Honours Report

Active Network Security

Theuns Verwoerd

5 November, 1999

Supervisor: Ray Hunt

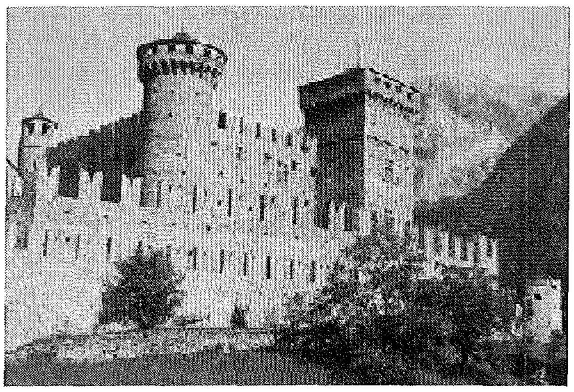
Abstract

Most discussions of network security focus on the tools and techniques used to fortify networks: firewalls, biometrics, access controls, encryption. This paper presents an outline of tools that assist an administrator in verifying and maintaining the security of a networked system – Active Security tools. It discusses why there is a need for such tools and how security mechanisms are attacked. The report also describes the main tools available in this field, with particular emphasis on Intrusion Detection tools – how they work, what is available, and how they are changing. Finally, it demonstrates some of the concepts in a practical firewall network simulation.

Keywords: Network Security, Intrusion Detection, Firewalls, Cryptography, Network Attacks

There are a few people I'd like to thank, since I don't often get the chance. First, my parents – you more or less made me as I am (I just hope to be worthy of your efforts, someday). Next, my God – I have never been alone. Mr. Ray Hunt – who spared no effort to ensure that I had everything I needed (and then some). Finally, all the others who gave their time and energy – properly thanking them would shift the size of this document well into triple digit page counts.

Introduction



A network could be likened to a medieval town; with clusters of houses (systems) connected by roadways, separated by perilous terrain. Like any town, different landmarks can be seen: private homes, businesses, warehouses, stronghouses and fortifications.
The function of this simile is to illustrate two main points. The first is to note that walls do not adapt well with change. As the town grows, walls become constrictive, and do not provide sufficient protection. Secondly, while walls make frontal assault more difficult, they do not prevent all attacks – and unmanned walls do not prevent burglary. In networks, Active Security takes the role of a policing force: ensuring the integrity of the town's defences, while curbing excesses by the town's residents.

Computer networks are becoming fundamental to the functioning of modern organisations. As the dependency on networks increases, the need to control networked resources becomes increasingly critical. At the same time, networks are becoming ever more valuable – in terms of their function, the resources they offer, and the information they contain. In this way, they become not only more valuable to an organisation itself – they also become an attractive target for hostile parties (both in and outside of an organisation).

The concepts of protecting assets are not new; security of physical assets is a well-developed part of any organisational structure. With the rampant growth of internetworks – typified by the Internet – the logical assets of an organisation are increasingly exposed, however. It is now possible for an attacker to penetrate a system, steal or vandalise a company's most valuable assets, and leave – all without leaving any physical trace.

In the past decade, a wide variety of security mechanisms have been developed, aimed at safeguarding the logical assets of an organisation: access controls, firewall technologies, encryption and cryptographic authentication, biometrics and the like. These measures have one common factor in that they attempt to prevent unauthorised access to resources – they could be likened to the locks and secure doors used in physical security. What is missing is a responsive element – the security guards, monitoring and alarm systems present in physical security structures.

Active Network Security is comprised of a number of techniques that address this shortcoming. The goal is not only to reduce the number of successful abuses of a system, but also to give early warning of abuses in progress. Finally, the objective is to ensure that misuse of the system does not go unnoticed – that, should all of the security mechanisms fail, a record exists to allow corrective action.

The mechanisms involved fall into two main groups: those aimed at inspecting a system to ensure its security, and those aimed at monitoring a system in use. The first group, that of System Verification Tools, is discussed in Section 8. The second, that of Intrusion Detection Systems, is the focus of much of the remainder of this report.

1.1 The need for active network security

As noted, a wide variety of security tools and mechanisms are currently available. This begs the question: Why is there a need for active security? In order to answer this, let us consider the 1999 CSI/FBI Computer Crime and Security Survey [CSI99][1].

The survey (dated March 1999) was conducted over 521 US companies from a range of industry sectors, with sizes ranging from under 100 employees to over 10,000. These companies had a variety of security structures in place, as shown in Table 1.

Table 1 Security Measures in Place

Access Control	89%
Biometrics	8%
Encrypted login/sessions	44%
Firewalls	88%
Physical Security	88%
Intrusion Detection	40%

In spite of these measures, 61% of these companies reported experiencing unauthorised use of their computer systems. Twenty percent did not know if their systems had been abused. While 30% of organisations reported outside penetration of their systems, 55% reported insider abuse. Many of the organisations were unable to quantify their losses due to intrusions – for the 163 organisations that were, the total losses exceeded US$123 million. Clearly, in spite the presence of security mechanisms (with the vast majority of organisations having access controls and firewalls in place), abuse of systems continue – sometimes without the organisation even being aware of the breach.

As a specific case, consider an organisational Web site. As an organisation's most visible Internet system, these have long been favoured points of attack. With the development of e-commerce and the increasing use of the Internet as a source of information, an organisation's Web site is developing a significant commercial value. By the same token, attacks on these sites could do significant harm to an organisation – in loss of revenue, loss of customer confidence and damage to information systems.

A good illustration of the risks involved is the "Solar Sunrise" attacks on US government sites [Herald99] [CNN99]. During this series of attacks, a wide variety of web sites were defaced or disabled – including such sites as the FBI, the US Army main Web site, a number of government departments and universities, the US Information Agency, and the US Senate (twice).

Returning to the CSI/FBI survey, it is notable that 94% of those organisations have Web sites (29% offering electronic commerce via these sites, for annual revenues of US$617 million). Again, figures on the abuse of these sites are startling: 18% report abuse – while 30% do not know.

Clearly conventional, static security mechanisms such as firewalls are incapable of offering complete protection (discussed in more detail in Section 2). Active Security mechanisms such as Intrusion Detection should have a place in any secure network.

1.2 Active security mechanisms

Active network security, as described in this document, encompasses networking tools and systems that allow system administrators to observe, inspect and improve the security of their networks. Many conventional security mechanisms are effective in enforcing security in a system, but lack the responsiveness necessary to maintain security on an ongoing basis.

In recent years, a number of security tools have been developed that may best be classified under this heading: while these tools often have no direct effect in preventing misuse, they allow administrators to improve the overall security of their systems. Examples include:

[1] Concern is often expressed on the application of such surveys [ICSA98-2], [CSI99]. Clearly, a survey such as this will not completely model the real world. The figures quoted should therefore be considered as an approximate *lower bound* on the true problem.

- Intrusion Detection Systems (IDS) – Intrusion Detection Systems monitor the state of a system, attempting to recognise and report improper behaviour. These systems protect a network in much the same way as security cameras protect buildings: by letting security personnel keep an eye on what is going on.
- Network Security Scanners – Security scanning systems inspect a network or host system, looking for known weaknesses and possible misconfigurations. The best known example is probably the Satan system – it scans hosts and connected networks for a specific series of weaknesses, reporting any found, and suggesting solutions.
- System Integrity Checkers – Many of the ways in which systems are attacked involve changes to the host's software and data. Integrity checkers compare the contents of a system to a known safe state – allowing administrators to know exactly what has been changed.
- Honeytrap systems – If an IDS is a security camera, this is a burglar alarm; systems whose sole purpose is to be attacked. By closely monitoring these systems, network administrators can observe attackers in action – allowing them to repair, learn and strengthen security against future attacks.
- Special purpose tools – Specific tools have been developed to address security weaknesses present in systems. While not as generally applicable as those listed above, still deserve a place in every security administrator's toolkit. In Section 8, we will touch on two examples: password cracking systems and sniffer detector software.

In a world where security mechanisms were infallible, none of these systems would be necessary. In fact, none of these systems can, in itself, prevent an attack from succeeding. The function of these tools is to minimise the effect of an attack, mitigate resulting damage, enhance the effectiveness of other mechanisms, and ensure that future similar attacks do not succeed.

1.3 The History of Intrusion Detection

The subject field of Intrusion Detection is generally considered to have originated with a 1980 technical document by James Anderson [Anderson80]. In this he proposed a way in which audit information could be used to identify abuses occurring in systems – the original anomaly detection concept. In 1987 Dorothy Denning published a paper [Denning87] presenting a model of how an anomaly detection system could be implemented – a model that was applied in the IDES system. Interestingly, this paper also mentions the possibility of misuse detection – but discards the possibility as requiring too much world-knowledge.

Over the following years, a number of IDS tools were developed: IDES (1988), Haystack (1988), Wisdom & Sense (1989), ComputerWatch (1990), Distributed IDS (1991), Network Security Monitor (1990 - the original network-based IDS), USTAT (1992), IDIOT (1995), NIDES (1995), EMERALD (1997), Bro (1998), AAFID (1998) and Graph IDS (1999). In addition, a number of commercial offerings have been released such as Netranger [NetRanger99], Network Flight Recorder [NFR97], BlackICE Defender [BlackICE99], and numerous others.[2]

In spite of the large body of research that has been compiled in this field, Intrusion Detection is only starting to reach maturity. Many of the systems and techniques developed remain academic exercises, and a number of issues remain to be addressed.

1.4 Structure of this report

This report attempts to give a brief outline of the basic concepts and principles involved in applying active security mechanisms to a network. The first section, the introduction, briefly outlines the need for network security, and gives background information on some of the basic concepts.

Section 2 gives a description of the dominant types of static security mechanisms currently available, and describes why these tools do not address all of the security needs of modern networks.

[2] Michael Sobirey maintains an extensive list of IDS systems at http://www-rnks.informatik.tu-cottbus.de/~sobirey/ids.html, currently listing some 80 different systems.

Section 3 focuses on the attackers: who they are, how they attack, and what tools and attacks an administrator might expect to see used against his or her networks.

Section 4 describes a number of issues regarding security policy that are especially pertinent in the context of active security: aspects of the security policy that support active security, intrusion response policy, and the need for a dynamic review process in network security.

Section 5 is the first section that focuses directly on Intrusion Detection. In this section, the basic concepts underlying modern IDS techniques are described: anomaly and misuse detection, host vs. network IDS, sensors, monitors and distributed IDS.

Section 6 applies the concepts from Section 5 to a number of current and historical IDS systems. It describes how they work; strengths, weaknesses and innovations in their design; and where they fit into the framework outlined.

Section 7 gives an overview of the standardisation attempts currently in progress for Intrusion Detection; the Common Intrusion Detection Framework, the IETF Intrusion Detection Working Group, and the Intrusion Detection Systems Consortium.

Section 8 discusses Active Security tools: it outlines the main categories of tools available, with specific descriptions of some of the most popular examples..

Section 9 applies the principles described in this report to a specific example: the effect of a firewall in the interaction between network security scanning tools and intrusion detection. Some of the problems with static security are illustrated – specifically with the use of firewalls. We also demonstrate the effectiveness of specific Active Security techniques.

Section 10 contains a summary of the main points covered in this report, a few thoughts on the state of Active Security, and possible directions for future work.

Section 12 – the final section of this report – contains the bibliography an list of references for this report.

2. The Limitations of Static Security

2.1 What static security mechanisms are available[3]

Authentication. The core of many current security mechanisms, authentication encompasses the technologies used to identify and verify the authenticity of users, network components and processes. This ranges from simple password based schemes through to biometric and cryptographic mechanisms. The ultimate goal is to uniquely associate an entity external to a system with an identity stored inside the system. In most systems, this is done by requesting some identifying information from a client, for example a password, biometric reading or response to some challenge. This information is then verified against information held inside the system. Should the identifier and stored information match, the user is authenticated; otherwise the user is denied. Extensions of this scheme include the addition of timing or locality information in the identification data, and encrypting the dialogue – all aimed at making the synthesis of a counterfeit identification token more difficult.

Cryptography. With the recent increase in dependence on shared resources, especially public networks, the security of information in storage and transit has become a concern. Strong authentication may prevent active use of restricted resources, but passive interception of information can be as great a risk. In addition, where information is held in an untrusted system, ensuring that data remains unchanged in transit is also a concern. Cryptographic techniques are becoming increasingly prevalent in resolving these issues: ensuring that only authorised users can interpret sensitive information (encryption); and ensuring that vulnerable information is communicated intact (authentication) [Schneier96].

Encryption is the process of applying a transformation to data that can only be reversed using secret information. Depending on the application, one of two forms of encryption may be used: secret-key cryptography, where the transform and its reverse make use of the same secret, and public-key cryptography, where the encrypting transform does not require the use of secret information. Public-key cryptography bears a close resemblance to the authentication problem: a user may be defined as anyone capable of reversing a given transform, thereby authenticating a communication partner.

Cryptographic authentication involves the derivation of a message signature from a message, based on the use of secure hashing techniques. Should the message be modified in transit, the signature and resulting message will no longer match. In order to ensure that the message signature is not modified, encryption techniques are used (restricting the set of users capable of generating a message to those sharing a specific secret). In the case of a modified message, it is infeasible to generate a new encrypted signature that would decrypt to validate that modification. Therefore, if the signature matches the message, it is unlikely that the message was changed or counterfeited.[4]

Access controls. Authentication verifies the internal identity of external parties. Access controls define which resources those parties have access to – limiting the capabilities of those users. These controls are no stronger than the authentication mechanism underlying them, and have potential weaknesses independently of authentication failure.[5]

Firewalls. While firewalls could be considered a specific application of the mechanisms described above, they form one of the main pillars of current network security, and merit separate consideration. The function of a firewall is to separate networks with different security needs and policies – in the most general case, to separate the internal, controlled network and any external public networks. Effectively, a firewall acts as a filter on network traffic – controlling what goes into, or comes out of, a network.[6]

---

[3] A few good references on these subjects include [Sandhu96], [Harris98], [Schneier96] and [Siyan95]
[4] See [Bellovin96] [Schneier98] and [RSAFAQ] for details on how these methods can be attacked.
[5] See [Tanenbaum92] Section 4.5 for more details on different models of Access Control.
[6] Full information on the techniques and implications of firewalls can be found in [Cheswick94], [Chapman95], [Siyan95] or [Hunt98].

2.2 What do static methods offer

The static methods described here, perfectly applied, are effective in ensuring the security of any network. Even in realistic environments, static security mechanisms are capable of significantly improving the security of networked resources.

- Static mechanisms can increase the security of networks in the context where they apply.
- These mechanisms can increase the technical expertise and resources required to compromise the security of a network.
- Static methods can reduce the range of attacks that Active Security mechanisms must deal with.
- Static methods can combine with Active methods to provide a synergetic improvement in security.
- Static methods can prevent attacks from succeeding.

2.3 The limitations of static security

In spite of the wide variety of security mechanisms available, intrusions continue to occur. Based on this fact, a number of limitations in static security mechanisms can be identified:

- The protection offered by these mechanisms is limited in scope. While these mechanisms may be effective in the context in which they are applied, they do not offer universal protection. For example, firewalls, while being effective against external attack, offer no protection against internal abuse – which, as shown in a previous section, is a significant risk factor. The same type of argument applies to other mechanisms: authentication is vulnerable to trust networks, where the authentication mechanisms are bypassed. Encryption only protects information while in an encrypted form. All of the current static mechanisms can be bypassed, negating their effect.
- The security mechanisms themselves are sensitive to technical and implementation problems. Such systems can become vulnerable due to theoretical advances (such as the DES encryption standard, which can no longer be considered completely secure [RSAFAQ]), or poor implementation (for example Microsoft PPTP [Schneier99]).[7]
- Even if theoretically sound and correctly implemented, security mechanisms must be correctly applied in order to be effective. [Gula99] describes an organisation that had its web server defaced – while their firewall was hidden deep inside their network, acting as a log server. Many of the security mechanisms available are very complex (both in structure and in application), and a single mistake may be enough to nullify the efficacy of the system. An example of this is the use of dial-in lines allowing direct access to a trusted network. No matter how good the firewall blocking official connections to that network is, it is still vulnerable.
- Static security mechanisms, by their very nature, are prone to silent failure. Often, the first sign that your security has failed comes when it is far too late (such as when an entire server is wiped clean – an effective method for an intruder to erase a history of his actions). Even when a system's security has not yet been penetrated, that may lead to a mistaken sense of security. In general, these mechanisms also cannot recognise when they are under attack – at best, an attack is logged as a series of failed transactions.
- Associated with the previous point is the issue of remedial information. Once a failure is identified, it may be difficult or impossible to trace the cause of that failure. Information on the identity and methods of an intruder may allow the effects of an intrusion to be mitigated – but none of the mechanisms described offer any such capabilities inherently. The audit information collected by some tools, while being useable, does not have sufficient detail to allow this type of diagnostic[8].
- Finally, the security mechanisms can themselves be subject to attack. Authentication servers can be corrupted, firewalls crashed or circumvented, and cryptographic distribution channels can be compromised. In many cases it is a simple exercise to disable system by attacking its underlying infrastructure. A good illustration of this is the number of tools that are freely available, aimed at allowing users to circumvent the restrictions applied by security mechanisms – anonymous proxies, network tunnelling applications and the like.

[7] An entertaining review of insecure security is available in [Wietse96].
[8] See Section 9 for an example of the kind of audit information offered by our testbed firewall.

The essential problem with many of the mechanisms listed above is that they are essentially passive. While this may be sufficient for a degree of security, it does not hold up in the imperfect world of modern networks, where network administrators are often over-worked, do not have the necessary specialised skills, and where the attacks on networks are ever-escalating in complexity and intensity.

3. Outline of an Attack

3.1 Sources of attack

Script Kiddies: This is the name given to the masses of relatively unskilled hackers that use the tools written by others, without necessarily having any real skill. They are typified by having endless time to spend probing networks for victims to their latest exploit tool[9] – it is on these that the common perception of hackers is based.

This is not to say that they do not pose a risk, however – far from it. These hackers often have an array of tools available, and keep up to date with the latest new exploit software that becomes available. In addition, since they often have no specific aims in mind (beyond the trophy of having hacked a system), they will not necessarily target the most visible or valuable machines – obscurity is no defence.[10]

Employees: Possibly the most dangerous group of potential attackers are the very people who use the networks every day – the staff. They know what in a network is of value, what defences are in place, and have a ready foothold from which to escalate their control. It is a telling statistic that, in the CSI/FBI survey [CSI99] discussed in Section 1, 86% of respondents consider disgruntled employees as a likely source of attack (compared with 74% for independent hackers). Also, recall that 55% of respondents reported inside abuse of their networks.

Mistakes: Not all anomalies in your network have hostile intent. Many "attacks" might be result from a lack of user expertise or from simple user error. This is does not imply that such errors are not dangerous: the case of the 1980 ARPAnet collapse [RFC789] is a clear example of how devastating a simple mistake can be.

Automated Agents: This category includes such things as worms (such as the infamous 1988 Internet Worm[Spafford91]), automated hacking tools, viruses, and trojan software. There does not need to be a human active in order to attack systems – a good example of this is the recent Melissa [Melissa99] macro virus. With minimal modification, the Melissa virus would be capable of sending whatever document is being worked on to an email address – effectively leaking information.[11]

Expert hackers: A number of expert hacker groups have been in the media over the past few years – as government witnesses, software developers [cDc98], and network security experts [Schneier99]. These groups do not merely use exploits written by others; they produce tools of their own[12]. They constitute the highest skill level that network security will be faced with; an administrator can expect to see completely new attacks, if any signs remain at all.

The reason behind a given attack may differ wildly: recreation, industrial espionage, fraud, and attempts by foreign governments to destabilise national infrastructure have all been proposed as causes for intrusions. [Joyal96] [Kyas97 Chapter 2] [Law&Net99]

To place this discussion into context, consider some specific reports:

- "However, the hackers of the cases on which this paper is based are known. All of them were male, and computer science students doing their master's. They all had access to the Internet, and were reasonable well acquainted with UNIX. All of the hackers, except one, had the level of an ordinary UNIX programmer with a little bit more understanding of network software" [Doorn94]
- "A sixteen year-old from the U.K. entered a plea bargain and paid a $1900 fine while another twenty-two year old pled not guilty and was acquitted on all charges in February 1998. The 16

[9] Getting hold of such tools is surprisingly simple – for example, see ftp://technotronics.com.
[10] See [Remsing96], [Doorn94], [BlackICE99] or [Shadow98] for descriptions of attack patterns that could be expected.
[11] See [Kyas97] for the differences between worms, viruses and trojans.
[12] For example, [Hobbit97]

year old was operating on a home computer in his parents' house and had a "C" grade average in his high-school computer class" – Rome Labs, March 1994 [DOD99]
- "The attackers were two teenagers from California and one teenager from Israel. Their motivations were ego, power, and the challenge of hacking into U.S. DoD computer systems." – SOLAR SUNRISE, February 1998 [DOD99]

It would appear as if the common preconception of hackers being young, male and bored holds. However, real information is scarce – though a question would be whether experienced hackers get caught.

3.2 Outline of an attack

The process involved in gaining control of a system generally follows a number of discrete stages [Ruiu99], outlined below. One of the aspects that make internal abuse so dangerous is that the attacker can often bypass the early (and from an intruder's point of view, dangerous) stages, and proceed directly to escalating their control over a system[13].

1. **Exploring the target.** The first step in any intrusion is generally to build up an image of what potential targets a network contains. A number of different techniques are available to hackers, including:
    - **Network scanners.** These tools send specially constructed packets to addresses in the range being scanned. Based on the nature of the reply, it can be deduced which addresses correspond to active machines, and often even more information can be extracted: the operating system running on such systems, open ports, and the presence of intermediary network filters (such as firewalls). Detecting such sweeps has, in the past, been relatively simple: they generate a large number of similar events in system logs, within a short period of time. Increasingly, however, more complex tools are becoming effective in obscuring the details of such scans.

      Tools exist that allow scans to be conducted slowly, using only a few packets per hour or day [Shadow98] [ZDNet99], or conduct a scan co-operatively from different source addresses [Coord98]. One common tool, *nmap* [NMap], allows the source of a scan to be masked by generating a number of fake scans (from spoofed addresses), and has a number of stealth scan mechanisms. One of these, a TCP ACK scan (described in Section 3.3), has been found to be effective in penetrating our testbed firewall.
    - **DNS Zone Transfer.** By retrieving all information available for a network from the DNS hierarchy, an attacker can retrieve a list of all externally accessible points for that network. In addition, if the internal DNS servers are accessible externally, an attacker has access to a wealth of information: a map of the host names and addresses of all machines on the network, and possibly even account details for the system maintainer.[14] [Ruiu99]
    - **Tracing the system neighbourhood.** Using the DNS and addressing information and tools such as *traceroute*, an attacker can determine what machines are in a network neighbourhood. Compromising a machine on the external path of a target network, a number of attack forms become available – ranging from simple traffic snooping to TCP session hijacking [Harris98]. Compromising a machine that the target network depends on, such as a DNS cache server, similarly opens the door for attacks on the target network – and that machine may be significantly less secure than the protected network [Bellovin95].
    - **Public Information.** The information on an organisation's external presence can offer a significant amount of information. From the services and formats offered, an attacker can deduce which operating system may be in use, and identify possible weaknesses. From URLs and email addresses, an attacker can deduce machine names, accounts that may have administrative privileges, and naming schemes used. Based on the header information on emails and HTTP requests from a site, an attacker can extract the operating systems used, and a wealth of information on the SMTP structure of a network. In addition, some sites offer

[13] ZD Net's [ZDNet99-2] describes an expert attack on a web server in detail.
[14] An organisation may not even be aware of the transfer if it uses an external DNS server.

details on the systems they run on their web sites – greatly simplifying this step for an attacker.

- **Predictable names.** Host and service names are often chosen to maximise their convenience: using sequenced host names, naming themes, NIS domain names that correspond to Internet domain names, predictable account names and details (e.g. root), and IP allocations based on the service hosted. Any such features allow attackers to make intelligent guesses as to network structures. [Remsing98]

Once an attacker has a map of a target network, an attack may not be immediately forthcoming: such network maps are often stored, distributed, and used at a later stage.

2. **Vulnerability Identification.** The second step in preparing for an attack consists of determining which of the machines located in the initial exploration may have exploitable vulnerabilities. These often take the form of wide sweeps, looking for machines vulnerable to a given attack – often using an exploit script just released[15]. An alternative mechanism is to match the network information from Step 1 against the set of available exploits – picking viable attacks for a specific network.

   Favourite targets for these sweeps are the external and support services offered by a network: FTP, DNS, SMTP and HTTP servers. Recognising these sweeps can be simple, using local knowledge of a network: repeated probes on port 143 (IMAP) (for example), on machines not running mail software is reason for suspicion[16].

3. **Penetration.** The goal of this step is to gain an executing process on the target system. A vast number of exploits are known (with more being discovered every month) allowing an unauthorised user to gain a foothold on the victim host. Examples include server buffer overflows [Spafford91], poorly written CGI scripts[17], system backdoors (such as the BackOrifice trojan [cDc98]), and weak authentication or access control mechanisms [Doorn94]. Section 3.3 discusses some specific examples of well known attack techniques.

   It is this phase that IDS attempts to recognise – therefore it is also at this point that monitoring systems are likely to be attacked. Using denial of service (DoS) attack, or customised exploits, an attacker may attempt to disable the security mechanisms in a network. Alternatively, an attacker would use his knowledge of the organisation's traffic patterns to hide the attacking traffic in normal traffic streams – making filtering and detection more difficult. For example, a CGI exploit disguised as a normal HTTP request is likely to bypass any filtering mechanisms in place (as demonstrated in the firewall experiments).

4. **Escalation.** Once an attacker has a foothold on a system, the next step is to escalate to control over the system. In this step, the goal is to gain sufficient administrative privileges to allow the next step, Embedding, to proceed – or to do damage. This often takes the form of a bootstrapping process: initially, the attacker starts with minimal privileges. Then, using a succession of exploits and attacks, an attacker gains successively greater privileges until he has complete control over the system [Farmer93]. Alternatively, this could be bound to the Penetration step: many services run with extensive privileges, and grant an attacker those privileges when compromised (effectively allowing an attacker to bypass this step – which is why most services run with as few privileges as possible).

5. **Embedding.** Having gained control of a system, an attacker will cement his control over a system, so that later intrusions do not require the dangerous Penetration and Escalation steps to be repeated. This step involves removing all records of the initial intrusion, bypassing or disabling the reporting mechanisms, and building access routes that will allow the attacker to resume control of the compromised system at a later time. This ensures that the attack and access routes are not detected – ensuring that backdoors remain accessible.
   Examples of embedding techniques include: modifying access control files to allow the attacker access (e.g. adding accounts to a system); modifying access control mechanisms so that they do

[15] This is typical of script kiddie behaviour – effective where network security is out of date.
[16] See Section 8.1 or [Spitzner99] for details on how such a detector is implemented
[17] See CERT advisories CA-97.07.nph-test-cgi_script and CA-97.12.webdist for examples

not apply to the attacker (e.g. adding a master password to the login program). Another mechanisms is to place tools that allow rapid escalation into low-privilege accounts (and ensuring that those remain accessible) – these may be harder to detect. An example of this method is the placement of SUID-root command shells (under Unix) – allowing the user to instantly gain complete control over a system. [Backdoor97] A final mechanism is placing a server process on the machine that will accept commands from the attacker – Back Orifice [cDc98] is a good example.[18]

6. **Extraction.** At this point, the attacker has effectively gained complete control over the system. In many cases it is at this point that an attacker would extract information from the system, or attack the information held on the system (such as vandalising a web site hosted from a compromised server). Security systems such as firewalls may no longer hinder an attacker – many techniques exist for communicating invisibly through filtering systems.

7. **Relay.** Once an attacker has completed modifying or extracting information from a system, he will often retain that system for use as a springboard for further attacks. Tracing an attacker backward through the complex interconnected networks available is very difficult – attackers make use of multiple systems to obscure the true source of attack. In addition, tools are emerging that allow distributed attack and scanning of systems – not only obscuring the attacker, but making the attacks harder to detect and counter.
   An emerging trend is for attackers to target home machines permanently connected to the Internet. Such machines often have very low security, and are ideal as staging areas for further attacks. Who would be liable for damage done from such a compromised machine is unclear – what is clear is that systems need protection, whether or not they contain critical resources.

3.3 Typical attack techniques

- **Scanning a network.** The first step in an attack is reconnaissance – finding out as much as possible about the target. Many tools are available for investigating a network – ranging from simple scripts to commercial network mapping tools, to dedicated scanning applications[19]. In essence, these tools send a packets to a potential host, and deduce information about that host from any reply. Mapping a network consists of checking every possible address for that host. In particular, a number of scan types can be distinguished [Nmap99]:
  - **Ping scan**: The simplest form of scan, an attacker sends an ICMP echo request packet to every candidate machine (which is the same way the *ping* tool works). Any addresses that respond are noted as active.
  - **TCP Connect() scan**: Another simple scan, an attacker attempts to open a standard TCP connection to a typical port on the candidate machine (such as the HTTP port 80). Any machine where such a connection succeeds is noted as active. Since many systems log any connection attempts, this type of scan is relatively easy to recognise from standard audit data.
  - **TCP SYN (Stealth) scan**: This scan sends a connect request to every candidate machine (similar to the Connect() scan), but does not complete the connection by sending a final SYN/ACK packet. In this way, the connection fails and does not generally show up in the system logs – hence a "stealth" scan. Since this scan has a similar signature to a SYN flood attack [Schuba96], many security systems now log such occurrences.
  - **Stealth FIN, Xmas, ACK and NULL scans**: These scans all form part of the same family of variations on the SYN scan techniques. Each sends a special packet to a candidate address, deducing whether a port is open or not from RST reply packets (which indicate a closed port). If not reply is received the port is open – or the request lost in transit, such as being discarded by a firewall. FIN scans consists of packets with the FIN flag set, Xmas scans of packets with the FIN, URG and PUSH flags set, and NULL scans of packets with no set flags. The ACK scan consists of packets with the ACK flag set (generally denoting replies), and so are often capable of penetrating firewalls – as demonstrated in section 9.
  - **UDP scans**: This scan consists of sending UDP packets to likely ports on candidate machines – at worst, scanning for any open UDP ports. Since UDP is connectionless, such attempts are

[18] See also [Ruiu99] for details on how these services function *through* firewalls and filters.
[19] An example of this type of tool, *nmap*, is described in section 8.1 and in [Nmap], [NMap98] and [NMap99]

harder to control using filtering firewalls, and may be capable of finding unprotected services and hosts.

Many variations on these scanning techniques exists – including scans using fragmented packets, and scans spread across a long period or a number of source machines. In practice, completely blocking scans is probably infeasible – but may give an administrator early warning of an impending attack.

- **Buffer Overflows**. This is actually rich category of specific attacks, all using similar weaknesses in software. The core of the attack is to pass an unusually structured (often very long) value as a parameter to a system, when it is expecting something else – for example, requesting an FTP server to change the working directory to an extremely long filename. What happens, in general, is that the parameter overflows its storage buffer, overwriting commands that would later be executed – allowing an attacker to have arbitrary commands executed by the remote server. These commands can then be used to do any number of things – typically, creating an interactive shell, modifying access restrictions, or retrieving sensitive information, such as a password list. Refer to [spyrit99] for details on this technique.
- **Open doors and abused trust**. In order to simplify authentication and access control, many systems accept assertions made by trusted systems. For example, the *rsh* series of commands accepts the remote machine's claims to user identity, if the remote machine is authorised to make such claims. This allows a number of attack techniques, based around abusing the assumptions made in such systems. One technique described in [Bellovin95] involves an attacker assuming the identity of a trusted machine, allowing it access to the trusting system. Another is based on the fact that under some systems (such as some Unix variants), users can control which other machines are trusted (using the *.rhosts* file). A common escalation step in attacking such a host is to modify this file, to allow the attacker free access. See [Farmer93] for an example of the process involved.
- **Social Engineering**. This type of attack is one of the oldest, and most effective way of bypassing security mechanisms: fool somebody with the ability to do it for you. Variations range from guessing information based on the attacker's knowledge of the target involved (see [Remsing98]), to impersonating personnel, and more. The only way to protect an organisation is to ensure that it has a sufficiently clear security policy, and that its users are educated – no technical measures can prevent this type of attack. For a good example of how effective this can be, see [Hafner91].
- **Application Attacks**. These attacks depend on convincing an application to do something it was not expected to – overwrite files, execute commands it should not, or give away information that should be hidden. In addition, these attacks are notable since they can often penetrate even the best developed security mechanisms – the only defence is to keep the applications themselves secure. Examples include requesting password files via FTP or HTTP, attempting to overwrite sensitive files via the same, or passing unexpected information to server applications – such as any of the range of CGI exploits available. For a good example of how this type of attack proceeds, refer to [ZDNet99-2].
- **Trojan software**. The problem of computer viruses is well-known; but the techniques used for propagating these programs can also be used to compromise security. A good example is the Back Orifice system – once an infected application is run on a system, it installs a backdoor on the system, allowing the attacker free access [cDc98]. Preventing this type of attack is difficult – it requires user education, and security to be deeply embedded into systems [Schneier99-2].

4. Policy issues for Active Security

4.1 What is Security Policy

An organisation's Security Policy defines and outlines the measures present to ensure that the confidentiality, integrity and availability of systems remain intact[20]. This includes such items as:

- **System review**: What systems are in place and in need of protection.
- **Risk assessment**: What the risk factors affecting such systems are, and how vulnerable the organisation is to harm should one of these risks be realised.
- **General intent**: How the policy is to be interpreted, and how to resolve issues not directly covered in the policy.
- **Measure selection**: A listing of what measures are in place, describing their placement, configuration, and operational parameters.
- **Operational protocols**: What steps are to be taken under specific circumstances, such as system update protocols and change management, intrusion response and general operations.
- **Responsibility allocation and authority**: Who is responsible for specific actions or parts of the systems, and what authority they bear.
- **Security policy information**: When and how the policy is reviewed, where it is kept, and what authority underwrites it

In effect, the security policy of an organisation circumscribes the measures taken by an organisation to ensure that computing systems are protected under operational and adverse circumstances. Two main techniques are generally used to ensure that resources are adequately protected: baseline protection and customised protection [ITSEC99].

Baseline protection implies the application of security mechanisms across the entirety of a system or subsystem, without regard for the specific needs of components. This requires minimal risk assessment, and may offer acceptable security in low-risk environments, but generally will not offer the most cost-effective protection or adequately protect sensitive systems. In addition, certain safeguards may actually reduce the security of a system (in terms of the critical factors mentioned above). For example, encryption improves the confidentiality of systems, but decreases availability. Therefore, for systems where high availability supersedes confidentiality (e.g. internal email systems), the use of this mechanism reduces overall security.

Customised protection is the application of security mechanisms based on a detailed risk assessment, in order to address the particular needs of a system. This ensures the most efficient allocation of resources, and avoids the problem of inappropriate security measures, but requires a more complex assessment of the needs of an organisation. In addition, an incomplete assessment would result in a mismatch between the actual and estimated needs of a system, creating gaps in the security present.

A method that is often used is to combine the techniques described above: using baseline security to increase overall protection, and protecting critical or sensitive systems with custom measures. This offers many of the advantages of both worlds: a common base of protection system-wide, sufficient protection for vulnerable systems, protection against changes in risk patterns, and simplified administration.

Intrusion Detection and Active Security mechanisms lend themselves to both baseline and customised security. Applying these measures system-wide allows the system to be protected against general misuse, but may require significant resources. By optimising the placement and configuration of these tools, it is possible to offer both increased protection for sensitive systems, and more context-sensitive detection, at the cost of general protection. For example, IDS deployment often concentrates monitors in high-risk areas, such as network ingress points (e.g. adjacent to firewalls), or in the presence of valuable resources (such as network server farms) [Medina98].

[20] For more detail on creating a Security Policy, refer to [DSD98], [Hunt98], [ITSEC99], [Kyas97] and [Sommer97]

4.2 The relationship between Active Security and Security Policy

The Active Security tools discussed in this document are capable of being used as part of a baseline security strategy. This is also effectively what an organisation defaults to, when no formal Security Policy is set out. In order to be used to greatest effect, however, these tools need to be deployed and configured with knowledge of the needs and behaviour of the specific systems involved. [Ranum97]

As an illustration, IDS can function on any network or host system, attempting to recognise generally known abusive behaviour (such as invalid network traffic). Such a system will not be capable of recognising misuse, where such misuse does not correspond to anomalous or illegal activity. For example, such an IDS would offer no protection against users attempting to access resources in an inappropriate manner: for example, Joe from Sales attempting to read the personnel database (using a syntactically legal query).

Embedding information from the security policy into such tools can greatly improve their efficacy. To extend the example, if it is known that certain actions are precluded by the security policy, the IDS and other tools could be configured to include this information. Knowing that nobody outside the personnel department can access that database, an IDS could easily detect Joe's attempt. The IDS can report the problem to security personnel – whether this is a case of internal abuse, or Joe's identity has been compromised and abused.

In addition, Active Security tools can only function correctly if they are constantly maintained and monitored. As such, they depend on a security policy that defines how, and by whom, they are to be cared for – these tools rapidly lose their function if they are ignored. As described more fully in the next section, the reporting capabilities of these tools also imply the need for policies to be set out, in order to handle the changing system.

The security policy may also develop from the results gained from Active Security measures. These tools offer rich detail on the security state of a system: which areas are weak, which areas are being attacked, and the general behaviour of a system. This allows the system administration to extract system-specific information on the real security needs of the system, and modify the security policy accordingly. The information gained from these tools can show not only security problems – but also performance, management and configuration problems, and may give early warning of system failures [Tripwire94].

4.3 Intrusion Response Policy

Active Security in general, and Intrusion Detection in particular, is aimed at identifying problems in computer network systems. In order to make effective use of the results of this type of tool, an organisation needs to have policies and procedures in place before an intrusion occurs. This ensures that critical systems are not mistakenly disconnected, that personnel handling an intrusion have authority and guidance for appropriate corrective action, and avoids later problems in the admissibility of evidence gathered during the episode.

In order to ensure that an intrusion has minimal impact on the functioning of an organisation, a number of specific decisions must be made and documented [CERT99][21]:

- First, determine the basic stance to be taken: to protect and restore the system, or to gather information to allow future prosecution and repair. [Siyan95] lists a number of factors influencing this decision; for non-critical systems, allowing an intrusion to proceed (for the time being) may allow entry techniques, and other resources compromised, to be identified.
- What series of actions should be taken in response to an intrusion, and the relative priorities of such actions. Options include notifying a security administrator, documenting the intrusion[22], identifying the point of entry, ejecting (or restricting) an intruder, repairing damage and bringing the system back online. Information on how the priorities of these options depend on the situation should also be present.

[21] For more information, refer to [Siyan95] pg.109-116, [Chapman95] pg.413-434, and [DSD98] section 14.
[22] [Sommer97] includes a description of computer forensic techniques.

- What authority an intrusion handling team has – what actions they may take without further authorisation, what actions they require authorisation for, and how authorisation could be obtained.
- What resources are available to response teams: links to organisations such as CERT, local law enforcement, system vendors and ISPs; administrative staff with specialised skills; tools and documentation available on the system.
- How and when affected systems should be repaired (or replaced) and restored to use: restoring a machine may destroy evidence and leave it open to a repeat of the intrusion, but disconnecting critical systems would have an additional impact.

In order for such policies to be effective, personnel monitoring security should be trained in the prescribed procedures, and should have ready access to the security policy during an attack[23]. [DOD99] In addition, this policy should be maintained on a regular basis, as software and systems deployed change.

A particular problem in responding to intrusions involves ensuring that the evidence gathered during the course of an intrusion is acceptable in a court of law. Intrusion Detection systems, in principle, should provide a solution to this problem. In order for the information produced by an IDS to be effective, however, it needs to be complete, accurate, and have a clear chain of custody. In particular, [Sommer98] notes a number of specific requirements:
- Evidence should take the form of multiple independent corroborating streams, rather than a single unified stream.
- Some form of synchronisation between streams is necessary, typically a synchronised clock record.
- The defence may require disclosure of the complete details of an evidence-gathering tool, including configuration details. With commercial tools and sensitive network information, this may be problematic.
- The evidence would have to be formally "produced" by responsible parties – this includes rules as to the production of information generated on a team basis.
- An IDS would need to ensure that information gathered during an attack cannot be compromised by attackers, drawing its applicability into doubt.
- Where evidence is based on derived data, the raw data must be available for disclosure – possibly requiring the collection of large volumes of system logs or network traces.
- There needs to be a clear "continuity of evidence" from gathering to presentation.

These requirements need to be taken into account when drafting an intrusion response policy, in order to ensure that legal action is possible, if the need arises. For more details, refer to [Sommer97], [Sommer98], [IDS-Faq99] section 3.7-9, [DSD98] section 14, [Siyan95] chapter 3, [Chapman95] chapter 13 or such organisations as CERT[24] and FIRST[25].

4.4 Policy Review

Security in computer networks is a rapidly changing field: tools such as firewalls and IDS have only entered the mainstream in the last few years, and new challenges are emerging on a monthly basis. In addition, the network structure and systems used by organisations continually evolve. For this reason, it is critical that the security policy be reviewed on a regular basis.

For Active Security, the need for regular review is even more fundamental. Firstly, the field is at the forefront of current development, with new tools and problems being developed continually. An out-of-date system offers a false sense of security, as techniques develop that bypass it. Secondly, active security tools have close ties to the configuration and behaviour of the system it aims to protect. In order for changed systems to be offered correct protection, active security systems need to be changed to reflect the change in network configuration.

[23] An example of how an Intrusion Response should work can be found in [ICSA98] Case 1.
[24] http://www.cert.org
[25] http://www.first.org

5. How Intrusion Detection works

5.1 The goals of Intrusion Detection

Intrusion Detection has as its primary goal the detection of abuse of computer systems. The ideal IDS would be capable of detecting intrusive behaviour in progress, notify security personnel of the problem, and be capable of taking independent action to minimise the risk posed by such abuse.

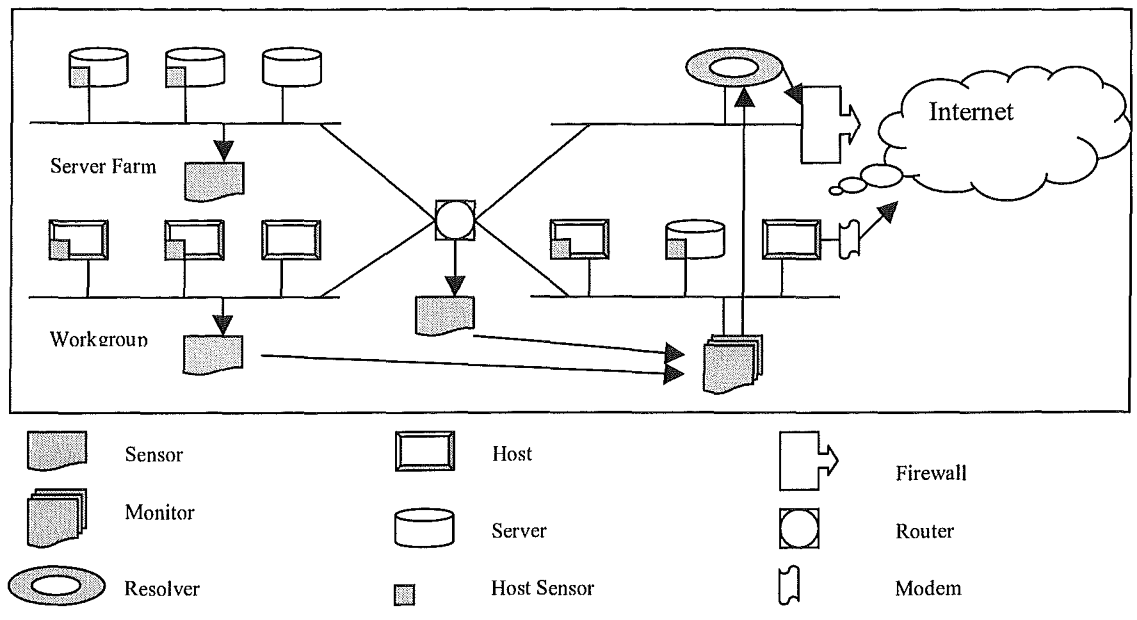
A second, less obvious goal of IDS is to collect data on system behaviour, in order to facilitate recovery after intrusions, identify the source and methods involved in an attack, and serve as legal evidence in the case of a prosecution in the aftermath of an episode.

These goals can be broken down into the following specific points:

- IDS must be capable of accurately differentiating normal or acceptable user behaviour from potentially damaging actions.
- IDS should be capable of scaling across the large composite networks increasingly present in the real world.
- IDS should be capable of handling the complex structures and interaction typical of modern heterogeneous networks, and should be capable of deployment across a variety of network and system architectures.
- IDS should be capable of adapting in response to new attacks and usage patterns, ideally with minimal administrative intervention.
- IDS should offer reports of attacks in real time, ideally as the intrusion is in progress – allowing security personnel to take corrective action.
- IDS should co-operate with other security mechanisms, increasing the overall security of systems. Ideally, IDS should be capable of detecting failures or attacks on other security mechanisms, forming a second level of defence.
- IDS should be capable of responding to intrusive behaviour: by increasing its monitoring in the relevant sections, increasing the security in relevant sections, or by excluding or restricting intrusive behaviour.
- IDS should recognise abusive behaviour in all sections of a system.
- An IDS should protect itself against attacks, ensuring that the integrity of the greater system, and audit information up to the point of compromise remains intact, and ensuring that a compromised or hostile component cannot adversely affect the functioning of the system as a whole.
- An IDS should continue to function in the presence of network failures, unreliable transmission, high system loads, and denial of service attacks.
- IDS should have a minimal impact on normal system behaviour: it should use limited system and communication resources and it should not interfere with legal behaviour. This implies that the level of false positives should be minimised, especially in the presence of response capabilities.
- IDS should generate audit information in a manner and form that is amenable to later use for network profiling and use in the recovery of intrusions – in particular, IDS should generate information in a manner that would allow it to be admissible as evidence in a court of law.
- IDS should reflect the security policy of the organisation in which it is deployed, allowing the priorities of that organisation to shape the level and form of monitoring present.

5.2 An architectural outline

Intrusion Detection systems have evolved from monolithic batch-oriented structures to complex, distributed real-time networks of components. In this development, a basic general model has emerged, allowing discrete functional components to be distinguished. [Debar99]



Intrusion Detection Network Structure

A typical IDS structure consists of the following components:

- **Sensors**: These components form the data-gathering section of an IDS. Sensor modules take the form of monitoring processes on networked hosts (extracting information from the hosts' event logs, audit information, application logs, and general state), or of dedicated network monitors connected to an observation point on a network segment. From there, a network monitor inspects all visible network traffic, synthesising event logs from observed traffic. These systems also filter the event logs, generating summaries that are forwarded to IDS Monitors.
- **Monitors**: The processing segment of an IDS, these systems receive and interpret event summaries received from sensors. These event summaries are then inspected for anomalous or suspicious activity, and suspicion reports are generated. The suspicion reports are forwarded to higher level monitors, or to resolver units.
- **Resolver**: These modules receive suspicion reports from monitors, and are responsible for determining appropriate responses – reporting to an administrator, changing the behaviour of lower level components (for example, increasing sensitivity or logging on subsidiary monitors or sensors) or reconfiguring other security mechanisms such as firewalls.
- **Controller**: In a distributed IDS, configuration of components is possible via centralised controllers. These modules, while not involved in the ongoing functioning of an IDS, simplify administration, and allow administrative personnel to rapidly reconfigure IDS components (for example, increasing the records kept in the case of an intrusion).

The division between these modules in contemporary IDS is often indistinct. Early, single-system IDS had all four components functioning as a single unit. With the development of distributed IDS, however, these components are becoming more distinct. Systems such as GrIDS (section 6.7), AAFID (section 6.8), and EMERALD (seciton 6.9) extend this model by applying these components in a cascading fashion – allowing higher level system overviews to be gained as a user ascends through the tree.

5.3 Intrusion Detection Techniques

Intrusion Detection methodologies can be broken down into two major categories: Misuse Detection and Anomaly Detection. In addition, a number of lesser classifications are possible based on the location of sensors, the nature of events reviewed, the execution timing of monitors, and the correlation of results between resolver units.

Misuse Detection (M-IDS) attempts to match observed behaviour against known intrusive behavioural patterns. A variety of techniques have been used to model and recognise attack patterns, such as expert systems [PBEST99], signature analysis (used in [NFR97], [NetRanger99], [BlackICE99), Petri nets [Kumar95], state-transition analysis, and genetic algorithms [GASSATA98]. A common element between these techniques is that they attempt to represent the essential nature of a known attack in such a way that variations on that attack can be distinguished from normal behaviour. Anything that is not recognised as an attack is accepted as legal behaviour.

In commercial systems, the dominant form of misuse detection used is signature analysis, due to the simplicity of representation and efficiency of implementation possible. A limitation of this approach, and M-IDS in general, is that the signature set requires constant review as new attacks develop. In addition, as more attacks and attack variations become available, the number of rules against which an event stream must be checked becomes larger – leading to scaling difficulties [IDSList].

Anomaly Detection (A-IDS) attempts to model the expected behaviour of objects (users, processes, network hosts and the like). Any action that does not correspond to expectations is considered suspicious. The strength of these methods lies in their ability to differentiate normal user behaviour, anomalous acceptable behaviour, and intrusive behaviour. Techniques used for constructing models include statistical measures (static or adaptive) [Anderson95], expert systems [Frank92], neural networks [Debar92], and user behaviour patterning [Lane97]. Any observed behaviour is compared to known patterns or expected behaviour – large deviations are noted as suspicious.

Few commercial systems currently use this approach – systems using these methods generally stem from academic projects (e.g. IDES, EMERALD). The main reasons for this include:

- System overhead involved in maintaining and checking complex behavioural models.
- Overhead involved in maintaining profiles for every object involved in large systems.
- Difficulty distinguishing valid changes in user behaviour from intrusive behaviour.
- Problems modelling complex heterogeneous systems accurately.
- Generation of large numbers of false positives as models adapt to behaviour changes.
- The ability for attackers to train adaptive models to ignore intrusive behaviour.
- Difficulty in customising a system to take security policy into account.

Location of sensors: IDS sensors are generally either network or host-based. Network based sensors form the mainstay of current commercial IDS products, since they place no processing overhead on network hosts, and no audit or logging requirements for hosts monitored. In addition, network sensors are more difficult to compromise in the event of an attack, and can monitor an entire network segment from a single sensor. [Ptacek98] shows, however, that an intruder can generate traffic that is observed differently by a sensor and attacked host. In addition, network sensors have difficulty in handling modern networking technologies, such as switched networks, high-speed network links, and encrypted communication.

Host based detection avoids many of the difficulties of network sensors, since the sensor is guaranteed to observe the same traffic as the host monitored (bypassing insertion and encryption problems, and scaling issues on high-speed networks). In addition, a host sensor has access to event and audit information local to the host system, allowing localised misuse to be detected. However, host-based monitoring places an additional overhead on every system monitored, requires platform-specific sensors, and may become a liability if a host becomes compromised. In any case, sensor reports from a compromised host are, at best, unusable – at worst, it serves as a springboard for attacking other IDS modules.

Due to the costs involved in monitoring every host or segment on a network, IDS sensors are often deployed selectively, focusing on locations that contain valuable or high-risk resources. Examples include placing sensors near gateways between trusted and untrusted networks (e.g. just inside a firewall), placing sensors near valuable network resources (such as critical servers), or placing network sensors in natural network convergence points (such as on a central router or backbone) [Medina98]. While such limited coverage reduces the capabilities of IDS tools to recognise and trace distributed attacks, this allows a more cost-effective deployment of security mechanisms. Refer to section 5.2 for a diagram illustrating possible IDS component placements.

Monitor processing patterns: The ideal IDS would be capable of detecting all attacks in real time, and offer comprehensive historical summaries. In practice, IDS systems often break down into real-time or batch-oriented systems. Real-time systems, while offering quicker response to intrusions, suffer from performance issues (such systems must be capable of inspecting large amounts of information in real time), and have difficulty recognising complex or distributed attack patterns. The ability of real-time detection to observe and respond to intrusions in progress, potentially preventing or minimising resulting damage, can however be of great value. Most commercial products appear to fall within this group: a notable exception is the Shadow system (refer to Section 6.2). In addition, real-time systems effectively require a sensor and its primary monitor to reside on the same host, due to communication overheads.

Batch or periodic review systems collect event data and inspect the collected traces at regular intervals. Such inspections allow more complex analysis, since a large window of information is available, and do not suffer from many of the performance problems inherent in real-time processing. Since this technique places a delay between the occurrence and detection of intrusive behaviour, it is most appropriate to low-threat environments, and where security personnel are not continually available. In addition, this style of review places a lower processing load on sensor modules, and allows storage overhead (which can be significant) to be centralised. Finally, the availability of historical information surrounding an intrusion may greatly simplify the repair and strengthening of security weaknesses.

Distributed correlation: Many of the emerging IDS products offer distributed correlation of attack results. This correlation ranges from simple composition, where results from different resolver units are presented via the same interface (e.g. Shadow), to hierarchical structures where higher level views of attacks are available (e.g. GrIDS, EMERALD). Composition of results offers slightly greater convenience, but no improvement in the resulting output (e.g. Shadow). The next tier, where a centralised resolver compounds results (e.g. DIDS) allows recognition of distributed trends, but suffers from scaling issues. At the cutting edge, systems are being developed where resolver systems are deployed in a hierarchic fashion, each abstracting lower level information to offer a more general image (e.g. EMERALD, AAFID, GrIDS).

5.4 Capabilities of Intrusion Detection Systems

Intrusion Detection and Active Security mechanisms offer a number of benefits to an organisation:

- Intrusion Detection systems can offer a second level of security for other security mechanisms. In many cases, an attack will target security mechanisms directly. Should the mechanism or the system underlying it fail, Intrusion Detection systems can trigger alerts, allowing the problem to be repaired and resulting damage to be minimised.
- Intrusion Detection systems allow system administrators to form a clearer view of what the true security state of their systems is. Audit trails and system logs often contain valuable information, but are generally in a format that are unusable to all but the most expert of users. As a side effect of their interpretation of this information, IDS can offer comprehensible summaries of this information, possibly alerting operators to problems even before they happen[26].
- Intrusion Detection systems are designed to extract information useful in tracing intrusions. This enables them to identify when system abuse occurs, as well as trace the abuse to an entry point to the system: either to some external network host or, in the case of an internal malefactor, directly to the responsible party.
- In addition to recognising the source of abuse, IDS can often identify the exact nature of that abuse. This allows steps to be taken to repair or mitigate the effects of such abuse, and to update procedures and systems to prevent future recurrences. For example, intrusions commonly include the modification of system files to facilitate future access [Backdoor97], and to erase signs of the intrusion. In addition, the aim of the intrusion itself might be sabotage or alteration of information; such changes can be extremely difficult to identify and repair. IDS, and particularly System

[26] For example, [Tripwire94] describes an Integrity Checker detecting a hard drive failure where even drive diagnostics did not. [Ranum97-2] describes a similar occasion for an A-IDS.

Integrity Checkers (refer to Section 8.3), can simplify this task greatly by indicating which files were, or were not, modified.[27]

- There are a number of complications in using computer-generated logs in legal proceedings. Should the need arise to prosecute an intruder, the data held in IDS logs may be more likely to offer acceptable evidence – particularly if the IDS was designed with this goal in mind. This is discussed in more detail in Section 4.3, and in [Sommer98].
- Intrusion Detection tools may be able to recognise system misconfigurations or failures. Many attacks are based on creating illegal input to systems; other sources of illegal data would also be recognised by an IDS. [Bellovin93] describes a number of examples of such anomalies detected using Intrusion Detection techniques.
- When combined with network security scanners and similar tools, IDS can identify security issues in networks before they become hazardous. For example, finding out that a firewall is vulnerable to a specific attack while configuring security allows early preventative action. Refer to Section 9 for an example of how such tools may be combined.
- IDS systems can help to identify which attacks are being used against your systems, and what system resources are being targeted. This allows system administrators to boost security where it is needed, instead of where it may be needed.
- Every month, new attacks are being discovered. Misuse-detection tools come with extensive libraries of attack signatures, which are constantly being updated. This relieves the system administrators of the responsibility of keeping track of what new attacks might be implemented against them – that is a function of the experts maintaining the M-IDS's signature database.
- Keeping track of the security of a network is a complex task. IDS products have embedded knowledge on network security, which allows less specialised administrative personnel to maintain network security effectively.
- In order to use Active Security tools effectively, the organisational security policy must be well developed (refer to Section 4 for more detail). By offering detailed information on the security status and behaviour of a network, IDS can help in establishing a comprehensive security policy. In addition, many Active Security tools include recommendations giving guidance in formulating and refining a Security Policy.

5.5 Limitations of current Intrusion Detection

Clearly, Intrusion Detection systems offer a number of advantages in terms of network security and management. However, IDS does not offer a complete solution to network security. In particular, there are a number of limitations and problems that restrict the usefulness of current IDSs:

- An IDS cannot stop ongoing intrusions. While an IDS may be capable of detecting an intrusion while it is occurring, it is essentially a reporting tool – it cannot directly disconnect abusive connections. Many current IDS claim the capability of responding and blocking intrusions, but these capabilities generally depend on reconfiguring other security mechanisms already in place (for example, having a firewall block a specific site from access)[28].
- An IDS cannot trace intrusive behaviour in environments with poor authentication and identification structures[29]. Where it is possible for a user to gain anonymity in a system, an IDS might be capable of isolating the intrusive behaviour, but cannot trace back the intrusion beyond the point of anonymity[30]. In addition, many intrusions consist of discrete steps – an IDS may be unable to correlate these steps where these do not have a common source. This may lead to false negatives – intrusive activity going unreported.

[27] [Tripwire94] and [ICSA98] contain examples of how valuable this differentiation can be.
[28] In addition, current IDS systems suffer from false positive and negative results – making automatic responses likely to adversely impact legitimate users.
[29] Pattern-based methods of recognising users, while being researched ([Lane97]), have not yet sufficiently matured to offer a solution.
[30] The Internet is an excellent example of an environment capable of offering anonymity to its users. This may change when IPv6 comes into general use, however. [King98]

- IDSs are designed to collect information on intrusive behaviour, and attempt to trace such behaviour to its source. However, due to the current nature of networking protocols and systems, the best an IDS can generally do is to trace an intrusion to its point of entry into the protected system. In the same way, IDS will attempt to identify the nature of an intrusion. However, by the very nature of the subject, it will often not be possible for an automatic system to fully comprehend the nature of an attack. Therefore, while an IDS is an invaluable tool diagnosing an attack, human specialist knowledge will generally be required for incident handling[31].

- In order to fully protect an organisation, an IDS should be aware of the security policy of that organisation. In particular, every IDS has a particular mechanism for distinguishing acceptable and unacceptable behaviour, originally based on a general, baseline approach. Unless an IDS is specifically configured to recognise specific, additional actions as intrusive (for example, by defining new rules for an M-IDS), it will not flag those actions. For example, browsing through other users' files may be against an organisation's security policy, but it will not generally trigger an IDS response (unless otherwise intrusive action was taken to gain access to such files). By the same token, an IDS cannot function optimally in the absence of a security policy – in these cases, it becomes, essentially, part of a baseline approach to security.

- Attackers are often very aware of the presence of IDS capabilities on a network, and will often directly attack such systems [Phrack98]. An IDS cannot function correctly if the information it receives is corrupted. Should an attacker succeed in disabling an IDS sensor, the system will, at best, retain records up to the loss of contact. A more dangerous scenario is where an attacker takes over and impersonates a sensor: no alert will be generated from losing contact, and an attacker can then feed arbitrary information to the monitor. While IDS protocols and modules are designed to resist attack, the reports of an IDS is only as good as the information it is fed.

- IDSs generally depend on seeing all traffic on a network segment, or all of the event logs for a host-based IDS. With the current increasing use of network bandwidth, it is becoming impossible for any machine to dependably monitor a network link under heavy load[32]. This implies that some parts of an attack may be missed. A similar problem is the increasing use of switching technology in networks – where an IDS sensor would have to be embedded into the switch hardware in order to ensure that it can inspect all traffic[33]. One possible solution to this is to place IDS sensors in particularly sensitive places, or on natural network bottlenecks (such as next to a firewall). In any case, the coverage of the IDS becomes incomplete, and there is opportunity for unreported abuse.

- In order to recognise attacks, an IDS has to model the effect of an event on the systems it is protecting. Particularly in network IDS, the heterogeneity of systems monitored may cause problems. In particular, since different systems respond differently to the same events [Ptacek98], it becomes impossible for an IDS to accurately predict the effect of any given sequence[34]. This implies that an IDS needs to maintain a detailed state of the network it is guarding (which is clearly infeasible), or make assumptions as to the effect of observed events. The end result is that it becomes possible to have effective attacks being obscured from IDS systems – again, leading to false positives and false negatives.

- IDS have problems recognising low-bandwidth attacks. In order to recognise attacks consisting of multiple events, most systems retain state information about recent event sequences. Due to limitations on hardware resources, however, attacks that consist of widely spread events will be ignored. For more detail on a system that attempts to address this problem, refer to section 6.2 on the Shadow/Step IDS.

- New attack forms are continually being discovered. Current IDS systems have limited capabilities for detecting attacks that differ significantly from previously known attacks – exactly those attacks that systems are most vulnerable to. A-IDS may have some success in detecting such attacks, but IDS tools must be updated and maintained continually to ensure that their coverage remains intact.

---

[31] [Sommer98] includes a case demonstrating the difficulty of tracing an attacker – involving a London-based attacker illegally using an exchange in Bogota (Columbia) to contact a Seattle ISP. From a free shell account at this site, a number of military networks were attacked.
[32] According to [ICSA98], the current maximum is 100% analysis coverage at 65 Mbps.
[33] Even there, monitoring may not be feasible: the effective bandwidth of a switch often far exceeds its nominal rates, as parallel independent dialogues are handled. [IDSList]
[34] For example, different OSs handle overlapping fragments differently. Without knowledge of the system involved, and IDS would be unable to simulate the messages received by such a host.

- Finally, current IDS technology suffers from scaling problems. Modern networks are continually becoming larger and more connected, and attacks are emerging that make use of distributed sources, and attacking wider groups of targets. Where the behaviour observed by a localised sensor might not reflect intrusive behaviour, the global picture may be entirely different. As an example, consider a password-guessing attack. An IDS would be likely to notice a large number of failed authentication attempts with a common source. However, making use of distributed authentication systems (such as NIS), an attacker could spread the probes across a wide range of machines and networks. At any given single location, this would not be observed as intrusive behaviour (a small number of failed authentication attempts is generally acceptable), but viewed across the entire network, this should be recognised as an attack.

  Much of the recent research in Intrusion Detection has been aimed at developing mechanisms for sharing intrusion information between a large number of systems. In particular, EMERALD (Section 6.9), AAFID (Section 6.8) and GrIDS (Section 6.6) are all recent systems designed to address this issue. In addition, the current efforts in standardising IDS communications would allow distributed systems to be constructed using components from different vendors.

5.6 Current areas of development

- Distributed IDS: Making IDS technology more scalable, and allowing sharing of intrusion information between a network of IDS modules [EMERALD] [GrIDS]
- Use of Artificial Intelligence (AI) techniques in IDS: Traditionally, anomaly detection has been based on statistical measures and heuristics. With the use of AI techniques, many of the basic limitations in current A-IDS could be addressed: performance problems due to the statistical collection, the choice of appropriate measures to model populations, management of model evolution to allow for change in user behaviour, and user profile folding.
  In addition, AI techniques could be used to improve the pattern matching capabilities of M-IDS, improving the recognition of new and variant attacks, and reducing the dependence on human-generated expert rules.
- Embedded IDS: Security is becoming ever more of an issue on all levels of networking. Building IDS capabilities into network devices, such as routers, switches and firewalls is becoming an issue [Cheung97].
- Application of IDS techniques in non-network environments: Examples include the use of IDS techniques to monitor telephone traffic and credit card transactions. [Anderson98]
- Adapting IDS to new technologies: New networking protocols and products are continually being developed, affecting the functioning of IDS technologies.
- IDS Standards: As IDS technology becomes more mainstream, efforts to standardise the communication between IDS modules, and IDS interfaces, are developing. Refer to Section 7 for more detail.
- Automatic recognition of new attacks: Current M-IDS tools are not capable of recognising new attacks, nor significant variations on existing attacks. Adaptive and AI techniques for recognising previously unseen attacks are still being developed.
- Current IDS have little capability for responding to attacks in progress. There are some efforts to create IDS mechanisms for responding to suspicious behaviour, and automatically act to minimise damage.

6. Contemporary Intrusion Detection Systems and Products

This section describes a number of tools and techniques currently used in intrusion detection. Due to space constraints, some of these tools are only briefly described, but include references to more complete descriptions. The ordering of sections loosely follows the development of IDS tools – from simple manual techniques to complex distributed systems.

6.1 Manual review techniques

In many applications, full-scale IDS systems may not be appropriate. The systems involved may not be especially vulnerable, resources may be scarce, or the scope of the system may make the overhead imposed by IDS systems unacceptable – such as the case for hobby or home systems. There are a number of techniques available for adding detection capabilities to existing systems without the need for additional resources – two of which will be discussed here. Note that the discussion focuses on Unix systems – similar techniques could be applied in other cases, however.

The first method is essentially a specialised form of misuse detection. On a typical system, there are a number of services that will not be in use – for example, IMAP (143) or HTTP (80). Any attempts to connect to such services would be considered suspicious. Techniques based on this approach are described in [Bellovin94] chapter 7, and in [Spitzner99-4].

To briefly describe how this would work, consider a system with the IMAP (143), SMB (139), and HTTP (80) ports unused. By connecting a dummy service to each of these, it appears to an attacker as if this is a valid port. Any connections to these ports trigger a script (or use a similar mechanism to raise an alert), which emails details of that connection to a security officer. This system can easily be refined by extending the fake clients – a dummy HTTP port, that always responds with a fixed error 404 message (address not found), could easily be envisaged.

The second method, taken from [Ranum97-2][35], makes use of the log files and audit information already being gathered on the host. It essentially boils down to a host-based anomaly detection system – where any event not explicitly filtered is reported.

The first step is to set up a list of event patterns that are uninteresting – the method Ranum describes essentially searches the log for distinct messages. It is then up to an administrator to decide which of these messages are not interesting, and define distinguishing patterns for these. These patterns take the form or regular expressions – essentially, expressions that would result in *grep* returning only these lines. The IDS system then consists of scripts that retrieve new log events, and append them to a critical log (if they do not match any of the filters set up). This produces extremely personalised filtering of the event logs – reducing them to a manageable size – and ensures that no events are ignored (unless they are configured to!).

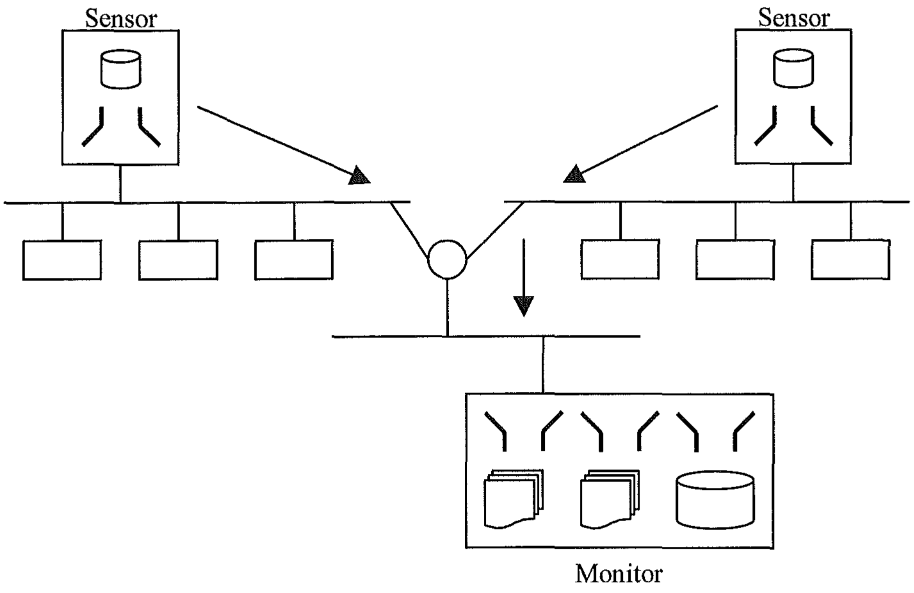
These techniques, while corresponding to many of the typical mechanisms used in heavier IDS tools, can markedly increase the effectiveness of the security already in place on a system – which is exactly the goal of intrusion detection.

6.2 Shadow/Step/CIDER [Shadow98]

The CIDER (Co-operative Intrusion Detection Evaluation and Response) toolkit is a series of public domain tools, aimed at automating the information gathering and traffic analysis components for intrusion detection systems. The SHADOW (SANS's Heuristic Analysis system for Defensive Online Warfare) system is constructed from these freely available components. As a result, setting up a Shadow intrusion detection system involves minimal cost and user expertise, while the system is easily customisable - unlike many of the complex commercial systems described in this section.[36]

[35] [Spitzner99-3] describes a similar method for misuse detection, using *swatch* (ftp://ftp.stanford.edu/general/security-tools/swatch).
[36] The Shadow toolkit and documentation is available from http://www.nswc.navy.mil/ISSEC/CID.

Structurally, the Shadow system consists of a number of Perl and shell scripts layered on top of commonly available Unix tools. It uses *tcpdump* (based on the libpcap library from the Network Research Group as Lawrence Berkeley Laboratory) for traffic capture and analysis[37], *SSH*[38] for secure communications between sensors and monitors, and the *Apache*[39] web server as a reporting tool. In addition, the Shadow package includes specialised scripts to detect low-bandwidth and distributed attacks, and a series of *tcpdump* filters as M-IDS signatures.



The structure of the Shadow/CIDER System

The Shadow system makes use of a series of distributed Sensors, collecting and reducing traffic observed on their local network segments. These sensors essentially consist of Unix machines running *tcpdump*, storing headers from all packets with TCP flags set or directed to TCP port 25 (SMTP). Every hour, a file containing the most recent set of observations is uploaded to the Monitor (using SSH[40]), where it is further processed.

Under this approach, an attacker would be unable to gain much information on the IDS capabilities of the system from a compromised sensor, and the individual sensors require minimal resources. This system is, however, only capable of traffic analysis: content-based analysis would require more information to be forwarded. At that point, network load between a sensor and monitor pair becomes a major issue. Finally, due to the simple nature of sensors, improving the detection capabilities of the system is unlikely to require modifications to the distributed components, simplifying management.

The Monitor is a central Unix system that collects traffic summaries from a number of Sensors. These reports are then filtered using a range of *tcpdump* filters, each of which corresponds to a specific anomalous network event type. Matches on these filters are collected in html files (organised by hour), where they can be reviewed at the administrator's leisure. This review depends heavily on the expertise of the person reviewing: the filters only extract events that are suspicious, but security personnel must extract whether such patterns are significant.

Once specific suspicious activities are identified, an administrator can review the collected traffic traces looking for similar events. For example, should a login attempt originate from an unusual source machine, an administrator can then request for other activity by that source during the past day.

[37] Available from ftp://ftp.ee.lbl.gov/
[38] Available from http://ns.uoregon.edu/pgpssh/sshstart.html
[39] Available from http://www.apache.org/
[40] Should a Sensor be compromised, an attacker may have a path into the Monitor system – due to the way in which SSH is configured in this system – and injecting false reports would be possible.

Similarly, specialised scripts are included to facilitate searching for low-bandwidth or distributed probes [Coord98].

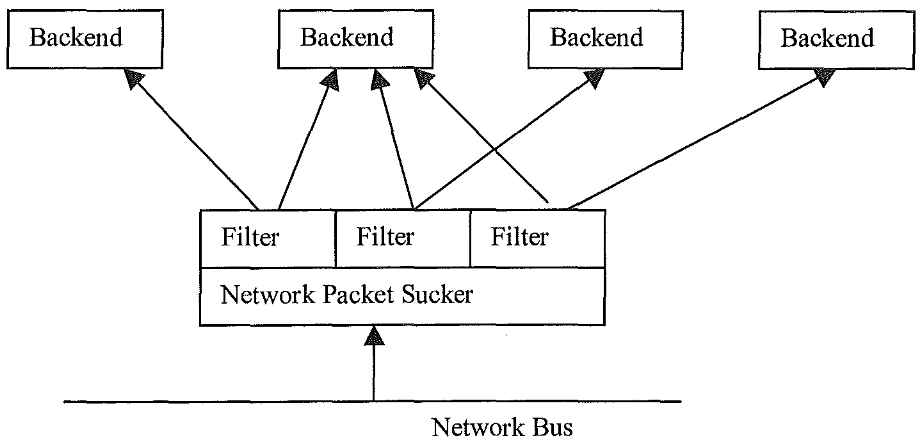
A few comments on this system:

- Shadow uses periodic traffic analysis for intrusion detection (specifically, the first 96 bytes of each packet, by default). This greatly restricts the use of such a system, in that it is incapable of recognising valid (but intrusive) traffic – for example, it would be unable to recognise any of a wide range of CGI exploits. In addition, this system does not lend itself to immediate response to intrusions – on average, results from a sensor will only be available half an hour after intrusive activity.
- The system uses a pull-based reporting structure, where security staff retrieve results whenever they have time. This differs from most current security systems, but makes sense in terms of the periodic nature of analysis. It does, however, require that personnel actively monitor the system, which may not be feasible in small organisations.
- The relatively raw nature of the results produced mean that reviewing personnel have to be extremely familiar with both the network behaviour, and the nature of possible attacks. This effectively restricts the use of this system to security specialists.
- Due to the construction of this toolkit from commonly available components and simple scripts, this tool is highly customisable. It lends itself well to low level analysis of network behaviour.
- This system depends heavily on context-specific recognition of illegal or unusual behaviour. It virtually requires a detailed security policy to be in place, in order to allow security personnel to differentiate legal and intrusive behaviour.

6.3 Network Flight Recorder (NFR)

The Network Flight Recorder network monitoring system (described in [Ranum97], and available from http://www.nfr.net) is one of the most discussed new IDS systems. It combines content-based network monitoring and an efficient filtering mechanism, with some support for distributed review using a centralised concentration point.[41]

NFR is an outgrowth of an in-house series of IDS tools written around the NNStat statistical network analysis tool. It retains much of the essential nature of its origins: it consists of a network monitor based on the libpcap libraries, with a series of filtering and correlation layers built on top. Referring to the previous discussion of IDS techniques, NFR is a pure network monitor – with all the advantages and problems that entails.



Schematic Structure of the NFR architecture

The basis of the NFR architecture is a network packet capture system based on a modified version Berkeley packet filter (BPF) and the libpcap packet capture interface. Captured packets are then passed through a decision engine, consisting of a number of filters. Using these filters, information

[41] Also notable about NFR is the fact that one of its authors, and CEO of NFR, is Markus Ranum – an active contributor to the IDS mailing list, and credited with the firewall phrase "Bastion Host".

from the captured packets is then forwarded to a selection of backends, where the final packet processing occurs: statistical information is compounded, packet event logs are generated, and the IDS state is maintained. TCP stream and packet reassembly occurs in the decision engine, which reduces the complexity of filters.

The NFR architecture includes a number of significant optimisations, allowing it to attain high capture rates without packet loss. The first of these is the customised BPF capture module, allowing minimal operating system overhead when capturing packets. Secondly, the system has a highly modular structure: it is segmented into independent capture, decision engine, and backend layers.

A further significant technique is the use of a compiled bytecode language, named N-code, which is used to specify the filters used to classify network packets. Many IDS systems use some form of decision structure to identify packet types – by implementing these as a compiled language, NFR gains a marked performance boost. In addition, since only a subset of the information for a packet (as defined in the filter) is passed on to the backend, this allows a marked reduction in information retained, and allows the backends to be written independently.

The backends themselves are essentially special-purpose collection units, collating statistical information, keeping watch for specific events, or maintaining a behavioural state for the system. For example, one backend is described as a histogram module – counting the number of occurrences of a given attribute tuple. One possible use of this type of backend would be to collect network usage patterns (which machines connect to which others), possibly as a building block for an anomaly detection system. The independent nature of the backend modules allows them to be developed with relative ease – allowing the system to rapidly develop responses to novel attack mechanisms.

It can be readily seen that the NFR system bears a marked resemblance to the Shadow system described in Section 6.2. A number of differences are notable however. Firstly, NFR is a real-time detection system, processing packets during capture, whereas Shadow is essentially an offline system. Secondly, while both NFR and Shadow are derived from the libpcap library, NFR adds a packet and stream reconstruction layer (used in content inspection). This allows NFR to detect fragmented and application-level attacks, but makes it sensitive to issues surrounding the reconstruction of data streams. Finally, while both Shadow and NFR use filtering rules to extract pertinent events from the raw packet stream, Shadow uses essentially interpreted rules, while NFR uses compiled N-code. Where Shadow does not do further processing on filtered packets, NFR's backends allow the system to be used for network profiling as readily as misuse detection.

6.4 BlackICE

BlackICE is a host-based network IDS, running on a Microsoft Windows platform. Though aimed at the consumer market, it has a number of interesting features (from a technical point of view). These include an extremely simple installation system, back tracing of intruders, and the ability to block intrusive connections [BlackICE99].

Possibly the most interesting aspect of the system, however, is the manner in which it bypasses the problems inherent in many host-based or network-based systems. Host based systems depend on the operating system for event reports – which is often not complete or dependable. Network based systems have performance problems on high-bandwidth links, and have difficulty handling encrypted traffic. BlackICE sidesteps this problem by linking into the protocol stack of the host application – effectively allowing it access to information at any protocol level.

Effectively, BlackICE is a hybrid between current IDS methodologies. By being host-based, it is capable of effectively protecting that single host with a minimal performance cost. Using network information allows it to gather as much information as is available – and, in particular, ensures that the traffic observed corresponds exactly to that observed by the targeted machine.

6.5 BRO

This IDS, the result of a research project, is a good example of the general techniques used in building a network-based monitor. As such, it bears many similarities to systems described previously – so this will be brief. For more detail, see [Bro98].

The Bro system (named after Big Brother from George Orwell's 1984) essentially consists of a network monitor and review engine. Like NFR, it includes an embedded control language for representing policy and pattern rules. It explicitly separates the software engine and configuration rules, allowing it to be customised with relative ease. Finally, it has a number of design features aimed at protecting the system against attack: overloading detection and handling; a monitoring system that restarts Bro if it should crash (and sets off logging of the cause of the crash); and explicit detection of attempts to obscure traffic streams (for example, using overlapping fragments).

6.6 DIDS

The DIDS system [DIDS91] is an early attempt at bridging the gap between single-module IDS tools and interconnected networks – a prototype distributed IDS. It makes use of a distributed series of conventional IDS installations, and a centralised global monitor. This global monitor receives event reports from its subsidiary IDS components, compounding these into a system-wide security state.

The DIDS system also introduced the concept of a NID – a network ID. This is an identifier used to associate actions taken by a single external user, independent of the usercode or identity set by the operating system. In other words, an external login is allocated an NID, and all transaction for that user is associated with the NID. Even if the effective user changes (for example, the user logs into a different account on another machine), it retains the same NID. This allows DIDS to inspect the behaviour of a specific user on a system-wide basis.

One issue with this approach is the recognition of globally significant events. The fewer events that are passed on by point IDS tools, the better the system as a whole will scale. However, locally insignificant events can be notable on a global scale (consider the case of a network scan). This tension is a problem in many distributed IDS models.

6.7 GrIDS

A more recent attempt at a global distributed IDS, GrIDS (Graph based IDS) uses event graphs to model network activity. These graphs are organised into a hierarchic fashion – at a given level, lower level subsystems are represented as single nodes, and the interaction between peers of the same level modelled. By retaining external edges from the system node to other nodes, an inter-system picture of network activity can be formed.

While the system depends on a global installation of the GrIDS system, it solves the scaling problems associated with DIDS and other flat hierarchies. By refining the hierarchical decomposition of systems, it is possible to control the amount of detail available at each level. The retention of inter-system information implies that distributed intrusive behaviour will be detected at an appropriate level, avoiding the loss of detail that plagues other hierarchic distributed systems.

6.8 AAFID[42]

Many of the problems present in the application of IDS can be solved by creating lightweight, specialised IDS components. An example of this approach is the BlackICE system – by monitoring the network activity of a single host, it bypasses many of the problems found in general network IDS.

The AAFID system [AAFID98] takes this approach to the extreme. It makes use of a wide hierarchy of agents (which have a similar function to a single-point IDS – detecting local events and attacks), monitors (which compound information received from agents), transceivers (control units for the agents on each host) and user interfaces.

[42] Copies of the AAFID system are available from [AAFID]

Agents are essentially small detection modules – translating observed events into suspicion reports. These range from the extremely simple – a monitor that checks for single attacks, such as 0-length packet fragments – to full-blown IDS tools. All communicate with monitors via the per-host transceiver units.

Monitors receive event reports from agents, correlating behaviour across a number of agents and hosts, producing event summaries for higher-level monitors, and possibly initiating responses. By applying a hierarchy of monitors, this structure is capable of scaling across large and complex networks.

6.9 Emerald

The EMERALD (Event Monitoring Enabling Responses to Anomalous Live Disturbances) [Emerald97][Emerald99] system bears many similarities to the AAFID structure, described above. Where AAFID uses specialised component architectures, however, EMERALD uses a single component model.

The EMERALD architecture is based around the concept of a generic monitor unit, which includes event acquisition, anomaly and misuse detection subsystems built around a taret-specific translation layer. This allows a single monitor system to be used on a variety of different platforms, and at different levels in a detection hierarchy.

Like AAFID, EMERALD uses a hierarchy of IDS monitor units for distributed detection. A monitor receives event reports from lower level components, and passes event summaries and suspicion reports to higher level monitors.

7. Standardisation efforts In Intrusion Detection

As IDS technology becomes mainstream, the need for standardisation of IDS interfaces has begun to be addressed by a number of bodies. Standardisation would offer a number of definite advantages:

- The ability for users to combine the strengths of different IDS tools from different vendors
- The ability to share intrusion information across different organisations, facilitating detection of widespread abuse
- The ability to develop new IDS methods more rapidly, making use of existing components
- Standardised communications would facilitate the development of distributed IDS systems, a pressing need in modern large networks.

7.1 Common Intrusion Detection Format (CIDF)

An outgrowth of the DARPA Information Survivability program, the CIDF is one of the first IDS standardisation efforts. It aims to present a suite of protocols, application interfaces and communication mechanisms to facilitate interoperability of IDS tools [CIDF]. Specifically, it will present four components:

- **Communication in the Common Intrusion Detection Framework**[CIDF98-3]

  This document describes a matchmaking service for distributed IDS components, allowing a single point of administration and dynamic configuration. Essentially, this is a component that stores addressing and capability information for IDS components. On request, it can return to a component a list of suitable communication partners, based on an identity-based or capability-based lookup, filtered by component categorisations.

  In addition, the specification describes a series of message formats and communication protocols, for use in communicating components. This protocol defines encryption, authentication and security aspects for the communication – the IETF equivalent would be the Intrusion Alert Protocol and associated protocols.

- **A Common Intrusion Specification Language**[CIDF98][CIDF98-2]

  The CIDF CISL defines the format used to represent intrusion alerts communicated between IDS components. It uses an encapsulating tree structure (based on the S-grammar [Rivest97]) very familiar to LISP users to represent different levels of detail. The simplest way to understand this would be to consider a simple example (from [CIDF98]):

```
(Insequence
    (Login
            (Context
                    (Time '14:57:36 24 Feb 1998')
            )
            (Initiator
                    (HostName 'big.evil.com')
            )
            (Account
                    (UserName 'joe')
                    (RealName 'Joe Cool')
                    (HostName 'ten.ada.net')
                    (ReferAs 0x12345678)
            )
    )
    (Delete
            (Context
                    (HostName 'ten.ada.net')
                    (Time '14:58:12 24 Feb 1998')
            )
            (Initiator
                    (ReferTo 0x12345678)
            )
            (Source
                    (FileName (ExtendedBy UnixFullFileName) '/etc/passwd')
```

```
                )
        )
        (Login
                (CriticalContext
                        (ReturnCode (ExtendedBy CIDFReturnCode) Failed)
                        (Comment '/etc/passwd missing')
                )
                (Context
                        (Time '15:02:48 24 Feb 1998')
                )
                (Initiator
                        (HostName 'small.world.com')
                )
                (Account
                        (UserName 'mworth')
                        (RealName 'Mary Worth')
                        (HostName 'ten.ada.net')
                )
        )
)
```

A quick translation of this would be the sequence of actions where:

- 'joe' ('Joe Cool') logged into 'ten.ada.net' from 'big.evil.com'
- 'joe' then deleted the file '/etc/passwd'
- Later, a login attempt by 'mworth' ('Mary Worth') from 'small.world.com', failed due to a missing password file.

Without going into the detail of how the grammar is structured, it is easy to recognise the tree structure inherent in this format. The hierarchic details of entity descriptions allow components to extract only the information needed from alerts – facilitating the use of components with different complexities. The CISL specification describes in detail how these structures are built up, includes a range of examples, and defines a lexicon for use with this language.

- **CIDF APIs: Their Care and Feeding (not yet available)**
- **The Common Intrusion Detection Framework Architecture (not yet available)**

7.2 IETF Intrusion Detection Exchange Format Working Group (IDWG)

The IETF Intrusion Detection Exchange Format Working Group (IDWG) [IDWG99-4] was established to define data formats and protocols for sharing information between IDS systems. In particular, the group aims to define:

1. A list of high-level requirements for the communication between IDS systems, and between IDS and management systems [IDWG99]
2. A common intrusion language and data format specification [IDWG99-3]
3. A framework document defining intrusion communication protocols and their relation to the data format [IDWG99-2]

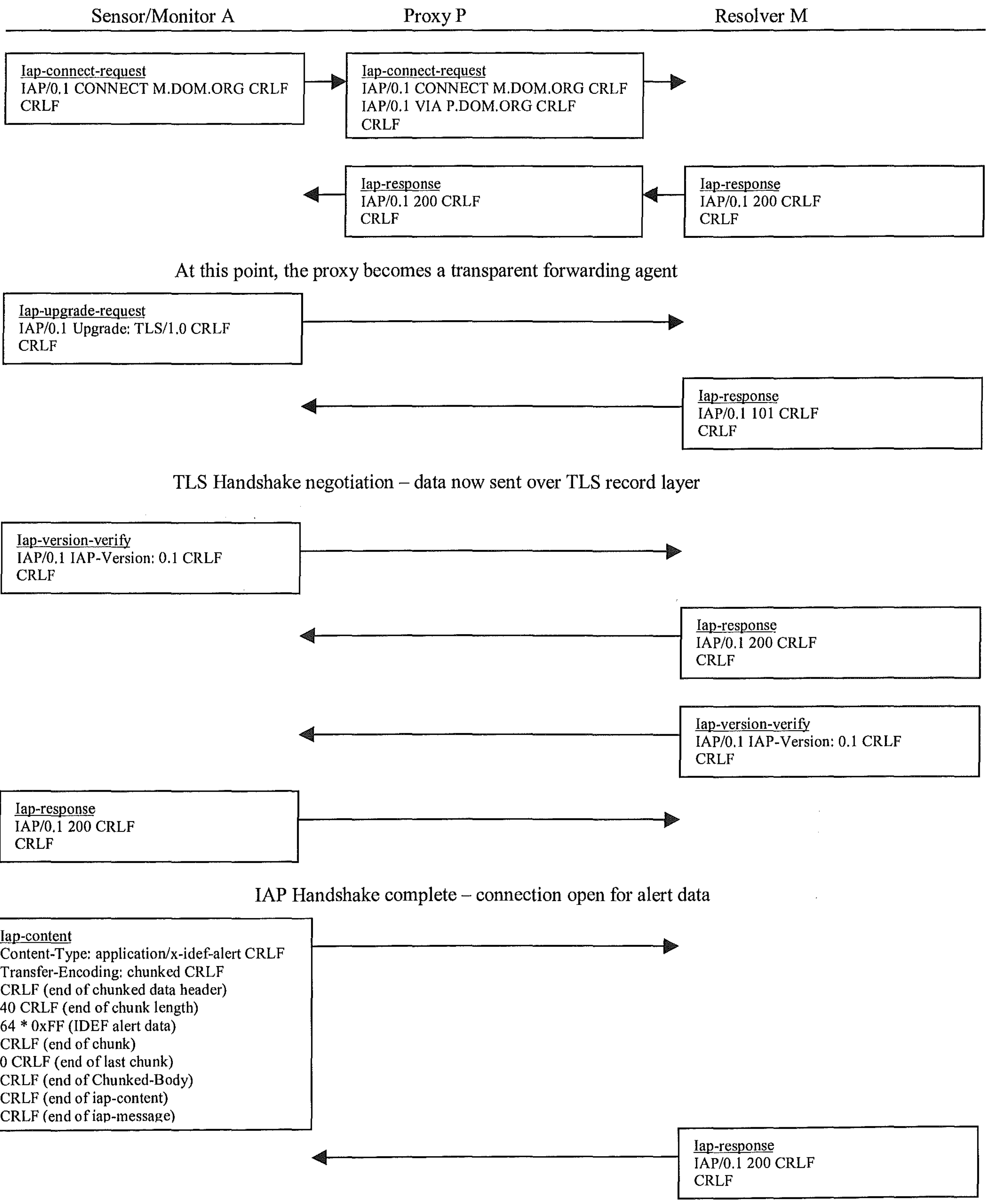
This group has a large overlap with the CIDF project, described above – to the point having a common chair in Stuart Staniford-Chen. The results produced, however, are independent – there are marked differences between the approaches used in the CIDF and IDWG proposed standards.

The output of this group is still very much a work in progress – the IDWG data format draft [IDWG99-3], for example is dated 15 October 1999. In the remainder of this subsection, we will briefly cover the three documents that have been released.

7.2.1 Intrusion Alert Protocol

The Intrusion Alert Protocol (IAP) defines an application-level protocol, running over TCP, for carrying intrusion alert information. It emphasises the reuse of components from other protocols – it uses TLS [TLS99] for link encryption, MIME content types, and an HTTP-like message structure and error codes. A such, the protocol definition is much simpler than the comparable CIDF document.

The simplest method to describe the protocol would be to make use of an example communication setup (taken from [IDWG99-2]). In this example, sensor/monitor A wishes to send alerts to resolver B via proxy P.



Iap-connect-request
IAP/0.1 CONNECT M.DOM.ORG CRLF
CRLF

Iap-connect-request
IAP/0.1 CONNECT M.DOM.ORG CRLF
IAP/0.1 VIA P.DOM.ORG CRLF
CRLF

Iap-response
IAP/0.1 200 CRLF
CRLF

Iap-response
IAP/0.1 200 CRLF
CRLF

At this point, the proxy becomes a transparent forwarding agent

Iap-upgrade-request
IAP/0.1 Upgrade: TLS/1.0 CRLF
CRLF

Iap-response
IAP/0.1 101 CRLF
CRLF

TLS Handshake negotiation – data now sent over TLS record layer

Iap-version-verify
IAP/0.1 IAP-Version: 0.1 CRLF
CRLF

Iap-response
IAP/0.1 200 CRLF
CRLF

Iap-version-verify
IAP/0.1 IAP-Version: 0.1 CRLF
CRLF

Iap-response
IAP/0.1 200 CRLF
CRLF

IAP Handshake complete – connection open for alert data

Iap-content
Content-Type: application/x-idef-alert CRLF
Transfer-Encoding: chunked CRLF
CRLF (end of chunked data header)
40 CRLF (end of chunk length)
64 * 0xFF (IDEF alert data)
CRLF (end of chunk)
0 CRLF (end of last chunk)
CRLF (end of Chunked-Body)
CRLF (end of iap-content)
CRLF (end of iap-message)

Iap-response
IAP/0.1 200 CRLF
CRLF

In the above dialogue, the response codes correspond to HTTP response codes, and many of the other formatting elements were directly drawn from HTTP/1.1 [HTTP99]. One interesting aspect of this standard is the support for transparent proxies, aimed at allowing IAP links to cross security boundaries, such as firewalls. Also, the protocol has a simple method of handling unexpected data, such as TCP urgent messages – the receiver should terminate the connection. This is aimed at preventing masquerading and similar attacks against the IDS structure form succeeding, but may leave it open to denial of service attacks. Finally, the protocol requires the use of mutual authentication during the TLS setup phase – ensuring that an attacker cannot easily inject IAP content into a system.

7.2.2 IDEF Data Model

Unlike the CIDF CISL tree-based model, the IDEF data model is based on an object-oriented paradigm, with inheritance structures defining the associations between similar alerts or attacks. The standard proceeds to define a number of data types and alert classes – the details of which are omitted here, due to space constraints.

7.2.3 IDEF Requirements

[IDWG99] contains a number of specific requirements, aimed at ensuring that the resulting IDS structure is simple to implement, reliable, co-exists with modern networking and security mechanisms, applicable across all IDS methodologies, and will be resistant to attack. These requirements correspond in many cases to items described in section 5.1, and so will not be repeated here.

7.3 Intrusion Detection Systems Consortium (IDSC)

The ICSA Intrusion Detection Systems Consortium was established in 1998 as a forum for commercial IDS developers, to facilitate co-operation in attaining common goals [ICSA]. These goals include creating industry standards, encouraging and enhancing product interoperability, educating users, and maintaining product and marketing integrity [ICSA98].

Notable among the results so far attained by the IDSC is the release of [ICSA98] and [ICSA98-2]. For more details, refer to http://www.icsa.net/services/consortia/intrusion.

8. Tools supporting Active Security

8.1 Network Mappers

A variety of commercial and free network discovery tools are currently available – examples include Cheops (http://www.marko.net/cheops), and *nmap* [NMap]. These tools use many of the same techniques described in section 3 to explore the content of networks: DNS zone transfers, scanning the address and port space, requesting information from hosts found, and promiscuous monitoring of a network. In fact, many of these tools are now used by attackers – *nmap*, for example, was an invaluable aid in inspecting the exact coverage of the firewall policy during our experiments.

As an example of how a typical network mapper works, consider the *nmap* tool. It is a powerful aid in exploring networks – not only because it offers a wide variety of scanning options (including every scan type described in section 3.3), but also due to its unique ability to identify a wide variety of hosts systems, down to the operating system, and sometimes version.

Nmap works by sending packets with a wide variety of special characteristics to hosts being investigated: packets with specific (often illegal) flags set, ICMP echo packets, fragmented packets (again, sometimes with illegal fragmentation), etc. Every host has a particular style of responding to such packets – by combining these response characteristics, it is possible to narrow down exactly what system is present on the interrogated host. In fact, *nmap* uses a signature analysis system which bears some similarity to that used by IDS systems to recognise specific attacks – allowing the tools to easily extend its library of recognised systems.

For example, it is possible to recognise Linux systems with older kernels than version 2.0.35 by the fact that, presented with a packet with the SYN flag and an illegal flag set, these systems retain the illegal flag in their response.

Scanning a network generates a mass of highly anomalous packets – alerting any good IDS tools present – and may have unwanted side effects. Because of the use of unusual traffic patterns, these tools are capable of damaging a network system – certain types of fragmentation patterns, for example, crash specific systems when received[43].

8.2 Network Security Scanners

Configuring networks and network hosts to be secure is a difficult task: validating that such a system is secure may be even more difficult. A single security weakness in a configuration is all an attacker needs: a single weak password, a single outdated server, or a single vulnerable port. Network mapping tools go some way towards allowing an administrator to verify systems. Network security scanners (also known as vulnerability assessment tools) take this a step further – they actively test the security of a system against a number of attack scenarios, reporting on the location, severity, and solution to weaknesses found.

These tools have had a contentious history – from the early COPS system, to the controversial Satan tool, to the current range of freely available toolkits, such as Nessus (http://cvs.nessus.org), Internet Security Scanner (http://www.iss.com) and Cybercop Scanner (http://www.nai.com). Because these tools are capable of automating the vulnerability identification phase of an attack, it was felt by some that releasing such tools encourage script kiddies to attack systems. In practice, similar tools are available in the hacker community – sscan (http://www.ben2.ucla.edu/~jsbach/sscan.tar.gz) being a good example.

Like IDS systems, these tools come in two varieties: host-based and network-based systems. Host-based systems (such as COPS) analyse the security mechanisms in place on a system, looking for possible misconfigurations or dangerous settings. Examples include accounts with weak passwords, excessively trusting systems, and applications with unusual privileges (which may simply be a

[43] For example, see CERT Advisory CA-97.28 for details of two such attacks: Land and Teardrop

misconfiguration, or may be indicative of a past intrusion). This review is generally extremely system specific, but allows a wide range of issues to be checked across many user accounts – a potentially significant saving for overworked administrators.

The second class, that of network-based systems, check hosts for secure networking policies. Tests include weak passwords for well-known accounts, the presence of services known to be dangerous (e.g. NFS available from outside a firewall), and unnecessary services (e.g. NFS without shared file systems). In addition, these tools include libraries of exploits, which are tested against subject systems – checking whether such systems are susceptible to the specific weaknesses. In effect, the tool attempts to break into the subject system – if it succeeds, there is clearly a security flaw.

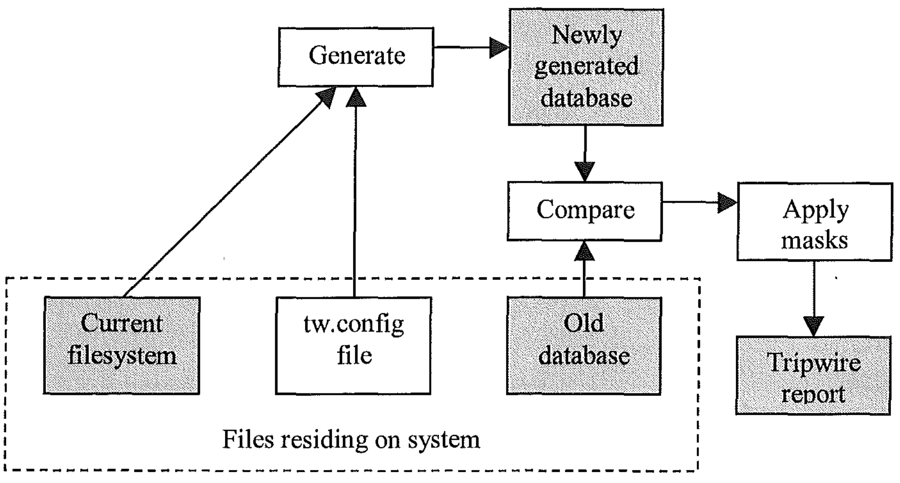
Finally, network-based systems are presently developing mechanisms for reviewing other security systems, such as IDS and firewalls. In particular, these systems can simulate the techniques used by attackers, allowing an administrator to verify that these are blocked or detected by the firewall or IDS, as appropriate.

One issue with such systems that is sometimes overlooked is that these systems must be kept up to date constantly – ensuring that a network is secure against last year's attacks does not offer any benefit against current risks. As the attack techniques used against systems evolve, these system should be updated, and the systems re-inspected.

8.3 System Integrity Checkers

Once a system is compromised, one of the first actions taken by an intruder involves changing system files: to disguise the intrusion, facilitate future penetrations, or support escalation in control over the system. In addition, there is a variety of events that will result in unauthorised changes to system files – ranging from viral infection, unauthorised changes by administrative personnel, or failing hardware.

A tool developed to address this problem is the well-known Tripwire package [Tripwire94]. Written by Gene Kim and Eugene Spafford and released on November 2, 1992, it has since become a standard component in many system administrator's toolkits. In essence, the Tripwire system stores a hashed snapshot of file system features and content, compares this to the current system state, and reports any discrepancies.



Structure of the Tripwire system

As can be seen from the above diagram (adapted from [Tripwire94]), a Tripwire configuration consists of two main components: the Tripwire configuration files, and a previously generated reference database for that system. The configuration files consist of a series of file or directory paths and attribute masks (defining which attributes of a file may safely be ignored), or of M4-style preprocessing commands (similar to those used by the *cpp* C-language preprocessor). Using these features, it is possible to create fine-grained configurations with support for host-specific variations.

The reference database is generated by Tripwire, based on some initial trusted file system. It is important to ensure that this initial generation is done on an uncompromised system – ideally, this should be created for a system after the initial configuration, but before that system is taken into use. Tripwire cannot detect preexisting problems – only changes that occur after its installation.

The security of the Tripwire system is based on a number of factors: the integrity of the Tripwire software itself, the integrity of the reference database, and the strength of the hashing algorithms used to identify files. Therefore, it is suggested that the reference database be stored in a secure location: on a different, secure system, or on read-only media. To minimise the chance of an attacker making undetectable modifications to files, Tripwire supports the use of up to 10 different, simultaneous hashing algorithms (by default: MD5, MD4, MD2, Snefru, SHA, POSIX 1003.2 CRC-32 and CCITT CRC-16 signatures are available). These algorithms offer a range of security/performance features, and the use of multiple signatures increase the difficulty of generating hash collisions greatly.

From an Intrusion Detection point of view, this type of tool is most useful as a last line of defence, and for recovering from an intrusion. These tools will only report changes already present in a system – at which point the attack may be in an advanced stage. In addition, these tools will only report that changes have been made – not what those changes were. For example, one of the first steps in controlling a system is to purge the system logs of evidence of the intrusion. While integrity checkers may detect that the logs have been modified, the nature of those modifications may not be evident.

System integrity checkers offer a strong deterrent, and can be of inestimable value in mitigating the effects of an intrusion, but they are best suited as a last line of defence. Once an intrusion has progressed to the point where system files are compromised, much of the potential damage could already have occurred – particularly where a loss of confidentiality is concerned.

8.4 Password Crackers

Many modern security systems have moved away from the user – password authentication scheme, using biometric identities, cryptographic schemes, one-time passwords, and the like. For the large group remaining, however, weak passwords remains a significant problem.

Password crackers are tools that attempt, through a combination of social engineering and brute force, to guess the password associated with a resource [Muffett92]. These tools are well-known as a major risk in the Unix world [Farmer93], but have recently found their way into many other systems – in fact, a password cracker is now available for virtually every system using key-phrase based protection, such system authentication and file encryption.

In addition, the computing power available to crackers is increasing the level of complexity needed for secure passwords – current recommendations [Kyas97 p 40] include passwords of at least 8 characters in length, and changed every 3-6 months. However, even with these recommendations in place, human nature tends to use the simplest solutions – hence the problem of weak passwords.

From a security point of view, password crackers allow an administrator to identify and address weak passwords before they become a problem. On the counter side, attackers also find such tools invaluable in gaining access to systems. Running periodic checks on passwords used, especially for sensitive accounts, can make a system more secure – but is not a replacement for user education, and stronger authentication mechanisms.

8.5 Sniffer Detection

Many of the current network protocols were designed to function in a trusted environment. Protocols such as Telnet, HTTP, FTP, and many others carry sensitive information in clear format – any person observing the network traffic can extract such information, a typical example being username – password pairs on a network login.

Attackers are well aware of this fact, and often place network monitoring tools, or sniffers, on compromised hosts. The traffic captured on such hosts can then be used to compromise better protected hosts, or gather sensitive information. Since there is often no need for special equipment for this monitoring, it can be very difficult to identify which hosts may be observing confidential exchanges.

In response to this problem, a series of tools have been developed – so-called sniffer detectors[44]. These tools use a number of techniques to attempt to isolate eavesdroppers [Graham99] [L0pht99]:

- **MAC / Protocol addressing mismatches**: Many network protocol stacks do not verify that messages received were actually sent to their addresses – they rely on lower levels in the stack for that. A machine in promiscuous mode may therefore respond to requests sent out to its correct protocol address – even if the lower level MAC address was incorrect. A machine will then only respond to such requests if the MAC address filtering is not active – or it is in promiscuous mode. Examples of such requests include ICMP Ping, UDP or TCP echo (or other ports that always respond), or requests that generate error replies. The core of the method is attempting to fool a host into replying to a request it should not have been capable of seeing.
- **DNS Test**: Many machines automatically do reverse DNS lookups on IP addresses not yet mapped. Therefore, by sending messages to fictitious hosts, and monitoring reverse lookups, a sniffer detector can recognise machines monitoring traffic.
- **Decoy method**: Attackers will often sniff networks looking for such items as remote login sessions in the setup phase of protocols such as FTP, Telnet, or POP. By generating false login transactions, and monitoring for attempts to make use of that information, it is possible not only to identify the presence of monitors, but also to verify that these are being used to attack a network. This technique was described more fully in [Dacier98].
- **Latency tests**: In most modern networks, the resolution of MAC addresses is handled by hardware on the network interface. Therefore, the workload of a machine on a heavily loaded network segment will depend only on the traffic destined for that machine – unless it is observing all traffic. By comparing the response patterns of machines on lightly and heavily loaded links, sniffer detector tools can determine whether a machine appears to be in promiscuous mode. A machine monitoring the segment will have to interpret every message on a heavily loaded link, placing a high processing overhead on that machine. Therefore, the response pattern for a monitor will differ greatly between light and heavy loads, while for normal configurations the patterns should be near identical.
- **Direct inspection**: Directly checking the state of a network adapter on host machines is possible – and may be the only way to detect which machines are in promiscuous mode under certain circumstances. This method may not be feasible on large networks, and on an ongoing basis, however.

Of course, tools have been developed to attempt to avoid detection – but the presence of an unauthorised monitor on a network is a strong indication that there is a security problem.

8.6 Honeytrap Systems

As pointed out in [Ranum97], current IDS methodologies have a number of shortcomings, including problems recognising novel attacks, the occurrence of false positives, and reporting of attacks that are of no interest (because the system is known to be invulnerable to these attacks). A tool which attempts to bridge these gaps is that of honeytrap systems – simulated or real systems that exist for the sole purpose of being attacked.

In essence, the goal of these systems is to act as bait – encouraging attackers to attack these in preference to more valuable parts of a network. Once such a system is attacked, an administrator knows that the network is under attack, and can closely monitor the attacker. Since the system is not generally used, the problem of false positives does not occur – any activity on that system is hostile. Since the system does not depend on recognising specific attacks, and the limited activity levels allow thorough review of all activity on that system, novel attacks can be observed and studied. Finally, the fact that an attacker has penetrated the honeytrap system implies that other systems on the network are vulnerable.

One of the most important aspects of a honeytrap system is that it should not be recognisable as such – to an attacker, it should look and behave as a real system would. In addition, in order to learn from

[44] A recent example of such a tool is the recently released L0pht Antisniff application – see [L0pht99]

such a system, it should be configured similarly to the real systems on a network, allowing lessons learned there to be applied directly in improving the security of more valuable machines. While software tools are available that simulate networks and hosts, in this section we shall focus on honeytraps built out of dedicated systems [Spitzner99-2].

The first step issue in setting up a honeytrap is to ensure that it does not reduce the security of other systems on the network. Should a honeytrap system be compromised, it must be ensured that this system cannot be used to attack other systems (on the same network, or any other). Many of the mechanisms used in firewalls apply here – aimed at keeping the intruder in, rather than out.

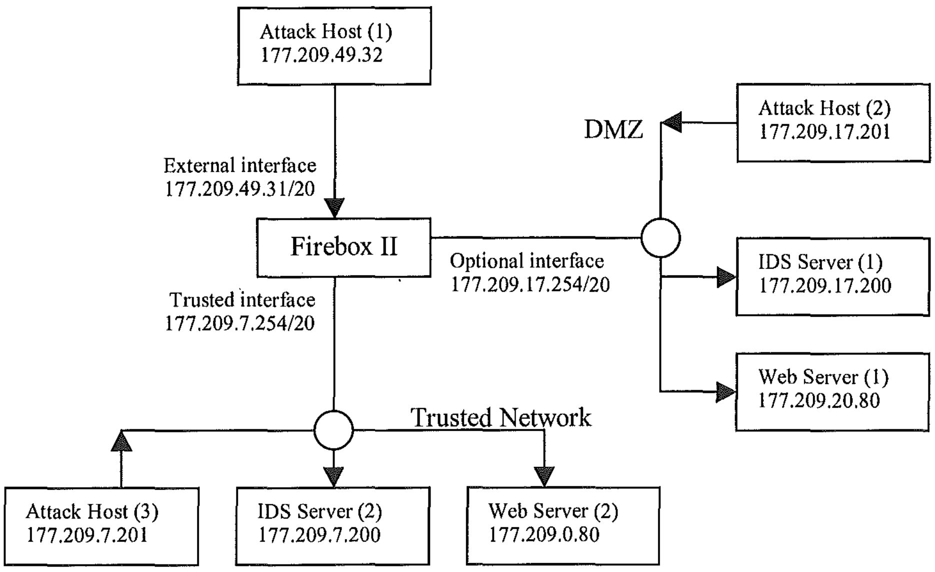
Secondly, an administrator should ensure that the honeytrap system gathers as much information as possible. In addition, this information should be kept in a safe area – since it is assumed that the honeypot will become compromised. In addition, this increased logging should be hidden from an attacker, to avoid them focusing on "less protected" – and more valuable – systems.

For an eminently readable description of how a honeytrap works, and what its value in a system under attack can be, refer to [Cheswick92] or [Bellovin94]. The legality of using such systems has been a subject of some discussion – the conclusion of which was that a honeytrap is no more illegal than a burglar alarm [IDSList].

9. Intrusion Detection experiments

The discussion thus far has been predominantly theoretical. In order to demonstrate some of the techniques and security aspects that have been discussed, we built a small network based around a Watchguard Firebox II firewall. The experiments themselves consisted of reviewing the interaction between the firewall, IDS tools installed at various locations, and network scanning tools applied against these systems. In particular, the lab attempts to demonstrate the need for additional security measures, even in the presence of mechanisms such as firewalls.

9.1 Testbed Configuration



Attack Host (1) 177.209.49.32
External interface 177.209.49.31/20
DMZ
Attack Host (2) 177.209.17.201
Firebox II
Optional interface 177.209.17.254/20
IDS Server (1) 177.209.17.200
Trusted interface 177.209.7.254/20
Web Server (1) 177.209.20.80
Trusted Network
Attack Host (3) 177.209.7.201
IDS Server (2) 177.209.7.200
Web Server (2) 177.209.0.80

Schematic outline of the firewall testbed

The testbed was set up to simulate a simple small office network: a single public server (located in the firewall's DMZ), a limited set of machines on the firewall's trusted network, and an unspecified group of machines on the external network[45]. The configurations used were based on recent versions of common software systems: in particular, no special effort was spent to weaken or harden machines against attack. It is believed that this closely resembles the reality of networks - where minimal effort is put into securing individual machines, and security mechanisms are centralised around the firewall.

Specifically, the following components were used:

1. Watchguard Security System 3.3, including a Watchguard Firebox II (running driver version 3.30.B293) [WG99]
2. IDS Server: A Windows NT Server 4.0 (Service Pack 5) system running:
   - Internet Information Server version 3.0
   - Microsoft FTP server 3.0
   - AbirNet SessionWall-3 Version 1 Release 4 (Network based IDS)[46]
   - Network ICE BlackICE version 1.8.6.4 (Host based IDS)[47]
   - Microsoft Network Monitor version 1.1 (NetMon network capture software)
3. Web Server: A Linux RedHat 6.0 system running:
   - Apache 1.3.6 HTTP server[48]

[45] For information on how firewalls work, refer to [Hunt98] or one of the references for Section 2.1
[46] Available from http://www.abirnet.com/products.html
[47] Available from http://networkice.com/Products/BlackICE
[48] Available from http://www.apache.org/

- wu-ftpd 2.4.2 FTP server[49]
- tcpdump 3.4 (network capture software)[50]

4. Attack host: A Linux Red Hat 6.0 system running:
   - Nessus 0.98.1 (network scanning tool)[51]
   - Satan 1.1.1 (network scanning tool)[52]
   - Nmap 2.12 (network mapping tool)[53]
5. Attack host: A Windows NT Workstation 4.0 (Service Pack 3) system running:
   - Cybercop Scanner version 5.0 (network scanning tool)[54]

The firewall policy was set up as follows:

- Ping (ICMP) traffic was allowed in and out without restriction
- Incoming FTP traffic was allowed (via a proxy) only if destined for 177.209.20.80 - the public server located in the firewall DMZ
- Outgoing FTP traffic was allowed without restriction (via a proxy)
- Incoming HTTP traffic was allowed (via a proxy) only if destined for 177.209.20.80
- Outgoing HTTP traffic was allowed without restriction (via a proxy)
- Incoming SMTP traffic was allowed only to 177.209.49.31 (the external firewall interface)
- Outgoing SMTP traffic was allowed only from 177.209.0.25 (a hypothetical SMTP server on the trusted network)
- Configuration access to the firewall was allowed only from the internal networks
- IP Masquerading was disabled
- Port Autoblocking was disabled
- All other ports andservices were (in theory) blocked

The experiments themselves consisted of running Nessus, Cybercop Scanner and Satan (located on Attack host, from positions 1 to 3) against IDS Server and Web Server (located on positions 1 and 2 for different tests). It was then noted which attacks were reported on IDS Server (by SessionWall and BlackICE), and what traffic was observable via NetMon and tcpdump. This gives an idea of how effective the firewall was in filtering out dangerous traffic, how susceptible the standard configurations are to attack, and what attacks would be capable of penetrating the firewall from various locations.[55]

9.2 Experimental Results

The exact scan configurations were as follows:

1. Scan Web server (2) and IDS server (2) from Attack host (3) (all machines are on a common network segment).
2. Scan Web server (1) and IDS server (1) from Attack host (3) (attack on DMZ from trusted network).
3. Scan Web server (2) and IDS server (2) from Attack host (2) (attack on trusted network from DMZ).
4. Scan Web server (1) and IDS server (1) from Attack host (1) (external attack on DMZ).
5. Scan Web server (2) and IDS-server (2) from Attack host (1) (external attack on trusted network).

Scan 1 gives a baseline of what attacks the IDS tools are capable of recognising, and corresponds to an internal attack on the trusted network. Scan 2 simulates the effect of an internal attack against the optional network. Scan 3 simulates the result if a machine in the DMZ is compromised, and attempts to attack internal machines. Scan 4 represents the most common case, where an external attacker attempts to access protected machines in the DMZ, while scan 5 is the same situation with respect to machines in the trusted network.

---

[49] Available from ftp://ftp.wu-ftpd.org/pub/wu-ftpd/
[50] Available from ftp://ftp.ee.lbl.gov/tcpdump.tar.Z
[51] Available from http://cvs.nessus.org/
[52] Available from http://www.fish.com/satan/
[53] Available from http://www.insecure.org/nmap/
[54] Available from http://www.nai.com/
[55] For information on more rigorous IDS testing, refer to [DURST99] and [Jackson99]. See also [Spitzner99-5] for firewall validation methods.

Each scan was originally done using all three scanning tools. For brevity, however, we shall only detail the results gained using Nessus – the results gained from the other tools were similar. In addition, Nmap was used to determine the visibility of target machines, where appropriate.

The following table contains a summary of the IDS report for scanning runs 1 to 5. Note that attacks with duplicate recognition patterns have in some cases been grouped or omitted, to reduce the size of the output. In the table below, Attack Name is the label given by an IDS for a specific probe[56]. The scans are numbered 1 through 5, as above; results for the Web and IDS servers are listed side by side. Columns headed **S** refer to results from the SessionWall package; **B** refers to BlackICE's detection.

Table 2 Scanning results

	Scan																			
	1				2				3				4				5			
	Web (2)		IDS (2)		Web (1)		IDS (1)		Web (2)		IDS (2)		Web (1)		IDS (1)		Web (2)		IDS (2)	
Attack Name	S	B*	S	B	S	B*	S	B	S	B*	S	B	S	B*	S	B	S	B*	S	B
TCP Port Probe								X[2]												
TCP SYN flood				X																
FTP PORT bounce				X																
TCP OS Fingerprint				X																
TCP ACK Ping				X				X				X				X				X
Back Orifice Ping			X	X																
CGI htmlscript				X				X												
ICMP subnet mask request				X																
HTTP URL very long				X				X												
IIS malformed HTR request			X	X																
CGI info2www				X				X												
CGI nph-test-cgi	X		X	X	X		X	X					X							
CGI perl.exe				X				X												
CGI perl	X		X	X	X		X	X					X							
CGI pfdisplay.cgi				X				X												
CGI phf	X		X	X	X		X	X					X							
HTTP cgi starting with php				X				X												
CGI sh, csh, bash, tcsh	X		X	X	X		X	X					X							
CGI ksh, ash	X		X		X		X						X							
CGI test-cgi	X		X	X	X		X	X					X							
CGI uploader.exe	X		X	X	X		X	X					X							
CGI webdist.cgi	X		X	X	X		X	X					X							

[56] See http://networkice.com/Advice/Intrusions/default.htm for descriptions

	Scan																			
	1				2				3				4				5			
	Web (2)		**IDS (2)**		**Web (1)**		**IDS (1)**		**Web (2)**		**IDS (2)**		**Web (1)**		**IDS (1)**		**Web (2)**		**IDS (2)**	
Attack Name	**S**	**B***	**S**	**B**	**S**	**B***	**S**	**B**	**S**	**B***	**S**	**B**	**S**	**B***	**S**	**B**	**S**	**B***	**S**	**B**
CGI webgais				X				X												
CGI websendmail	X		X	X	X		X	X					X							
Passwd file				X				X												
TearDrop2 attack				X																
FTP CWD very long				X																
TCP port scan				X	X[1]			X												
Fragment overlap				X																
Teardrop attack	X		X	X																
FTP Port Difference	X		X																	
HTTP ../.. exploit	X		X		X		X	X					X							
Inetd Newline Vulnerability	X				X															
NULL Linux FTP Backdoor	X				X		X						X							
Cold Fusion sample URL								X												
Cold Fusion Application Server	X				X		X						X							
Windows WebSite buffer overflow	X				X		X						X							
FTP ROOT Attempt	X				X		X	X					X							
WS_FTP Server Remote DoS	X				X		X						X							
Glimpse HTTP	X				X		X						X							
IRIX /cgi-bin/handler	X				X		X	X					X							
SMTP/mailer exploits	X																			
IIS FTP Exploit/DoS	X												X							
IIS sample URL								X												
CGI win-c-sample.exe								X												
CGI campas								X												
CGI faxsurvey								X												
CGI finger								X												
FTP SITE EXEC command								X												

[1] But with reversed source and destination: reports that Web is scanning Attack.

[2] This attack is reported as originating from the firewall interface
[3] No open ports were found, abbreviating the scan
* BlackICE, being a host-based IDS, is unable to recognize attacks not directed at its host.

While scanning the Web server, Nessus found the following security issues:

Scan 1:

- The FTP service allows anonymous logins
- A number of dangerous services are in place, including Telnet (it is vulnerable to sniffing), Finger (it gives away user details), ident (allows attackers to identify what the privileges of services are – what user id it runs under) and rlogin (vulnerable to spoofing, sniffing).
- The SMTP server responds to EXPN and VRFY queries, allowing attackers to gain information on the user structure of a system
- The HTTP server responds to http://hostname/?open queries, matching a security flaw in the Domino HTTP server. This is a false positive (the additional parameter actually gets ignored), but it will be useful in identifying which scans would also allow application-level attacks.
- A number of open ports were found. These include DNS (53), sunrpc (111), nntp, netbios-ssn, printer (515), and a number of ports corresponding to unspecified services (including port 6000 – probably an X-server).
- A number of NFS-oriented services were running (nlockmgr, rquotad, statd), but no file systems were exported.
- The scanner recognized the remote operating system as Linux 2.1.* or 2.2.* - in fact it is Linux 2.2.5.

This should be the maximal set of security flaws visible for the server itself – in the next scans, we should be able to observe the effects of the firewall.

Scan 2:

- Ports 256-258 were found open, which matches a Checkpoint Firewall/1, according to nessus. This would allow an attacker to infer the presence of a firewall, if mislabeling the exact model.
- The FTP server still allows anonymous logins, but is now also reported to allow unlimited PASV commands (leading to a possible DoS attack). It would appear as if the proxy software on the firewall has introduced a new vulnerability.
- The HTTP ?open false positive occurs again – the firewall seems to have no effect on this application level attack.
- Again, nessus lists a number of open ports and dangerous services – no change there.
- The remote OS is identified as Linux 1.3.* or 2.0.0-2.0.34. This corresponds to the firewall itself, rather than to the host scanned – but still implies potential problems, as discussed in section 9.3. (The actual firewall kernel version is Linux 2.0.33)

Scans 3 and 5 produced no results, since no open ports were found. Using TCP ACK scanning, however, it was possible to identify the IP numbers of machines present in the scanned network – including the firewall interfaces.

Scan 4:

- FTP server allows anonymous logins
- The Apache HTTP server accepts ?open requests – but ignores the parameter. While this is a false positive, it demonstrates that application-level exploits would blithely penetrate the firewall.

This scan produces minimal results – as would be hoped, given that it corresponds to the exact attack a firewall is designed to counter. It is worth noting, however, that there are still vulnerabilities found – but only on services explicitly allowed through the firewall.

9.3 Conclusions

- **Out-of-box configurations may be dangerous.** The test systems were configured using the default installation settings. The fact that a number of potentially dangerous services were set up demonstrates a significant problem in current system design: security is an afterthought.
- **Firewalls protect inaccessible machines well.** The sudden drop probes that reach the scanned host, demonstrates that the firewall is effective in what it was designed to do: blocking external

access to a network. It is encouraging to note that the DMZ has as little access to the trusted network as the external network – correctly limiting the risk of compromised machines in the DMZ.

- **Firewalls do not protect against application-level attacks.** Both of the public services that were allowed through the firewall made use of the proxies provided – and both remained susceptible to attack. A quick look at the application logs confirms this: a large number of attacks were allowed through to the application. This demonstrates the fact that firewalls, while effective against low-level attacks, cannot protect a system against application level attacks. The same paths that are used by authorised external users allow attackers access to the system.
- **The firewall configuration may not be what you expect.** While configuring the firewall, we noticed an interesting point: the DMZ has very little or no protection against machines in the trusted network. Intuitively, we would have expected that access to the DMZ would be subject to the same rules as external access – this was not the case. Without the use of the network scanners and inspection tools, it is unlikely that we would have become aware of this problem. No matter how simple the configuration, or how dependable the firewall may be, it clearly pays to verify the workings of a configuration. This is where many of the tools described in this report comes into their own.
  Consider the following scenario: A policy decision has been made that Telnet connections and the rsh utilities may only be used internally, due to the danger of sniffing attacks. Therefore, the firewall is configured to block such outgoing connections. The caution is well-founded: an attacker has already compromised a public server in the DMZ, and installed a network monitor. Since the firewall does not consider connections to the DMZ to be external (at least, according to the operation of its rules), low-security protocols are used to connect to machines in this network. The result: the combination of unexpected rule behaviour, user patterns, and a compromised server allows an attacker to gain information that would otherwise be inaccessible.
- **Firewalls are themselves vulnerable to attack.** At one point during our testing, we noticed that the firewall ceased responding about midway through every scan. Testing showed this to be the result of one of the DoS attacks being tested against remote machines – specifically, the *nestea* and 0-length fragment attacks. Because of a bug in the fragmentation handling code in the firewall, these attacks were capable of disabling the firewall completely. Even though the firewall itself was not directly targeted, an attacker could still easily launch an extremely effective Denial of Service attack against the network.
  (The firewall did, however, fail in a safe manner – by blocking all connections)
- **Firewalls do not always do what they are told.** Another problem we noticed with the firewall, was the fact that some features (specifically, the auto-blocking behaviour) were impossible to disable. Firewalls are complex pieces of software, and subject to many of the same problems as the systems they protect. Just because it looks like a hardware device, does not mean that it is bug-free.
- **IDS tools can recognise many attacks.** As shown in the above table, the IDS tools used were effective in recognising a wide range of attacks – particularly application level attempts. Clearly, the IDS tools are effective in recognising some attacks.
- **Different IDS tools have different detection sets.** Looking at the table, it is clear that the different IDS tools recognise different attacks – and classify the same attacks in different ways. While it is still in the early stages, there has been some discussion on standardising attack names. By combining different IDS tools, it is possible to improve the chances of detecting every attack.
- **Different IDS systems have different protective ranges.** In the scan results, it is notable that the BlackICE system offered no protection to the Web server – while the SessionWall system had virtually identical coverage on both targets. It has been noted that host-based IDS are limited in coverage; this clearly demonstrates that point. In the same manner, however, neither IDS would offer any protection against attacks that does not coincide with its monitoring range – the specific host or broadcast segment.
- **Network IDS recognise attacks from their area of coverage.** During testing, it was interesting to note that the SessionWall IDS was as effective in recognising attacks originating from its network segment, as recognising attacks directed into its area. An interesting concept would be to monitor possible sources of attack – rather than vulnerable assets.
- **Network scanning tools are susceptible to false readings.** The only weakness found in the Apache Web server was the ?open parameter – a weakness associated with the Domino software. Just because a network scanner reports a problem, does not necessarily imply that it is present.
- **The link between a firewall and it log host is subject to disruption.** The Watchguard firewall allows an administrator to monitor it in real time, and offers remote logging facilities. During

testing, it became obvious that this link was rather fragile – it was continually being disrupted by the scanning activity. Protection against this type of problem is one of the core aspects of the IDS communication protocols under development – but in the mean time, how secure is the reporting in a firewall really?

- **Firewalls offer minimal detection capabilities.** Firewalls are in an excellent position to inspect external network traffic, and many firewalls offer some form of IDS capability. In the Watchguard firewalls, this came in two forms: reporting of attempts to access blocked resources, and Auto-blocking of remote machines that make such attempts. These capabilities are, however, mostly applicable to the early scanning phase of an attack. Against higher-level attacks, the all a firewall offers is the following type of log:

```
DO_NOT_EDIT_THIS_LINE_version_1.05
415   //_this_is_the_number_of_entries
11/01 13:27:26    * 0   deny   in   eth0    tcp  177.209.49.32       177.209.20.80
1364    256     syn (default)
(… for ports tcp 7, udp 7, tcp 8080, tcp 23, tcp 0)
11/01 13:28:07    * 0   deny   in   eth0    tcp  177.209.49.32       177.209.20.80
25839   0       fin
11/01 13:28:07    * 0   deny   in   eth0    tcp  177.209.49.32       177.209.20.80
25841   0       fin syn (blocked port)
11/01 13:28:07    * 0   deny   in   eth0    tcp  177.209.49.32       177.209.20.80
25842   0       psh
11/01 13:28:07    * 0   deny   in   eth0    tcp  177.209.49.32       177.209.20.80
25843   0       syn (blocked port)
(… and many more such port probes)
11/01 13:31:19    * 1   ftp-proxy[133] No access to command STOR .nessus_test from
177.209.49.32
(… application level attack: what did nessus attempt to STOR?)
11/01 13:31:24    * 1   ftp-proxy[134] No access to command STOR .nessus_test_2 from
177.209.49.32
11/01 13:31:30    * 1   ftp-proxy[135] Command from 177.209.49.32 too long
11/01 13:31:30    * 1   ftp-proxy[135] Command from 177.209.49.32 too long
11/01 13:31:30    * 1   ftp-proxy[135] Command from 177.209.49.32 too long
(… buffer overflow attempts)
11/01 13:31:36    * 0   deny   in   eth0    tcp  177.209.49.32       177.209.20.80
1412    14582   syn (default)
11/01 13:31:56    * 1   ftp-proxy[139] No access to command SITE exec /bin/sh -c
/bin/id from 177.209.49.32
(… more port probes)
11/01 13:33:35    * 0   deny   in   eth0    tcp  177.209.49.32       177.209.20.80
1446    25      syn (SMTP)
11/01 13:36:58    * 1   kernel fragment (possible attack) from 177.209.49.32.
11/01 13:36:58    * 1   kernel Oversize fragment (possible attack) from 177.209.49.32.
(+ 274 lines of the same)
(… more diverse probles)
11/01 13:37:04    * 0   deny   in   eth0    tcp  177.209.49.32       177.209.20.80
1490    25      syn (SMTP)
11/01 13:37:06    * 0   deny   in   eth0    tcp  177.209.49.32       177.209.20.80
(… and so on)
```

While these entries may be useful for statistical purposes, they offer little or no value for intrusion detection – too much information is discarded. In addition, the sheer volume of reporting, and the fact that the logs are produced in a format which does not lend itself to direct manipulation (getting the above required extracting information from the native log format), significantly reduces the value of the logs as a detection tool.

Finally, then, we have reached a number of conclusions. First, we have shown that firewalls do not completely protect network resources. Secondly, we have shown that IDS tools are capable of improving the security of a network. This was, however, only a brief investigation of the issues involved –for a fuller review of a number of IDS systems, refer to [Jackson99].

10. Conclusion

Hopefully, this report has given the reader an overview of the field of security tools that do more than just keeping attackers out. Intrusion Detection is often considered to be a field equivalent to the one described here, and is developing by great strides at the moment.

Areas of future interest particularly include distributed IDS – breaking away from the hierarchic filter paradigm, IDS in devices – offering error detection and security on a whole new level, and intelligent security systems – systems that recognise and respond to attacks independently. The next few years promise to be interesting.

One issue that would appear to have been left behind in the development of this field is the application of small scale, simple systems for detection. A wide variety of powerful distributed systems are available, but have too much administrative load and complexity for many small networks. There are systems that promise lightweight sensors – trading off administrative complexity for power.

At the other end of the scale, a number of powerful single-point detection systems have been developed. These have, however, fallen out of fashion in recent years – and are still too complex. There remains a definite split between systems that offer powerful detection, and systems that are configurable.

The most powerful form of detection is one that knows and understands the network it is protecting – information embodied in the organisational security policy. The next logical step, therefore, is policy-focused detection, and systems that scale well to small networks.

11. Glossary

Due to space limitations, this section has been dropped – please refer to http://www.sans.org/NSA/glossary.htm for details on specific terms.

12. Bibliography

[AAFID] CERIAS Autonomous Agents for Intrusion Detection Group, "COAST Autonomous Agents for Intrusion Detection", Project homepage, http://www.cs.purdue.edu/coast/projects/autonomous-agents.html, 7 September 1999

[AAFID98] Jai Balasubramaniyan, Jose Omar Garcia-Fernandez, E. H. Spafford, Diego Zamboni, "An Architecture for Intrusion Detection using Autonomous Agents", COAST Technical Report 98/05, ftp://coast.cs.purdue.edu/pub/COAST/papers/diego-zamboni/zamboni9805.pdf June 11, 1998

[Anderson80] James P. Anderson, "Computer Security Threat Monitoring and Surveillance", Technical Report, James P. Anderson Co., Fort Washington, PA, April 1980.

[Anderson95] Debra Anderson, Thane Frivold, Alfonso Valdes "Next-generation Intrusion Detection Expert System (NIDES) A Summary", SRI-CSL-95-07, May 1995, http://www.sdl.sri.com/nides/reports/4sri.pdf

[Anderson98] Ross Anderson, Abida Khattak "The Use of Information Retrieval Techniques for Intrusion Detection", 10 June 1997, http://www.cl.cam.ac.uk/ftp/users/rja14/raid.ps.gz, RAID'98

[Backdoor97] Christopher Claus "Backdoors", http://www.rootshell.com/docs/backdoors.txt, April 1997

[Bellovin93] Steven M. Bellovin, "Packets Found on an Internet", 23 August 1993, Computer Communications Review, July 1993, Vol. 23, No. 3, pp. 26-31. ftp://ftp.research.att.com/dist/smb/packets.ps

[Bellovin95] Steven M. Bellovin, "Using the Domain Name System for System Break-ins", June 1995, Proceedings of the Fifth Usenix UNIX Security Symposium, ftp://ftp.research.att.com/dist/smb/dnshack.ps

[Bellovin96] Steven M. Bellovin, "Problem Areas for the IP Security Protocols", June 1996, Proceedings of the Sixth Usenix UNIX Security Symposium, ftp://ftp.research.att.com/dist/smb/badesp.ps

[Belloving94] Steven M. Bellovin, William R. Cheswick "Firewalls and Internet Security – Repelling the Wily Hacker", Addison-Wesley, 1994, ISBN 0-201-63357-4

[BlacICE99] Andrew Plato, Network ICE "BlackICE Defender User's Guide version 1.0", http://networkice.com/support/Docs/BlackICEDefUG.pdf, 1999

[Bro98] Vern Paxson "Bro: A System for Detecting Network Intruders in Real-Time" USENIX Security Symposium, January 1998, ftp://ftp.ee.lbl.gov/papers/bro-usenix98-revised.ps.Z

[cDc98] Cult of the Dead Cow Communications "Back Orifice Remote Administration System v1.20", http://www.cultdeadcow.com/tools/bo.html, August 1998.

[CERT99] CERT "Establish policies and procedures for responding to intrusions", http://www.cert.org/security-improvement/practices/p044.html.

[Chapman95] D. Brent Chapman, Elizabeth D. Zwicky "Building Internet Firewalls", O'Reilly & Associates, 1995, ISBN 1-56592-124-0

[Cheswick92] Bill Cheswick, "An Evening with Berferd In Which a Cracker is Lured, Endured, and Studied", Proc. Winter USENIX Conference, January 1992. ftp://ftp.research.bell-labs.com/dist/ches/berferd.ps

[Cheung97] S. Cheung, K. N. Levitt, "Protecting Routing Infrastructures from Denial of Service Using Cooperative Intrusion Detection" Proc. New Security Paradigms Workshop 1997, Cumbria, UK, September 23-26, 1997. http://seclab.cs.ucdavis.edu/papers/nsp.pdf

[CIDF] CIDF "Common Intrusion Detection Framework", http://seclab.cs.ucdavis.edu/cidf/, 12/3/98

[CIDF98] Kahn, C., Porras, P., Staniford-Chen, S. and Tung, B. "A Common Intrusion Detection Framework". Submitted to the Journal of Computer Security, 15 July 1998 http://seclab.cs.ucdavis.edu/cidf/papers/jcs-draft/cidf-paper.ps

[CIDF98-2] Rich Feiertag (TIS), Cliff Kahn (Open Group), Phil Porras (SRI), Dan Schnackenberg (Boeing), Stuart Staniford-Chen (UC Davis), Brian Tung (ISI, editor) "A Common Intrusion Specification Language (CISL)", 16/10/98, http://gost.isi.edu/projects/crisis/cidf/cisl_current.txt

[CIDF98-3] Clifford Kahn (Open Group), Don Bolinger (Open Group), Dan Schnackenberg (Boeing) "Communication in the Common Intrusion Detection Framework v0.7", 8/6/98, http://seclab.cs.ucdavis.edu/cidf/comm/cidfcomm.txt

[CNN99] CNN "Pentagon 'at war' with computer hackers", 5 March 1999, http://cnn.com/TECH/computing/9903/05/pentagon.hackers/

[Coord98] Naval Surface Warfare Center, "SHADOW Indications Technical Analysis: Coordinated Attacks and Probes", 14 Dec 1998, http://www.nswc.navy.mil/ISSEC/CID/co-ordinated_analysis.txt

[CSI99] Computer Security Institute "1999 CSI/FBI Computer Crime and Security Survey", Computer Security Issues & Trends Vol. V, No. 1, Winter 1999, March 1999, http://www.gocsi.com/prelea990301.htm

[Dacier98] Marc Dacier, Stephane Grundschober "Design and Implementation of a Sniffer Detector", RAID'98, September 1998, http://www.zurich.ibm.com/~dac/Prog_RAID98/Full_Papers/sniffer_detector.html/index.htm

[Debar92] H. Debar, M. Becker, D.Siboni "A Neural Network Component for an Intrusion Detection System", Proc. IEEE Symposium on Research in Computer Security and Privacy, 1992.

[Debar99] Herve Debar, Marc Dacier, Andreas Wespi "Towards a taxonomy of intrusion-detection systems", Computer Networks 31 (1999) 805-822, (http ref depends on userid)

[Denning87] Dorothy E. Denning, "An Intrusion-Detection Model", IEEE Transactions on Software Engineering, Vol SE-13, No. 2, February 1987 p 222-232

[Denning96] Dorothy E. Denning, "Protection and Defense of Intrusion", 5 March 1996, Presented at Conf. on National

Security in the Information Age, US Air Force Academy, Feb. 1996., http://www.cosc.georgetown.edu/%7Edenning/infosec/USAFA.html

[DIDS91] Steven R. Snapp *et al*, "DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype", Proc. 14th National Computer Security Conference. Washington, DC, Oct. 1991, pp. 167-176., http://seclab.cs.ucdavis.edu/papers/DIDS.ncsc91.pdf

[DOD99] Sam Cox, Ron Stimeare, Tim Dean, Brad Ashley, "Information Assurance - the Achilles'Heel of Joint Vision 2010?", http://www.airpower.maxwell.af.mil/airchronicles/cc/ashley.html, March 1999

[Doorn94] Leendert van Doorn, "Computer Break-ins: A Case Study", ftp://coast.cs.purdue.edu/pub/doc/misc/Leendert_van_Doorn-Computer_Brakins.ps.Z June 1994

[DSD98] Defense Signals Directorate, "Security Guidelines for Australian Government IT Systems", Australian Communications-Electronic Security Instructions 33, April 1998, http://www.dsd.gov.au/infosec/downloads/acsi33.pdf

[Durst99] Robert Durst, Terrence Champion, Brian Witten, Eric Miller and Luigi Spagnuolo "Testing and Evaluating Computer Intrusion Detection Systems", Communications of the ACM, July 1999, Vol. 42 no. 7, p. 53-61

[EMERALD97] Phillip A. Porras, Peter G. Neumann " EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances", http://www.sdl.sri.com/emerald/Emerald-NISS97.ps.gz, 1997 National Information Systems Security Conference)

[EMERALD99] Peter G. Neumann, Phillip A. Porras, "Experience with EMERALD To Date" First USENIX Workshop on Intrusion Detection and Network Monitoring, April 1999, http://www.sdl.sri.com/emerald/det99.ps.gz

[Farmer93] Dan Farmer, Wietse Venema "Improving the Security of Your Site by Breaking Into it", ftp://ftp.porcupine.org/pub/security/admin-guide-to-cracking.101.Z, December 1993.

[Frank94] Jeremy Frank, "Artificial Intelligence and Intrusion Detection: Current and future Directions", 9 June 1994, http://www.raptor.com/lib/ncsc.94.pdf

[GASSATA98] Ludovic Me (SUPELEC, France) "GASSATA, a Genetic Algorithm as an Alternative Tool for Security Audit Trails Analysis", RAID'98, September 1998, http://www.zurich.ibm.com/%7Edac/Prog_RAID98/Full_Papers/gassata_paper.pdf

[Graham99] Robert Graham, "Sniffing (network wiretap, sniffer) FAQ", Version 0.2.8, 28 September 1999, http://www.robertgraham.com/pubs/sniffing-faq.html

[GrIDS99] S. Cheung, *et al* "The Design of GrIDS: A Graph-Based Intrusion Detection System." U.C. Davis Computer Science Department Technical Report CSE-99-2, 1999, http://seclab.cs.ucdavis.edu/arpa/grids/grids.pdf

[Gula99] Ron Gula, "Impressing your friends with Network Security", July 1999, http://www.securitywizards.com/papers/imp.html

[Harris98] B.A. Harris "Firewalls and Virtual Private Networks" University of Canterbury Master's Thesis, 1998

[Herald99] Mark Mueller "Computer 'crackers' set sights on .gov for chaos",1 August 1999, Boston Herald, http://www.bostonherald.com/bostonherald/nat/hack08011999.htm

[Hobbit97] Hobbit, "CIFS: Common Insecurities Fail Scrutiny", January 1997, ftp://ftp.technotronic.com/rfc/cifs.txt

[HTTP99] R. Fielding, *et al* "Hypertext transfer protocol – HTTP/1.1", RFC 2616, IETF, June 1999, http://ietf.org/rfc/rfc2616.txt.

[Hunt98] Ray Hunt "Internet/Intranet firewall security – policy, architecture and transaction services", Computer Communications 21 (1998) 1107-1123, April 1998.

[ICSA] ICSA "About the Intrusion Detection Systems Consortium", ICSA Intrusion Detection Homepage, http://www.icsa.net/services/consortia/intrusion/

[ICSA98] Rebecca Bace, "An Introduction To Intrusion Detection & Assessment", ICSA, March 1999, http://www.icsa.net/services/consortia/intrusion/intrusion.pdf

[ICSA98-2] M. E. Kabay, "ICSA White Paper on Computer Crime Statistics", ISCA, http://www.icsa.net/library/research/crime.PDF

[IDS-Faq99] Robert Graham "FAQ: Network Intrusion Detection Systems Version 0.6.4", 22 September 1999, http://www.ticm.com/kb/faq/idsfaq.html

[IDSList] IDS Mailing list, 1999. Mail majordomo@uow.edu.au to subscribe

[IDWG99] Mark Wood, "Intrusion Detection Message Exchange Requirements", draft-ietf-idwg-requirements-02.txt, October 1999, http://www.ietf.org/internet-drafts/draft-ietf-idwg-requirements-02.txt

[IDWG99-2] Herve Debar, Ming-Yuh Huang, David J. Donahoo "Intrusion Detection Exchange Format Data Model", 15 October 1999, draft-ietf-idwg-model-00.txt, http://www.ietf.org/internet-drafts/draft-ietf-idwg-data-model-00.txt

[IDWG99-3] Dipankar Gupta, "IAP: Intrusion Alert Protocol", draft-gupta-idef-iap-00.txt, 15 September 1999, http://www.ietf.org/internet-drafts/draft-gupta-idef-iap-00.txt

[IDWG99-4] IDWG "Intrusion Detection Exchange Format (idwg) Charter", IDWG Homepage, http://www.ietf.org/html.charters/idwg-charter.html, 29 Sep 1999

[ITSEC99] ISO "Information technology – Security techniques – Guidelines for the management of IT security (GMITS) – Part 5: Safeguards for external connections", ISO/IEC PTDR 13335-5, February 1999

[Jackson99] Kathleen A. Jackson, "Intrusion Detection System (IDS) Product Survey", Los Alamos National Laboratory report LA-UR-99-3883, June 1999, http://lib-www.lanl.gov/ls-pubs/00416750.pdf

[Javitz93] Harold S. Javitz, Alfonso Valdes "The NIDES Statistical Component: Description and Justification", March 1993, SRI International, http://www.sdl.sri.com/nides/reports/statreport.ps.gz

[Joyal96] Paul M. Joyal "Industrial Espionage Today and Information Wars of Tomorrow", October 1996, 19th National Information Systems Security Conference, http://csrc.nist.gov/nissc/1996/papers/NISSC96/joyal/industry.pdf

[Kumar95] Sandeep Kumar, "Classification and Detection of Computer Intrusions", August 1995, Ph.D. Dissertation, Purdue University, ftp://coast.cs.purdue.edu/pub/COAST/papers/sandeep-kumar/kumar-intdet-phddiss.ps

[King98] Steve King *et al*, "The Case for IPv6", Internet Draft, http://search.ietf.org/internet-drafts/draft-ietf-iab-case-for-ipv6-05.txt

[Kyas97] Othmar Kyas "Internet Security – Risk Analysis, Strategies and Firewalls", International Thomson Computer Press, 1997, ISBN 1-85032-302-X

[L0pht99] L0pht "L0pht Antisniff - Technical Details", http://www.l0pht.com/antisniff/tech-paper.html 19/7/1999

[Lane97] Terran Lane and Carla E. Brodley, "An Application of Machine Learning to Anomaly Detection", NISSC'97, 14 February 1997, ftp://coast.cs.purdue.edu/pub/COAST/papers/carla-brodley/brodley-lane-nissc97_paper.ps

[Law&Net99] Law and the Net, "Infrastructure Threats from Cyber-Terrorists", 19 March 1999, http://www.steptoe.com/WebDoc.NSF/Law+&+The+Net+All/Infrastructure+Threats+from+Cyber-Terrorists

[Lunt93] Teresa F. Lunt. "Detecting intruders in computer systems". 1993 Conference on Auditing and Computer Technology, 1993. http://www.raptor.com/lib/canada93.ps

[Medina98] Marcelo Medina "A Layered Framework for Placement of Distributed Intrusion Detection Devices", http://csrc.nist.gov/nissc/1998/proceedings/paperD2.pdf

[Melissa99] CERT "CERT Advisory CA-99-04-Melissa-Macro-Virus", http://www.cert.org/advisories/CA-99-04-Melissa-Macro-Virus.html, March 1999

[Muffertt92] Alec D. E. Muffett "Crack Version 4.1 - A Sensible Password Checker for Unix", July 1992, http://www.ja.net/CERT/Muffett/Crack_4.1_readme.txt

[NetRanger99] Cisco Systems "NetRanger Intrusion Detection System Technical Overview", http://www.cisco.com/warp/public/cc/cisco/mkt/security/nranger/tech/ntran_tc.htm, 1999

[NFR97] Marcus J. Ranum, Kent Landfield, Mike Stolarchuk, Mark Sienkiewicz, Andrew Lambeth, Eric Wall "Implementing a Generalized Tool for Network Monitoring", LISA'97, http://www.nfr.net/forum/publications/LISA-97.htm

[NMap] Fyodor, "Nmap - The Network Mapper", http://www.insecure.org/nmap/index.html

[NMap98] Fyodor, "Remote OS detection via TCP/IP Stack FingerPrinting", 18 October, 1998, http://www.insecure.org/nmap/nmap-fingerprinting-article.txt

[NMap99] Fyodor, "Nmap network security scanner man page", http://www.insecure.org/nmap/nmap_manpage.html, 1999

[PBEST99] Ulf Lindqvist, Phillip A. Porras, "Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST)", Proceedings of the 1999 IEEE Symposium on Security and Privacy, May 9-12, 1999. http://www.sdl.sri.com/emerald/pbest-sp99-cr.pdf

[Phrack98] solar designer "Designing and Attacking Port Scan Detection Tools", Phrack Magazine volume 8, Issue 53, 8 July 1998, http://www.2600.com/phrack/p53-13.html.

[Ptacek98] Thomas H. Ptacek, Timothy N. Newsharn "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection", http://www.secnet.com/papers/IDS.PDF.

[Ranum97] Marcus J. Ranum "Intrusion Detection: Challenges and Myths", 1998, http://www.nfr.net/forum/publications/id-myths.html

[Remsing96] Steve Remsing, "Computer and Network Security: Information for Everyone", 21 August 1996, http://lheawww.gsfc.nasa.gov/%7Esrr/prog_forum.html

[Remsing98] Steve Remsing, "Dealing With Your First Break-in: Mistakes Made, Lessons Learned", SANS'98, May 1998, http://lheawww.gsfc.nasa.gov/~srr/sans98.pdf

[RFC789] Eric C. Rosen, "Vulnerabilities of Network Control Protocols: An Example", RFC 789, IETF, January 1981, http://ietf.org/rfc/rfc789.txt

[Rivest97] Ronald Rivest "S-expressions", Internet Draft draft-rivest-sexp-00.txt, 1997, http://www.ietf.org/internet-drafts/draft-rivest-sexp-00.txt.

[RSAFAQ] RSA Laboratories "Frequently Asked Questions about Today's Cryptography", ftp://ftp.rsasecurity.com/pub/labsfaq/faq4pdf.zip, 1999

[Ruiu99] Dragos Ruiu, "Cautionary Tales: Stealth Coordinated Attack HOWTO", 19 Aug 1999, http://www.nswc.navy.mil/ISSEC/CID/Stealth_Coordinated_Attack.html

[Schneier96] Bruce Scheier "Applied Cryptography", John Wiley & Sons, 1996, ISBN 0-471-11709-9

[Schneier98] Bruce Schneier, "Security Pitfalls in Cryptography", 1998, http://www.counterpane.com/pitfalls.pdf

[Schneier99] Bruce Schneier, Mudge "Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2)", 19 October 1999, http://www.counterpane.com/pptpv2.pdf

[Schneier99-2] Bruce Schneier "The Trojan Horse Race" September 1999, Vol. 42 No. 9 Communications of the ACM p. 128

[Schuba96] Christoph L. Schuba, Ivan V. Krsul, Markus G. Kuhn, Eugene H. Spafford, Aurobindo Sundaram, Diego Zamboni "Analysis of a Denial of Service Attack on TCP", ftp://coast.cs.purdue.edu/pub/COAST/papers/christoph-schuba/schuba-krsul-kuhn-spaf-sundaram-zamboni-synkill.ps

[Shadow98] SANS Institute, "Building a Network Monitoring and Analysis Capability" 1 July 1998, http://www.nswc.navy.mil/ISSEC/CID/step.htm

[Siyan95] Karanjit Syan, Chris Hare "Internet Firewalls and Network Security", New Riders Publishing, 1995, ISBN 1-56205-437-6

[Sommer97] Peter Sommer, "Computer Forensics: an introduction", 1997, http://www.virtualcity.co.uk/vcaforens.htm

[Sommer98] Peter Sommer, "Intrusion Detection Systems as Evidence", RAID98, http://www.zurich.ibm.com/pub/Other/RAID/Prog_RAID98/Full_Papers/Sommer_text.pdf

[Spafford91] Eugene H. Spafford, "The Internet Worm Incident", Purdue Technical Report CSD-TR-933, Proc. Of the 1989 European Software Engineering Conference, September 1991. ftp://coast.cs.purdue.edu/doc/morris_worm/spaf-Iworm-paper-ESEC.ps.Z

[Spitzner99-4] Lance Spitzner, "Intrusion Detection", 28 August 1999, http://www.enteract.com/~lspitz/ids.html

[Spitzner99-5] Lance Spitzner, "Auditing Your Firewall Setup", 26 September 1999, http://www.enteract.com/~lspitz/audit.html

[Spiztner99] Lance Spitzner, "Know Your Enemy", 18 August 1999, http://www.enteract.com/~lspitz/enemy.html

[Spiztner99-2] Lance Spitzner, "To Build A Honeypot", 10 October, 1999, http://www.enteract.com/~lspitz/honeypot.html

[Spiztner99-3] Lance Spitznet, "Watching Your Logs", 28 August 1999, http://www.enteract.com/~lspitz/swatch.html

[spyrit99] dark spyrit "Win32 Buffer Overflows (Location, Exploitation and Prevention", Phrack 55, September 1999, http://ww.insecure.org/news/p55-15.txt

[Sundhu96] Ravi Sandhu, Pierangela Samarati "Authentication, Access Control, and Audit" ACM Computing Surveys, Vol.28, No. 1, March 1996 (off ACM.org)

[Tanenbaum92]	Andrew S. Tanenbaum, “Modern Operating Systems”, Prentice Hall, 1992, ISBN 0-13-595752-4
[TLS99]	T. Dierks and C. Allen “The TLS protocol version 1.0”, RFC 2246, IETF, January 1999, http://ietf.org/rfc/rfc2246.txt
[Tripwire94]	Gene H. Kim, Eugene H. Spafford, "Experiences with Tripwire: Using integrity checkers for intrusion detection", Systems Administration, Networking and Security Conference III, USENIX, 1994. Ftp://coast.cs.purdue.edu/pub/COAST/Tripwire/Tripwire-SANS.ps.Z
[Watson98]	Bret Watson "Current Practices in Audit Automation", TACS-Asia 1998, 29 June 1998, http://www.ticm.com/kb/papers/tacspaper98.pdf
[WG99]	Watchguard Technologies “Watchguard Security System User’s Guide Version 3.3”, April 1999, included with the Watchguard FireBox II
[Wietse96]	Wietse Venema "Murphy's law and computer security", 7 June 1996, http://www.fish.com/security/murphy.html
[ZDNet99]	Robert Lemos, “Cyber Attacks – both old and new”, 19 October 1999, http://www.zdnet.com/zdnn/stories/news/0,4586,2376768,00.html?chkpt=hpqsnewstest,
[ZDNet99-2]	Pankaj Chowdhry “Attacked and hacked!”, PC Week Labs, 11 October 1999, http://www.zdnet.com/pcweek/stories/news/0,4153,2350743,00.html.