoming. For example, it is possible to monitor the use of IP ports [53], or IP addresses which normally connect to given destinations. In fact, these two methods can be combined, giving a record of hosts connecting to which services, on which hosts, for a given ‘learning’ period. However, research by Huberman

& Adamic (1999), would suggest that this technique would be inadequate for many variables, as the observed 'rules' would simply continue to evolve and change, however long the 'learning' period!

"For example, the range of Internet hosts running Web servers (HTTP) alone, has a Pareto distribution" Huberman, Adamic, (1999). [54]

The significance of this is that, as this is what mathematicians refer to as a 'power law distributed random variable', in which it is impossible to establish a 'pseudonormal' set of values regardless of the sampling interval! This also implies that it is impossible, given the pattern of network traffic, to establish accurately a 'mean' figure for events in a given short time period, for example, a mean number of packets detected per second/minute/hour of monitoring, or bandwidth. Although theoretically a longer-term average would demonstrate increased accuracy, we still face the difficulty that network traffic tends, on both the longer and shorter timescale, to occur in 'peaks and troughs', where a peak or trough can have a time duration ranging from milliseconds to months! These are superimposed on top of each other; the resulting patterns are only partially predictable.

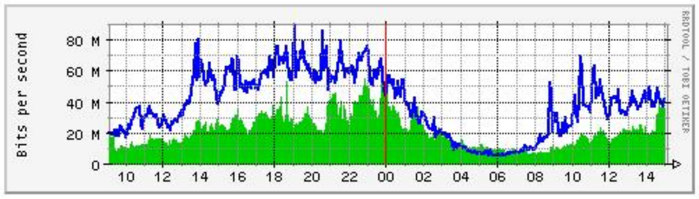
These example traffic graphs (Fig. 46), which are generated by the Multi Router Traffic Gopher, [55] and have been obtained from a network node illustrates the extent of the difficulty. Although there are some predictable underlying patterns, for example, the evening peak which would be anticipated in student accommodation, [56] and the noticeable reduction during holiday periods, there remains a great deal of fluctuation between identical times on various days of a given week.

A further complication arises from the data format – although the Multi-Router traffic gopher gives a fairly accurate indication of the traffic flow per sampling interval, (for example 5 minutes for the Daily graph), this does not demonstrate the short-term fluctuations, the duration of which is measured in seconds.

Resnick, (1997), describes the problem: *"We are given a set of instances, (e.g. network packets or user sessions), some of which have category labels, e.g. normal or hostile. The problem is to assign labels to the remaining instances."*

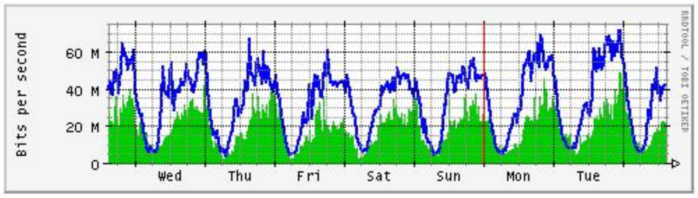
In general, the main difficulty results from the assumption that an adaptive system assumes, often wrongly, that events encountered during the learning phase are normal, whereas events encountered subsequently are hostile.

`Daily' Graph (5 Minute Average)



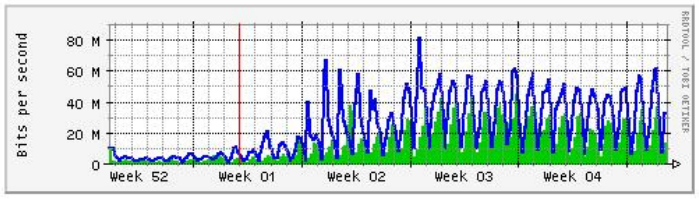
Max In: 59.7 Mb/s (6.0%) Average In: 21.8 Mb/s (2.2%) Current In: 31.3 Mb/s (3.1%)
Max Out: 88.8 Mb/s (8.9%) Average Out: 38.4 Mb/s (3.8%) Current Out: 39.6 Mb/s (4.0%)

`Weekly' Graph (30 Minute Average)



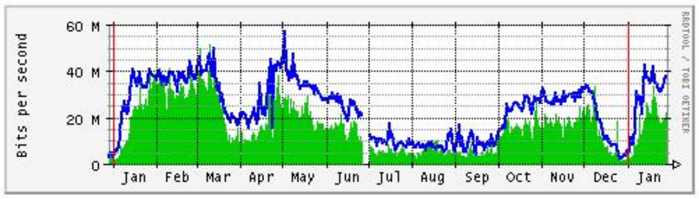
Max In: 48.9 Mb/s (4.9%) Average In: 19.8 Mb/s (2.0%) Current In: 37.1 Mb/s (3.7%)
Max Out: 71.3 Mb/s (7.1%) Average Out: 33.8 Mb/s (3.4%) Current Out: 41.7 Mb/s (4.2%)

`Monthly' Graph (2 Hour Average)



Max In: 55.6 Mb/s (5.6%) Average In: 14.1 Mb/s (1.4%) Current In: 20.7 Mb/s (2.1%)
Max Out: 80.6 Mb/s (8.1%) Average Out: 23.3 Mb/s (2.3%) Current Out: 38.9 Mb/s (3.9%)

`Yearly' Graph (1 Day Average)



Max In: 51.5 Mb/s (5.2%) Average In: 16.5 Mb/s (1.7%) Current In: 22.0 Mb/s (2.2%)
Max Out: 57.0 Mb/s (5.7%) Average Out: 23.6 Mb/s (2.4%) Current Out: 37.6 Mb/s (3.8%)

Fig. 46: Example Output for the 'Multi Router Traffic Gopher'

IX. Detection Thresholds

It is possible for 'legitimate' connection attempts to be made to a small number of disparate IP ports. An organization may be intentionally offering a number of services to the Internet at large on various ports, which may include a public web server, FTP services, email servers for internal users, and possibly SSH or Rlogin servers, intended for employees who may sometimes work from home.
Although many detection systems utilise configurable thresholds, the use of linear alerting procedures will result in a large number of false positives. Furthermore, there will always be the difficulty of attempting to find the 'best compromise' between the possible missing of events of interest, and an unacceptably high level of 'false positives'.

For this reason, I have experimented with a logarithmic scale for categorizing levels of alert. Logarithmic functions have the useful property of having a high sensitivity for low linear quantities, whereas each order of magnitude only doubles the logarithmic value. [57] The equation itself, however, must meet the following criteria:

- Although logarithmic, values must be contained within a ‘ceiling’. This is because, if, for example, we wish to rate the likelihood of an event being connected with hostile activity on a scale of, for example, 0 – 100.
- A logarithmic scale provides a means by which to differentiate between ‘low’ and ‘medium’ anomaly levels, for example, ‘low’ levels may be ignored entirely, whereas ‘medium’ levels may be recorded or used for calibration purposes, but may not trigger an urgent alarm.
- Another benefit of logarithmic scaling is that the frequency of events spanning multiple orders of frequency can be easily represented on the same plot. For example, it is possible to view trends relating to IP port usage on the same time based graph, even though the expected or actual, traffic levels may be different by several orders of magnitude.

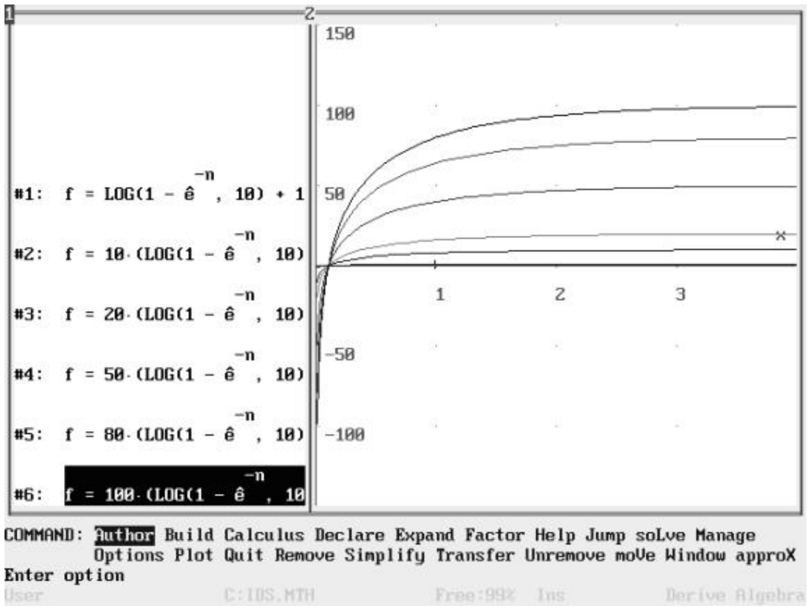
The formula I have developed is as follows: [58]

$$F = \sum_{n=1}^{n_{events}} S\left(\mathrm{LOG}_{10}\left(1 - e^{-dn}\right) + 1\right)$$

The variable definitions are:

- ‘S’ is a scalar, which determines the scale over which we wish to categorise events. For example, if we desired a scale of 0 – 10, S would be equal to 10.
- ‘n’ is the number of occurrences of a given event.
- ‘d’ is a delaying factor. This allows the ‘time constant’, τ. This makes it possible to configure the number of events required to reach a given rating.

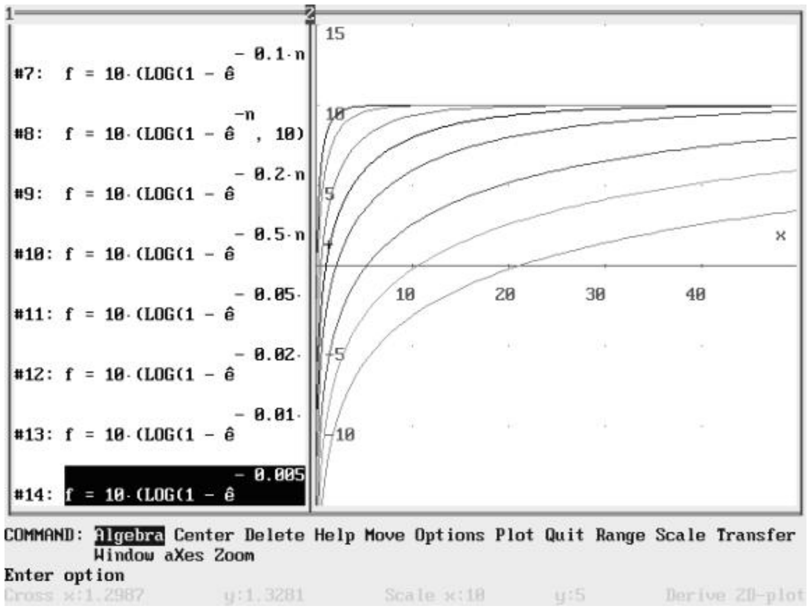
The following graph demonstrates the effect of the variable ‘S’:



```
#1:  f = LOG(1 - ê^-n , 10) + 1
#2:  f = 10·(LOG(1 - ê^-n , 10)
#3:  f = 20·(LOG(1 - ê^-n , 10)
#4:  f = 50·(LOG(1 - ê^-n , 10)
#5:  f = 80·(LOG(1 - ê^-n , 10)
#6:  f = 100·(LOG(1 - ê^-n , 10
COMMAND: Author Build Calculus Declare Expand Factor Help Jump soLve Manage
         Options Plot Quit Remove Simplify Transfer Unremove moVe Window approX
Enter option
User              C:IDS.MTH          Free:99%  Ins          Derive Algebra
```

Fig. 47: Graph of the above equation

For each ‘S’ value, ‘F’ never exceeds ‘S’. The effect of ‘d’ is also summarised:



```
                               - 0.1·n
#7:  f = 10·(LOG(1 - ê

                               -n
#8:  f = 10·(LOG(1 - ê  , 10)

                               - 0.2·n
#9:  f = 10·(LOG(1 - ê

                               - 0.5·n
#10: f = 10·(LOG(1 - ê

                               - 0.05·
#11: f = 10·(LOG(1 - ê

                               - 0.02·
#12: f = 10·(LOG(1 - ê

                               - 0.01·
#13: f = 10·(LOG(1 - ê

                               - 0.005
#14: f = 10·(LOG(1 - ê

COMMAND: Algebra Center Delete Help Move Options Plot Quit Range Scale Transfer
         Window aXes Zoom
Enter option
Cross x:1.2987      y:1.3281        Scale x:10      y:5        Derive 2D-plot
```

Fig. 48: Effect of changing the ‘delay’ variable

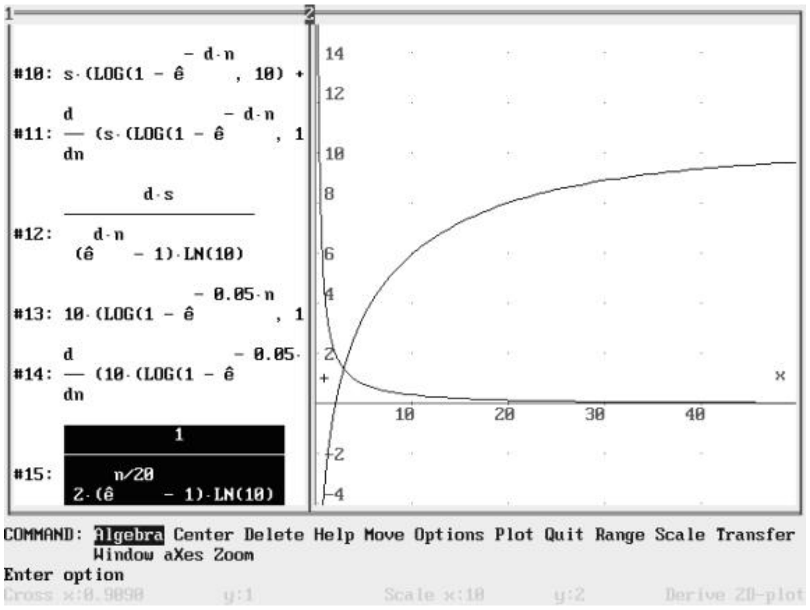
The ‘n’ axis illustrates the effect of ‘d’, on τ. The initial negative values are not a concern, since any negative ‘f’ value is considered equal to 0. However, it is important not to attempt to calculate ‘f’ whilst ‘n’ is zero, as f = - ∞ at this point, as is common to all logarithmic functions.

The exponential decay component is required to prevent ‘f’ from ever exceeding ‘S’, for any value of ‘n’. Theoretically, ‘f’ never quite achieves the value of ‘S’, since as ‘n’ tends to infinity, the gradient of the graph tends to 0. The gradient at any given point can be found by differentiating the equation with respect to ‘n’. The general case is:

$$\frac{dF}{dn}\left(S\left(\mathrm{LOG}_{10}\left(1 - e^{-dn}\right) + 1\right)\right) =$$

$$\frac{dS}{\left(e^{dn - 1}\right)\left(\mathrm{LN}\ 10\right)}$$

This is best illustrated by plotting the derivative on the same scale:



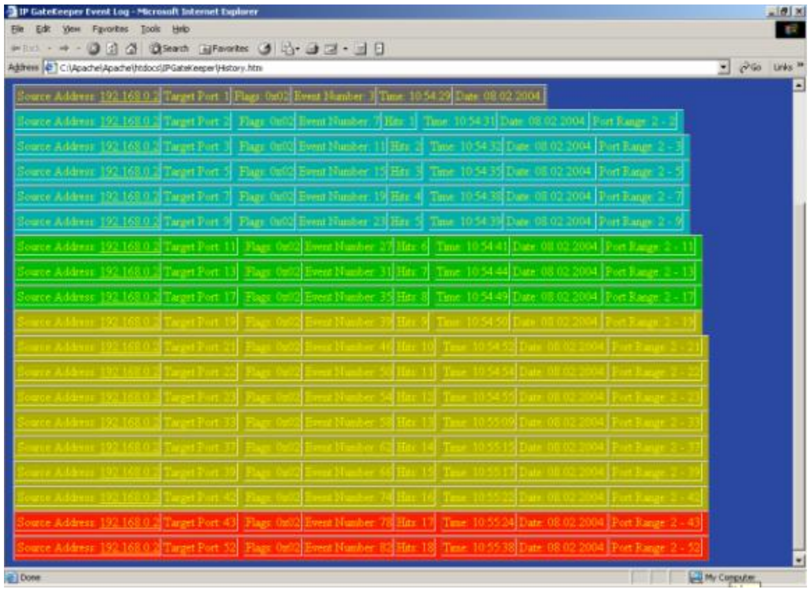
```
                    - d·n
#10: s·(LOG(1 - ê        , 10) +

     d                    - d·n
#11: ── (s·(LOG(1 - ê          , 1
     dn

              d·s
     ─────────────────────
#12:     d·n
      (ê      - 1)·LN(10)

                       - 0.05·n
#13: 10·(LOG(1 - ê             , 1

     d                       - 0.05·
#14: ── (10·(LOG(1 - ê
     dn

                   1
     ──────────────────────
#15:         n/20
       2·(ê       - 1)·LN(10)
```

```
COMMAND: Algebra Center Delete Help Move Options Plot Quit Range Scale Transfer
         Window aXes Zoom
Enter option
Cross x:0.9090      y:1          Scale x:10       y:2        Derive 2D-plot
```

Fig. 49: Derivative (gradient) demonstration

The graph shown uses an 'S' value of 10, in order to use a rating scale of 0 – 10, the 'd' value of 0.05 would appear a reasonable starting value for testing. It is important to realise that Derive automatically simplifies equations, expressing fractional values in their lowest terms.

The performance of the above equation was tested simply by using a TCP connect scan. The first connection attempt is simply recorded as a 'single' connection attempt since no record of previous connections from host 192.168.0.2 is present. [59] However, as demonstrated below, the second connection attempt meets the requirement for the 'multiple port detection criteria' – there is now a record of this same host connecting to more than one port of a target system.



IP GateKeeper Event Log - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Apache\Apache\htdocs\IPGateKeeper\History.htm
Done
My Computer

Source Address: 192.168.0.2	Target Port: 1	Flags: 0x02	Event Number: 3	Time: 10:54:29	Date: 08.02.2004		
Source Address: 192.168.0.2	Target Port: 2	Flags: 0x02	Event Number: 7	Hits: 1	Time: 10:54:31	Date: 08.02.2004	Port Range: 2 - 2
Source Address: 192.168.0.2	Target Port: 3	Flags: 0x02	Event Number: 11	Hits: 2	Time: 10:54:32	Date: 08.02.2004	Port Range: 2 - 3
Source Address: 192.168.0.2	Target Port: 5	Flags: 0x02	Event Number: 15	Hits: 3	Time: 10:54:35	Date: 08.02.2004	Port Range: 2 - 5
Source Address: 192.168.0.2	Target Port: 7	Flags: 0x02	Event Number: 19	Hits: 4	Time: 10:54:38	Date: 08.02.2004	Port Range: 2 - 7
Source Address: 192.168.0.2	Target Port: 9	Flags: 0x02	Event Number: 23	Hits: 5	Time: 10:54:39	Date: 08.02.2004	Port Range: 2 - 9
Source Address: 192.168.0.2	Target Port: 11	Flags: 0x02	Event Number: 27	Hits: 6	Time: 10:54:41	Date: 08.02.2004	Port Range: 2 - 11
Source Address: 192.168.0.2	Target Port: 13	Flags: 0x02	Event Number: 31	Hits: 7	Time: 10:54:44	Date: 08.02.2004	Port Range: 2 - 13
Source Address: 192.168.0.2	Target Port: 17	Flags: 0x02	Event Number: 35	Hits: 8	Time: 10:54:49	Date: 08.02.2004	Port Range: 2 - 17
Source Address: 192.168.0.2	Target Port: 19	Flags: 0x02	Event Number: 39	Hits: 9	Time: 10:54:50	Date: 08.02.2004	Port Range: 2 - 19
Source Address: 192.168.0.2	Target Port: 21	Flags: 0x02	Event Number: 46	Hits: 10	Time: 10:54:52	Date: 08.02.2004	Port Range: 2 - 21
Source Address: 192.168.0.2	Target Port: 22	Flags: 0x02	Event Number: 50	Hits: 11	Time: 10:54:54	Date: 08.02.2004	Port Range: 2 - 22
Source Address: 192.168.0.2	Target Port: 23	Flags: 0x02	Event Number: 54	Hits: 12	Time: 10:54:55	Date: 08.02.2004	Port Range: 2 - 23
Source Address: 192.168.0.2	Target Port: 33	Flags: 0x02	Event Number: 58	Hits: 13	Time: 10:55:09	Date: 08.02.2004	Port Range: 2 - 33
Source Address: 192.168.0.2	Target Port: 37	Flags: 0x02	Event Number: 62	Hits: 14	Time: 10:55:15	Date: 08.02.2004	Port Range: 2 - 37
Source Address: 192.168.0.2	Target Port: 39	Flags: 0x02	Event Number: 66	Hits: 15	Time: 10:55:17	Date: 08.02.2004	Port Range: 2 - 39
Source Address: 192.168.0.2	Target Port: 42	Flags: 0x02	Event Number: 74	Hits: 16	Time: 10:55:22	Date: 08.02.2004	Port Range: 2 - 42
Source Address: 192.168.0.2	Target Port: 43	Flags: 0x02	Event Number: 78	Hits: 17	Time: 10:55:24	Date: 08.02.2004	Port Range: 2 - 43
Source Address: 192.168.0.2	Target Port: 52	Flags: 0x02	Event Number: 82	Hits: 18	Time: 10:55:38	Date: 08.02.2004	Port Range: 2 - 52

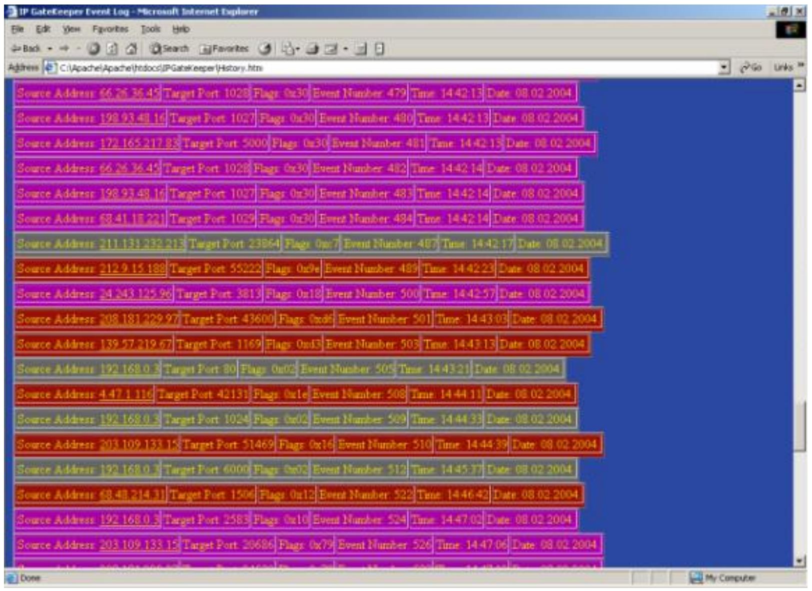
Fig. 50: Example Output for a TCP connect scan

As this may not in itself be suspicious (it is not unusual for a single host to follow a URL from a web page to an FTP site, for example), the 'probability' factor is still rated as low (shown in blue). However, further

connection attempts escalate the probability.
As no conditions are placed on the exact destination address (the ports probed can belong to one or more destination addresses), the procedure is equally valid for a single machine, or across the full range of machines accessible through the gateway. [60]
This use of multiple (ramped) thresholds is less simplistic than the use of a single threshold, and may reduce the changes of false positives, for example if an alert were only generated once the high/very high probability category was reached (which in itself is unlikely to be accidental).
Incidentally, any connection attempts, which appear from other source addresses whilst 192.168.0.2 continues the port scan, will not be included in the count. They will show up separately as 'single connection attempts' [61] colour coded in gray. The key reason to record this 'apparently normal' network activity is not so much to facilitate testing as to detect the instances of probes which involve a single destination port (and which may be replicated across large numbers of machines). Also, this ensures the recording of events, which are spaced too far apart in time for the IDS to 'remember' previous connection attempts by the same host. This is in contrast to the 'multiple port' probing which the 'threshold' based routine detects. [62]
Not all reconnaissance activity uses fast scanning techniques such as those employed by 'Mscan', which scanned at a rate of 10 packets to separate ports per second (and which IPGateKeeper would detect easily!).
Northcutt, (2003) states *"Scan detection boils down to detecting X events of interest across a Y sized time window".*
However, since it is always possible to defeat the longest time window, it is my view that it is best to:

1) Record the IP address of any host which has recently attempted to connect to more than a single port;
2) Compare this with any future addresses doing likewise;
3) Alert where the total unique connection attempts exceed a suitable threshold;
4) It is unavoidable that, at some stage, the process needs to be reset. This could take place after, for example, 1000 occurrences of event 1, *or* after 24 hours.

The obvious outcome of the above procedure is to resize the time window to fit the scanning rate. Since it is necessary to reset the window periodically (for example, after 20 minutes); it is still possible to remain undetected, however to scan even a single subnet at such a rate would take months!
Since not all scans rely on actual connection attempts, the count should not be restricted to TCP Syn packets alone.
The output resulting from the scanning activity (above) can be contrasted with that of normal traffic (Fig. 51).



```
Source Address: 66.26.36.45 Target Port: 1028 Flags: 0x30 Event Number: 479 Time: 14:42:13 Date: 08.02.2004
Source Address: 198.93.48.16 Target Port: 1027 Flags: 0x30 Event Number: 480 Time: 14:42:13 Date: 08.02.2004
Source Address: 172.165.217.83 Target Port: 5000 Flags: 0x30 Event Number: 481 Time: 14:42:13 Date: 08.02.2004
Source Address: 66.26.36.45 Target Port: 1028 Flags: 0x30 Event Number: 482 Time: 14:42:14 Date: 08.02.2004
Source Address: 198.93.48.16 Target Port: 1027 Flags: 0x30 Event Number: 483 Time: 14:42:14 Date: 08.02.2004
Source Address: 68.41.18.221 Target Port: 1029 Flags: 0x30 Event Number: 484 Time: 14:42:14 Date: 08.02.2004
Source Address: 211.131.232.213 Target Port: 23864 Flags: 0xc7 Event Number: 487 Time: 14:42:17 Date: 08.02.2004
Source Address: 212.9.15.188 Target Port: 55222 Flags: 0x9e Event Number: 489 Time: 14:42:23 Date: 08.02.2004
Source Address: 24.243.125.96 Target Port: 3813 Flags: 0x18 Event Number: 500 Time: 14:42:57 Date: 08.02.2004
Source Address: 208.181.229.97 Target Port: 43600 Flags: 0xd6 Event Number: 501 Time: 14:43:03 Date: 08.02.2004
Source Address: 139.57.219.67 Target Port: 1169 Flags: 0xd3 Event Number: 503 Time: 14:43:13 Date: 08.02.2004
Source Address: 192.168.0.3 Target Port: 80 Flags: 0x02 Event Number: 505 Time: 14:43:21 Date: 08.02.2004
Source Address: 4.47.1.116 Target Port: 42131 Flags: 0x1e Event Number: 508 Time: 14:44:11 Date: 08.02.2004
Source Address: 192.168.0.3 Target Port: 1024 Flags: 0x02 Event Number: 509 Time: 14:44:33 Date: 08.02.2004
Source Address: 203.109.133.15 Target Port: 51469 Flags: 0x16 Event Number: 510 Time: 14:44:39 Date: 08.02.2004
Source Address: 192.168.0.3 Target Port: 6000 Flags: 0x02 Event Number: 512 Time: 14:45:37 Date: 08.02.2004
Source Address: 68.48.214.31 Target Port: 1506 Flags: 0x12 Event Number: 522 Time: 14:46:42 Date: 08.02.2004
Source Address: 192.168.0.3 Target Port: 2583 Flags: 0x10 Event Number: 524 Time: 14:47:02 Date: 08.02.2004
Source Address: 203.109.133.15 Target Port: 20686 Flags: 0x79 Event Number: 526 Time: 14:47:06 Date: 08.02.2004
```

IP GateKeeper Event Log - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Apache\Apache\htdocs\IPGateKeeper\History.htm
Go Links
Done
My Computer

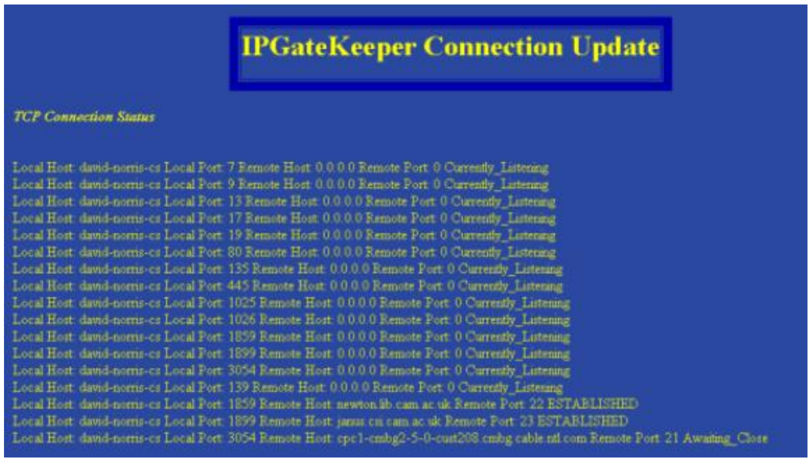
Fig. 51:Appearance of 'Normal' network activity, for a generalised configuration

X. Local Connection Status

Although the IDS is designed to run on a gateway, [63] it is useful to maintain a record of connections established to/from the gateway itself. This is useful for testing purposes, and because the gateway itself can be compromised. (The gateway is the only system visible externally where NAT is in use). This allows a 'profile' of actual connections to be maintained. Although the 'Netstat' command can be used on the gateway itself to list all contemporary connections, there is no means by which this can be accomplished automatically.

For this reason, I took a decision to add this capability to IPGateKeeper. The code for this function was based partially on an open source program, Netstatp by Mark Russinovich, (2003). The derived endpoint monitoring code is dependent on the dynamic link library, 'Inetmib1.dll', which must be present on the search path.

This function is called on startup, and thereafter when an incoming connection is made to the gateway or standalone system itself. [64] This generates a connection history (Fig. 52), in the file 'status.htm'.



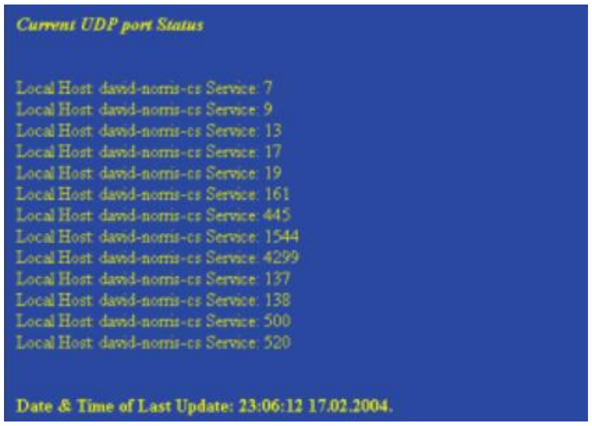
IPGateKeeper Connection Update

TCP Connection Status

```
Local Host: david-norris-cs Local Port: 7 Remote Host: 0.0.0.0 Remote Port: 0 Currently_Listening
Local Host: david-norris-cs Local Port: 9 Remote Host: 0.0.0.0 Remote Port: 0 Currently_Listening
Local Host: david-norris-cs Local Port: 13 Remote Host: 0.0.0.0 Remote Port: 0 Currently_Listening
Local Host: david-norris-cs Local Port: 17 Remote Host: 0.0.0.0 Remote Port: 0 Currently_Listening
Local Host: david-norris-cs Local Port: 19 Remote Host: 0.0.0.0 Remote Port: 0 Currently_Listening
Local Host: david-norris-cs Local Port: 80 Remote Host: 0.0.0.0 Remote Port: 0 Currently_Listening
Local Host: david-norris-cs Local Port: 135 Remote Host: 0.0.0.0 Remote Port: 0 Currently_Listening
Local Host: david-norris-cs Local Port: 445 Remote Host: 0.0.0.0 Remote Port: 0 Currently_Listening
Local Host: david-norris-cs Local Port: 1025 Remote Host: 0.0.0.0 Remote Port: 0 Currently_Listening
Local Host: david-norris-cs Local Port: 1026 Remote Host: 0.0.0.0 Remote Port: 0 Currently_Listening
Local Host: david-norris-cs Local Port: 1859 Remote Host: 0.0.0.0 Remote Port: 0 Currently_Listening
Local Host: david-norris-cs Local Port: 1899 Remote Host: 0.0.0.0 Remote Port: 0 Currently_Listening
Local Host: david-norris-cs Local Port: 3054 Remote Host: 0.0.0.0 Remote Port: 0 Currently_Listening
Local Host: david-norris-cs Local Port: 139 Remote Host: 0.0.0.0 Remote Port: 0 Currently_Listening
Local Host: david-norris-cs Local Port: 1859 Remote Host: newton.lib.cam.ac.uk Remote Port: 22 ESTABLISHED
Local Host: david-norris-cs Local Port: 1899 Remote Host: janus.csi.cam.ac.uk Remote Port: 23 ESTABLISHED
Local Host: david-norris-cs Local Port: 3054 Remote Host: cpc1-cmbg2-5-0-cust208.cmbg.cable.ntl.com Remote Port: 21 Awaiting_Close
```

Fig. 52 TCP Connection Status Record

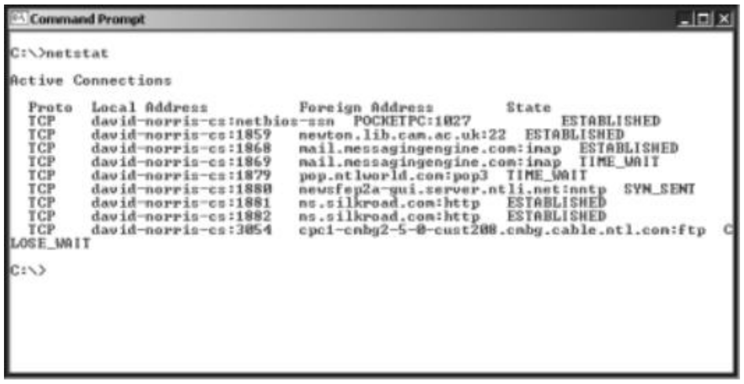
Conversion between IP addresses and names can be carried out using Gethostbyname() and Gethostbyaddr() in C.



```
Current UDP port Status

Local Host: david-norris-cs Service: 7
Local Host: david-norris-cs Service: 9
Local Host: david-norris-cs Service: 13
Local Host: david-norris-cs Service: 17
Local Host: david-norris-cs Service: 19
Local Host: david-norris-cs Service: 161
Local Host: david-norris-cs Service: 445
Local Host: david-norris-cs Service: 1544
Local Host: david-norris-cs Service: 4299
Local Host: david-norris-cs Service: 137
Local Host: david-norris-cs Service: 138
Local Host: david-norris-cs Service: 500
Local Host: david-norris-cs Service: 520

Date & Time of Last Update: 23:06:12 17.02.2004.
```

Fig. 53 Listening UDP Services

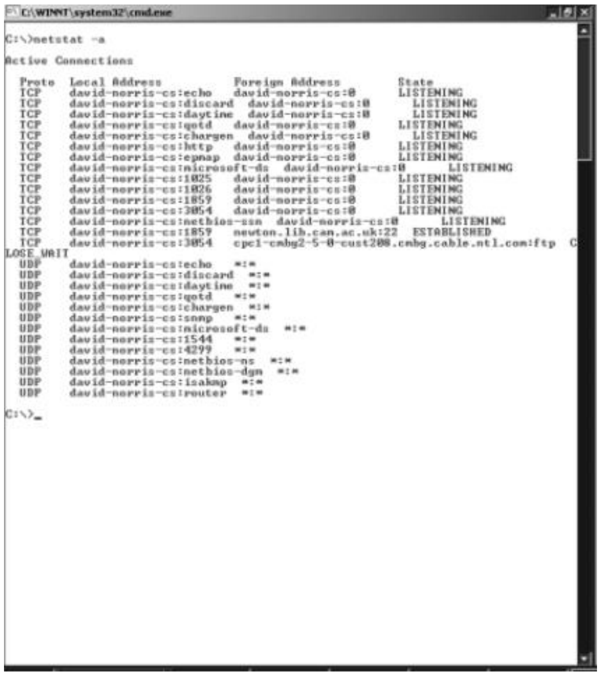
This procedure cannot be carried out for UDP connections, as UDP is 'connectionless'. It is however possible to maintain a list of listening ports, which would, for example, give away a 'Trojan horse' utilizing UDP for transport. The accuracy of the reports can be tested using the 'Netstat' command (at the time of update!).



```
C:\>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    david-norris-cs:netbios-ssn  POCKETPC:1027          ESTABLISHED
  TCP    david-norris-cs:1859   newton.lib.cam.ac.uk:22  ESTABLISHED
  TCP    david-norris-cs:1868   mail.messagingengine.com:imap  ESTABLISHED
  TCP    david-norris-cs:1869   mail.messagingengine.com:imap  TIME_WAIT
  TCP    david-norris-cs:1879   pop.ntlworld.com:pop3  TIME_WAIT
  TCP    david-norris-cs:1880   newsfep2a-gui.server.ntli.net:nntp  SYN_SENT
  TCP    david-norris-cs:1881   ns.silkroad.com:http   ESTABLISHED
  TCP    david-norris-cs:1882   ns.silkroad.com:http   ESTABLISHED
  TCP    david-norris-cs:3054   cpc1-cmbg2-5-0-cust208.cmbg.cable.ntl.com:ftp  C
LOSE_WAIT

C:\>
```

Fig. 54 Comparison with MS Windows 'Netstat' command

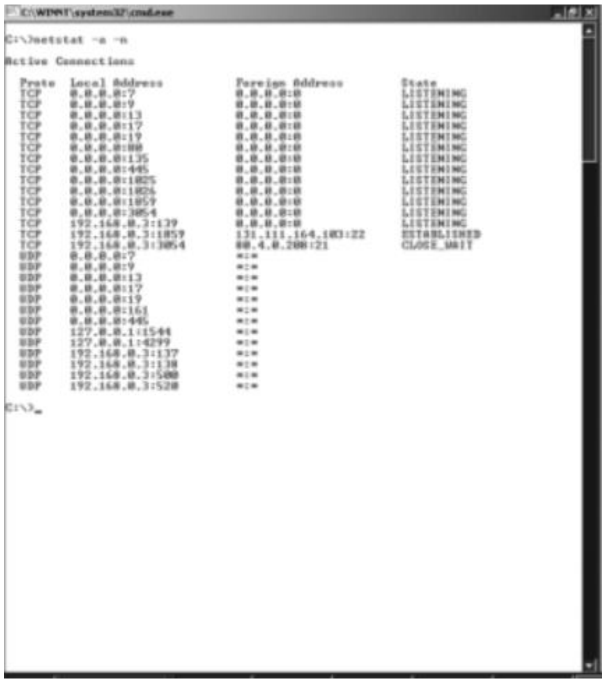
The list in fig. 54 would appear to confirm accuracy of reporting. A list of listening TCP and UDP ports can be obtained using the '-a switch (Fig. 55).



```
C:\>netstat -a

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    david-norris-cs:echo    david-norris-cs:0      LISTENING
  TCP    david-norris-cs:discard  david-norris-cs:0      LISTENING
  TCP    david-norris-cs:daytime  david-norris-cs:0      LISTENING
  TCP    david-norris-cs:qotd    david-norris-cs:0      LISTENING
  TCP    david-norris-cs:chargen  david-norris-cs:0      LISTENING
  TCP    david-norris-cs:http    david-norris-cs:0      LISTENING
  TCP    david-norris-cs:epmap   david-norris-cs:0      LISTENING
  TCP    david-norris-cs:microsoft-ds  david-norris-cs:0      LISTENING
  TCP    david-norris-cs:1025    david-norris-cs:0      LISTENING
  TCP    david-norris-cs:1026    david-norris-cs:0      LISTENING
  TCP    david-norris-cs:1059    david-norris-cs:0      LISTENING
  TCP    david-norris-cs:3054    david-norris-cs:0      LISTENING
  TCP    david-norris-cs:netbios-ssn  david-norris-cs:0      LISTENING
  TCP    david-norris-cs:1059    newton.lib.cam.ac.uk:22  ESTABLISHED
  TCP    david-norris-cs:3054    cpc1-cmbg2-5-0-cust208.cmbg.cable.ntl.com:ftp  C
LOSE_WAIT
  UDP    david-norris-cs:echo    *:*
  UDP    david-norris-cs:discard  *:*
  UDP    david-norris-cs:daytime  *:*
  UDP    david-norris-cs:qotd    *:*
  UDP    david-norris-cs:chargen  *:*
  UDP    david-norris-cs:snmp    *:*
  UDP    david-norris-cs:microsoft-ds  *:*
  UDP    david-norris-cs:1544    *:*
  UDP    david-norris-cs:4299    *:*
  UDP    david-norris-cs:netbios-ns  *:*
  UDP    david-norris-cs:netbios-dgm  *:*
  UDP    david-norris-cs:isakmp  *:*
  UDP    david-norris-cs:router  *:*

C:\>_
```

Fig. 55: 'Netstat –a' Shows all listening services by name where recognised, and their status

The 'state' column refers to the TCP connection status (se Fig. 56).



```
C:\>netstat -a -n

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:7              0.0.0.0:0              LISTENING
  TCP    0.0.0.0:9              0.0.0.0:0              LISTENING
  TCP    0.0.0.0:13             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:17             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:19             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1026           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1059           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3054           0.0.0.0:0              LISTENING
  TCP    192.168.0.3:139        0.0.0.0:0              LISTENING
  TCP    192.168.0.3:1059       131.111.164.103:22     ESTABLISHED
  TCP    192.168.0.3:3054       80.4.0.208:21          CLOSE_WAIT
  UDP    0.0.0.0:7              *:*
  UDP    0.0.0.0:9              *:*
  UDP    0.0.0.0:13             *:*
  UDP    0.0.0.0:17             *:*
  UDP    0.0.0.0:19             *:*
  UDP    0.0.0.0:161            *:*
  UDP    0.0.0.0:445            *:*
  UDP    127.0.0.1:1544         *:*
  UDP    127.0.0.1:4299         *:*
  UDP    192.168.0.3:137        *:*
  UDP    192.168.0.3:138        *:*
  UDP    192.168.0.3:500        *:*
  UDP    192.168.0.3:520        *:*

C:\>_
```

Fig. 56: 'Netstat –a –n' Shows all listening services, and current TCP connection status

A numerical listing can be obtained using the '-n' switch. Local TCP ports currently awaiting a connection are shown with a foreign address of 0.0.0.0.

XI. Testing & Evaluation

The performance and usability of any software application can only be fully assessed "in the field". This is largely due to the fact that, at a 'pre-implementation' stage, you have yet to have a sufficiently complete application for testing to commence. Shneiderman, (1987), states *"At this early post-implementation stage the only factor that can confidently be assessed is the degree of subjective satisfaction among users".*

Preece (1993), makes a distinction between formative evaluation, which is primarily concerned with influencing the design of the product during the development stage, and summative evaluation, concerned with testing the actual functionality of the system, following it's implementation. Unfortunately, the test procedure is fraught with difficulty. There is no standard means by which to carry out testing, except to demonstrate responses to known attacks. "While intrusion detection systems are becoming ubiquitous defenses in today's networks, currently we have no comprehensive and scientifically rigorous methodology to test the effectiveness of these systems." Mell, (N.d). There is no practical means by which to carry out performance testing with regard to future threats. This is contraversial. Real environments offer a realistic traffic scenario, without the requirement for any user activity analysis. [65]

However, since this environment is likely to contain some intrusive data, therefore there is little opportunity to sample 'exclusively normal' network activity.

Simulated environments offer freedom from such contamination; however it is difficult to accurately simulate the 'real world'. This takes a great deal of estimation regarding the statistical distribution of network traffic generated by thousands of users. For example, it has been suggested by some researchers that the characteristics of network traffic have a Gaussian distribution; others disagree. The real difficulty is that little simulation has been attempted in the analysis of IDS performance.

For this reason, I decided to carry out testing in both 'real' and 'simulated' environments.

The 'real' environment consisted of traffic actually obtained from the Internet, by placing the computer running the software in the DMZ for one week. The simulated test involved isolating this computer from the Internet, and simulating an environment within the private network alone. As the number and distribution of 'events' in the two scenarios are invariably different, however, the two methods are mutually independent. [66] For this reason, the results cannot be compared and contrasted.

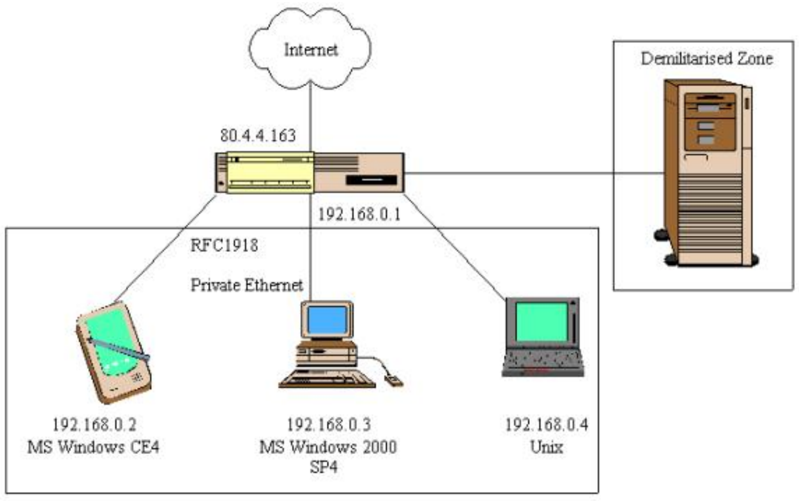


Fig. 57: Home Network Topology

These are the bandwidth performance results, in terms of packets dropped as a percentage of the total throughput:

Bandwidth	IPGateKeeper	Snort(Outside X-	Snort(Under X-

		Windows)	Windows
Quiescent	0%	0%	3%
500kbps, Omni-Dir	11.5%	4%	13.2%
10Mbps, Bi-Dir	21%	12%	37%

Clearly, Snort has lower packet loss rates when run alone on the Unix system, giving better performance than IPGateKeeper. It has to be bourne in mind, however, that IPGateKeeper is at a considerable disadvantage, since Windows requires that Windows be running (!) whereas all Unix operating systems can run fully without X-Windows. Snort exhibits far worse performance when run in a Konsole Window under X-Windows.

It is also essential to consider the speed of the system CPU when making such comparisons. The Windows 2000 system and Unix system have CPU clock speeds of 500 and 133MHz respectively; both systems have 64MB of RAM installed. Therefore, the test results are somewhat arbitary; they do illustrate the difficulty of making a scientific observation without ‘all circumstances being equal’.[67]

It is, however, worth noting that the test results shown are ‘worst possible case’, since in both cases the 500kbps Cable Modem and 10Mbps Internal LAN were running at 100% utilization. In practice, most networks, including Ethernet, will tend to saturate at approximately 85-90% of the maximum throughput supported by the hardware, as pointed out by Stallings, (1999). This saturation point does vary between networks of various specification, such as Ethernet, (IEEE 802.3), and Token Ring, (IEEE 802.5). For the vast majority of the time, networks run at a small percentage of saturation.

For signature detection, (which involves the analysis of network packets as far up the OSI model as the presentation layer), the performance is expressed as the percentage of signatures detected, of the total sent across the network using the Echo protocol[68]:

Quiescent	500kbps, Omni-Directional	10Mbps, Bi-Directional
100%	76%	64%

The conclusion to be drawn from the above, is that, at low levels of utilization, signature detection is very reliable. However, at saturation levels, the CPU loading becomes sufficient to cause the IDS to drop packets.

XII. Conclusion

Although I am slightly disappointed with the Packet losses encountered under high traffic loading, I am satisfied with the overall performance of IPGateKeeper. One of the earliest issues which I spent a great deal of time attempting to resolve, was associated with ‘real-time’ operation. Except under low utilization conditions, the packet handling time exceeds δT. Without utilizing multithreaded operation, the percentage of packets successfully processed averages to 1/(Process time / δT). This implies that, where, for example, the processing time is equal to 12δT, the percentage of packets successfully handled will be equal to:

1/(12δT/δT) Packets = 8.333% of all packets. This is not a good performance!

Multithreaded operation dramatically improves the performance of the IDS. This proved problematic; I eventually decided to utilise the Winposix[69] regular expression library to assist in the processing of packet data where required as per the configuration.

A later problem concerned the monitoring of TCP connection states, in order to detect many types of 'stateful' attack. This requires that an accurate record of connection states be retained. The difficulty proved very difficult to overcome, simply by virtue of the time domains and storage requirements in question. Disk storage is limited only by the available disk space, as far as capacity is concerned. However, the use of disk storage effectively equates to the use of virtual memory, which offers access times several orders of magnitude slower than RAM storage. Furthermore, as IPGateKeeper has to compete with any other applications running for use of disk access, and with the Windows' Swap file, access times can become even more problematic!

RAM has the advantage of being accessible in near real-time; however, it has limited storage capability. Although I considered the use of dynamic memory allocation to conserve memory, it became clear that the overheads involved were too severe, both in terms of time and CPU overhead.[70] Arrays represent a relatively inefficient use of RAM, since not all elements declared may be utilised; however, in this application I faced the opposite problem – any array becomes full! Therefore, however large an array I declared for the purpose of storing such data as TCP connection states, it would become full, and I had the choice of either re-using the array elements cyclically, (and thereby discarding data, so that, for example the 'memory' of a TCP Syn Packet could be lost, making the further analysis of the connection inconclusive), or, by not reusing the available capacity, to allow the available memory to become full, rendering the IDS ineffective after 100% array usage! In the first instance, a slow scan would defeat the IDS; in the latter, a flooding attack would quickly have the same outcome. Using array storage for 'short-term' events, and disk storage in the longer term appeared only a partial solution, since retrieving data from large files (of the order of 100MB+ after several days) is very resource intensive.

Eventually I decoded to opt for a statistical method in lieu of the 'stateful' method. This involved counting the number of connection attempts and responses (Ack/Syn or Ack/Rst in the case of TCP), [71] detected in a given 'time window' and alerting when a threshold was exceeded. This process was intentionally made 'non address specific', since the use of decoy addresses (which many reconnaissance tools use as a matter of course!), would render the method ineffective.

As discussed earlier, I decided to provide summaries of activity in a readable, colour coded format, with an 'event specific' record to provide 'supporting evidence'. Generally, irregularities within individual packets are reported in the 'packet specific' reports as per the configuration; whereas events involving sequences in packet arrival are provided in the 'History Report'. The reasoning behind this reasoning is documented in the logbook.

To sumarise, I believe that, in spite of the shortcomings of software based IDS, they have a useful role in today's increasingly mobile and decentralised computing environment. However, wherever feasible, I would conclude that a hardware based IDS, with updateable firmware, is preferable. In such a system, the hardware is dedicated to the IDS process alone. Nowadays, when non-volatile RAM capacity (Such as 'flash' memory) is available in capacities measured in gigabytes, I believe that this combination of mobile software IDS, and fixed hardware IDS, is the way ahead.

Glossary

δT: In this context, the average time duration between packet arrivals. This varies with both bandwidth and average packet size, within a given time window.

ACK: Acknowledgement (TCP Flag). This is set as an acknowledgement of an incoming TCP connection. All TCP packets, except for the initial 'SYN' packet, which initiates a connection, have this flag set.

AND: Boolean 'AND'. This condition is true, only when all input conditions are true. In 'C', a zero value evaluates to true, whereas all non-zero values evaluate to true.

Anomaly: An event or sequence of events, which do not occur under normal operating conditions. For example, the occurrence of connection attempts by a single host to multiple ports, TCP packets with nonsensical flag settings, or multiple incomplete TCP connections between hosts, are anomalous events.

ARP: Automatic Routing Protocol. This is used to match IP addresses to MAC addresses.

b: Suffix indicates binary number. Binary numbers have a base of 2; therefore all bits can take either a value of 0 (false), or 1 (true).

Bind: The Berkeley Internet Name Deamon.

Buffer Overflow: Reading of data into a memory space (such as an array), which is too large to fit into the memory space. This is generally caused by a program failing to test whether the data will fit into the memory allocated in advance. This problem is common in programs written in 'C', since the language does not perform a bounds check on any variable access. However, this problem is less common in programs written in other languages, such as Modula2 and Java.

BUG: An unintentional property of a software component, which may cause it to fail under certain conditions. For example, earlier versions of MS Windows, and some versions of Unix, would attempt to assemble a fragmented TCP packet, the total size of which was above the legal maximum of 65535 bytes in size. This caused a buffer overflow to occur.

CGI: Common Gateway Interface. Scripts providing an interface between a web interface, and internal databases.

Command: Single Operating Instruction. For IPGateKeeper, *commands must be separated by spaces, but must not contain spaces.*

CWR: Congestion Window Reduced (TCP Flag): This facility was added to TCP in order to reduce the number of packets dropped due to congestion. This is set in response to the remote system setting the 'ECN Echo' flag.

d: Suffix indicates decimal number. This is an 'ordinary' number to base 10. The suffix avoids any possible confusion with hexadecimal numbers (base 16).

DOS: Denial of Service attack: Attacks which aim to prevent operation of a network, either by saturating its bandwidth, causing legitimate packets to be dropped, or by causing a break in connectivity.

DDOS: Distributed Denial of Service attack. This involves the launching of a denial of service attack from multiple compromised systems. These systems may be found manually, or more commonly, by automatic means. Home users, who access the Internet via 'broadband' Internet services, are particularly vulnerable to this type of exploitation. DDOS attacks are often known as 'Man in the Middle' attacks.

DMZ: Demilitarised Zone. A system in a DMZ (on which an IDS would be run), would not (generally) be protected by a firewall or network address translation.

DNS: Domain Name Service. This is the standard service which returns the IP address associated with a particular host name. The Domain Name Service resides at the session layer of the OSI model.

DOS: Denial of Service. An attempt to exploit shortcomings in the design of a software component, in order to provoke failure.

ECN: ECN Echo (TCP Flag): Set in order to request a slowing in the rate of data transmission by a remote system. The remote system will set the CWR flag indicating acknowledgement of congestion.

FIN: Final Acknowledgement (TCP Flag): This is set in order to request the closure of a connection. This cannot be legally set without the 'ACK' flag.

Flag: A single bit, which serves as an indicator. Usually, a whole byte is set aside providing 8 flags. In TCP, the flag settings indicate the state of a connection.

h: Suffix indicates hexadecimal number, base 16. This suffix is provided in order to avoid possible confusion with decimal values, for example 10h = 16d.

HCI: Human/Computer Interaction: The Science of Human/Computer interaction is central to the 'ease of usage' philosophy. Systems should be designed with the end user in mind.

HEX: Hexadecimal: Number having a base of 16. This is useful as each digit (hexadecimal place) is equivalent to four binary bits. For example, 3Fh = 00111111b.

HTTP: Hypertext Transfer Protocol. The protocol employed by 'web' servers to supply pages written in HTML (Hypertext Markup Language). The Hypertext Transfer Protocol resides at the session layer of the OSI model.

The specifications are defined in RFC's 2616/2617.

HTTPS: As HTTP, only HTTPS encrypts data before sending it across the network. For this reason, it is used in secure e-commerce and banking transactions. The Secure Hypertext Transfer Protocol resides at the session layer of the OSI model.

ICMP: Internet Control Message Protocol: This is used for the testing of routing and reachability. ICMP is protocol 1 when used over IP. The Internet Message Control Protocol resides at the transport layer of the OSI model. The specifications are defined in RFC792.

Ident: Identification Protocol. This provides a means by which a server can obtain some information regarding the 'owner' of a TCP connection, on a multi-user system. The Ident Protocol resides at the session layer of the OSI model. The specifications are defined in RFC1413.

IDS: Intrusion Detection System: A system, which 'generally' monitors network traffic passively, recording any suspicious events which deviate in some way from the expected behavior of the network. This is as opposed to a firewall, which actively blocks 'abnormal' packets.

Illegal: In this context, this effectively means 'Out of specification'. This is by virtue of non-compliance with meaningful values or settings, as defined in relevant documentation.

IMAP: Internet Message Access Protocol. Retrieves incoming electronic mail; similar to the post office protocol, but has extra functionality. The Internet Message Access Protocol resides at the session layer of the OSI model.

The specifications are defined in RFC2060.

IP: Internet Protocol: The standard, ubiquitous, 'language' of the Internet. This operates at the network layer of the OSI model. The IP packet 'wraps' packets of the transport layer protocols, such as TCP, UDP and ICMP. IP is so widely used that even dedicated network operating systems, such as Novell Netware, which have their own built in protocols (IPX for Netware), support IP in addition. The Internet Protocol resides at the Network layer of the OSI model. The specifications are defined in RFC760.

IPV4: Internet Protocol Version 4. This is the current (at time of writing) version of the Internet protocol. The main limitation of IPV4 lies in it's 32 bit addressing scheme, which supports 'only' 2^32 (4294967296d), (100000000h), addresses, which are fast running out.

IPV6: Internet Protocol Version 6: This updated version of the Internet Protocol has a number of refinements, including support for 2^128 (> 3.4 X 10^38d) addresses!

Kerberos: A secure authentication protocol utilizing secret-key cryptography to identify client-server identities, across an insecure connection.

MAC: Media Access Control: Each network interface has a unique hexadecimal number, worldwide, known, as it's MAC address. The allocation of MAC addresses is regulated worldwide.

Malware: A general term for malicious software. This term covers 'viruses', 'Trojan Horses', 'Worms', and such like. In general, software which is intended to cause loss, damage or disruption, whether or not the programs have the capability to 'self replicate'.

MTU: Maximum Transmission Unit. The maximum packet size supported by a network. For Ethernet, this size is 1500 bytes. Packets larger than this will be fragmented, and reassembled at the destination.

NAT: Network Address Translation. A scheme in which (usually) a single public IP address is 'translated' in such a way that hosts inside a private network (using IP addresses reserved for this purpose (see RFC1918)) are not directly connected to the Internet, but can access the Internet via the gateway. This has two benefits; firstly, it saves public IPV4 addresses, which are currently in short supply; secondly, hosts inside a private network are invisible externally, except via ports which are 'forwarded' through the gateway, or generally, where they are placed in a DMZ. Unfortunately, some services, for instance videoconferencing, cannot operate through NAT as there is no direct return channel to the internal host.

NFS: Network File System. On Unix, this allows remote Unix systems to 'mount' part of the local file system as though it were local. This appears as a subdirectory of the directory tree. The Network File System Protocol resides at the session layer of the OSI model.

The specifications are defined in RFC1094

NIA: Network Interface Adapter: The hardware, which interfaces a particular computer or other network capable device to the network itself.

NTP: Network Time Protocol. This provides an accurate time reference (typically of the order of 1-30 milliseconds), depending on the stratum level of the server, and the electrical distance from the client to the server. The Network Time Protocol resides at the session layer of the OSI model. The specifications are defined in RFC1305.

NNTP: Network News Transport Protocol. This is similar to those protocols utilised by electronic mail; however, the same protocol is employed by user-to-host and host-to-host exchanges. The Network News Transport Protocol resides at the session layer of the OSI model. The specifications are defined in RFC977.

OR: Boolean 'OR': This condition is true if at least one input condition is true. Which input(s) are true are unimportant. (This is actually the inclusive OR).

OSI: Open System Interconnection: This model illustrates the 'layered' network design, which effectively 'insulates' each layer from those below it. For example, the Transmission Control Protocol operates equally well over IP, or Novell IPX, and IP operates equally well over Ethernet or Token Ring based networks.

POP: The original electronic mail retrieval protocol. The currently used version is version 3, POP3. The specifications are defined in RFC1939.

Port: Effectively, a 'logical endpoint' for a connection. The majority of services listen on an assigned port number; if a service is running, then a port is said to be 'open' to connections.

Posix: Portable Operating System Interface. This is a standard adhered to by virtually all Unix (but not Microsoft) operating systems.

PSH: Push (TCP Flag): This flag is set in order to ensure that the packet is passed as quickly as possible to the application receiving it. This is always set in combination with ACK.

RFC: Request for Consultation. In despite their name, the specifications of well-known protocols and other parameters are defined in 'request for consultation' documents.

Rlogin: Remote Login. This provides a virtual terminal connection similar to Telnet, but which provides greater transport of terminal environment semantics than Telnet. Also, it can be configured not to require password authentication from trusted hosts. The Remote Login Protocol resides at the session layer of the OSI model. The specifications are defined in RFC1282.

RST: Reset (TCP Flag): The reset flag indicates that immediate closure of a TCP connection is required; for example, due to congestion.

Rule: A configuration of one or more commands defining event characteristics. IPGateKeeper expects rules not to span lines; i.e. each line in the configuration file is treated as a separate rule.

Signature: An identifiable, characteristic, string found in one or more network packets, which is related to a particular attack. For example, the following string:

"GET /scripts/..%255c%255c../winnt/system32/cmd.exe?/c+dir" attempts to exploit a vulnerability in the MS Windows NT scripting host, in order to obtain a directory listing via the Command Prompt.

SMTP: Simple Mail Transfer Protocol. The ubiquitous protocol for outgoing mail transfer. The Simple Mail Transfer Protocol resides at the session layer of the OSI model. The specifications are defined in RFC821.

SNMP: Simple Network Management Protocol. This provides a means to control and/or monitor large or complex networks. Although normally used to monitor network hardware, there are implementations for full-blown operating systems. Access to ‘agents’, as its servers are known, is controlled via ‘community strings’, which are, in effect, passwords. The Simple Network Management Protocol resides at the session layer of the OSI model. The specifications are defined in RFC1157.

SNTP: Simple Network Time Protocol. A minimal version of NTP, simpler to configure than NTP, but is suitable in applications not requiring the accuracy of NTP. The Simple Network Time Protocol resides at the session layer of the OSI model. The specifications are defined in RFC2030.

Spyware: A general term for software, which is designed to leak personal or other data from a computer or network, generally for commercial purposes. In the majority of cases, this takes place behind the scenes, without either the user’s knowledge or consent.

SSH: Secure Shell. Essentially, a secure version of the Telnet remote login protocol, which encrypts passwords and other data. Secure Shell resides at the session layer of the OSI model.

SSL: Secure Socket Layer. The industry standard for encrypting network traffic against interception. Netscape Communications Corporation developed SSL.

SYN: Synchronise (TCP Flag): This is used to initiate a TCP connection. The first packet sent has the SYN flag set, as does the first packet sent in reply (if the requested service is running on the remote system). Thereafter, the SYN flag is reset.

TCP: Transmission Control Protocol: this is protocol number 4 over IP. This provides a reliable connection, since packets are acknowledged upon receipt. The majority of Internet traffic utilises TCP. The Transport Control Protocol resides at the Transport layer of the OSI model. The specifications are defined in RFC793.

Telnet: The (almost) original remote login protocol. Allows the remote execution of commands on a given system; can be configured to perform other functions, ranging from information provision, to database searches. The Telnet Protocol resides at the session layer of the OSI model. The specifications are defined in RFC854.

TTL: Time To Live. This byte is decremented by one on passing through a router. Should the TTL reach 0, the packet is dropped, and the sending host informed of the delivery failure. This is to prevent ‘orphaned’ packets from being endlessly circulated around the Internet.

UDP: User Datagram Protocol: This provides a ‘connectionless’ means of transport, without a requirement for acknowledgement. This is less reliable than TCP; however UDP acquires little performance overhead. It is utilised by simpler network services, such as the domain name service. The User Datagram Protocol resides at the Transport layer of the OSI model. The specifications are defined in RFC768.

URG: Urgent (TCP Flag): This flag is set indicating that a packet has high priority. It is always used in conjunction with the ACK flag.

XOR: Exclusive OR. This condition is true where at least one, but not all, input conditions are true.

References

Those references described as a ‘White Paper’ are academic papers and journals, which have been located using a search of the databases available via the ‘Athens’ system: http://www.athens.ac.uk/ . I have not included the URL for these, since these are database generated and invalid outside of the Athens system.

Adkins, Major B.N, USAF (2001) [White Paper], The Spectrum of Cyber Conflict from Hacking to Information Warfare: What is law Enforcement’s role?

Axelsson, S, (2000), The base-rate fallacy and the difficulty of intrusion detection.

Bace, R.G, (2000), Intrusion detection, Indianapolis, Ind. Macmillan Technical

Bandy, P, Money, M. (2000), [WWW], Why is Intrusion Detection Required in Today’s Computing Environment?
http://www.sans.org/resources/idfaq/id_required.php

Barrus, J, Rowe, N.C. (1998), [WWW], A Distributed Autonomous-Agent Network-Intrusion Detection and Response System
http://www.cs.nps.navy.mil/people/faculty/rowe/barruspap.html

Beckers, J, Ballarini, J.P, (2003), [White Paper], Advanced Analysis of Intrusion Detection Logs.

Bierman, E, Cloete, E, Venter, L.M. (2001), [White Paper], A comparison of Intrusion Detection Systems

Boehm, BW, (1981), Software Engineering Economics, London, Prentice-Hall

Buzzard, K, (1999), [White Paper], Computer Security, What Should You Spend your Money On?

Cohen, F, (1997), [White Paper], 50 Ways to Defeat your IDS

Dautlich, M. (N.d), {White Paper], Penetration Testing – The Legal Implications

Didio, L. (1997), [White paper],Network Security Interest on Rebound

Douligeris, C, Mitrokotsa, A. (2003), [White Paper], DDoS attacks and defense mechanisms: classification and state-of-the-art

Edwards, S. (2002), [White Paper], Vulnerabilities of Network Intrusion Detection Systems: Realizing and Overcoming the Risks

Free Software Foundation, (1998), [WWW], GNU General Public License

Gunning, S, (2002) [WWW], Wireless London is Wide Open, http://news.bbc.co.uk/1/hi/sci/tech/1892510.stm

Hasii, B, Malababa, S, Pandey, R, Bishop, M, (2000), [White Paper], Supporting reconfigurable security policies for mobile programs

Helmer, G, Wong, J.S.K, Honovar, V, Millar, L, Wang, Y. (2002), [White Paper], Lightweight agents for intrusion detection

Hinde, S, (2003), [White Paper] Time Cost$ Money

Hunter, P. (N.d) [White Paper], Distributed Intrusion Detection Systems, (DIDS) can make Security More Adaptive

Giacinto, G, Roli, F, Didaci, L. (2003), [WWW] Fusion of multiple classifiers for intrusion detection in computer networks, Pattern Recognition Letters(24), University of Cagliari, Cagliari, Italy. www.elsevier.com/locate/patrec

Heberlein, L.T, Bishop, M. (N.d), [White Paper], Attack Class, Address 'Spoofing'

Huberman, B, Adamic, L, (1999), [CD-ROM], Growth Dynamics of the World-Wide Web

Mell, P, Hu, V, Lippmann, R, Haines, J, Zissman, Marc. (N.d), [White Paper], An Overview of Issues in Testing Intrusion Detection Systems

Newsham, T, Ptacek, T, (1998). [WWW], Insertation, Evasion & Denial of Service: Eluding Network Intrusion Detection www.robertgraham.com/mirror/Ptacek-Newsham-Evasion-98.html

Ning, P, Jajodia, S, Wang, X.S. (2001), [White Paper], Design & Implementation of a Decentralised Prototype System for Detecting distributed attacks

Northcutt, S., Novak, J., 2002. Network Intrusion Detection: an analyst's handbook, 3rd Edition, Indianapolis, Ind. New Riders Publishing

Northcutt, S, Cooper, M, Fearnow, M, Frederick, K. (2003), Intrusion Signatures & Analysis, 2nd Edition, Indianapolis, Ind, New Riders Publishing

Pichnrczyk, K, Weeber, S, Feingold, R. (1994), [White paper] Unix Incident Guide: How to Detect an Intrusion,

Preece, J, (1993), A Guide to Usability - Human Factors in Computing, Wokingham, Addison-Wesley.

Proctor, P.E, (2001), Practical intrusion detection handbook, London, Prentice Hall.

Resnick, M, (1997), Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds (Complex Adaptive Systems), [Whitepaper] Massachusetts Institute of Technology.

Russinovich, M. (2003), [WWW], 'Netstatp' Endpoint Monitoring Source. http://www.sysinternals.com/files/netstatp.zip

Securenode Inc (2003), [WWW], White Paper on Basics of Networking. http://www.securenode.com/resources/whitepapers/TCP_summary.pdf

Schumacher, H.J, Ghosh, S. (1997), [White Paper], A fundamental framework for network security

Schultz, E. (2003), [White Paper], Pandora's Box: spyware, 'adware', autoexecution, and NGSCB

Thompson, H.H, (2002), Intrusion Detection: Perspectives on the Insider Threat.

Shneiderman, B, (1987), Designing the User Interface, Wokingham, Addison-Wesley.

Sommerville, I, (2000), Software Engineering, Leicester, Addison Wesley.

Verwoerd, T, Hunt, R. (2001), Intrusion Detection Techniques & Approaches, University of Canterbury, Cristchurch, New Zealand.

Wood, (N.d), Why Information Security is now multi-disciplinary, multi-departmental, & Multi-Organizational in Nature.

Zona Research, (1998), Security Policies Inadequate

Bibliography

Alhir, SS, (1993), Learning UML, Sebastapol, CA, O Reilly

Bandy, P, Money, M. (2000), [WWW] What is a Honey pot? http://www.sans.org/resources/idfaq/honeypot2.php

Bandy, P, Money, M. (2000), [WWW] Should communication between the sensor & the monitor be encrypted? http://www.sans.org/resources/idfaq/communication.php

Beckers, J. & Ballerini, J.P, Technology Consultants, Internet Security Systems EMEA, (www.iss.net), N.d. Advanced Analysis of Intrusion Detection Logs

Biermann, E, Cloete, E, and Venter, L.M. (2001). [White paper] A comparison of Intrusion Detection systems

Butenhof, D,R. (1997), Programming with POSIX threads, Reading, MA. Addison-Wesley

Caasi, R. (2000), [WWW] What is a Honeypot and how is it used? www.sans.org/newlook/resources/idfaq/honeypot.htm

Cohen, F, (N.d), [White Paper] Novelty Detection

Criscoulo, P.J. (2000), [White Paper], Distributed Denial of Service Trin00, Tribe Flood Network, Tribe Flood Network 2000, CIAC-2319

Davis, R, (1998) Win32 Network Programming, Harlow, Addison-Wesley Developers Press

Dautlich, M. (2004) Penetration Testing – The Legal Implications

Deering, S, Hinden, R. (1998), [WWW], RFC 2460 - Internet Protocol, Version 6 (IPv6) Specification, http://www.faqs.org/rfcs/rfc2460.html

Deitel H.M, & Deitel P.J, (2000) C How to Program, Third Edition, London, Prentice Hall

Donahoo, M.J, & Calvert, K.L, (2001) TCP/IP Sockets in C

Edwards, S. (2002) Vulnerabilities of Network Intrusion Detection Systems: Realising and overcoming the Risks

Enterasys Networks (2003) [WWW] Intrusion Detection Methodologies Demystified,

http://www.enterasys.com/products/whitepapers/9013198.pdf

Enterprise Snort Installations , (2001), [White Paper] 'Snorting the Enterprise"

Eschelbeck, G, Kreiger, M. (2003), [White Paper], Elliminating Noise from Intrusion Detection Systems

Fraiser, B, (1997), Site Security Handbook, RFC 2196, http://www.ietf.org/rfc/rfc2196.txt

Frederick, K, (2004) [WWW], Network Intrusion Detection Signatures, Part One, 10 January 2004, http://www.securityfocus.com/infocus/1524

Friedl, J.E.F, (1997), Mastering Regular Expressions, Sebastapol, CA. , O' Reilly & Associates

The Honeynet Project, (2003), [WWW], Know Your Enemy: Sebek - A kernel based data capture tool.

http://www.honeynet.org

Information Assurance Technology Newsletter (1998), [Newsletter vol. 1, No. 3], Defending Against C2W & IW Attack

Iheagwara, C, Blyth, A, Kevin, T, Kinn, D. (2003), [White paper], Cost effective management frameworks: the impact of IDS deployment technique on threat mitigation

Information Sciences Institute, University of Southern California (1981), RFC 793, http://www.faqs.org/rfcs/rfc793.html

Information Sciences Institute, University of Southern California, (1981) [WWW], INTERNET PROTOCOL, http://www.faqs.org/rfcs/rfc791.html

Jackson, K.A, (1999), [White Paper] Intrusion Detection System Product Survey

Lemonnier, E. (2001), [White Paper], Guidelines for a Long Term Competitive Intrusion Detection System

Kamara, S, Schultz, E, Kerschbaum, F, Frantzen, M, (2002), [White Paper], Analysis of Vulnerabilities in Internet Firewalls

Kumar, S & Spalford, F.H, Purdue University (1996) [WWW]

A Software Architecture to Support Misuse & Intrusion Detection,

http://gunther.smeal.psu.edu/kumar95software.html

Lackey, Dr J, Roths, A, Goddard, J, (2003), [White Paper], Wireless Intrusion Detection

McHugh, J. (2000), State of the Practice of Intrusion Detection Technologies, Software Engineering Institute, Carnegie Mellon University

McKenner, B. (2002), [White Paper], Web services set to provoke new threats

Mondal, A. S. (2001), [White Paper] Porting Applications Across Unix & WIN32 Platforms

Nowak, E, Ingersoll, W. (1997), [White Paper], Tracking Down Trouble

NSS Group, (2002), [White paper] Intrusion Detection Systems Group Test, Third Edition

Ollmann, G. (N.d), [White Paper], Intrusion Prevention Systems (IPS) destined to replace legacy routers

Orvis, W.J, Smith, J, Krystosek, P. (N.d), [White Paper], Connecting to the Internet Securely; Protecting Home Networks CIAC-2324

Philip, H, N.d. Distributed Intrusion Detection Systems (DIDS) can make security more adaptive, Intrusion diagnosis in an uncontrolled environment.

Postel, J. (1983), Echo Protocol, http://www.faqs.org/rfcs/rfc862.html

Postel, J. (1981), Internet Message Control Protocol, http://www.faqs.org/rfcs/rfc792.html

Postel, J. (1983), Transmission Control Protocol, http://www.ietf.org/rfc/rfc793.txt

Postel, J. (1980), User Datagram Protocol, http://www.faqs.org/rfcs/rfc768.html

Rankin, J. (1993), Technical guide to POSIX : open systems technology transfer, Oxford, Blackwell Publishing

Ranhum, M.J, (2001), [WWW], Coverage in Intrusion Detection Systems, www.nfr.com

Recent advances in intrusion detection : 5th International Symposium, (2002), Zurich, Switzerland. Berlin/London Springer Publishing.

Schultz, E. Ph.D., CISSP, University of California/Berkeley Lab, N.d [WWW]
Demystifying Intrusion Detection: Sorting through the Confusion, Hyperbole and Misconceptions,
http://www.insight.co.uk/downloads/whitepapers/Effective%20Intrusion%20Detection%20(White%20Paper).pdf

Seo, H.S, Cho, T.H, (2003), [White Paper], An application of blackboard architecture
for the coordination among the security systems

Sharpe, A, Russell, C. (2004), [White Paper], Regulation of Electronic Networks & Communication in the UK

Sommer, P, (1999) [CD-ROM]. Intrusion Detection Systems as Evidence, Abstract from Computer Networks 31

Stallings, W, (1999) SNMP, SNMPv2, SNMPv3 and RMON 1 & 2: Practical Network Management, Reading, Mass. Addison Wesley

Stein, L.D & Stewert, J.N, (2002), [WWW], The World Wide Web Consortium, The WWW Security FAQ, http://www.w3.org/Security/Faq/

Stevens, R.S, (2003), TCP/IP Illustrated, Toronto, Addison Wesley

Sommer, P. (1999), [White Paper] IDS as Evidence

Templeton, S. J, Levitt, K. E, (N.d), [White Paper], Detecting 'Spoofed' Packets

United States Air Force (1999), [White Paper], Docterine Document 2-5.1: Electronic Warfare

Vannoorenberghe, P. & Postaire, J.G, N.d., Laboratorie d'Instrumentation du signal image et des reseaux, Calais. Detection of Intrusion using a Background Structural Model.

Wack, J, Tracy, M, Souppaya, M, Computer Security Division, National Institute of Standards and Technology. (2003), [White Paper], Guideline for Network Security Testing, NIST Special Publication 800-42

White, G, Pooch, V, (1996), Cooperating Security Managers: Distributed IDS

The White House, (2000), [Consultation Document}, Defending America's Cyberspace National Plan for Information Systems Protection Version 1.0

Appendix I. Existing Intrusion Detection Methodologies

With regard to computing, security is regarded as having three classifications:

- Availability is concerned with ensuring the preservation of data, such that those who legitimately require access to that data may reliably access it;
- Confidentiality is concerned with the prevention of unauthorised access to, or disclosure of, data held;
- Integrity is concerned with the prevention of data modification by unauthorised parties.

In order to maintain these three objectives, it is necessary to define a set of 'rules' relating to who is allowed access to the data, and under what conditions. It follows that, in order to ensure that security requirements are met, events, which are potentially hostile, must be detected. There are various detection techniques currently in use.

Anomaly-Based Detection:

This is an adaptive method, which involves ‘learning’ the normal characteristics of traffic, which is ‘normally’ present. The major difficulty inherent in this method is that, if abnormal activity is present during the learning phase, this may be misclassified such that this activity will remain undetected into the future. Additionally, Anomaly based detection is of no use during the time taken to ‘learn’ the characteristics of the network. A longer ‘learning’ phase offers higher accuracy, however, it will also take longer for the IDS to generate useful results. Furthermore, should any changes be made to the network topology, or the services employed, the learning process must be repeated.

Protocol-Based Anomaly Detection:

This utilises knowledge of protocol characteristics. Since this ‘knowledge’ is ‘hard coded’, the ‘learning’ phase is eliminated. For example, the presence of fragmented IP packets, the sum of which exceeded 65535 bytes, would generate an alert as such packets violate the protocol specification. Another obvious example of protocol based anomaly detection involves the detection of TCP packets with illegal flag settings, as discussed previously. Also, the monitoring for violations of protocol specifications will not require the user to configure any ‘attack specific rules’. This is a considerable advantage of the Protocol Based Anomaly Detection method.

A further variant of anomaly detection is statistical anomaly detection. For example, an increase in UDP traffic of a whole order of magnitude is highly suspicious. It is, however, important to realise that the IDS must either learn or be programmed, with threshold values for each type of traffic, which is (not) likely to be present under normal operation conditions.

In general, the advantage of ‘adaptive’ methodologies lies in the fact that, as little configuration is required, they offer a great deal of convenience to the end user. In addition, they are not restricted to the detection of known types of attack. However, results are often inconclusive, and adaptive detection systems are highly dependent on what they learn to be ‘normal’ network traffic characteristics.

Heuristic Detection Methodologies utilise mathematical techniques in order to ‘detect’ abnormalities in network traffic. This method is ideally suited to the detection of ‘port scans’, involving connection attempts to multiple, diverse, TCP or UDP ports on one or more target systems. It is very unusual for normal connection attempts to be made to multiple ports in a short space of time, therefore it is a near certainty that this activity indicates an attempt to discover which services are running on a given machine. This can be confirmed for certain if all packets detected have their Syn flag set (For TCP). Although there may be a requirement for all packets to originate from a single source address, this option should never be ‘hard coded’ since it is not unknown for hackers to work in groups, in order to make it more difficult for an intrusion detection system to correlate the data.

Heuristic Detection

Heuristic detection is the only viable detection method for some suspicious activity; however, algorithms may require a considerable investment of development time in order to maximise their effectiveness and minimise the generation of false alerts. It may also be necessary to design separate algorithms for various types of network.

Pattern Matching:

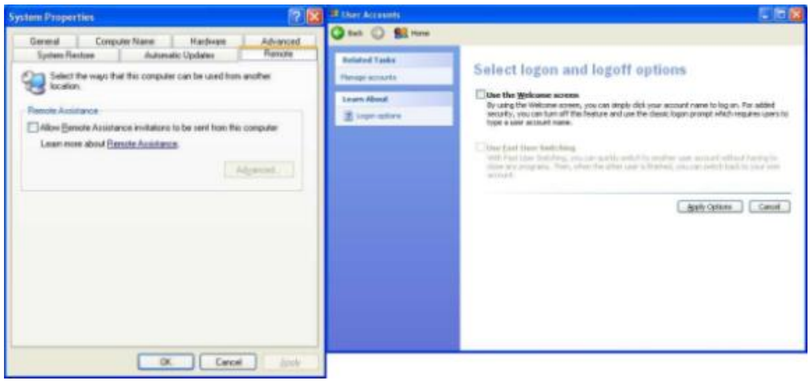
This is the simplest method of detection. It is very decisive in it’s approach, and system overheads are lower than those of more complex methods of detection. For example, the detection of the character string “GET /default.ida?” in a TCP Syn packet, targeting port 80 would indicate, with almost 100% certainty, that the

‘Code Red’ worm was responsible. This principle is generally of high accuracy, generating very few false alerts.
However, where another, legitimate service also shared this signature, this technique would certainly produce a very high number of false alerts! Another disadvantage is due to the fact that a slight modification of the attack, which is common between variants of the same virus, for example, would result in a missed event. Also, as this method only analyses individual packets, it does not readily adapt to the analysis of multiple packets forming an individual packet. It can be subverted simply by spanning signatures across multiple packets, or by fragmenting packets. Although it is possible to ‘queue’ packets until a complete analysis can be performed, this would greatly increase the performance overhead.
Also, there is the difficulty that many network services and intrusion tools do not operate on specific TCP or UDP port numbers. Although exceptions to a rule are rare as far as far as ‘well known’ services are concerned, many services listen for connections in the dynamic port range, some of which may allocate ports dynamically. For example, MS SQL Server listens for connections on port 1433 by default, but is easily reconfigured to use any port. It is also common for web services to listen on ports other than 80; for example, port 8080 is commonly used for HTTP proxy servers. This presents difficulties for pattern matching detection.
Stateful Pattern Matching:
This variant of the above technique recognises the order of packet arrival and extends detection across packet boundaries. This does imply the storage of incomplete signatures until a complete match can be made.
Protocol-Decode Analysis:
This method addresses the difficulty of detecting segmented signatures by decoding the protocol itself, in using exactly the same method as the software, which normally decodes the data. This allows for complete, accurate analysis of a given exploit, as it is able to detect violations of a given protocol to a high degree of accuracy. However, as not all ‘real-world’ software adheres strictly to RFC’s, it can produce regular false alarms as a consequence. Furthermore, this is a complex form of analysis, which is difficult and costly to implement.
It is apparent that no single method of detection is perfectly suited to the detection of every possible form of known attack. Furthermore, as the struggle between IDS designers, and those attempting to subvert or evade detection is ongoing, any intrusion detection system designer is permanently engaged in an ‘arms race’ in attempting to keep their product up to date. A further problem faced by designers of intrusion detection products involves the fact that there is no standard definition of a network intrusion. This results in a great deal of difficulty when comparing products and methodologies.
A general description of the attacks, and their objectives, can be found in Appendix 2.

Appendix II. Definitions of Attack Types

Why are systems vulnerable to attacks?
Once, hacking almost exclusively involved machines running the various versions of Unix, which often offered a great deal of existing services, which provided opportunities for hacking attacks. Up until approximately 1996, DOS/Windows based systems were relatively simple, offering few such opportunities. However, three factors have changed this situation. The first was the explosion in the numbers of home and business computers running various Microsoft Windows versions, many of which were not secure. [72]

Furthermore, Microsoft has tended to design for ‘ease of use’, rather than with security in mind.
Secondly, the emergence of Windows NT, whilst advertised as ‘secure’, offered a more challenging target. However, many users, whether home users or business users, see security as getting in the way of convenience, and therefore continue to run ‘potentially secure’ versions of MS Windows (NT/2000/XP) in the least secure way possible, making themselves just as vulnerable as if they were running Windows 9.X.
Furthermore, many ‘convenience’ features such as ‘Fast User Switching’ and ‘Remote Assistance’ (Fig. 58) provide all the assistance a that malicious user needs to compromise MS Windows systems.



Select the ways that this computer can be used from another location.

Remote Assistance

☐ Allow Remote Assistance invitations to be sent from this computer

Learn more about Remote Assistance

Advanced...

Select logon and logoff options

☐ Use the Welcome screen

By using the Welcome screen, you can simply click your account name to log on. For added security, you can turn off this feature and use the classic logon prompt which requires users to type a user account name.

☐ Use Fast User Switching

With Fast User Switching, you can quickly switch to another user account without having to close any programs. Then, when the other user is finished, you can switch back to your own account.

Fig. 58: In spite of its recent policy of ‘Taking security seriously’, Microsoft is continuing to provide assistance to malicious users, in the name of ‘Ease of Use’.

Finally, the widespread recent adoption of ‘broadband’ Internet connections has consequences for security – unprotected ‘home computers’ are connected permanently, often with fixed IP addresses, via fast connections. This has resulted in the emergence of widespread DDOS attacks.
Probably the single biggest problem at time of writing, however, is that many home users neither know nor understand that security is their concern –no Internet connected computer exists as an ‘island’; whereas people talk about being ‘connected to the Internet’, they fail to realise that by definition, it is equally true that *the Internet is connected to them.* [73].
Therefore, although Unix systems continue to be targets for malicious activity, the current focus of malicious activity has shifted towards Microsoft Operating Systems. It seems reasonable for this trend to continue. Although the majority of intruders are currently ‘amateurs’ who tend to choose very easy targets, experience has shown that such activity tends to capture the attention of ‘professional’ criminals, such as fraudsters. No amount of encryption of network traffic associated with electronic commerce is of any protection whatever, if the same customers computers can be easily accessed by fraudsters. [74]
Brief descriptions of the attacks follow.
1) Probes
These are reconnaissance methods, which attempt to identify vulnerable systems. Examples include:

- ICMP Echo (Ping) sweeps across a given address range, for example, 80.4.0.0 to 80.4.7.255, in order to discover which systems are up at the time (There is little point in attempting further reconnaissance on ‘down’ hosts!). Alternatively, it is possible to obtain lists of hostnames via DNS. This gives away plenty of information, as administrators tend to name their systems according to the services they provide, and often their location. The hostname may even give clues as to when the host may be up! For example, a name beginning in ‘ftp’ would imply not only that this particular machine offers the file transfer protocol service, but also that it is likely to be up 24 hours per day!

- Port Scans, which test one or more ports for the presence of servers. These range from simple TCP connect scans, to ‘stealth’ scans, which attempt to evade detection. For example, a ‘FIN’ scan sends TCP packets with only the ‘FIN’ flag set. Open TCP ports usually generate no response to this packet, however closed ports respond with a ‘reset’ packet, having the ‘Ack’ and ‘Rst’ flags set. Other techniques include the sending of packets from a range of faked IP addresses, in addition to the correct address. This makes it very difficult (but not impossible) to determine the actual source of the attack. [75]
- Vulnerability Testing: Although a port scan reveals which ports of a given system are open, they do not tell the attacker which vendor or version of a service is running. As, for example, vulnerabilities affecting Microsoft Internet Information Server are not common to all HTTP servers, the attacker needs this information in order to attempt an attack. This type of reconnaissance reveals whether, for example, a system has the latest patches installed.

Fingerprinting: This type of scan monitors a system’s responses to various events, such as the receipt of illegal TCP packets, comparing these with the known responses of various operating systems. This may possibly provide an indication of which operating system is running. The information obtained would allow the malicious computer attacker to identify if the targeted host is vulnerable to a certain exploit aimed at a certain service version running on a certain operating system.

- If an attacker already has a compromised host, they may run the scan remotely from this remote system. This simply makes the compromised system the actual ‘source’ of the attack.

2) Password Attacks

One of the traditional difficulties involving passwords concerns the ability of the user to memorise them. The longer, and more complex a password, the more difficult they are to guess, or even break. However, a stronger password is difficult to remember, particularly when it has only recently been assigned, and users will often either write them down, or change them to something easier to remember, such as their username capitalised or reversed; they may also use other easily guessed passwords, for example, their car registration number.

There are many software tools, which are freely available, which attempt to break passwords. These may use a ‘dictionary’ attack, or simply use sequential techniques.

Even if strong passwords are used, many protocols such as telnet, Netbios, FTP, POP3 and IMAP transmit passwords over the network without encryption! This allows an intruder, having compromised a single host, to gain access to other systems on the same network.

This problem can be addressed using ‘secure’ hubs, in order to minimise the number of points at which passwords can be intercepted, and where possible, using protocols, which encrypt passwords, for example, secure shell in lieu of telnet.

Where protocols are unencrypted, password breaking attacks may be detected by constructing an IDS configuration rule, which monitors the relevant port (for example, port 23, TCP for Telnet, or port 143 for IMAP), and a ‘signature’ which is identical to the username – as the username must be used in conjunction with the password. For example, the following IPGateKeeper rule “Format=1 Dest_Port=139 Record=Netbios.htm Text=DavidN” would detect such an attack on a Netbios network share. This log will normally be indicative of typical usage patterns; an anomalous usage pattern, such as multiple failed connection attempts, would indicate a ‘brute force’ attack. This log must, however, be maintained in a secure location. [76]

3) Denial of Service Attacks

In recent years, a new type of attack has become common. DOS attacks are not intended to obtain unauthorised access to systems, or the data stored on them. Rather, they attempt to render a system or network inoperable, either by flooding them with bogus traffic, or alternatively by sending malformed packets in an attempt to exploit a ‘bug’. A ‘DDOS’ attack involves the mounting of such an attack from multiple compromised systems. A worrying trend is for ‘viruses’ and ‘worms’ to carry out DDOS attacks. The ‘MyDoom’ virus is the most recent at time of writing. Therefore, it is naive to assume that the IP address(S) which is/are recorded by a firewall or IDS belongs to the original perpetrator.

‘Syn Flooding’ is another variant of a DDOS attack, in which multiple TCP connections are initiated, but never completed, causing a server to run out of resources.

4) Viruses, ‘Trojan Horses”, and other ‘Malware’ [77]

In pre-Internet days, viruses and other such programs were generally either application infectors, which were mainly distributed on floppy disks and other removable media, and spread when infected programs were run or copied, or boot sector infectors, which spread when a user inadvertently booted a computer from the infected media. The rate at which such programs spread was limited by the distribution method itself. Unfortunately, the Internet offers the scope for viruses and other forms of malware to be spread worldwide in a matter of hours, as the example below demonstrates.

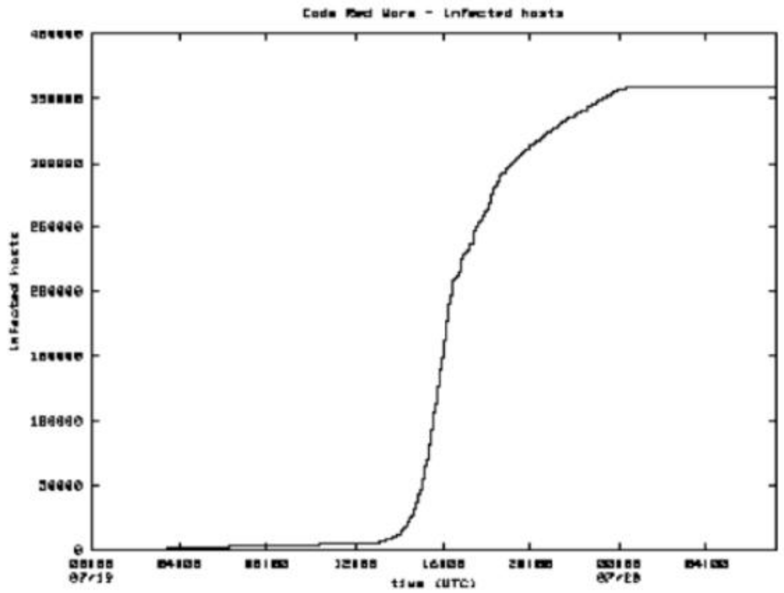
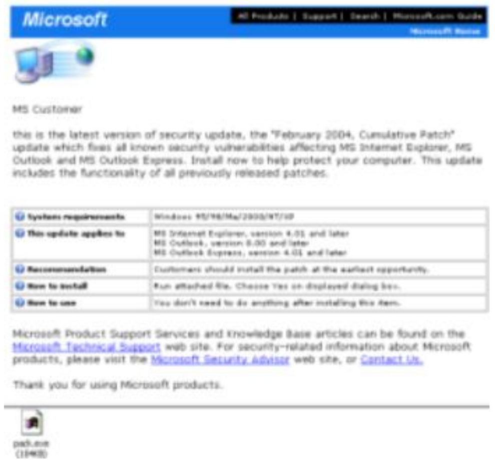


Fig. 59: Exponential spread of the ‘Code Red’ worm, which exploited flaws in MS Internet Information Services

Moore, D, (2001).

This graph demonstrates how the ‘Code Red’ worm spread on July 19th, 2001. As the rate of distribution at any given point is proportional to the number of compromised hosts, the number of compromised systems grows exponentially, until the number of vulnerable systems, which have not been infected, has diminished sufficiently. However, even at time of writing, I am still logging probes from infected machines on occasion).

There is also a trend for viruses, rather than to directly destroy files or data, to instead attempt to carry out DOS/DDOS attacks on networks and facilities. Some such programs spread by exploiting security vulnerabilities in network services (‘Code Red’, via a vulnerability in Microsoft Internet Information Server, and the recent ‘MSBlast’ worm which exploited a flaw in Microsoft’s implementation of the Endpoint Resolution protocol, are examples.). Others may rely on human activities to spread; for example, the Swen.A worm spreads via email, and relies on the user to open the attachment, which is the worm. (‘Opening’ an executable attachment actually runs the program). This is the appearance of this message:



Microsoft

MS Customer

this is the latest version of security update, the "February 2004, Cumulative Patch" update which fixes all known security vulnerabilities affecting MS Internet Explorer, MS Outlook and MS Outlook Express. Install now to help protect your computer. This update includes the functionality of all previously released patches.

System requirements	Windows 95/98/Me/2000/NT/XP
This update applies to	MS Internet Explorer, version 4.01 and later MS Outlook, version 8.00 and later MS Outlook Express, version 4.01 and later
Recommendation	Customers should install the patch at the earliest opportunity.
How to install	Run attached file. Choose Yes on displayed dialog box.
How to use	You don't need to do anything after installing this item.

Microsoft Product Support Services and Knowledge Base articles can be found on the Microsoft Technical Support web site. For security-related information about Microsoft products, please visit the Microsoft Security Advisor web site, or Contact Us.

Thank you for using Microsoft products.

Fig. 60: Typical appearance of a virus carrying email message, purporting to be a security update from Microsoft!

One reason for such viruses to be so successful is that the average user does not really understand how computers, and networks, operate. As an example, a fairly high percentage of users cannot identify which types of file are ‘dangerous’, when sent as email attachments. A survey carried out by Pentasafe Security Technologies, [78] discovered that as many as 10% of users would open an executable (.exe) attachment contained in an email message from someone they knew, whereas as two thirds would open a Microsoft Word document! ‘High risk’ file types include visual basic scripts (.vbs), MSDOS batch files, and Microsoft Word and Excel documents, which can contain macro viruses. However, files which cannot contain program code, such as images, plain text files, and certain formatted documents (for example, Adobe PDF) are safe to open. The fact that many ‘home users’ are completely unprotected from viruses compounds the problem of malicious software. Another common difficulty is that, even where they do have antiviral software installed, many users fail to appreciate that it will need to be updated regularly, as it gives no protection from viruses newer than the definition file.

Unfortunately, other than continually exhorting users to keep their antiviral software updated, and to avoid opening unexpected email attachments, there is little, which can be done to reduce the danger from malware. For example, it may be feasible to block messages with executable attachments at the server, but it is likely to be impractical to take this precaution regarding MS Word documents, as there are genuine reasons for users to need to exchange such documents. As few users genuinely need the Windows Scripting Host, it may be desirable to disassociate the .vbs file extension with the scripting host in order to provide protection from malicious scripts.

From an intrusion detection point of view, it is possible to detect virus related activity in several ways:

- Email provides an easy route for malware to penetrate internal networks, even those utilizing private addressing and NAT, which are not directly accessible from the public Internet. It is possible, however, to detect strings such as .vbs, and .exe in packets as they pass through the network. This makes it possible to identify messages containing dangerous attachments. The same principle can be applied to suspicious text in the message itself. For example, a number of recent viruses have purported to be service packs from Microsoft!

- Viruses which exploit weaknesses in network facing software must scan for vulnerable systems. Therefore, not only is this scanning activity in itself a detectable event, the packets also contain 'signatures', for example, GET /default.ida? In the case of the 'Code Red' worm.
- Network traffic related to viruses may itself present anomalous patterns. For example, the sudden increase in the usage of rarely used IP ports may be indicative of virus related activity. For example, large numbers of connection attempts to port 135 (DCE Endpoint Resolution) are characteristic of the 'MSBlast' virus.

5) Miscellaneous Attacks

In addition to the commonest attacks discussed so far, there are a number of other types, worth a brief mention:

- Miss-configured Services: Many services install with very weak default settings, or may be unintentionally miss configured by the user. For example, SMTP listeners having a default configuration allowing open relay of mail to anywhere; NFS exports exporting file spaces to 'everyone', SNMP 'agents' which are left with default passwords (which are included in the vendor's documentation), and telnet/rlogin services which have not been re-configured since installation.
- An example of a user error is the setting up of an X server, using the command 'xhost +', which allows users all over the Internet access to X sessions.
- Poorly written software: Bugs in network facing software can often be exploited, allowing an intruder to run commands on a remote system. Certain SMTP, SSH, IMAP, and DNS implementations have been found to contain such flaws. Processes running as root/administrator present the greatest risks. Some CGI scripts, installed by default, have also been exploited in this fashion.

"It is a sad fact that , although many people can write functioning software, very few can write software that functions securely." Northcutt, (January, 2001).

6) 'Backdoors': an intruder, who wishes to regain access to a system at a later date, whether or not the original vulnerability has been discovered, installs these. A 'root kit' is a well-known example. These may also attempt to further hide evidence of a break-in, for example, they may contain a fake version of the Netstat command, which does not list listening ports associated with the backdoor.

Appendix III. Desirable Characteristics of a Software Component

The requirements for a software component depend, in part; on it's application, and the circumstances in which it is to be used. For example, a firewall should ideally 'fail closed', rather than 'fail open'. However, there are certain considerations which are at least of some importance to all software applications:

- Dependability: refers not only to the statistical probability of failure, but also the consequences of failure. In the case of IPGateKeeper, attention to such design details as the usage of array elements, has resulted in very few crashes. Where failure does occur, this failure should be immediately obvious.

Furthermore, failure should not result in damage to data, or the creation of security holes. The last consideration is of great importance given that IPGateKeeper must run with administrative privileges.

- Usability: refers to the ease with which the user can interact with the end product. In this particular application, it is necessary for the program to run with a minimum of attention (except where alerts are concerned), and should have adequate documentation defining the meaning and significance of each alert type generated. Also, the generation of 'false positive' alerts (which waste the users time, and give the impression of 'crying wolf too often'), and 'false negative' alerts (which fail to report correctly an event of significance), should be avoided. However, due to the complexity of the operating environment, there is no means by which this problem can be completely eliminated. [79]
- Efficiency: It is essential to make economical use of processor time, disk space, and memory usage. However, this presents some difficulty, due to conflicting operating requirements. As the application must operate in 'real time', the use of disk storage for transient data, and the use of dynamic memory allocation techniques (such as linked lists) for temporary storage, imposes unacceptable overheads in terms of disk read/write operations, and processor time respectively. Therefore, it is necessary to introduce the concept of 'primary' and 'secondary' resource usage considerations.
- Maintainability: As new threats emerge, it is necessary to make the software adaptable where reasonably practical. For example, although I made the decision to allow certain parameters to be configurable via the configuration file, other parameters for which configurability is not essential, I omitted from the configuration. As an example, I decided not to make the treatment of TCP flag settings configurable, as legal flag definitions do not change greatly on a timescale comparable with program version updates. (The need to allow the user to configure the system must be balanced with the need not to make the configuration system needlessly complicated; this is both from the viewpoint of HCI, and that of making it overly easy to miss-configure the system, possibly resulting in false reporting!).

Whereas the majority of software applications must be maintainable with regard to changing business requirements, in this particular application, changing security threats are of paramount importance.

Appendix IV. Development & Testing Environment

In order to quantify the test results, it is appropriate to provide some basic information relating to the conditions under which the software has been developed and tested. This is to provide the background to experimental results, and to the reasoning behind the findings and conclusion.
Although the software is designed to run on a computer acting as a router or 'gateway', between a private (RFC1918) [80] network and the public Internet, my own network is actually set up somewhat differently. The actual gateway is a solid-state gateway, which is configured to allow certain IP ports to be forwarded to a single computer inside the network. The computer I have developed the software on is running MS Windows 2000 Professional, with service pack 4 installed. The IP address of this computer is 192.168.0.3 – this will appear in log files generated. As port forwarding is in use, it will appear as though this computer is directly connected to external computers; this is not the result of a software 'bug'. The topology of the network is as follows:

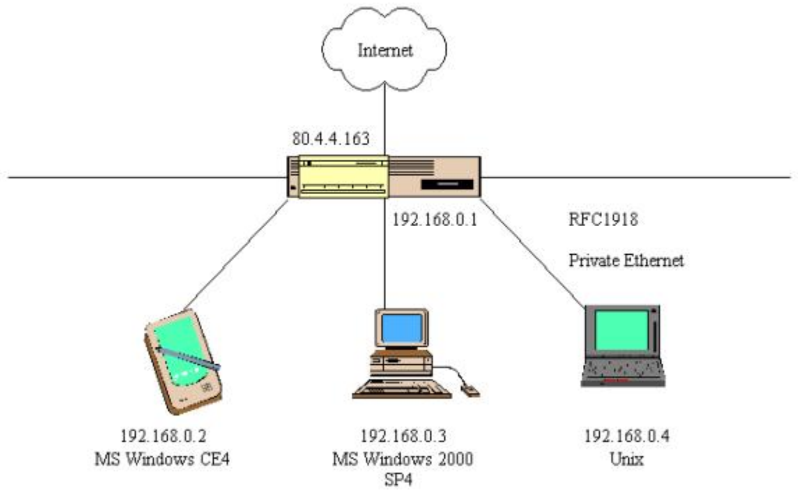
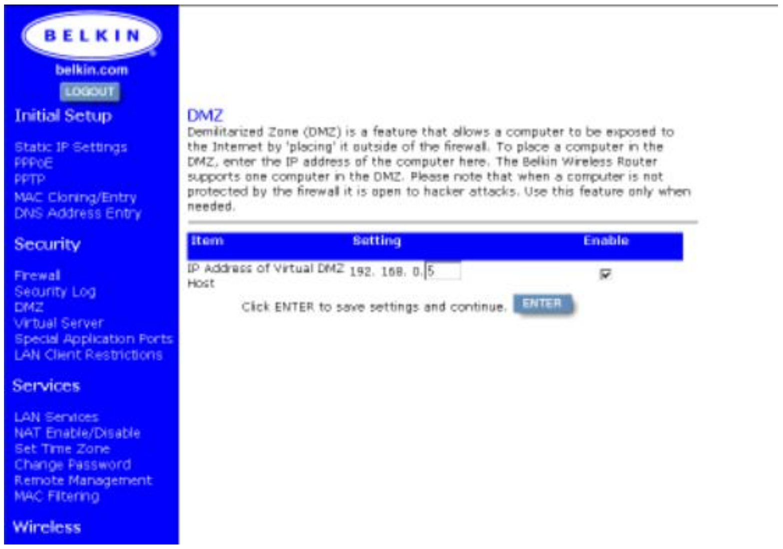


Fig. 61: Topology of the Network on which IPGateKeeper was developed and tested

This map was drawn using Smartdraw 6.2. IP addresses are static, and Port forwarding is set up via the gateway (192.168.0.1). The network itself is a standard Ethernet conforming to the IEEE 802.3 standard.
A single computer can be placed in the DMZ using the gateway configuration settings:



DMZ

Demilitarized Zone (DMZ) is a feature that allows a computer to be exposed to the Internet by 'placing' it outside of the firewall. To place a computer in the DMZ, enter the IP address of the computer here. The Belkin Wireless Router supports one computer in the DMZ. Please note that when a computer is not protected by the firewall it is open to hacker attacks. Use this feature only when needed.

Item	Setting	Enable
IP Address of Virtual DMZ Host	192. 168. 0. 5	✓

Click ENTER to save settings and continue. ENTER

Fig. 62: Administration page for the Gateway DMZ

The purpose of this section is to demonstrate how the network architecture can in itself affect the traffic ‘seen’ by an IDS. It is worth noting that the network architecture may present formidable topological difficulties for any IDS. In the following example, the location of the IDS is unimportant:

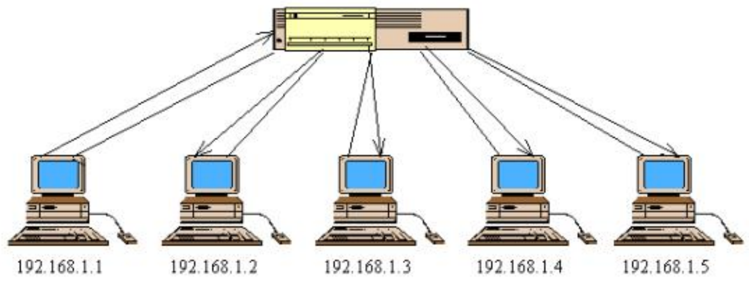


Fig. 63: Re-broadcasting of network traffic through all ports of a ‘hub’

In this case, the network traffic is routed through a hub. This has consequences for security, since network traffic from any one host (for example 192.168.1.1) is visible on all ports (and can be monitored by all other computers connected). [81] This implies that were any of these computers to be compromised, a network ‘sniffer’ could intercept password and user identifiers for all connected computers! Also, broadcast-based attacks, such as ICMP ‘smurf’ attacks are most destructive in a non-switched environment. [82]However, an IDS could be run on any of the connected computers (or a hardware IDS connected to any port) and could ‘see’ all activity.

However, where the computers are connected via a ‘switch’, [83] the situation is different:

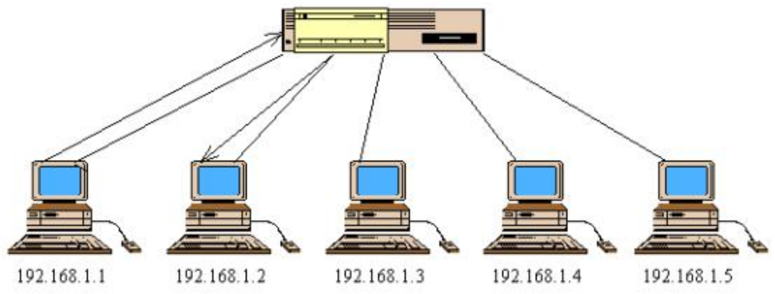


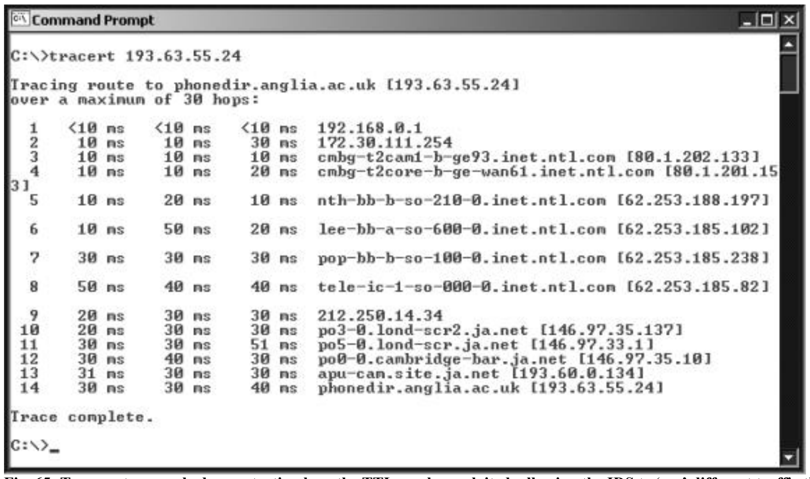
Fig. 64: A switch only broadcasts packets via the 'relevant' ports

As the switch is able to recognise the individual devices connected to it, traffic is only sent to single ports. For example, if 192.168.1.1 sends a packet to 192.168.1.2, this packet is only sent to 192.168.1.2. Consequently, a switch is much more economic with regard to bandwidth, and offers a better degree of security, as a compromised machine connected to other ports would not 'see' traffic directed to the other machines. However, this makes the installation of an IDS difficult, since the IDS needs to be able to monitor all traffic. For this reason, many manufacturers provide 'mirror' ports, which are reserved for monitoring purposes.
It is worth pointing out that my own gateway has some properties of a switch, in the regard to the way traffic is routed. The gateway provides both wired and wireless Ethernet connectivity, having three wired ports. (The numbers of wireless hosts, which can be connected, are limited only with regard to bandwidth). Traffic to wired hosts is only routed to the port to which it is connected. Therefore, the system running the IDS (192.168.0.3) will only be able to monitor traffic for which 192.168.0.3 is the source or destination. This is true for both internal traffic, and traffic to/from the Internet. However, it is possible for an internal host to 'see' all incoming traffic from the Internet, if it is placed in the DMZ.

Appendix V. Security Considerations

Given the purpose of an IDS, it is reasonable to expect that, given the opportunity, an intruder would wish to subvert, evade or even compromise the IDS itself. Such attempts may range from a simple attempt to evade conditions which result in the IDS recording evidence of suspicious activity, to denial of service attempts on the IDS itself.
For example, one technique, which could be employed by the intruder, would involve the sending of small, or widely spaced, packets, which are part of the actual attack, concealed amongst innocuous packets, in an attempt to fool the IDS. For example, signatures could span multiple packets, interspersed between which are thousands of 'legitimate' packets. Although in theory it is possible to 'queue' packets in order to resemble them, in practice, few systems can afford the performance overhead.
Newsham and Ptacek (1998) state *"It is not possible for an IDS to 'see' exactly the same traffic as the target(s), without an impractical level of knowledge of the target environment".*

Fig. 65 provides insight into how the TTL operates:



```
C:\>tracert 193.63.55.24

Tracing route to phonedir.anglia.ac.uk [193.63.55.24]
over a maximum of 30 hops:

  1    <10 ms   <10 ms   <10 ms  192.168.0.1
  2     10 ms    10 ms    30 ms  172.30.111.254
  3     10 ms    10 ms    10 ms  cmbg-t2cam1-b-ge93.inet.ntl.com [80.1.202.133]
  4     10 ms    10 ms    20 ms  cmbg-t2core-b-ge-wan61.inet.ntl.com [80.1.201.15
3]
  5     10 ms    20 ms    10 ms  nth-bb-b-so-210-0.inet.ntl.com [62.253.188.197]

  6     10 ms    50 ms    20 ms  lee-bb-a-so-600-0.inet.ntl.com [62.253.185.102]

  7     30 ms    30 ms    30 ms  pop-bb-b-so-100-0.inet.ntl.com [62.253.185.238]

  8     50 ms    40 ms    40 ms  tele-ic-1-so-000-0.inet.ntl.com [62.253.185.82]

  9     20 ms    30 ms    30 ms  212.250.14.34
 10     20 ms    30 ms    30 ms  po3-0.lond-scr2.ja.net [146.97.35.137]
 11     30 ms    30 ms    51 ms  po5-0.lond-scr.ja.net [146.97.33.1]
 12     30 ms    40 ms    30 ms  po0-0.cambridge-bar.ja.net [146.97.35.10]
 13     31 ms    30 ms    30 ms  apu-cam.site.ja.net [193.60.0.134]
 14     30 ms    30 ms    40 ms  phonedir.anglia.ac.uk [193.63.55.24]

Trace complete.

C:\>_
```

Fig. 65: Traceroot example demonstrating how the TTL can be exploited, allowing the IDS to 'see' different traffic than the hosts. This is because the TTL may vary due to the network topology (the IDS may not be the same number of 'hops' from the intruder as the target).

The ‘traceroot’ (‘tracert’ utility under MS Windows NT), operates using the ‘Time to Live’ feature of the Internet Protocol. Normally, operating systems tend to utilise default values for the TTL; for example for Windows NT operating systems, the default is 128 (80h):

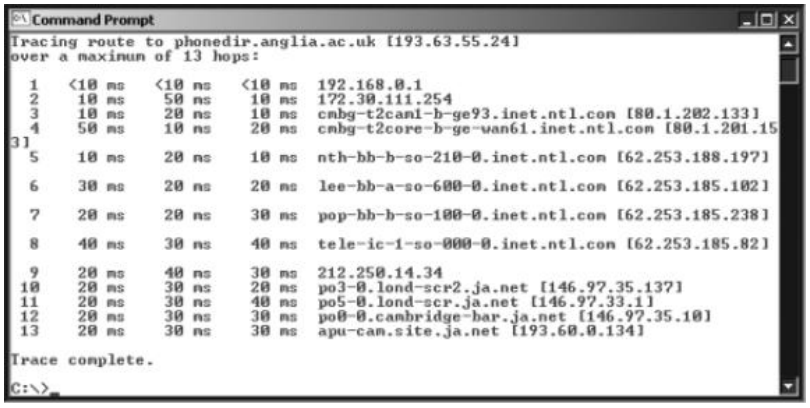
```
0000  00 10 7a 4e 4a 7a 00 10  a7 10 e5 50 08 00 45 00   ..ZNJz.. ...P..E.
0010  00 3c 95 b1 40 00 80 01  e3 b9 c0 a8 00 03 c0 a8   .<..@.█. ........
0020  00 02 00 00 a8 ab 02 00  01 00 45 45 45 45 45 45   ........ ..EEEEEE
0030  45 45 45 45 45 45 45 45  45 45 45 45 45 45 45 45   EEEEEEEE EEEEEEEE
0040  45 45 45 45 45 45 45 45  45 45                     EEEEEEEE EE
```

However, traceroot sends ICMP echo requests to the specified host using incremented TTL values. On the first attempt, the TTL is set to one:

```
0000  00 10 a7 10 e5 50 00 10  7a 4e 4a 7a 08 00 45 00   .....P.. ZNJz..E.
0010  00 3c 05 01 40 00 01 01  f3 6a c0 a8 00 02 c0 a8   .<..@.█. .j......
0020  00 03 08 00 a0 ab 02 00  01 00 45 45 45 45 45 45   ........ ..EEEEEE
0030  45 45 45 45 45 45 45 45  45 45 45 45 45 45 45 45   EEEEEEEE EEEEEEEE
0040  45 45 45 45 45 45 45 45  45 45                     EEEEEEEE EE
```

On reaching the first router, the TTL is decremented to zero, and the packet dropped. The router then notifies the sending host, giving its IP address. For each subsequent attempt, the TTL is increased by one, until the destination is reached. This constructs a list of the intermediate routers. Note that in the case of an incoming trace root, the sequence would be reversed. [84]

By manipulating the TTL or checksum[85] values, the attacker could cause all of the packets to be dropped at the gateway, except for those involved in the actual attack, possibly also using false source addresses. An insertation/evasion attack uses an invalid checksum. Where the IDS validates checksums, but not the target, an IDS can drop a packet containing part of a signature. Where the target does not validate the checksum, it accepts the packet, completing the signature not seen at the IDS! On the contrary, where the target host validates checksums, but not the IDS, an evasion attack would consist of a packet containing a string inserted into a signature. The IDS, not validating the checksum, accepts the packet, and never ‘sees’ the signature, but the target drops this packet, and therefore acts on the signature.
In figure 51, an IDS running on apu-cam.site.ja.net [193.60.0.134] would ‘see’ packets sent from the same host from which the traceroot was initiated, having a TTL of 13, whereas the host phonedir.anglia.ac.uk [193.63.55.24], being 14 hops away, would not. Another traceroot, with the maximum TTL set to 13, demonstrates how the ICMP packet expires one hop short of the target:



```
Tracing route to phonedir.anglia.ac.uk [193.63.55.24]
over a maximum of 13 hops:

  1    <10 ms   <10 ms   <10 ms  192.168.0.1
  2     10 ms    50 ms    10 ms  172.30.111.254
  3     10 ms    20 ms    10 ms  cmbg-t2cam1-b-ge93.inet.ntl.com [80.1.202.133]
  4     50 ms    10 ms    20 ms  cmbg-t2core-b-ge-wan61.inet.ntl.com [80.1.201.15
3]
  5     10 ms    20 ms    10 ms  nth-bb-b-so-210-0.inet.ntl.com [62.253.188.197]

  6     30 ms    20 ms    20 ms  lee-bb-a-so-600-0.inet.ntl.com [62.253.185.102]

  7     20 ms    20 ms    30 ms  pop-bb-b-so-100-0.inet.ntl.com [62.253.185.238]

  8     40 ms    30 ms    40 ms  tele-ic-1-so-000-0.inet.ntl.com [62.253.185.82]

  9     20 ms    40 ms    30 ms  212.250.14.34
 10     20 ms    30 ms    20 ms  po3-0.lond-scr2.ja.net [146.97.35.137]
 11     20 ms    30 ms    40 ms  po5-0.lond-scr.ja.net [146.97.33.1]
 12     20 ms    30 ms    30 ms  po0-0.cambridge-bar.ja.net [146.97.35.10]
 13     20 ms    30 ms    30 ms  apu-cam.site.ja.net [193.60.0.134]

Trace complete.

C:\>_
```

Fig. 66: The traceroot expires short of the target, since the TTL is too low

It is worth noting that traceroot can only measure the number of routers on the way to a target which is actually routable from the Internet – for example, it is not possible for a traceroot from 193.63.55.24 to reach the initiating host 192.168.0.3, since this host is inside a private (RFC1918) Intranet; it is not routable from outside.

Another potential difficulty involves the actual recording of the data itself. Regardless of the size of the gateway system's hard disk, it would eventually become full. The use of compression [86] can dramatically increase the space required to store the event logs – however here also there is a real difficulty. An IDS has to monitor and record events in real time, and in any real time system, time delay is de-stabilising. It is unlikely that, during the progress of a DDOS attack, for example, the records could be updated in real time, if compression were in use.

Whether by filling up the hard disk, or by sending traffic at such a rate that it could not be recorded (possibly even causing the IDS to crash!), it is entirely possible that the IDS could, in itself, become the target of a flooding attack.

In addition, the fact that the IDS must run with Administrative levels of privilege implies that it must not in any way be accessible to the outside Internet. For example, the executable and configuration files must not reside in publicly accessible directories, nor must any sensitive output be publicly accessible. [87]

Appendix VI. Intrusion Detection Systems as opposed to Firewalls

One frequently asked question regarding intrusion detection, is what perceived benefit an IDS has, given that an organization already has a firewall? There are many possible reasons.

A network-based firewall is a hardware device, which is placed between a network and the global Internet, in order to enhance the security of the network. As a rule, it is configured to block all types of traffic other than those specifically required. Configuration can be either protocol based, or address based. However, although a firewall can provide a useful enhancement to network security, by virtue of restricted access to servers, which are running accidentally, it cannot entirely guarantee security of the equipment behind the firewall, since some network traffic must be allowed to pass through. It is best regarded as a supplement to other security measures, rather than a substitute.

"The very existence of the IDS products is in part based on the assumption that at some Point the firewall will fail." Sample, Nickle, & Poynter, (2000)

The majority of hardware firewall manufacturers make the use of NAT a condition of their design. This effectively conceals the internal topology of a network behind the firewall, the logic being that it is considerably more difficult to attack internal systems if the intruder cannot discover the addresses or topology employed within the network. NAT also has the benefit of reducing the number of global IP addresses required by an organization, in the light of current IPv4 address shortages. However, this denies the network designer the option of non-translation, using a 'one to one' mapping of hosts, which require protocols which do not operate over NAT. [88]

Also, many organisations, requiring the compilation of network traffic statistics, would not be able to determine the origin of traffic from hosts behind the firewall, as all traffic would be traceable only to the firewall itself. Furthermore, as time is of the essence during a security incident, it may be important to quickly identify or block a single host. This could not be achieved easily where NAT was in use.

One other major problem associated with firewalls, is that although they do provide protection against intrusions via services, which may be running accidentally, they provide no protection at all from attacks on services which their configuration allows external access to. For example, traffic directed to a public web server must by necessity be allowed to pass through; otherwise the web server is of no use whatsoever!

Therefore, the firewall will give no protection against attacks on the web server itself. An IDS, however, will detect such attacks on permitted services as violate it's own configuration, without effecting access to the permitted services.
The truth is that a firewall only restricts access at the network perimeter. It therefore follows that they give no protection from internal network misuse, or where other external access is available behind them, for example, via modems or, in particular, wireless access points, as a survey sponsored by the International Chamber of commerce demonstrates. "Although wireless networks do have some basic built-in security features, the vast majority of networks found during this latest survey had not turned them on.

Even the few that had turned on the basic encryption system were using default settings, making it easy for an attacker to guess the key needed to unscramble data." Gunning, (2002).

Finally, the presence of a firewall can impact a security audit process, causing it to generate misleading results. For example, it may be possible for an unauthorised user to hide behind the firewall itself, by connecting portable equipment to the network behind a NAT firewall. The connection of unauthorised equipment would not be detectable outside the firewall, as would otherwise be the case. [89] This is particularly pertinent to wireless networks, which do not even require the intruder to obtain physical access to hardware.
It therefore follows that the security benefits of the firewall must be weighed against the possible disadvantages. However, an IDS, which passively monitors network traffic, in theory has no effect on the behavior of the network itself.
As firewalls do not protect against attacks, which do not happen to contravene their rule base, it is advisable to situate an IDS both inside and outside the firewall.

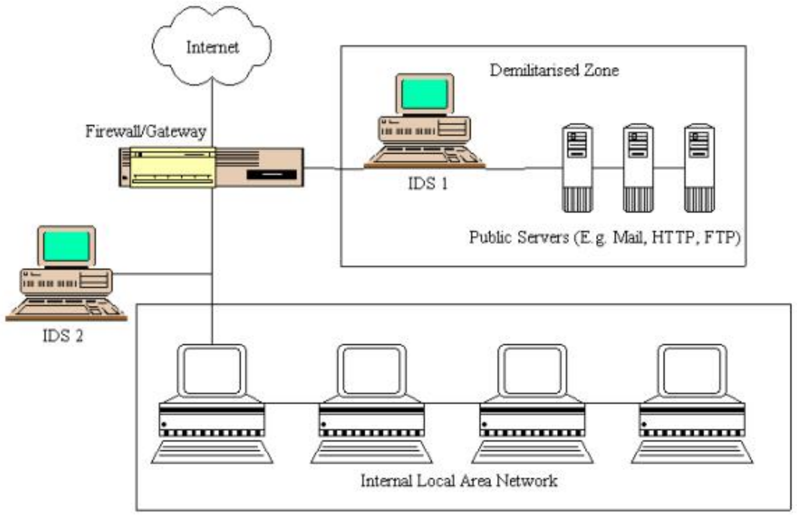

Internet
Demilitarised Zone
Firewall/Gateway
IDS 1
Public Servers (E.g. Mail, HTTP, FTP)
IDS 2
Internal Local Area Network

Fig. 67: The IDS in the DMZ, 'sees' all incoming traffic, and is able to generate an 'early warning' of probing attempts, whereas the IDS inside the firewall is able to concentrate exclusively on those attacks which do not violate any of the firewall rules

IDS 1, located in the Demilitarised Zone, will 'see' all traffic arriving from the global Internet, whereas IDS 2 will only 'see' the traffic, which has passed the firewall. This combination of 'internal' and 'external' monitoring has the following advantages:

- It is possible to determine whether hostile activity originated internally or externally;
- It is possible to differentiate the records from the internal and external IDS, in order to configure any new firewall rules;

- The internal IDS can be configured to detect intrusions from sources other than the primary Internet connection, such as modems and wireless access points;
- Should the firewall fail, or be miss configured, the internal IDS should detect this;
- The external IDS may be able to detect DOS attacks on the router or gateway itself. Also, it may give warning of any scanning activity, much of which will not pass through the firewall.

"A critical rule of thumb is redundancy; in other words, if information sources are key to monitoring a critical system, we should have multiple monitoring points with a good bit of overlap"

Bace, (2000).

VIII. List of Illustrations

All illustrations are original except where otherwise referenced.

Fig. 1: Languard Security Scanner, by GFI Software LTD
Fig. 2: Original IPGateKeeper Deployment UML Class Diagram; drawn using 'SmartDraw' (Prior to implementation).
Fig. 3: IPGateKeeper UML Operating Sequence Diagram
Fig. 4: Simplified IPGateKeeper UML Interaction Diagram
Fig. 5: WSA Access Error 10013
Fig. 6: IPGateKeeper UML Activity Diagram
Fig. 7: IPGateKeeper UML Operating Sequence Diagram
Fig. 8: The seven layer OSI Model
Fig. 9: An IP packet contained within an Ethernet Frame.
Fig. 10: The first IP byte
Fig. 11: The transport layer protocol number
Fig. 12: Destination address – 32-bit word
Fig. 13: The source port – 16-bit word
Fig. 14: The TCP Flag Byte
Fig. 15: An 'out of specification' TCP packet
Fig. 16: Packet generated by a 'FIN scan'
Fig. 17: NMAP 'XMAS' scan packet
Fig. 18: Example of a TCP connect scan
Fig. 19: Flowchart for a TCP Connect Scan. Many firewalls prevent any response to probing attempts (stealth).
Fig. 20: TCP State Transition Diagram. Based on SecureNode Inc (2003).
Fig. 21: UDP Scan Flowchart
Fig. 22: HTTP (Left) is unencrypted; therefore the traffic is not secure. However, an IDS can monitor the traffic. In this example, a signature of 'Apache' could have been used in conjunction with Dest_Port=80 to detect connections to Apache servers. However, HTTPS (Centre) and SSH (Right) are strongly encrypted, preventing signature detection.
Fig. 23: IPGateKeeper record for an Echo protocol test
Fig. 24: Chargen protocol packet
Fig. 25: Echo Protocol packet
Fig. 26: Examples of such scripting requests arrive at my own server with alarming regularity.
Fig. 27: An example of a 'non-standard' service
Fig. 28: IP Gatekeeper configuration design. Traffic attributes relevant to IDS design are selected from RFC's 768 (UDP), 791 (IP), 792 (ICMP) & 793 (TCP)..
Fig. 29: Example signature detection record
Fig. 30: Netbios Name Service query
Fig. 31: Netbios Broadcast, using the Netbios Datagram Service (UDP port 138)
Fig. 32: Startup Summery
Fig. 33: A typical NTP Query. The lower the stratum number, the higher the accuracy; however network latency must be taken into account also.
Fig. 34: Output log generated by the 'Format=0' command
Fig. 35: Output log generated by the 'Format=1' command
Fig. 36: Output log generated by the 'Format=2' command
Fig. 37: Output log generated by the 'Format=3' command
Fig. 38: Output log generated by the 'Format=4' command
Fig. 39: Output log generated by the 'Format=5' command
Fig. 40: Output log generated by the 'Format=6' command
Fig. 41: Falsified ('Spoofed') IP addresses
Fig. 42: Configuration Command Activity Diagram (Using 'Logic Trees')
Fig. 43: 'Logic Tree' for a single rule
Fig. 44: Sample automatically generated rules in response to events
Fig. 45: Notification of 'anomalous' port usage
Fig. 46: Example Output for the 'Multi Router Traffic Gopher'
Fig. 47: Graph of the above equation
Fig. 48: Effect of changing the 'delay' variable
Fig. 49: Derivative (gradient) demonstration
Fig. 50: Example Output for a TCP connect scan

Fig. 51:Appearance of 'Normal' network activity, for a generalised configuration
Fig. 52 TCP Connection Status Record
Fig. 53 Listening UDP Services
Fig. 54 Comparison with MS Windows 'Netstat' command
Fig. 55: 'Netstat –a' Shows all listening services by name where recognised, and their status
Fig. 56: 'Netstat –a –n' Shows all listening services, and current TCP connection status
Fig. 57: Home Network Topology
Fig. 58: In spite of its recent policy of 'Taking security seriously', Microsoft is continuing to provide assistance to malicious users, in the name of 'Ease of Use'.
Fig. 59: Exponential spread of the 'Code Red' worm, which exploited flaws in MS Internet Information Services, Moore, D, (2001).
Fig. 60: Typical appearance of a virus carrying email message, purporting to be a security update from Microsoft!
Fig. 61: Topology of the Network on which IPGateKeeper was developed and tested
Fig. 62: Administration page for the Gateway DMZ
Fig. 63: Re-broadcasting of network traffic through all ports of a 'hub'
Fig. 64: A switch only broadcasts packets via the 'relevant' ports
Fig. 65: Traceroot example demonstrating how the TTL can be exploited, allowing the IDS to 'see' different traffic than the hosts. This is because the TTL may vary due to the network topology (the IDS may not be the same number of 'hops' from the intruder as the target).
Fig. 66: The traceroot expires short of the target, since the TTL is too low
Fig. 67: The IDS in the DMZ, 'sees' all incoming traffic, and is able to generate an 'early warning' of probing attempts, whereas the IDS inside the firewall is able to concentrate exclusively on those attacks which do not violate any of the firewall rules.

---

[1] IPGateKeeper is written in 'C', as opposed to C++ or Java.

[2] However, in my own case this approach is of limited meaning except where background research is concerned; IPGateKeeper has no way of knowing what local or network countermeasures have been utilised!

[3] As an example, as a probe by the 'Code Red' worm significant? Certainly, where an unprotected instance of MS IIS is concerned, but this is of little urgency where your web servers only run Apache. The configuration allows such probes directed at non-IIS servers to be ignored where desired.

[4] This is an approximation of the perceived threat posed by the nature of the service offered. This takes into account factors such as the appropriateness of exposure of a service to the global Internet, the nature of the service itself (for example Netbios, anonymous FTP and SNMP are considered high risk), whether the IP port concerned is associated with known malicious code, and whether implementations of a service have a previous history of secondary exposures, such as buffer overflow vulnerabilities in Bind.

[5] Normal packets are produced indirectly, via the system protocol stack. However, 'Raw Sockets' bypass the protocol stack, allowing packets to be produced 'manually' and written directly to the Network Interface. This requires Root/Administrative privileges.

[6]http://www.hmso.gov.uk/acts/acts1990/Ukpga_19900018_en_2.htm#mdiv1

[7] http://www.hmso.gov.uk/acts/acts1998/19980029.htm

[8] http://www.hmso.gov.uk/acts/acts2000/20000023.htm

[9] At the very minimum, port scanning is contrary to nearly all Internet Service Providers' Acceptable Use Policies.

[10] This can be proven by means of the IP addresses involved; the addresses allocated for use within private Intranets are specified in RFC1918. In my own case, the addresses concerned are in the 192.168.0.X subnet. Note that a minority of tests use false addresses; where this is the case, it has been clearly stated.

[11] This is the 'Transport Layer' as defined in the OSI Model. The protocols of relevance to this project are Transport Control Protocol (TCP), User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP).

[12] An example of such a vulnerability, affecting Microsoft's implementation of the RPC Portmap Service, (defined in RFC 2616), was exploited by the 'MSBlast' virus in 2003.

[13] Although a 'naive' IDS methodology can itself be responsible for the generation of false alerts, there is the additional problem that many result from 'abnormal' traffic, which does not result from an attack. For example, broadcast packets from remote sites, packets having invalid header values, and packets addressed to nonexistent hosts or networks (for example, 'private' (RFC1918) ranges via the public Internet), can all result from miss configured domain name servers, bridges and routers. Whilst not hostile, the reporting of such activity may be regarded as helpful simply as it may assist in locating the miss configured device responsible.

[14] In software design, preventative action is far preferable to remedial action!

[15] (The UML diagrams have been produced using SmartDraw, which is copyright © of SmartDraw http://www.smartdraw.com . This is a general-purpose diagrammatic drawing package, which reduces the time required to produce technical and business drawings.)

[16] Although there are potentially 128 transport layer protocols (as dictated by the protocol number byte in the IP packet header), TCP, UDP and ICMP are the only common transport protocols of interest.

[17] In engineering terms, precision implies consistency of results (but not necessarily accuracy), whereas accuracy implies 'correctness' of results.

[18] Unfortunately, the Runas command is available only under Windows 2000 and higher.

[19] This topic is described in Microsoft Knowledge Base article 195445.

[20] The 'Desirable Characteristics' of a software component are discussed in appendix 3.

[21] Posix – Portable Operating System Interface.

[22] The network time protocol is the most accurate time reference; each entity in a distributed system should be synchronised using the same stratum.

[23] The Ethereal network analyzer can be obtained from www.ethereal.com. A list of authors is available here:

http://www.ethereal.com/introduction.html#authors.

[24] The Internet Assigned Numbers Authority manages the assignment of the 'Well Known' ports in the range 0-1023. A listing of the assigned services is available on IANA's website at http://www.iana.org/assignments/port-numbers.

[25] In fact, the flag settings are the only means by which to determine the state of a connection.

[26] A third possibility is that certain ports are 'stealthed' by a software firewall. In this case, no response is received at all! This is poor 'netiquette'; however where an intruder happens to scan ports, which are 'stealthed', it would appear that no computer exists at the IP address in question! Also, the scanner is delayed for the remainder of its timeout period whilst waiting for a response.

[27] The tern 'Trojan Horse' refers to a software application, which typically performs operations in addition to, or instead of, it's declared operation. Although in theory many such programs can be used in legitimate remote monitoring and control of a host, in practice they are normally put to dubious use.

[28] The term 'most significant', when dealing with binary numbers, implies 'highest value', rather than greatest priority. In an eight bit binary number, the most significant bit has a value of 128 if set. (A byte is an eight bit binary number).

[29] Unreliable, in this respect, implies that as UDP is 'connectionless', the sending host is not notified of dropped packets. It is the responsibility of session layer protocols to generate any acknowledgements when listening processes time out.

[30] The home network has a total bandwidth of 100mbps, which applies only to internal traffic. The Internet connection is capped at 1Mbps.

[31] One potential solution to the throughput problem would involve the use of a number of load balanced intrusion detection systems, running in parallel. However, this introduces other difficulties, including the co-ordination of events between the individual IDS's. This is particularly true where packet fragmentation is involved.

[32] There is an example of such a list at http://www.net.apu.ac.uk/bad-hosts.txt.

[33] However, signature-based analysis is certainly of value when taking into account intrusions at personal, rather than network, level. Harassing e-mail messages are an example. Using the email address, or keywords, as the signature, it is possible to build up a profile of the sender/recipient as evidence.

[34] The Echo protocol is a simple protocol defined in RFC862. Echo servers simply rebroadcast the payload of received packets. The 'Chargen' (Character Generator) protocol, defined in RFC864, generates an ordered list of ASCII characters. Both protocols are useful only for testing purposes.

[35] The majority of intrusions via web servers tend to target scripting programs, rather than the web server itself. It is therefore advisable to detect requests for scripts which are either known to be insecure, or which are known to have been poorly written.

[36] The Internet Assigned Numbers Authority has a comprehensive listing at http://www.iana.org/assignments/port-numbers.

[37] In practice, only processes running as 'root' can open ports in the 'privileged' range (1-1023) on most Unix systems.

[38] This is significant as specific exploits target specific services. For example, the 'Code Red' worm targeted only specific implementations of Microsoft Internet Information Services. Although packets destined for port 80, containing the "GET default IDA?" signature were characteristic of a scan by this worm, it may be helpful to determine whether a vulnerable server actually *responded* to the scan!

[39] Although an intruder would have to 'port scan' a machine in order to discover a service known to be running on an unassigned port, it is certainly possible to discover the location of such a server!

[40] Although it is possible to have a number of analysis processes running in parallel, experience has demonstrated that such a design presents an unacceptably high impact on system resources.

[41] Since the sole purpose of a Syn packet is to initiate the connection itself.

[42] Although ports are officially registered for session layer services, for both TCP and UDP, only a small minority actually deploys both TCP and UDP.

[43] Snort can be obtained for MS Windows and Unix/Linux systems, from http://www.snort.org/. A list of contributing authors can be obtained from http://www.snort.org/team.html.

[44] However, many Netbios Name lookups result from reconnaissance activity; the Netbios Names may in themselves provide useful information to an intruder.

[45] As many users miss configure 'Windows File and Printer Sharing', allowing users throughout the Internet access to their network shares. Therefore, on a typical Cable Modem network, a large number of probes for Netbios services on ports 139 and 445 are to be expected. In addition, a number of 'Worms', which spread via Windows File Sharing, may perform Netbios Name lookups on port 137 also.

[46] Some traceroot utilities provide a facility by which a DNS server can be queried to supply a list of all hosts in its database. Although a useful audit tool for an administrator, intruders can use the information to their advantage. This is because systems are often named according to their function. Whereas normal queries are performed using UDP, zone transfers use TCP for transport.

[47] The analysis is arranged in a hierarchy, with 'low level' analysis at packet level taking place first. Where any matches to the conditions declared in the configuration occur, the details are passed to subsequent procedures, which attempt to identify patterns between the packets.

[48] This is an unsolvable problem for an Internet Service Provider – it may appear that the scan originated from an address, which was not even allocated at the time! In this case, they are likely to accuse the complainant of providing inaccurate logs.

[49] It is certain that this address is false since the event originated from the public Internet, and this address is in a private (RFC1918) range. All addresses within my own network are in the 192.168.0.x subnet. Also, the flag bit value of 7Bb, or 1111011b, has both the Syn and Fin flags set! The remaining flags may be set in an attempt to fool firewalls or intrusion detection systems.

[50] For IPGateKeeper, a 'Rule' is a complete line in the configuration file. A rule may contain one or more commands.

[51] The shortcoming of a rule based IDS is centered around 'false negative' alerts, resulting from slight rule mismatches and non-implemented rules; in the case of adaptive systems, false alerting arises from 'mis-adaption' during the 'learning phase'.

[52] The word 'probably' is used here as a cautionary measure; this fact alone does not prove that this host is running Netbus! The truth is that users, when configuring applications, tend to choose values, which are easily remembered, such as 12345! For example, Netbus runs on MS Windows systems alone; if this host turns out to be a Unix system, then the Netbus possibility is ruled out. For example, users may install peer-to-peer software, or license managers, configuring them to run on an 'easy to remember' port value.

[53] Simple 'port usage profiling' is a very effective means by which to detect 'rouge' servers or malware.

[54] This implies that the n'th commonest value occurs with a frequency of cr^-1/k, where c and k are constants. K is often ≅ 1; where k = 1, the n'th commonest value has a frequently equal to 1/r.

[55] Details are available at http://www.fi.muni.cz/~kas/mrtg-rrd.

[56] The actual node in question, route-cent-2.cam.ac.uk, was chosen as a fairly representative sample from the Cambridge University Data Network. As a general rule, the larger the network, (in terms of the number of hosts), the more representive the data.

[57] The human eye is an example of a logarithmic detection system; this is due to the fact that a high degree of sensitivity is required at low illumination levels, whereas during the daytime, the eye must be protected from damage due to high light intensity.

[58] The equations employed were devised from work relating to transfer functions, as part of the reference material for the undergraduate 'Control Systems' module.

[59] The reason for recording a single connection attempt is in order to detect scans which only probe for a single port per target system, and for general testing purposes.

[60] For this particular test criteria, connection attempts to the same port on more than one host count as a single attempt. Although it is easily possible to count each connection attempt separately, this is difficult to demonstrate or test using a single machine.

[61] It is important to realise that this particular method is applicable only to 'TCP connect' scans. Other types, such as Fin/Syn/Xmas/Null scans, and UDP scans, are dealt with elsewhere. This procedure will also detect Syn Flooding' attempts also, as this attack is initiated by partial opening of TCP connections using a Syn Packet. Syn flooding attacks on the gateway itself will also show up in the 'connection status' log.

[62] However, once the same host attempts connections to more than a single service, it is then 'handed over' to the threshold based detection scheme.

[63] One example of such a gateway is a Windows 2000/XP system running MS Internet connection sharing. This setup is similar to that often used by Unix/Linux systems, which is running as gateway. My own network is not set up in this way, since I am using a hardware gateway.

[64] For TCP, this occurs when a 'Syn' packet is detected.

[65] This assumes that the IDS is tested on a sufficiently large network. The law of averages dictates that the more entities are present in a group, the more statistically accurate the data.

[66] It is important to ensure that the conditions are otherwise identical, for example, the configuration is the same in both cases.

[67] In any event, the fact that the two programs are running under different operating systems makes such a comparison somewhat open to speculation, since Windows NT and Unix have an entirely different architecture.

[68] The Echo Protocol is specified in RFC 862; see http://www.faqs.org/rfcs/rfc862.html . This can operate over either TCP or UDP; it is not to be confused with ICMP Echo.

[69] This is based on the Unix Posix Regular Expression Library. This was required since Microsoft Operating systems are not Posix compliant.

[70] The problem inherent in the use of a 'linked list', is that the addition or removal of entries involves the re-arrangement of the whole list, involving frequent function calls and consequent usage of CPU time.

[71] The use of incoming traffic and responses is utilised using an exclusive OR, since this method allows for the quickest detection where both are present; whereas incoming traffic alone implies the presence of non-existent hosts or 'stealth' firewalls, which prevent responses. Responses alone indicate 'third party' traffic, resulting from a scan of a third-party system using your own IP address as the apparent source of the attack.

[72] Windows 9.X systems were, in no regard, secure. Not only do they lack any means to protect against network-based attacks; but also the file system offered no security either.

[73] Most systems connected to residential broadband networks are scanned several times daily.

[74] Although online banking facilities, for example, are intrinsically secure, the banking organizations are currently either unaware of this issue, or are intentionally ignoring it since they are unlikely to be legally liable for losses which are not the result of their own acts or omissions.

[75] Generally, one of the source addresses must be correct, if the attacker is to obtain any responses to the scan! However, in the case of denial of service attacks, the address is always falsified, as no response is required.

[76] Since the most common login failures are due to the user inadvertently responding to the username prompt with their password.

[77] The term 'Malware' in this context is a general term for software, which was written with dubious intent.

[78] www.pentasafe.com

[79] In practice, such a program would run as a Windows system process, started automatically at boot time. However, for testing and demonstration purposes, I implemented IPGateKeeper as a console application. Console applications are not synonymous with MSDOS applications (they have access to Windows operating system resources, such as dynamic link libraries), but have lower overheads than GUI applications.

[80] RFC1918 defines a set of IP address ranges, which may be used for constructing 'private internets', that is networks which are strictly local to an organization and whose traffic is not routed over the global Internet.

[81] This is due to the fact that a hub only monitors the data link layer, and does not have any way of knowing which machine is which. This implies that it has to rebroadcast all traffic on all ports.

[82] Broadcasts, in general, are resource-intensive since all hosts have to process each packet, up until the OSI layer at which they are addressed. For example, a UDP broadcast such as a Netbios Name broadcast, must be processed all the way to the UDP layer before a host may discard the packet if not ‘relevant’ to itself.

[83] As a network switch effectively operates at layers network and transport layers of the OSI model, it is able to recognise IP and MAC addresses, and to route traffic accordingly.

[84] Note that although network address translation is used, in practice, the IDS is intended to run on the gateway itself, in this case, 192.168.0.1. For testing purposes, the system running it may be placed in the DMZ.

[85] In practice, some hosts and intrusion detection systems do not validate the packet checksum, thereby not adhering the RFC’s.

[86] This compression could be implemented either by the IDS itself, or by the operating system. As IPGateKeeper runs on Windows NT, the standard NTFS file system compression could compress the data. However, this increases disk access times.

[87] The fact that I have been outputting some sample records to public web pages was for testing purposes; even so, I needed to be careful to ensure that the protocols used for demonstration either contained no important data, or else were encrypted, examples include SSH and HTTPS.

[88] Some services, notably videoconferencing, cannot operate over NAT, as there is no direct return channel to the host system. In addition, hosts in a private network cannot receive incoming connections from global hosts, except where they are placed in a DMZ, or where port forwarding is in use.

[89] Software packages are available for the detection of ‘rouge’ devices. An example of such a package is ‘ArpWatch’, which detects unfamiliar MAC addresses.