Priorities in the deployment of network intrusion detection systems

Master's Thesis

Marcin Dobrucki

Helsinki University of Technology	Teknillinen korkeakoulu
Department of Computer Science and Engineering	Tietotekniikan osasto
Telecommunications Software and Multimedia Laboratory	Tietoliikenneohjelmistojen ja multimedian laboratorio
Espoo 2002	

HELSINKI UNIVERSITY OF TECHNOLOGY	ABSTRACT OF MASTER'S THESIS

Author: Marcin Dobrucki

Name of thesis:
Priorities in the deployment of network intrusion detection systems

Date: May 27, 2002 **Pages:** 8 + 69

Department: Department of Computer Science and Engineering **Chair:** T-110

Supervisor: Professor Teemupekka Virtanen
Instructor: Jonna Särs, MSc

The purpose of this work is to study the priorities in the deployment of network intrusion detection systems in order to minimize costs and to optimize performance.

Network intrusion detection systems are hard to deploy efficiently due to difficulties in accurate detection of events of interest and high workload involved in their processing.

This thesis generalizes the intrusion detection process and describes the factors influencing it. Ways to prioritize deployment of features based on intrusion detection goals are suggested, as well as a number of available techniques which could be used to provide some intrusion detection system functionality without full scale deployment.

A number of small security tools have been evaluated and test results matched against benefits gained by deployment of the Snort open-source network intrusion detection systems. Evaluations have been based on actual network infrastructures when possible, and laboratory simulations otherwise.

The obtained results indicate that using small tools aimed at looking for specific problems, helps achieve network maturity required to deploy a full-scale intrusion detection system. Prioritization of deployment should focus on allowing correlation to minimize false positives at the detection phase and automation of human-performed tasks at the incident handling phase.

Keywords: intrusion detection, priorities, network security
Language: English

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Tekijä: Marcin Dobrucki

Työn nimi:
Tunkeutumisenhavaitsemisjärjestelmien käyttöönoton prioriteetit

Päivämäärä:	27. toukokuuta 2002	**Sivuja:**	8 + 69
Osasto:	Tietotekniikan osasto	**Professuuri:**	T-110
Työn valvoja:	Professori Teemupekka Virtanen		
Työn ohjaaja:	DI Jonna Särs		

Diplomityössä tutkitaan verkon tunkeutumisenhavaitsemisjärjestelmän käyttöönottoon liittyviä prioriteetteja. Tutkimuksen tavoitteina on kustannusten minimointi ja suorituskyvyn optimointi.

Lisäksi työssä tutustutaan tunkeutumisenhavaitsemisen yleisiin periaatteisiin ja esitellään tekniikoita, joita hyödyntämällä voidaan tarjota täyttä järjestelmää vastaava toiminnallisuus ilman järjestelmän käyttöönottoa.

Järjestelmien tehokkaan hyödyntämisen esteinä ovat tärkeiden tapahtumien erottamisen vaikeus ja niiden käsittelyyn liittyvä suuri työmäärä.

Työn aikana on testattu erilaisia verkonvalvontaohjelmia ja tuloksia on verrattu vapaan Snort–järjestelmän tarjoamiin hyötyihin. Vertailu on pääosin toteutettu todellisessa verkkoympäristössä.

Tuloksena voidaan todeta, että tunkeutumisenhavaitsemisjärjestelmän vaatima verkonvalvonnan taso voidaan saavuttaa pienimuotoisilla erikoisohjelmilla.

Tunkeutumisenhavaitsemisjärjestelmän tärkeimpiä ominaisuuksia ovat väärien hälytysten vähentäminen korrelaatiotekniikalla, ja ihmistyön korvaaminen automaatiolla turvavälikohtauksen jatkoselvitysvaiheessa.

Avainsanat: tunkeutumisenhavaitsemisjärjestelmä, tietoturva, toteutus
Kieli: englanti

Acknowledgements

Otaniemi, 27^{th} of May, 2002

I would like to thank the following people for making this work possible: Heikki Arppe, Teddy Grenman, Sanna Havanto, Riku Kalinen, Minna Kangasluoma, Kenny Jansson, Jaakko Kotimäki, Paolo Lucente, Kristiina Mod, Lasse Sundström, and Lea Viljanen. Furthermore, thanks to the IRC community and especially folk on the `#snort` (OpenProjects) and `#ids` (IRCnet and UnderNet) channels for providing insightful comments and opinions.

Special thanks to my tutor Jonna Särs, and professor Teemupekka Virtanen for going out of their way to help my research.

Finally, the warmest of thanks to my wife Minna for her understanding and patience while I was away.

Contents

List of Figures

Glossary

ARP Address Resolution Protocol, RFC826

CIRT Computer Incident Response Team, sometimes the terms RRT (Rapid Response Team), CERT (Computer Emergency Response Team), and SIRT (Security Incident Response Team) are used

DCC Direct Client Client connection, a protocol for exchanging files directly between two IRC clients without interaction through an IRC server

DHCP Dynamic Host Configuration Protocol, RFC2131

DMZ De-Millitarized Zone

DoS Denial of Service

ICMP Internet Control Message Protocol, and extension to the IP which supports error, control, and information messages, RFC792

IDS Intrusion Detection System, hardware or software used to discover unauthorized use of a network or system. See HIDS and NIDS

IP Internet Protocol, a connectionless datagram protocol for package delivery, RFC791 (IPv4), RFC2460 (IPv6)

IRC Internet Relay Chat, RFC1459

GSM Global System for Mobile telecommunications, a wireless telephone system

HIDS Host Intrusion Detection System, a general term for a system used to identify unauthorized entry into a host and modifications made to it by the intruder

HTTP Hypertext Transmission Protocol, a stateless application level protocol used by the World Wide Web

LAN	Local Area Network
MAC	Medium Access Control, protocols which provide control for transmission medium access in an orderly and efficient manner
MSSP	Managed System Security Provider, sometimes referred to as Managed Security Monitoring (MSM)
NFS	Network File System, RFC1813
NIDS	Network Intrusion Detection Systems, a general term for systems used to identify and report infiltration attempts in a network environment
PAL	Phase Alternation by Line, a video encoding standard
RFC	Request For Comments, a term coined by Stephen D. Crocker, used for naming Internet standards (see RFC1000), `http://www.rfc.net`
SMS	Small Message Service is the ability to send and receive short text messages through the GSM. A single SMS message can be up to 160 characters long
SMTP	Simple Mail Transfer Protocol, an application level protocol used for exchanging electronic mail, RFC2821
SSH	Secure SHell, a secure replacement for Telnet providing encrypted authentication and communication channels
TCP	Transmission Control Protocol, a transport layer protocol providing reliable data streaming over IP, RFC793
UDP	User Datagram Protocol, a transport layer protocol providing unreliable data transmission over IP, RFC768
VLAN	Virtual LAN, a network of computers that behave as if they are connected to the same wire even though they may actually be physically located on different segments of a LAN
VPN	Virtual Private Network, a network which uses encryption and other security measures to create secure connections over public lines

Chapter 1

Introduction

Imagine a large office building housing offices of different companies. Some of these firms make lollipops, while others develop alternative fuel cells. As the building caretaker we want to make sure our customers are protected, and adds salt to the lollipop mixture nobody or gets away with the plans for a portable fusion reactor. The typical mechanisms, we might use, to help us secure the building are fences, locks on the doors, guards at the front door checking people's ID's, burglar alarms and video surveillance systems.

The protection mechanisms we utilize represent different levels of building security. We can think of the fences, locks, and the front desk guard as passive perimeter defenses. Once the intruder finds a way to penetrate the perimeter defenses, the chances are they are not going to be much use anymore. Burglar alarms and video surveillance, on the other hand, form an active system which constantly monitors the state of our hallways and offices. Even if penetration of the building occurs, chances are a trail of the intrusion is recorded by the surveillance equipment.

Neither perimeter defenses or surveillance are fully sufficient on their own. Regardless of the height of the fence, the quality of the locks, and the dedication of the guard, someone will eventually find a way to bypass them. In a similar fashion, if we only deploy alarms and surveillance, the intruders would be granted unrestricted access to the control rooms, and hence likely to disable the systems before we had a chance to pinpoint the breach and react.

Together, our security measures form an intrusion prevention and detection system. For such system to be effective, it must be set up to best protect our building, updated regularly to keep up with the latest intrusion advances, maintained so that it functions properly, and supervised in order for someone to be able to react when bad things happen.

In thesis we aim to define priorities in the design, implementation, and deployment of network intrusion detection systems. It is our hope that these guidelines will help companies schedule resource allocation in augmenting their existing security systems with active defenses.

We have focused our research on identifying the cost factors in network intrusion detection and trying to find alternative methods for at least a subset of common problems found in modern corporate LANs. Evaluation of tools and techniques has been based on real environments whenever possible to provide a practical view on the subject area.

1.1 Background

Active security systems are very good at providing large volumes of data [25]. Alarms go off when someone forgets to close a door, entire shelves of video tapes exist documenting events on the corridors, and so on. On its own, this data is not very useful. What we need is some form of classification of events, so that surveillance data can be turned into meaningful information. In this section we glance at the definitions, the motivation, and the principles of extracting information from audit data.

The word *intrusion* means “a wrongful entry” or “the act of seizing, or taking possession of the property of another” [27]. By *Intrusion Detection*, we mean identifying potentially malicious or undesirable activity that may have occurred in a given environment as recorded in an audit trail [4]. Several steps make up this process: capture, analyze, classify, report, and possibly react[1] [5, 19, 26, 40] to the event. When intrusion detection is applied to computers, systems known as *Intrusion Detection Systems* (IDS) usually take care of automating these steps before informing the human supervisor of what has transpired. The aspect of automation is important, and it is a major difference between intrusion detection systems used in buildings and those used in computer systems.

Although we have already stated that neither perimeter defenses nor surveillance mechanisms provide sufficient security on their own, the reasons for this might not be fully clear. In our office building, the perimeter is guarded by fences, doors, locks, guards at the entrance, and so on. However, all of these protection mechanisms exist so that under certain conditions we allow employees to walk in and out of the building. In addition, a number of other people must have access: service personnel, supply&delivery, cleaning staff, guests, and so on. By impersonating someone with legitimate access to the building, or other means of disguise [18],

[1]These steps are deducted from several computer-related intrusion detection models.

an intruder may successfully penetrate the perimeter defense.

Furthermore, assuming that only one type of protective measures is deployed, our resources are suspectable to a single point of failure [9]. By forcing that lone barricade, an unauthorized individual is able to completely negate all of the security mechanisms. This problem is irrespective of whether our defense is a guard, alarm, a surveillance system, or something else. The strength of the defense grows when a number of systems work to correlate and prevent unwanted events.

Finally, a surveillance system usually provides an audit trail, a means of reconstructing events which occurred leading to the current situation. In case of a break-in to the office, or robbery, this might be video footage. The existence of an audit trail provides several benefits: it may help to identify the the path through which infiltration occurred, it may help provide clues as to what has happened during the events, and it may help reveal the identity of the intruder. By iterating the security deployment process, we are able to identify the weak links and prioritize resource allocation in the future.

There are several difficulties in detecting intrusions, and even more in automating such a process. Typical methods of detecting an unauthorized event is either through a use of a policy, or by matching certain behavior [4, 33]. For instance, our policy says that all employees must leave the building by 6PM, and may not enter again before 6AM. Then, if we observe someone wandering through the offices at midnight, we can immediately recognize this as unauthorized event. Similarly, we can define a given pattern of behavior which constitutes an attempt at an intrusion, for instance throwing a brick through a window.

Our ability to determine if the event is an intrusion depends on the capabilities of our security systems. The guard who inspects ID's at the entrance must be able to spot a fake ID. Such ability comes with training and experience. However, when we try to automate such a process, we must be able to provide means of describing the events and policies in such way that we minimize the false positive rate. Computer-based automated IDS usually look at a number of signatures[2] and check whether the matching event falls within acceptable limits.

Despite apparent benefits of automated systems which could detect intrusions and possibly stop them in time, IDS in computer networks are not very widely deployed [35].

[2] A number of other techniques exist, and are discussed in Chapter 2

1.2 Research problem

Our main research problem is finding the key costs and difficulties in network intrusion detection systems. By defining ways to optimize actions in these problem areas, we try to minimize resources required to manage network intrusion detection systems, and maximize their benefits.

Furthermore, we try to identify some of the common security breaches in corporate LANs based on reports and surveys. By isolating specific problems which account for significant part of the incidents, we look for ways to use small tools to combat them without the need to deploy full-scale NIDS. The goal is to evaluate benefits of such approach when working with highly limited resources.

Finally, we investigate scalability of existing infrastructures for times when companies grow beyond the limits of the original system designs.

1.3 Evaluation criteria

Below is a list of some of the most important criteria which we used during the evaluation of methods and tools for intrusion detection:

- Solutions to problems must be feasible. This means it must be possible to implement them with a reasonable amount of human and financial resources.
- We are looking for providing *Return on Investment*, as implementing security mechanisms for purely marketing value is misguided.
- Intrusion detection systems should be transparent to the users on the networks, and should not interfere in day-to-day tasks.
- Legal issues such as employee privacy. We use the “Law for protection of privacy in the workplace” No:477/2001[3] which came into power in Finland on 1st of October, 2001.
- We refer to small networks to mean a range of several hosts to several hundred of hosts. In our discussion, the size of the company is limited to what is known in Finland as *small* and *small-and-medium* sized – up to about 250 employees.

[3]Laki yksityisyyden suojasta työelämässä, No: 477/2001

1.4 Limitations of the topic

There are several limitations imposed on this work in order to narrow down the scope.

- Our discussion of IDS is limited to Network IDS (NIDS). Host-based IDS (HIDS) are discussed briefly at the end of Chapter 2.
- Deployment evaluation is based on real deployments. Although much more natural than simulated laboratory environments, it is impossible to have perfect conditions which might be used to estimate the theoretical best approach.
- Although scenarios are based on real-life companies, they certainly do not represent every organization of that size. When considering the suggested guidelines, requirements should be re-evaluated to meet the specific needs of the company deploying the IDS.
- We only test and evaluate a small subset of tools available. It is our hope though that these tools represent a good sample of what is being deployed in other networks.
- More thorough evaluation of results would require at least one cycle of the suggested IDS process, roughly three years. Such lengthy testing is beyond our reach.

1.5 Organization of this thesis

The rest of this thesis is organized as follows: in Chapter 2, we present a generic intrusion detection process and discuss each of the phases in detail. The technologies, main benefits, and key limitations are discussed, and an alternative approach is mentioned in Section 2.6. In Chapter 3, we discuss cost dependencies in intrusion detection systems, deployment priorities, and post-deployment options. In Chapter 4, we evaluate a number of security tools which can be used for low-tech intrusion detection, and compare them to a full-scale network IDS. Our main findings and results are presented in Chapter 5.

Chapter 2

Theory

We mentioned in Chapter 1 that automation of IDS in relation to computers is important. Human beings are incapable of dealing with the speed and amount of information which computers generate. They are also suspectable to error, misinterpretation, and it is very difficult to accurately predict their capabilities. Computers on the other hand precisely execute what they have been instructed to, are deterministic in that the execution of the same sequence of instructions will always produce the same result, and their processing capabilities can be estimated in advance.

The five steps of the intrusion detection process (see Section 1.1), form a cascade as depicted in Figure 2.1. Our ability to automate an IDS is highly dependent on the first two phases: our ability to collect quality data [34], and our ability to extract useful information from that data. *Classification* and *Reporting* phases influence the information processing capabilities of the automatic or human operator. The *Reaction* step is both a culmination of the process, and a beginning upon which to base system improvements.

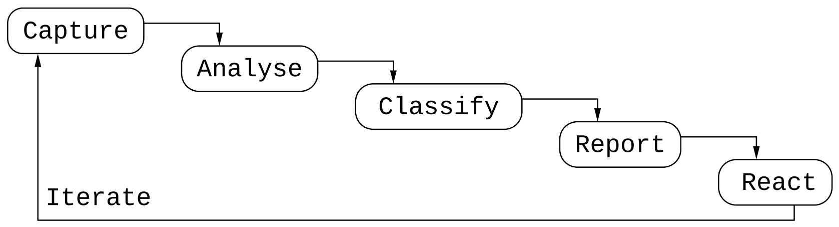

Figure 2.1: A generic, iterative intrusion detection process

The terms *quality data* and *useful information* are further explained in Sections 2.1 and 2.2 respectively. In general, by *quality* we mean that the event data is relevant and complete, but it is harder to determine what exactly is useful information. We use the feedback from the process at the reaction phase helps determine the usefulness of the collected data and fine-tune the system at the next iteration.

Our discussion of the theory is based on Network Intrusion Detection Systems (NIDS). NIDS look at network traffic, and analyze the contents based on some defined rules [32]. The following subsections each discuss one step of the IDS process from Figure 2.1. Typical techniques and methods are presented and limitations of each phase are analyzed in order to expose the difficulties in automated intrusion detection. We also briefly mention an alternative method of accomplishing similar tasks using Host Intrusion Detection Systems (HIDS). In Chapter 4, we briefly return to HIDS to show how it can complement some of our efforts in *pre-IDS* security stage (see Section 3.2.1).

2.1 Capture

The first step of the ID process is similar to that in our office building, the recording of event data. A well known axiom in computer science, GIGO, meaning *Garbage In, Garbage Out*, captures the essence of the problem. The quality of our recorded event data directly influences our ability to extract useful information from it. The following must be noted for successful recording of data:

- Surveillance should preferably cover the entire system we are monitoring. If such arrangement is not possible, we must identify the key areas of interest such as points of entry or high-security zones – this is part of risk assessment and analysis during design of security policies.
- The recording system should at least see the events which occur, and possibly record them as well. Some systems are designed to take samples[1] of the occurring events to increase system capacity. If our system takes samples, we must verify that samples provide good enough approximation of all events.
- The data capture equipment must be suited to the environment and be able to record all possible events which occur.
- We must be able to detect if the event capture equipment is actually functioning properly.

[1]For instance, basic video surveillance equipment may record only one frame per second instead of the usual 25 for PAL.

2.1.1 Methods

"TCP/IP. . . forms the base technology for a global internet" [10]. The data which we must capture (considered in this thesis) is TCP/IP traffic (packets) and assorted Media Access Control (MAC) protocols such as Ethernet. Specially configured hosts called *sensors* [32, 33] monitor the network in an attempt to record the traffic. For intrusion detection purposes, we are interested not only in the payload content of the traffic, but also associated header information for all underlying network protocols. In some cases, further analysis can be carried out by inspecting the associated link layer (MAC) protocols.

Sensors are usually dedicated hosts whose function is to capture data[2]. In order to decrease load on the analysis stations in multi-sensor environments, sensors often perform analysis of the captured data. Because we are interested in as wide coverage as possible, we must configure the sensor to accept data not only destined for itself, but for any host on the network[3] Figure 2.2 shows a typical configuration of an IDS sensor. The following factors influence the behavior of a sensor:

- placement
- performance
- recording capabilities
- network configuration

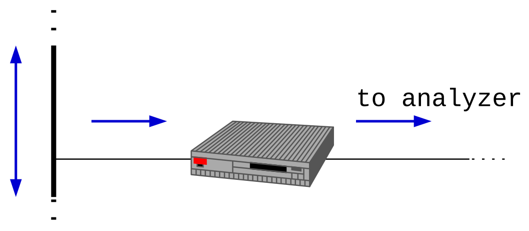

Figure 2.2: A sensor in dual-homed configuration

[2]As discussed in Chapter 3, minimal installations usually integrate a sensor, analysis station, and reporting facility.

[3]This will typically involve configuring more than just the sensor. Section 2.1.2 discusses switched networks, tunnels, VLANs, and other "problematic" designs.

Placement The placement of of the sensor depends on what we are trying to capture [33] and is highly dependent on the network configuration of the given site. In general, small networks have a similar design comprising of a firewall (FW), a De-Militarized Zone (DMZ) [31], one or several extranets, and the intranet. The design of the network is intended to isolate different functionality into separate zones[4]. In terms of sensors, these zones each carry unique traffic type, load, and are suspectable to different kinds of attack and intrusion attempts.

If our goal is to capture traffic going in and out of our network, we should look at placing the sensors outside the FW. In Section 2.2 we will see that the data capture outside the firewall will promote a "loud" system [37]. If the goal of the process is to capture successful intrusions, or misuse by insiders, then we need to capture the traffic behind the FW [33, 37].

In typical scenarios, it is useful to consider deploying several sensors in various parts of the network (outside the FW, in the DMZ, inside the perimeter, in the extranets, etc.). Data incoming from several sensors tends to be more complete than just a single one, and provides us with means of correlation at the analysis phase.

Performance The two performance issues for sensors are the ability to capture all the traffic on the segment, and the ability to record it. When considering current LANs which often use 100Mbit Ethernet, the amount of recorded data becomes large even at low load (at 10% load, aggregate traffic for one day would be around 100GB). Fortunately, not all zones require such high performance figures. Links connecting company networks to the Internet are typically much slower (order of several Mbps), and hence less demanding on the sensor performance capabilities.

In networks, where the traffic load cannot exceed the link capacity of the sensor (eg. Ethernet in broadcast mode), capture capabilities are limited only by the sensor itself. Performance of the network subsystem, processor speed, and memory requirements must be taken into consideration when selecting suitable hardware to perform the monitoring. If it is determined that the sensor is dropping packets, it should be either replaced with a more robust unit, or should be trimmed to filter unnecessary events. In switched environments, sensor link capacity may become a problem, see Section 2.1.2.

The recording of event data presents a somewhat different problem. Although it is not a problem to store 100GB of data using today's disks, it is expensive to store such vast quantity of information for prolonged periods of time or to perform efficient searches. When sensors perform both capture and analysis of traffic, it is

[4]Network zoning are discussed in more detail in Section 3.2.2

possible to greatly limit the recording requirements of the systems by recording only the processed events of interest.

Configuration The sensor configuration depicted in Figure 2.2, shows a host which is dual-homed. One of the interfaces is connected to the monitored network segment, and the other is used for communications with the analysis system [33]. Dual-home configuration brings both performance and security benefits to sensors.

By utilizing a separate interface to transmit the captured data to the analysis station, the sensor is able to utilize full link capacity for capturing data. In broadcast networks, this is somewhat irrelevant, but in switched environments, it helps limit dropping of packets beyond those which we are unable to process. We must realize though that using a separate interface does not guarantee 0% packet loss, only slightly improves our chances. Reasons for this are discussed below in Section 2.1.2.

Using dual-homed configurations for security purposes may not be obvious, but it allows us to configure sensors to be "invisible" [15]. We can configure the monitoring interface to either use a one-way cable, or run without an assigned IP address (IP=`0.0.0.0`)[5]. Because this renders the sensor unable to respond on that interface, its presence on the network is very hard to detect. The second interface is needed because the sensor must be able to communicate with the analysis station.

2.1.2 Limitations

A number of network design issues limit the recording capabilities of sensors:

- Switched networking [32] and virtual LANs
- Tunneling
- Encryption

Switched networking Switched networks present a problem for sensors as only traffic destined for the sensor is channeled to them. The situation is depicted in Figure 2.3. In a) the switch directs traffic destined to a specific host to its own

[5]This might not work on some systems

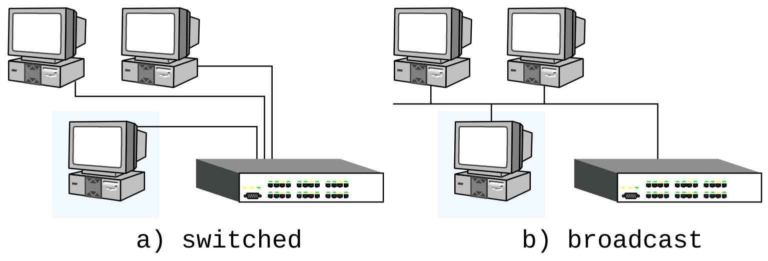

Figure 2.3: Switched vs. broadcast layout

cable. The IDS sensor (shaded host), is attached to another point, and does not see it. In broadcast mode as depicted in b), traffic is mirrored to each of the ports of a hub, and visible to all hosts connected to that network segment.

Typical solution for this problem is to configure switches to have one *span* port which receives all traffic passing through the switch, or to use taps [25, 32]. In most situations, this will work fine, however consider the following scenario: five hosts are connected to a switch, so that there is one sensor which monitors traffic, and two pairs of hosts which talk one-on-one. Because the network is switched, each of the pairs is capable of communicating at the network speed. Given that link utilization is, say, 70%, the aggregate traffic which needs to be delivered to the sensor by the switch is 140% of the link capacity. Hence, at least 40% of packets will not be recorded by the sensor[6].

Although we cannot do much about the dropped packets, here are two useful techniques which can be used to estimate the drop ratio in the above example. Switches usually maintain a record of the number of transmitted packets. The IDS can query a switch, and compare the data with what the sensor has managed to record. Alternatively, if our traffic analysis shows a blunt port scan of a block of addresses, the drop ratio can be estimated by the number of “missing” addresses [25].

Tunneling and Encryption The increased security awareness on the Internet has prompted the use of tunnels and encrypted channels for transferring data. Generally this increases the overall security of the network, but at the same time it presents a problem for network intrusion detection systems. Although we can usually capture the traffic, the analysis might be limited or impossible.

In terms of sensor technology, the only plausible solution might be a proxy system

[6]Actually, considerably more packets will be lost as it is impossible to achieve full link utilization. We are only trying to illustrate the the broadcast port on the switch will be overwhelmed.

which decrypts the traffic, analyses it, and then encrypts it again (and similarly for tunnels). However, its likely that such solution is more likely to bring more problems than benefits. A much more viable solution is to use Host-based intrusion detection and process encrypted data only after it has arrived at its final destination.

2.2 Analysis

Several methods exist for detecting anomalies in network traffic. The most intuitive, and the most commonly used [39] is signature matching (sometimes called “passive protocol analysis” [34]). Other techniques include protocol analysis, anomaly detection, statistical profiles, and policy misuse. Each of these methods can be used on its own, or in combination with the others. They also provide unique functionality and produce different kind of information.

The analysis phase is quite hard to automate. We are often limited by our ability to describe a certain event in an unique fashion, or properly distinguish between a malicious action and normal or spontaneous behavior. The analysis method which one selects is dependent on several factors such as the kind of data collected by the sensors, the storage and processing resources available for analysis, and the way in which we would like to use the resulting information..

2.2.1 Analysis methods

Signatures

Detection based on signatures is often referred to as *misuse detection*, but we would like to reserve the term to mean network abuse by insiders. A *signature* defines or describes a traffic pattern of interest [33]. Signatures can refer to both protocol headers and payload patterns and the choice of analyzing protocols patters or content (or both) will depend on the specific IDS in use and the focus of our interests.

IDS vendors take several approaches to signatures. Several products incorporate special languages which can be used to write custom filters while other vendors prefer to maintain the signature databases themselves. From the point of view of a security administrator who uses a given product, both approaches might be tempting. The decisions are mostly based on the capabilities of the operator, the resources available for IDS maintenance, and the site security policy.

Products which allow custom signatures are definitely more flexible. The security administrator is able to define his or her own signature set to tune the IDS for their network. However, writing signatures is not an easy job and requires broad knowledge of networking and security. Typically, a vendor will provide a set of base signatures which administrators may wish to expand themselves later on. In case of Snort[7] [36], the signature base is maintained by the Internet community, and two separate organizations maintain "official" signature databases (both freely available for private or commercial uses).

Vendor provided signatures on the other hand, only require the administrator to apply patches (or possibly an install of a new version) of the software, and an automated process handles the rest.

Nowadays, the issue of which approach is better seems mostly a thing of the past as in practice all makers of signature-based IDS base their signature sets on either Snort or Dragon rules [43].

Statistical analysis

Statistical analysis is based on building behavior profiles. Network traffic has several properties which can be used to detect anomalies in "average" traffic: load, round trip times, port access information, number of connections, and such. Additionally, we can assign identities to sources and destinations of the network traffic based on their network addresses[8].

One of the benefits of statistical analysis is that it offers some form of analysis of encrypted channels such as SSH or SSL protocols, as well as VPNs. It also provides some means for dealing with tunnels, though not necessarily too well. Utilizing signature matching on scrambled channels is rather futile, using statistical analysis we may be able to detect events that fall well beyond the boundaries of normal usage profiles. Such events can be unusually high number of connections, abnormal load levels for a given link, unusual load patters, or anything else that can be statistically described. Typically, statistical tools gather information on hourly, daily, weekly, and possibly monthly basis although this is dependent on what the administrator deems as viable analysis periods.

An intrusion detection example, using statistical analysis, was provided by Paolo Lucente of the Consiglio Nazionale delle Ricerche (`www.cnr.it`). A network router servicing one of the research labs has been configured to export network

[7]`http://www.snort.org`

[8]Depending on application, these can be MAC addresses, IP addresses, or something different altogether.

flows to a custom-built flow analysis software. An intrusion to one of the molecule modeling systems, running the IRIX operating system, was detected when the analyzer flagged an abnormally high number of connections being established with respect to the system profile. Upon further investigation of the pattern anomaly, an IRC bouncer used to forward IRC connections, has been found. The large number of netflows was traced to DCC traffic through the bouncer.

The graph[9] in Figure 2.4 shows a similar system which detected a DDoS attack against the host `mcdermott.sentor.se`. A more detailed analysis of this case can be found from Section 4.1.2.

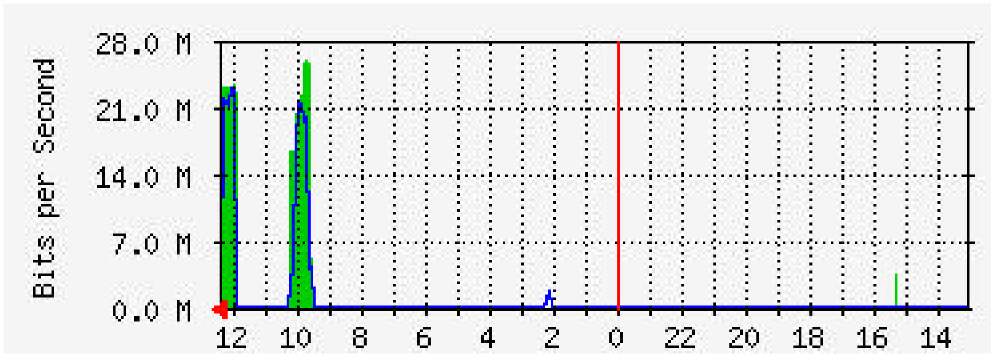

Figure 2.4: Example traffic profile from a distributed DoS attack

Protocol analysis

Protocol analysis is a form of advanced pattern matching based on protocol correctness [16, 30]. A system utilizing protocol analysis contains specification for each protocol which it can analyse. When it captures a packet, it decodes the embedded protocol information and analyses it for correctness. Depending on the system, network, transport, or application protocols may be analyzed.

At least two key issues make protocol analysis superior to plain signature matching. As we compare a packet against a pattern of a correctly built packet, any irregularity will be flagged regardless if we are actually looking for some particular attack or not. This means that a single pattern in the IDS is able to detect a family of attacks. Additionally, instead of static matching, packets are decoded as they would be by the system's TCP/IP stack. This helps combat several evasion techniques such as fragmentation or insertion [34].

[9]Courtesy of Kenny Jansson, Sentor Ltd., Sweden

Several minor issues are also solved by this advanced analysis technique. For instance, badly written signatures which flag all packets containing some given string (eg. `exploit.src`) can cause a good deal of false positives when one researcher sends mail to another describing this particular attack. Protocol decoding can perform some more in-depth analysis of how the given string occurs in the packet, and hence narrow down the detection to real attacks only.

Misuse detection

The term *misuse detection* is used in at least two contexts. It either refers to *signature matching* [34], or to detection of intrusions attempted by trusted insiders.

The difference in approach to detecting intrusions by insiders is that perimeter defenses are usually ineffective, and that intelligence gathering activities are greatly facilitated. Due to employee privacy issues, motivation, and technical difficulties, it might be best to try to combat insider misuse by other means such as employee screening, security policies, and tight access control. However, misuse of resources against foreign sites will likely reflect on the organization and not the guilty individual, and hence motivation exists for monitoring outgoing traffic for misuse attempts.

Even if we are not dealing with malicious activity by an insider, there is always a possibility that an outsider has managed to impersonate a trusted person or gained access to their system. In such cases, systems set up for misuse detection will also be beneficial.

2.2.2 Limitations

Signatures

The major issue with vendor-provided signature updates is the delay. IDS vendors have a long way to go before the process of applying signature patches is near that of virus protection software vendors [33]. Another potential problem is utilization of proprietary protocols, which might trigger some non-tuned signatures, or not be detected at all.

In case of systems which allow signatures to be written by the administrators themselves, one of the greatest dangers is that a given signature is ineffective. Custom written signatures must be well tested against the exploits in order to establish that they indeed trigger upon an intrusion and not some traffic by a misconfigured workstation or such (false positive).

Regardless of whether signatures are provided by the vendor, written by the site administrator, or obtained from a third party, their biggest limitation is that typically, they are oriented at picking up only the specific print for which they have been written. Therefore, each new attack requires that a signature be written for it, causing the signature databases to swell, putting performance strain on the analysis system [16]. While it is sometimes possible to write signatures for groups of related attack techniques, the multitude of software, architectures, and designs limits the potential. To illustrate the problem better, we can compare the couple of hundred signatures used by even the most advanced NIDS nowadays, with more mature, evolved areas such as virus scanners, which handle virus databases of several tens of thousands.

The implication of the above is that in general, we can only detect known attacks using signature matching. Any kind of unknown tools or attacks are likely to slip unnoticed. Further, the sensors which pick up the signatures must be constantly updated with the latest signatures to be able to detect the latest attacks. This causes a slight maintenance overhead in small environments with limited number of sensors, but can be a non-trivial task in large setups.

Statistical analysis

There are several points of notice if a statistical analyzer is used as an intrusion detection tool. Profiles are dynamic (otherwise, they are policies), and can be altered. By generating traffic within the profile threshold values, an attacker can slowly modify the profile to allow other traffic than that originally specified [4].

If a person using a system performs some unorthodox task not within their profile (for instance, uses a new software tool), an alert will be triggered. This will happen regardless whether the activity is malicious or not. For environments where such peeks occur often, building a good profile may be hard or even impossible.

If an attack is carried through a channel which is within a profile, no alert will be triggered. This generally means that statistical analysis is not useful to detect intrusions such as the Nimda worm [8] to sites which receive millions of hits a day.

Some networks pose a problem for statistical analysis of network profiles. Settings such as university computer laboratory classes, or companies with hot desking[10] will have varying daily profiles based on who is using a given workstation. Unless the IDS is somehow configured to match profiles against authenticated users

[10]Hot desking means that employees change their working stations daily, occupying the next free available spot upon arrival at the office

logged onto the workstation, it might be difficult to draw any conclusions based on usage profiles.

In contrast, statistical methods can be highly effective in certain environments such as government or military installations, where working hours, and places are strictly defined. Users are given a set of software to use, and user modifications to the systems are prohibited. Such environments allow the administrators to define very strict behavior patterns with tight threshold values. Anything out of the ordinary can be identified, and reported promptly.

Protocol Analysis

The main limitations, or rather difficulties, in protocol analysis are that a parser must be specifically built for each of the multitude of application protocols (and possibly lower-level protocols) and that decoding all traffic on application level is computationally intensive.

In signature matching system, we typically deal with lower level network protocols such as IP, TCP, UDP, and ICMP. Additionally, we look at the raw contents of the data and try to match strings which indicate some kind of attack. Protocol analysis on the other hand must provide filters for tens if not hundreds of different application-level protocols.

Typically, an IDS capable of protocol analysis will come pre-equiped with decoders for only some of the basic protocols such as HTTP or SMTP. If our intrusion detection requirements call for parsing of traffic not covered by the protocol decoders, additional time will have to be spent during the deployment phase to build one from scratch.

Protocol verification is much more processor intensive than basic signature matching. While a typical PC hardware is able to decode several tens or hundreds of simultaneous connections, it does not scale very well beyond that. This means that in some situation, as for instance, monitoring traffic to a popular web server, the sensor might be overwhelmed with traffic. Although it might continue to process at least a number of incoming connections, it would start dropping packets therefore creating a window of opportunity for the attacker to sneak an attack through undetected.

Misuse Detection

We can any of the mentioned techniques for misuse detection, or even engage the help of HIDS. Misuse detection tends to suffer from political rather than technical shortcomings. Because in effect, we are spying on our own employees, employee privacy issues come into play. The easiest way to resolve these is through the use of a policy which allows the company some employee monitoring for security purposes or limiting the monitoring in such a way as not to gather personal data which might lead to privacy infringements. The extent of misuse detection are dictated by the local laws.

2.3 Classification

Policy Misuse

Theoretically, *policy misuse* is a perfect technique as it generates zero false positives. The policy is aimed at strictly defining what is a normal event, and what is an anomaly. In practice however, policy-based IDS are not easy to implement. We need a powerful description language to define what is normal, or lack of normal, and then we must verify that the policy is not abused.

In policy misuse systems, one defines what constitutes legal traffic (the policy), and looks for events which are not defined in the policy. Because anything that falls outside the policy triggers a valid alert, the method can be seen as providing no false positives. Unfortunately, this is only partially true. This method however, provides a high rate of alerts as anything out of the ordinary (including misconfigured systems, broken hardware, unknowing users, etc) triggers an alert.

In many environments, it is near-impossible to define what constitutes *legal* traffic. Consider a university laboratory where computer science students learn to program network software. The amount, type, and even correctness of traffic is unpredictable.

Policy based systems are better suited as firewalls than intrusion detection systems, or at least should be only used to complement another system, not as the only form of detection.

Statistical limitations

A profile for a network node can be established either by capturing enough data for a statistical model, or by defining a profile by hand (similar to a policy definition). A profile established based on actual data is probably more accurate initially because it is based on facts, not assumptions. It is, however, suspectable to an attacker who manages to modify network behavior while the profile is being established. A profile which is defined by hand will probably need fine-tuning for several weeks in order to obtain the actual profile.

We should note that statistical profiles are typically triggered by “loud” attacks. This might allow attackers to carry out very low-and-slow attacks or information gathering without triggering alarms. Self-modifying profiles are also suspectable to being slowly altered by the attacker in order to allow for a louder, more prominent intrusion within threshold values.

Signatures and protocol analysis

Some systems integrate the classification into the signatures [36]. A signature is typically triggered either by the presence of some value (eg. a status flag in the header), or the presence of some defined number of events within a defined time span. Additionally, signatures may include information about actions which should be taken upon triggering.

Signatures are typically designed to look for patterns which do not occur in normal traffic (eg. two mutually exclusive flags in a header), however in some cases the classification must be based on data content, and it may cause triggering of rules with legal traffic (see Section 2.2).

Equally, the selection of threshold values when the rule should trigger is very difficult, and no exact guidelines exist. For instance, does a connection attempt to a busy server, which occurs once an hour constitute a port scan, or is it just some misinformed users attempting to connect to the wrong system?

Classification values are typically assigned during the tune-up stage of the deployment, when a well trained IDS administrator decides how loud the system should be based on their prior experience and knowledge.

In case of protocol analysis, the classification is much simpler because the validity of the received event can be more thoroughly verified through a more formal approach. We are no longer bound by shortcomings of string matching. However, protocol analysis is quite helpless against attacks which are syntactically correct, but carry malicious data.

2.4 Reporting

Reporting is a key phase of intrusion detection, as it is the main point of interaction of computers and humans. The immense amount of data gathered, analyzed, sorted, and classified by the IDS must now be presented to the human administrator. The ability of the human administrator to react to an intrusion and take appropriate actions, will depend greatly on the his ability to process the information reported by the IDS.

2.4.1 Methods

As discussed in Section 2.1.1, a typical installation will include several sensors placed in different areas of the network. One of the key ideas in reporting alerts is to centralize the reports received from these sensors using a single console. A centralized system makes it easy for the human administrator to keep an eye on all parts of the network simultaneously, correlate events, and prioritize response accordingly.

A conceptual scenario with multiple sensors in various network zones (Section 3.2.2) is illustrated in Figure 2.5. As the sensors are typically dual-homed (Section 2.1.1), the sensors and console form a separate, isolated network.

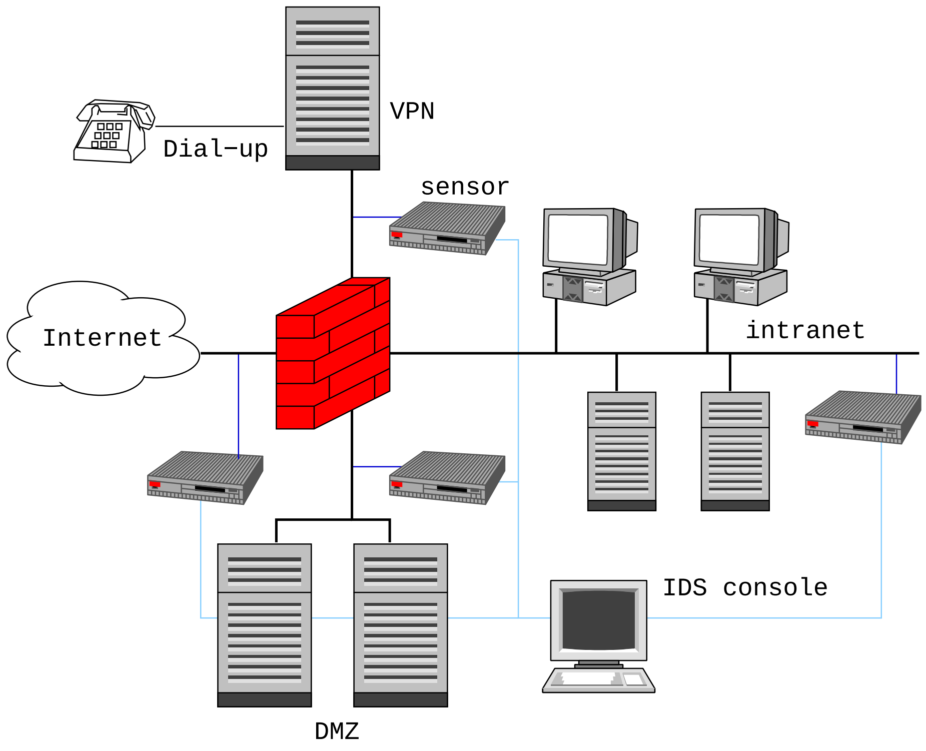

Figure 2.5: Typical sensor and console layout in multi-sensor configuration

The sensors and console may interact in either push or poll mode. In poll mode, the the console periodically queries the sensors for their capture data, while in push mode the sensors send data to the console, usually in *real time*. Both methods have advantages and disadvantages as discussed below in Section 2.4.2. The type of interaction of sensors and the console is largely governed by the kind of response designed into a system. Real-time or near-real-time response will need to use push technology, while delayed response can relay more on periodic polling.

A typical reporting console display [12, 13] contains an organized event summary. These events have already been filtered, and classified by the underlying phases. Today's reporting console software is also designed to correlate events from multiple sensors, as correlation improves the probability and accuracy of the detection [1]. Hopefully, in the near future, vendors will also support some uniform alert format [11] which will allow better multi-vendor cooperation of sensors and reporting consoles.

In some situations, the administrator of the IDS may wish to receive reports of some extremely crucial events in near real-time. Such alerts can be sent either by email, Short Message System (SMS), or some other fast notification method. This approach should be avoided unless the IDS is well configured as someone can launch a rather unpleasant attack against the human administrator by eliciting the IDS to send alert messages in the middle of the night. The inevitable result of such action would be the permanent disconnection of the service.

2.4.2 Limitations

Human factors The major limitation of the reporting system is the human administrator. If thousands and thousands of events scroll by, it is impossible for a human to understand them and take action. The impact of the overwhelming amount of information delivered to the administrator is combated with proper analysis and classification. If it is not handled properly, humans have a tendency to eventually switch off features which are too loud. It is hence plausible that an attacker will cause the system to go off in hope that the administrator modifies the classification rules to be more permitting (i.e. be harder to trigger).

Availability and Security Availability of the reporting console is a problem. Typically, the IDS net is not directly connected to the intranet and the reporting console must be accessed locally by the administrator. From a security standpoint, this is the preferred solution, but usability and availability of the IDS reports suffer greatly.

The reporting console is a focus point of the IDS. It receives all the alerts and typically is the only remote interface to the sensors. A compromise of the console would likely cause the IDS to be disabled, or worse, to hide the intruder's tracks. By connecting the console to the intranet, we create a point of entry to the IDS. On the other hand, a console which is only connected to the intrusion detection network suffers a great usability penalty as the administrator must be physically present in order to audit the system. Further, since there is no route to the mail server, e-mail alerting is not possible.

During deployment, these factors must be considered, and we must fine balance between security and usability. The choice is made somewhat easier if we start with with single-sensor/console combo as suggested in Chapter 3.

Push vs. poll Both push and poll consoles have their weaknesses. With push technology, the IDS may either flood the console or administrator with alerts, or be quiet. In case of floods, the administrators are likely to be overwhelmed by the information pouring in, causing them to miss events. If the system is quiet, there is no good way of verifying that it is actually operational. One plausible technique is to use timestamps in a fashion similar to syslog marks in Linux logs, however the problem is only decreased, not eliminated [31].

In polling, the obvious problem is that the IDS is always well behind the attacker. Despite high polling frequency, there is always a window of opportunity for the attacker to carry out their doings before the event is reported by the system.

2.5 Reaction and response

Whereas detection is the first step of our process, response to an alert is the first stage of incident handling. The purpose is to establish the best course of action to handle the event. For instance, it is unfeasable to expand resources on trivial port scan directed at a non-existent server, however we should definitely look closer at a well-targeted intrusion attempt into a web-server.

2.5.1 Methods

Automated response With automated response, the IDS administrators give the system power to take action upon detecting an intrusion attempt or an intrusion. The techniques range from increasing response delays in TCP handshakes, to blocking IPs, to resetting connections, and removing routes.

Automated response has a major advantage over a human operator, in that it operates fast enough to shut down an automated attack script. It can also deter probes and other reconnaissance methods which the attackers usually use to map the target network.

As we will discuss in Chapter 3, much of the IDS-related costs come from administrator salaries, and automating the response to minor problems helps direct the analysts attention towards the critical ones. Combined with the reaction speed at which the system reacts, we can view automation as a cost-effective way of IDS response. However when we are considering deployment, calculations of losses due to the system backfiring must be taken into account (see Section 2.5.2). Further, an automated system requires a deep understanding of the IDS and the environment, and must be proceeded by thorough tuning stage (see Section 3.1).

Manual response Many experts choose to do away with the automated response and concentrate efforts on optimizing manual response instead. We can view manual response as having somewhat different properties and goals than automated one. We cannot expect the reaction time to be near-real-time and we must have an operator who is trained in incident response. To our benefit though, the response will be tailored to the specific incident, can be followed by in-depth analysis and recovery, and lead to problem eradication.

If our goal is to deter intruders at any cost, then automated response is a plausible solution. If, on the other hand, our strategy calls for more in-depth investigation of intrusions, maintaining the systems online, and uninterrupted flow of operation, then manual response may prove a better option.

Whereas the automated response is often aimed at stopping the intrusion in progress, manual response strives to give a balanced, methodological approach to solving the intrusion problem. We can see the response as a four-step process [33]:

1. Containment
2. Eradication
3. Recovery
4. Lessons learned

A requirement for successful execution of these four steps is a well trained incident response personnel aided by proper documentation. The combination of personnel and response time are the main cost factor in manual response. The growing number of administrators required to minimize response time quickly becomes the dominant factor in IDS maintenance.

The procedure for manual response is typically as follows: locate the problem area, and the systems which have been compromised (Containment). Patch the security hole which allowed the intrusion to proceed, verify that other systems do not have this problem (Eradication). Finally, recover the systems which took part and document the incident (Lessons learned).

2.5.2 Limitations

Manual response The main limitation of manual response is the inevitable delay between an alert and human reaction. An automated exploit script may perform the tasks in about 30 seconds [31]. It is simply impossible to achieve this kind of response time from humans regardless of available resources.

If we would like to have a near real-time human response, we must assume, say, 5 minute delay. For a 24/7 system, it is feasible that a security console is monitored by at least 5 administrators[11] working shifts. We can clearly see that costs begin to mount up with such a scenario. Hence, for manual response in small corporations, we should assume a feasible “next business day” delay.

Sadly, the damaging factor in response may well be the human involvement. Without proper training, the response team may not only miss an incident, but by improper handling of it, destroy evidence, or misinterpret data, leading to costly recovery and follow-up procedures [12].

Automated response Ability of an IDS to automatically stop intrusions seems too good to be true, and in fact many experts see it as a sure way to shoot oneself in the foot [23, 33]. A typical scenario might be a spoofed attack with the source address pointing at the DNS server. An automated system responding to an attack may block access to the DNS server, hence disabling the entire network.

Automated response also falls in view of false positives [25, 33, 44]. We would like our system only to react to actual intrusions, not randomly shut systems down. It is advisable that automated response be limited, and that the administrator knows what they are doing before it is deployed.

[11]A yet more realistic estimate is 7 administrators which allows for incident response while maintaining watch over the console.

[12]Information provided by Ibas, a leader in data recovery, indicate that a considerable number of systems is damaged by administrative personnel who is not properly trained in incident handling and data recovery. `http://www.ibas.com`

2.6 Alternative approach

The general alternative method is to perform internal system auditing to detect attacks and intrusions. The systems which perform such functions are grouped under the name Host Intrusion Detection Systems (HIDS) [32]. In general, HIDS works by auditing system integrity, monitoring changes to files, performing process accounting, monitoring logs, and user behavior inside the systems. The major benefits are that it can be used at endpoints of encrypted channels, limit the capabilities of system misuse by insiders, and test system integrity.

Some cons of HIDS include the fact that it is highly platform dependent, and difficult to deploy in heterogenous environments and that once the system itself is compromised, the attacker might be able to replace the HIDS as to hide their presence. Typically, best effects are achieved when HIDS is combined with NIDS [33, 20, 21].

In Chapters 3 and 4 we briefly discuss the application of HIDS as a *pre-IDS* stage for NIDS. The idea is to show how simplistic use of login facilities can be used to improve network maturity.

Chapter 3

Deployment priorities

“One thing I think is missing... [is the] need for adequate network maturity before bothering to spend money on IDS.” [42]. Implementation of security features should be part of the network infrastructure design. The late recognition of security needs and ad-hoc implementations later on are likely to cause problems or fail altogether [28].

Each area of network infrastructure implementation is assigned a budget, and unfortunately in most cases, the budget it too small to cover the design wish-list. We must set priorities as to what to implement first. We must also remember that the system must be maintainable and provide some kind of return on investment, otherwise the resources which we spent on it are lost. Our prioritization should be based on risk analysis and assessment, our countermeasures, and ability to react. We must have a clear view of what we expect from our security measures.

Figure 3.1 suggests a very simplified view of stages in building of network security. Going from left to right, we expect the overall security level of the network to increase. The IDS is just one plausible step in this building process, and not necessarily a required one. Its implementation depends on security goals and resources.

The stages in Figure 3.1 are hard to define rigidly, but in general we can think of them as follows: the *Bare* stage represents a point where a company leases a line and connects a network to it without implementing any security measures. This is typical of very small, non-IT organizations where computer-related competence might be low. We define the *Basic* phase when the company implements a FW, and possibly divides the network into separate zones such as DMZ or extranets. The *basic* stage is what is typically found in company networks, and with the increasing popularity of personal firewalls, “hardware” firewalls, and other solutions, many companies who have been at the bare stage, are quite effortlessly

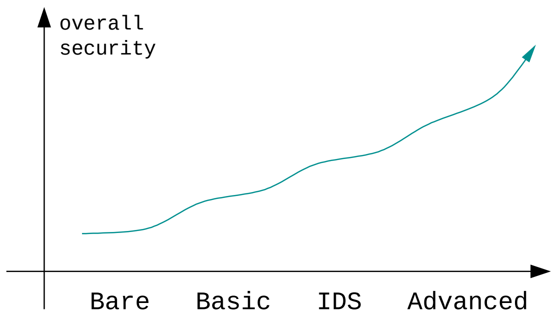

Figure 3.1: Security phases in small corporate networks

embracing *basic* solutions.

Beyond the *basic* stage, companies begin to look for active defenses. This typically involves active monitoring of network traffic, or HIDS. Whereas the gap between the *bare* and *basic* functionalities is quite simple to close, the deployment of an intrusion detection system is a large and difficult project. Without a proper design, and a proper deployment strategy, an intrusion detection system is likely to bring large costs mediocre benefits, which is the exact opposite of our goal.

Beyond the IDS phase we have defined an *Advanced* stage, where anything from large data-mining IDS systems to verified platforms can be found. Typically, this is were large organizations such as banks, military, or government institutions will be heading. Small companies rarely have the resources (or needs) to pursuit such designs.

In this chapter we focus at priorities in design and deployment of network intrusion detection systems. We start off with a review of factors which influence costs at various stages of the process, and their inter-dependencies. We then discuss how to prioritize features based on security requirements, and investigate possibilities of reducing costs by utilizing a *pre-IDS* stage. The chapter is closed with a discussion of things which should be considered after the initial deployment has taken place.

3.1 Costs dependencies

In analogy to our building office from Chapter 1, there are three ways one can approach the implementation of IDS in the network environment: build, buy, or rent. For companies and organizations which fall within the scope of our discussion, building custom IDS is not feasible. The question is narrowed down to whether purchasing a system from a vendor or outsourcing intrusion detection as a service from a Managed System Security Provider (MSSP) is a more appropriate path. We will place opensource systems into the same group as commercial software.

Below, we analyses the costs of different aspects when purchasing an IDS from a vendor or outsourcing it as a service from a MSSP (see Figure 3.2 for dependencies). The cost review will us help understand the prioritization at different phases of the process as discussed in Section 3.2.2. Failure to understand the dependencies in cost factors of IDS systems are likely to lead to short-term savings, and long-term cost increases.

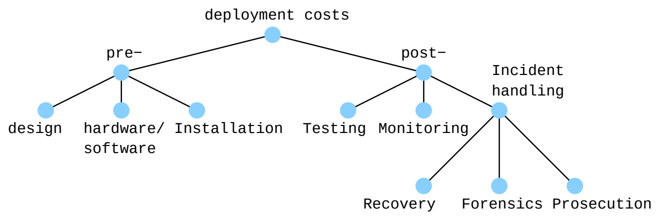

Figure 3.2: Costs layout for intrusion detection system deployment

Design The design of an IDS implementation involves a review the existing systems and infrastructures, developing functional and technical requirements, specifying the implementation procedures and defining the test plan. Although in the scale of the project, the costs compiled at this stage are insignificant, there are a number of issues which apply.

A successful implementation of an IDS requires that a *Security Policy*, *Incident Handling Policy*, and a *Chain of command* exist. We should pay attention that our IDS design does not infringe on any of the rules declared by those documents, such as employee privacy.

It is possible that at least one of the mentioned documents does not exist. We will need to account for that by increasing the IDS development budget to cover the implementation of the missing policies. Even with some preparatory work already in place (we assume that if an IDS is being considered at least some security measures have already been implemented) the work requirement of building policies is easily measured in man-months.

We should concentrate our efforts on building a feasible project plan, a set of requirement specifications (and associated acceptance criteria), as well as functional specifications and testing plan [22]. In the later stages of deployment, when we think about assigning values, the ability to quantify tasks, and compare them with our design documents allow us to estimate if we are really progressing in an efficient manner. We must also understand, that before the system is put in place, these documents form a link between management and the technical stuff in charge of the project.

The design should aim to create a system which could be implemented for a period of at least three years, with annual reviews concerning feasibility, functionality and effectiveness [24]. The period of three years has been chosen based on several criteria: the lifetime of software and hardware, predictable growth, and infrastructure changes. The period is long enough to render the costs associated with design low enough despite a fair amount of required work.

Software and hardware There are two classes of software we can utilize: commercial vendor IDS, and open source – free software. The problem is that free software does not necessarily mean it is actually free to deploy, and commercial software does not immediately mean it has to be expensive. What we must look at are the costs for the given software over its lifetime. It is hard to estimate to any degree of accuracy how long a given software will be used, but based on experience we can state that about three years is the period when software becomes obsolete[1].

In our computations of software costs, we should look at factors such as time spent for installation, whether all the needed features are required, the scalability of the software with respect to our company's growth strategy, trust, software continuity, and alike. Although for commercial software, the initial price tag may seem excessive, the cost should be spread out over the lifetime period.

Final software costs are brought on by scalability. Surprisingly many systems become expensive when we upgrade the number of licenses in use. Especially when license upgrades are only available in large increments, a small growth in the company size may require to a considerably overgrown, expensive license.

Typically, hardware becomes obsolete slower than software, but we must also count about three years as a plausible lifetime[2]. On the other hand, unless we plan to provide very long event history, or do complex trend analysis, we can limit initial hardware costs by utilizing older desktop machines.

[1]Estimate is based on version revisions of anti-virus software, secure authentication servers, firewalls, and other security-related software.

[2]Many of today's computers come with a 3-year warranty, which provides us with a reasonable life-time estimate

There are twofold opinions on using older hardware for IDS; if there is money to deploy an IDS, there should be money for up-to-date hardware – if you are going to be doing intrusion detection, you should do it right. However, in small environments, or for limited trial runs, decommissioned hardware can provide a cost-effective approach to evaluating IDS needs.

Small hardware-related costs are caused by incidents which we decide to pursue further. The reason is the need to take down and replace part of the hardware (harddisks) which serves as evidence of an incident. Typically, the hardware will be made obsolete by the time the legal procedures for the incident are completed, and the hardware might be reused. Because many companies choose not to report incidents, or only to store backups of the incident data, this remains a minimal cost in scale of the entire project.

An interesting idea is to expand slightly more resources on building event database systems with hot-pluggable drives. Apart from obvious reasons such as hardware failure, the benefit is increased uptime, as disks containing evidence data can be removed from the system without powering it down.

Installation The actual installation of the NIDS is not difficult or even time consuming. With proper preparation we can narrow it down to installing software and hardware for sensors and consoles, and rearranging the network cabling to allow for the intrusion detection -net to be separate from the rest of the network. Even in multi-sensor environments, this is one of the simplest and cheapest stages of the process.

Fine-tuning and testing Fine-tuning of the IDS is a highly critical phases as it has a heavy impact on all other future costs. It is also one which should be thought more of as a separate, ongoing process, rather than a one-time effort. Based on some testing performed by the Network Computing magazine [39], and other NIDS deployments, we should count spending a number of days at this stage.

Monitoring An IDS which is not monitored is a waste of time and money. We must hire administrators who watch over the alerts and handle incident response. Typically in small environments, we can count on one administrator who will periodically monitors the reporting console. His or her workload will depend on many factors such as how sensitive the IDS is, where the sensors are located, the accuracy of alerts, and others.

In ultra-secure environments, we might be required to run 24/7 intrusion detection monitoring, and this is were major staffing costs come into play. Effectively, we need at least seven[3] trained professionals to do little else than be on call monitoring and responding to incidents. In small companies, this is clearly infeasible, and such service is much better achieved through Managed System Security Providers (MSSPs). Although these are not cheap, they are more cost-effective as a MSSP can utilize the same administrators to monitor a number of networks simultaneously[4].

Some ideas about staffing costs in relation to incident handling are described in Figure 3.8.

Incident handling and aftermath Costs of incident handling and aftermath vary, but in general are quite high due to the amount of work needed, and the care which should be exerted. We can divide incident aftermath into three areas: the recovery phase, the forensics phase, and the prosecution.

System recovery phase is inevitable, but fortunately it is the least expensive of the aftermath stages. It involves rebuilding the damaged systems (typically from scratch), and patching the holes which allowed the intrusion. Most of the work falls within the duties of the system administrators, and they should be able to handle this in reasonable time. Depending on system criticality, the available resources and our further plans, we can either opt for rebuilding the damaged system from scratch, or building a mirror system while analysis of the incident takes place. In general we can say that limiting aftermath to recovery-only limits provides a bounded limit for the phase costs.

Forensic phase aims at analysis of the intrusion for the purpose of legal prosecution or academic research. Forensics is expensive. During the Honeypot Project Forensic Challenge [14], the reported time spent, for complete analysis by top experts, was about 80 working hours. Although legally-oriented analysis may require less technical depth, it requires great care and attention to preserving and verifying evidence. Costs associated with being sued because of false evidence can ruin a company – or at least badly dent the reputation.

Prosecuting the intruders is a lengthy process, and due to legal fees, a costly one. The positive side is that it is one plausible return on investment of the IDS. If the intruder is successfully prosecuted, the company will be able to claim damages and recover its costs. An excellent presentation about costs and difficulties in prosecuting computer crimes was given during the NordU2002 conference [29].

[3]Staff estimations for CIRT by Nixu Ltd.

[4]Information provided by Sentor Ltd, Sweden

Return on investment Return on investment is the value we associate with the benefits gained from our system. The benefits can vary from minimizing damage, to providing information which decreases workload of administrators, to providing evidence which can be used in a court of law to seek compensation from the attackers [7].

As office buildings might be equiped with sprinkler systems which aim to put the fire out before it spreads, so do intrusion detection systems aim to minimize the damaged caused by malicious intruders. These kinds of benefits are hard to define financially, however a number of studies has been made [7] which try to use regression testing to assign RoI values to network security measures. In the same way, video surveillance provides hard evidence which aids in identification of the burglar, or charging them with a crime in the court of law. IDS provide audit trails, which can be used as evidence or tool to trace security deficiencies..

3.2 Process breakdown and prioritization

3.2.1 Security prior to intrusion detection

The difficulties, costs, and maintenance of IDS systems lead to a question if there is some more cost-efficient ways to improve network security? Is it possible to implement any of the IDS functionality without an actual intrusion detection system. Can we isolate some common problems, and find small, easy, solutions? We introduce an idea of a *pre-IDS* phase, a set of improvements which are designed to mitigate most of the common problems related to network intrusions.

We make the following basic assumptions about the current security deployment:

- The firewall has been designed, configured, tested, and deployed successfully, and a policy exists which defines adjustments to rules.
- The network has been structured to isolate functionality, and create zones which could be considered as separate networks. The zones have been created based on functionality, host visibility, and security considerations.
- The currently deployed security mechanisms are maintained and supervised. This includes system log monitoring, and system updates upon patch releases from vendors.

Our capabilities are going to be limited, especially with respect to the classification and analysis stage (see Chapter 2). The strengths of the *pre-IDS* lie in isolating the events of interest with a low false-positive rate and reporting them in clear fashion. This allows a human to interpret the results quickly, and react accordingly.

Recent studies have shown that up to about 70-80% of corporate security problems originate from within the companies [38]. Although the detailed list depends on our security policies, some of the typical events of interest which we would like to detect are:

- Address spoofing. Are there any packets on our network with interesting looking source address?
- Address spoofing #2. Are there any new bindings between IP addresses and MAC addresses?
- Has any of our computers put any of their interfaces in promiscuous mode[5]?
- Has any of our computers started sending or receiving more traffic than usual?
- Has any of our computers started to listen on a port which was previously closed [41]?

Our goal is to transform the situation depicted in Figure 3.1, and make it more as the one shown in Figure 3.3. Although assigning values to security is hard (or impossible), the idea should be clear.

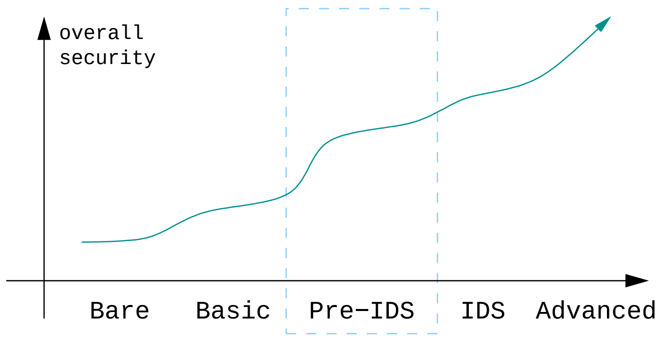

Figure 3.3: Introduction of pre-IDS phase

[5]In Ethernet, an interface in promiscuous mode listens not only for frames addressed to itself, but to all frames on the network.

Spoofing Spoofing can be used twofold for malicious purposes in LANs; either claiming an IP other than own, or the MAC address. Attackers will spoof for a variety of reasons, including masking their identity or switch trickery.

The *false positive* rate for detected spoofs is very low, because under normal circumstances, no legitimate action should require spoofing of any kind. Since new bindings of hardware addresses to IP addresses require physical changing of the network card, the administrators are likely to be aware of such action. The only major difficultly lies in using DHCP which might assign new IP addresses on too frequent basis. Typically the problem can be avoided by using long timeouts, or configuring DHCP to bind the same IP upon new request.

Spoofing can be detected in a variety of ways. A small utility called *ARPWatch* can be used to detect and report changes in ARP/IP address pairings. Additionally, routers and firewalls can be configured to block spoofed traffic passing through them. Spoofing can still be carried out within the local area network, but its impact can be limited by utilizing switching.

Spoofing detection applies to both pre- and post-intrusion situations. Since, usually, attackers will want to masquerade their intrusion attempts, there is a good chance that the attack will be noticed in time. On typical small LANs installations, spoofing detection will be useful in detecting misuse by insiders who want to masquerade their identity.

Because of its relatively low false-positive ratio, and simplicity, we recommend that anti-spoofing mechanisms be present continuously, and regardless of whether a proper IDS is implemented or not.

Sniffing Typically, Ethernet systems listen to traffic, and pick up frames addressed to them only. However, it is possible to put the system in promiscuous mode (as is done by the IDS sensors for instance), where all traffic is passed on to the TCP/IP stack. If on a given network, a machine goes into promiscuous mode, it is a strong indication of abnormal activity (though not necessarily malicious). There are a number of legitimate uses for systems entering promiscuous mode, however, these uses are mostly for maintenance purposes and suggest that the supervisor knows of this activity. As promiscuous mode typically requires administrative privileges, if the sysadmins are not aware of it, its a good indication of malicious activity.

We hence judge to treat machines which enter their network interfaces into promiscuous mode, as worthy of notice [17]. The suspicion is that such a machine has been compromised, and made to sniff network traffic in an attempt to capture useful information for the attacker (passwords, confidential information, etc).

Detection of interfaces in promiscuous mode is quite difficult, but in some cases possible [17]. Several techniques exist and tools have been written to exploit these techniques. Although a single tool might not provide a definite answer, correlation between tools using different techniques might give strong indication of a promiscuous interface. It should be noted that it is still quite possible to hide a system which is sniffing the network.

There are twofold problems with promiscuous mode detection: it is somewhat unreliable, and it is post-fact. If a hacker has put a system into promiscuous mode to sniff network traffic, an unknown time has passed since the system was actually compromised. The investigation of the machine is likely to be labor-intensive because the running system can no longer be trusted to provide accurate information (eg. a rootkit has been installed).

We recommend that anti-sniffing checks be run at least once in a while, perhaps in correlation with a vulnerability scan [41], and not as permanent deployment. A short evaluation is given in Section 4.1.5.

Traffic On typical networks, it should be possible to establish long-term statistical models for traffic [2]. Unexplainable variations to either netflows, load, or latency might indicate that some system is experiencing some abnormal activity. For instance, a drastic increase of netflows is a strong indication of a port-scan, or Denial of Service (DoS) attack. It can also indicate an increased activity from the system's side as mentioned in Section 4.1.2 leading to investigation by the administrator.

Variations in traffic patterns are not uncommon. For instance, downloading large files may temporarily cause high loads. Using correlation of different anomalies, we can increase the probability of a true positive. All prolonged variations from established patterns should be considered suspicious.

Traffic patterns can be observed either with hosts which monitor network traffic, or by utilizing network hardware such as routers and switches which can export traffic statistics using SNMP. In Section 4.1.2 we provide real-life evaluation of two cases of using pattern anomaly analysis.

Services Typically, in networked environments, systems will have open ports, with some services listening on them. It is unusual, however, to have service ports open up without interference from the system administrator, and may indicate that some backdoor has been set up by an intruder. By periodic scanning for open service ports, it might be possible to determine if any machine is running extra services, possibly malicious [41].

The false-positive rate in determining open ports varies greatly between systems, tools used for scanning ports, and environments [41]. We are prone to receiving false positives from open-connection ports, UDP ports, buggy scanners and so on. Additionally, because we are in effect performing an authorized malicious-like activity, any other monitoring system we have is likely to flag possible intrusions.

As with the *sniffer* issue above, scanning for ports is a post-event activity. If we find open ports on systems which should not have them, the intruder is already in control (possibly partial) of the system. Because port scanning does not provide an audit trail of activity, we can only guess as to the period during which the intrusion occurred if our scans are done at periodic intervals.

Service scanning is probably most feasible in the blue and green zones. In all but the smallest of environments, the process must be automated to be feasible, and a track record of previous scans must be maintained.

We see the major benefit of this method being in conjunction with vulnerability scanning, where we attempt to match the identity of found services against known attacks.

3.2.2 Prioritizing deployment

In this section we discuss priorities for a successful deployment of a network intrusion detection system in a small corporate network. At this scale, building custom IDS solutions is not a cost effective (or even plausible) option. We hence concentrate on using off-the-shelf software and through outsourcing the intrusion detection solutions from a MSSP.

We assume that, on average, a company that looks for deploying a system does not have a dedicated security analyst on staff, and the job of designing and deploying an IDS is directed to the system administrators. As the selection of the proper solution, and the right deployment is crucial to the success of the project, we suggest that the first step is to hire a security analyst who is familiar with deploying intrusion detection systems. There are several reasons why we should let an expert in the field handle the initial procedures:

- The analyst will be able to assess the current state of the network, and suggest improvements to facilitate the deployment.
- The analyst will have in-depth knowledge of existing intrusion detection systems, and will be able to assess which one is most suited for the environment at hand.

- The analyst will be able to provide policies for operation, management, follow-up and upgrade of the IDS.
- The analyst will provide company's own staff with enough training to maintain and administer the IDS.
- If we ever need assistance later on, a properly set up system will aid external experts after the project deployment.

The initial expenditure for hiring an expert to design and implement the IDS may seem like an extravagant way to start. Management will be sceptical, and probably suggest that the administrators find "a more cost-effective solution". As we pointed out before, patching up security designs at a later stage tends to be more expensive. By engaging professional help, we aim at eliminating the scenario where a non-functional solution is deployed providing both a false sense of security, and the need to re-implement the system into a more functional one later on.

Second general issue that we should consider is that of scale. How well should our IDS scale? In Section 3.1 we have presented the costs which are associated with the deployment of a NIDS. To recap, a large portion of those costs is the design, implementation and training required to operate the IDS. We must then aim for a period of at least 3 years as a feasible life-time during which our system should only need patching and maintenance upgrades.

In a three-year cycle, we recommend that the system be designed to scale an order of magnitude up. This will negate the need for costly redesigns when the rest of the network infrastructure is upgraded.

The rest of the discussion below follows the generic IDS process (see Figure 2.1 on page 6) from Chapter 2. The prioritization has been based on a typical scenario one might encounter. Adjustments for local peculiarities should be made as necessary.

Capture

By far the most popular type of NIDS currently is signature based [31], and therefore we suggest following this path as it provides the widest range of available products and support. Signature based systems benefit greatly from properly structured networks. The first step in the actual deployment of the NIDS is to verify network configuration. We need a layout which allows isolation of network segments in a way that helps decrease the complexity and workload of the sensors.

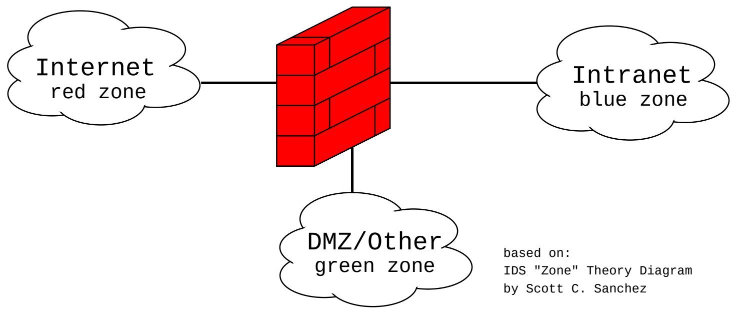

Figure 3.4: Network zones

Dr. Steven Bellovin in his recent post[6] on the Bugtraq mailing list [6] has summarized intrusion detection goals very well by stating that: "... you are under attack, more or less continuously... what you really want to know — that someone has gotten through your (other) defenses." In Figure 3.4, we present the idea of network zones [37]. The zones represent a simplified network layout. Typically, the firewall allows some traffic to the green zone (eg. public web server, mail, etc) and blocks all outside traffic directed at the blue zone (intranet). As Dr. Bellovin's comment, placing an IDS sensor in the *red* zone will tell only tell us what we already know. We are then left with a choice of sensor placement in either the *green* or *blue* zones (or in both).

The choice of which one to deploy first, in single-sensor environments, depends on the purpose our IDS serves:

- The sensor should be deployed in the *green* zone if our goal is the monitoring of our visible services.
- The sensor should be deployed in the *blue* zone if we are after misuse detection, and after-the-fact[7] security incidents.

By increasing the complexity of network design, we can partially combine the sensor placement benefits from both *blue* and *green* zones. We block traffic originating from the intranet at the firewall, and instead use proxy system in the DMZ (see Figure 3.5).

The layout presented in Figure 3.5 allows us to monitor traffic from both intranet, and DMZ with one sensor. However, this solution does not take into account

[6]April 19th, 2002

[7]The notion that someone has forced our firewall indicates we already have a serious security problem on our hands

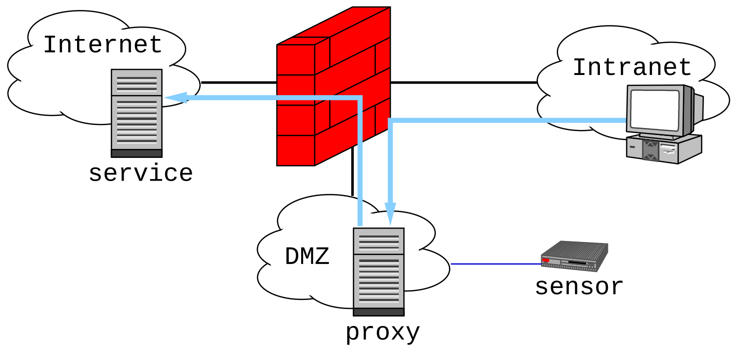

Figure 3.5: Proxying connections through the DMZ

that if the attacker can force the firewall directly, without going through DMZ, the sensor will not pick up the attack. Furthermore, it is impossible to monitor misuse detection where attacks originate in the intranet and are also directed at the intranet. As a recent survey [45] suggests that the largest threat is from the inside, we recommend that, in single-sensor environments, sensors be placed on the internal network first.

Deployment of multiple sensors should follow the same strategy as described above with respect to the single-sensor installment. Divide the network in such a way that it is brought down to the level presented in Figure 3.4, and then install at least one sensor per *blue* and *green* zone.

Installment of sensors in the *red* zone should be avoided unless the goal is either some type of data mining and correlation or ID research (see Section 3.3.3). Although the *red* zone would allow the capture of all but insider attacks directed at the intranet, the amount of "noise" far exceeds the usability and maintainability in the longer run.

An exception is a temporary installation of a sensor in the *red* zone to help assess attacks directed at our network [33]. This technique might be useful for either inexperienced IDS administrators, or as a periodic check to verify that our initial assessments do indeed hold. Applications are rather limited though, and we must keep in mind that the fast-changing exploit world will render any data gathered out of date very quickly.

Network zoning has an additional advantage for the following phases of the intrusion detection process. Zones isolate services into separate network areas, thus allowing us to only search for a relevant subset of signatures. For instance, webserver attacks are irrelevant in zones where no web servers exist. Of course, we would still like to know whether a given attack has penetrated the firewall (hence we could use a generic web-traffic rule match), but we no longer need to worry that a given attack noticed by the IDS was successful.

Figure 3.6 summarizes the discussion.

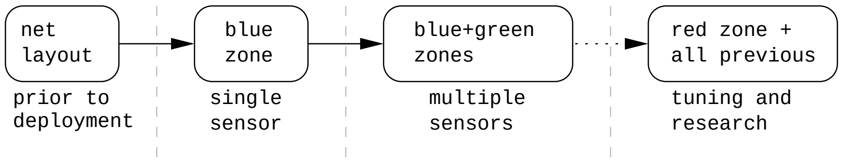

Figure 3.6: Summary of sensor deployment priorities

Analysis, classification and reporting

From the human perspective, the *Analysis*, *Classification* and *Reporting* of events captured by the sensors are typically combined and displayed on the IDS console. The console subsystem is responsible for delivering the information in such a way that the administrators can investigate alerts and prioritize their response tasks.

As discussed in Section 3.2.2, it is often plausible to start off with a single sensor. It should be no surprise that in such environment it makes sense to combine the sensor and reporting console into one system. The benefits gained are eased maintenance, no need to build a IDS network or distribute signatures to sensors, and only one bastion host need to be maintained.

We base the maintenance benefits on several issues. Only one system has to be configured and maintained as a bastion host thus reducing time the administrators spend keeping the ID systems ultra-secure. As the sensor is now effectively under human observation at the same time as the console, we minimize the risk that the sensor ceases to function without anybody noticing it. There is also no longer the need to ponder the push/poll problems (see Section 2.4.2) — polling makes little sense as it provides no benefits, and requires extra storage and management resources.

Although many systems support remote maintenance of sensors (this is especially easy with UNIX running `ssh`[8]), distributed sensors still require more attention during signature or operating system upgrades. As we now have one system, additional maintenance benefits are obtained. Although this does not make for large savings, it does play a part in cost saving in projects with highly limited budgets.

[8]Secure Shell, see `http://www.openssh.org`

At the time of this writing, NIDS typically come with proprietary sensors and consoles, which means that during the selection process, we must match both sensor and console capabilities against our requirements. The situation is changing as the Internet Engineering Task Force (IETF) working group has proposed a standard for sensor and console communications. Also some systems such as Motorola's *Intrusion Vision* are able to understand multiple sensors.

We propose the following three steps to aid in the selection of the reporting console:

1. Works with the sensors we plan to use (required)
2. Supports multiple sensors for when we expand our intrusion detection efforts
3. Supports multiple types of sensors

We mentioned before that humans are often the limiting factor in many computer systems due to their response time and data processing capabilities. Therefore, as a baseline, the decision what kind of IDS we can use is largely dependent on the ability of our administrators to interact with the reporting console.

If our intrusion detection project is built from scratch, and we must train our staff from the basics, our choice is broader. Although we must extend extra effort to train them, we have possibility of doing it right, and there are no "old habits" to break. There are some NIDS which utilize parts of other systems as consoles. If our administrators are familiar with using such a system, or possibly, if part of our network is managed by such a system, then we must consider it as a viable alternative. An example is Hewlett-Packard's *OpenView* system which can be used as a management console for several intrusion detection[9] systems.

Finally, how should we prioritize the reporting by the console. The obvious answer is that "critical alerts should receive priority". In practice, however, this is quite difficult to implement. The classification and reporting of events is largely based on our zoning.

With a single-sensor located in the *blue* zone, we should view all alerts as critical. If the alert is triggered by an insider abusing their privileges or if the perimeter defenses have been forced, we should look into both with equal care. Although typically, outsiders are considered the high threat, we need to understand that an insider is also quite dangerous as not only they have free access to many servers and resources, he or she may also be physically able to access some machines.

[9]It is also used as management system in areas other than intrusion detection

Further, they are more likely to be familiar with the network, and hence carry much more well aimed attacks.

With sensors in both *blue* and *green* zones, we need to consider what we are looking for, what is of importance to us. Are we mostly concerned about our web server being defaced, or are we suspecting attacks from within the intranet? Because the *green* zone is much more visible, chances are we will see most of the attacks there. It is hence our recommendation that we should still prioritize alerts from the *blue* zone as more crucial.

Some NIDS (for instance Snort [36]) help with the classification and reporting process by extending the signatures to include actions. The signature, then, not only includes the patterns to match, but also directives as to what to do with when triggered. Such systems are useful in that they can either be connected to some automated response (see next Section), or do follow-up analysis. They can also help inexperienced administrators in understanding and handling the event.

Deployment of multiple consoles in small companies should be avoided. The one-sensor/console environment is the minimum requirement for NIDS operation, and our next priority should be to extend the sensor coverage (multiple sensors, one console), while maintaining a centralized analysis and reporting center. Extending to multiple consoles, or consoles available from multiple access points creates problems in terms of network structure. As our goal was to isolate the IDS to protect it from being attacked, we would need to somehow link the reporting consoles back into our network, thus providing an entry point for intruders.

Final extension to the reporting facilities is adding a database which stores event information for future references and correlation. At the very least, a time-stamped quadruple, with information what triggered it, `<src addr, src port, dst addr, dst port>` should be stored. A general idea is presented in Figure 3.7.

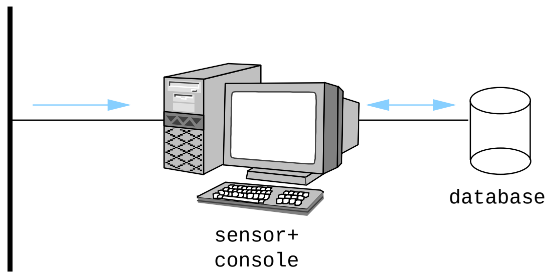

Figure 3.7: Sensor/console combo with a database back-end

Reacting

There are twofold priorities at the reaction phase of our ID process. We need to prioritize the procedures for incident handling and consider plausible reaction delay scenarios.

Delay The term *Real-Time IDS* has a nice ring to it, but we must face the reality. Stephen Northcutt presented an example [31] of an audit trail from an IDS as a host became compromised. The time it took from the start of an attack to the time when the intruder had access to the system was 16 seconds.

We have considered some factors which influence the cost of response time, and graphed[10] cost vs. delay as shown in Figure 3.8.

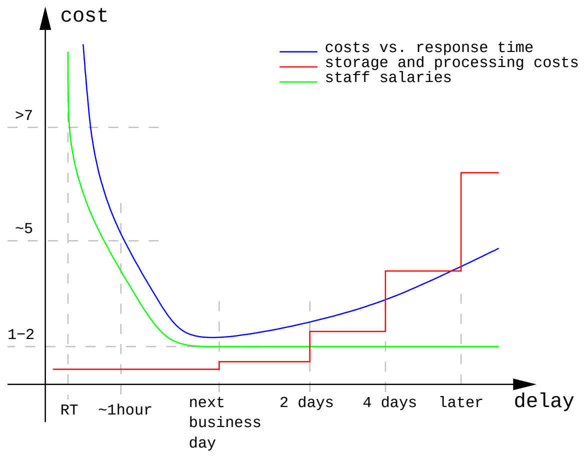

Figure 3.8: Response costs vs. response delay

Figure 3.8 shows correlation between number of administrators, IDS costs, and the delay between detection of an incident, and plausible response. Where does the graph in Figure 3.8 come from? The *real-time* advertised by IDS vendors should be considered real-time for detection, not response. With scripted exploits, it is very difficult to achieve real-time response even with automated systems. Additionally, when we add the possible losses caused by automated system disconnecting parts of our network, we note that real-time response is not possible.

The $\approx$*1 hour* mark indicated the 24/7 CIRT delay. The cost of this kind of response is still extremely high because we need to keep a number of highly qualified staff members on alert. This is highly infeasible for small companies. If the

[10]The axis in the graph are not meant to be linear

requirements call for this kind of response time, we suggest outsourcing the IDS needs from an MSSP.

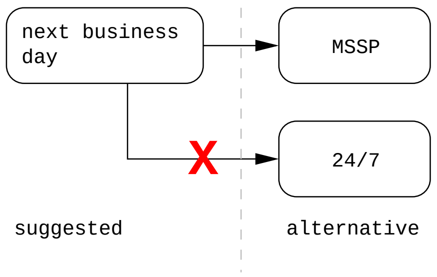

Figure 3.9: Reaction time prioritization

What we should aim at is “next business day” response. By this we mean that within business hours, incidents are handled as they come in, while anything that happens outside working hours is delayed until the following business day. A delay of about one day has been determined to be the optimum in terms of costs because it is plausible to achieve it using only one or two administrators. At this point, the costs of storing data are still quite low, and the recorded information can be processed with rational computational power. Beyond the one-day delay, the hardware costs (indicated by the red line) begin to grow as we must deploy large storage arrays, and powerful machines to process the old data. Beyond a certain limit, say, about one week, the data stored in those systems becomes mostly of historical value.

Procedures Whether we like it or not, chances are that eventually an attack launched against our network will succeed, and we will need to handle an incident. Our IDS is our best hope of making the incident response swift, accurate, and minimize the damage which has been caused.

Many current IDS come with the possibility to automate the response by shutting down or redirecting intruders. Although seen as a cost-effective solution [33], we recommend that you disable automatic response. Our prioritization encourages sensor deployment in the intranet at first. Automated response system would allow malicious insiders to play tricks on co-workers by eliciting the IDS to retaliate against some victim. In section 4.1.1 we present a method of combating this kind of behavior with some simple methods.

Assume a *fail-close* policy for both manual response, and automated response if you feel you must implement some. A *fail-close* response means that a breech of security causes shutting down of facilities to isolate the problem. This works twofold; it helps notice something went wrong when an automated system is in question, and helps isolate incidents disabling the attacker from causing any more damage. Recently there has been some talk of exploits which monitor network status, and erase the system upon detecting that they have been isolated (disconnected). To our knowledge, this is still rare.

Fail-close policy can also be deployed effectively by running the IDS in proxy mode (see Figure 3.10).

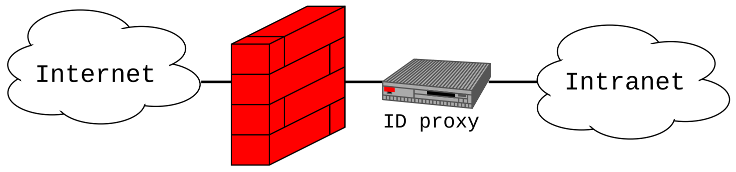

Figure 3.10: Intrusion detection pass-through proxy

In proxy configuration, we can not only monitor the traffic on the network, but also close outside communications in case of IDS failure or intrusion. Proxy mode IDS can either be deployed as a stand-alone system, or the functionality can be achieved by combining the IDS with the firewall.

This kind of setup can be equally dangerous as other automated response systems. It is however less “active” in its responses as it simply tightens the perimeter. The downside is that we need to have sensor interfaces which both receive and transmit traffic, thus allowing for attacks against the IDS itself.

What is the order of things to consider for the response phase then? The key to proper incident handling is an established response policy, and chain of command. This might sound like some kind of army related design, however in case of an incident, we must be sure that all the procedures are followed, and that the right people are doing them. This is the foundation of being able to carry out follow-up tasks such as forensic analysis or legal prosecution.

We recommend that the response policy be accompanied by checklists of tasks which need to be performed. There should always be a few copies of these checklists available in paper form just in case the system that has been hit is the one that we keep the files on. Also, the paper checklists should be filed in some safe place for future reference. A filled checklist must always be time-stamped, and signed at the end of the procedures.

Chain of command is required to verify procedures which have severe impact on the function of the network. In very small organizations, incidents are likely to be handled by one person, an administrator whose job is monitoring the IDS. In such case, we recommend the following actions be followed:

1. Validate the claim of the IDS.
2. Inform management or company's security officer, than an incident has occurred, and that investigation and recovery is going to be undertaken.
3. Inform the person or persons who are using the affected system, and isolate the system by disconnecting it from the network.
4. Investigate the incident, patch the hole which was used to enter the system.
5. Investigate the damage.
6. File a report.
7. Recover the system from a known good backup.
8. Report the incident to management, including what happened, why, the costs of the incident, plans for future preventive measures (and if any has been taken already), and IDS involvement.

We should not underestimate the importance of the first step; validating the claims of the systems. When the IDS is only a tool to point our problems, and the incident handling is limited to patching the holes, pursuing a false alarm is nothing more than a waste of time. However, if we plan to pursue legal action against alleged attackers, we must verify the validity and integrity of the alarm data.

How to improve on this setup? There is no need to add steps, the next objective is to improve the existing ones. A good idea is to pair staff working on incident response. This should be considered a nice-to-have feature if our IDS is aimed at only patching things up, but a must-have if we plan to do forensic analysis and seek compensation from the attackers. Work in pairs should be done so that one member carries out the tasks while other watches and verifies that correct procedures are completed. At the end of the investigation, both persons sign the report.

Honeypots Honeypots are systems which emulate real systems and try to lure the attacker in [32], but are closely monitored by administrators and the actions are logged. Honeypots are useful for gathering intelligence on the attacker, their techniques, exploit tools, and such.

Honeypot deployment is beyond the core of intrusion detection systems. However, honeypots can be successfully utilized in post-incident handling or research. Below we shortly describe applications of honeypots in incident aftermath.

If our IDS is limited to the *blue* zone, then the honeypot is only feasible if the intruder has caused considerable damage and we intent on prosecuting them but lack the evidence. By replacing the compromised system with a honeypot configured to mimic it, the administrators set a watch to capture the source of the intrusion. This is a costly and difficult process. We must isolate the affected machine, and make sure no further damage is caused, while maintaining the illusion that the intrusion has not been spotted. The system should be supervised round the clock so that no further damage to other systems is caused. Finally, the logging must be scrupulous to allow further action.

Replacing compromised systems with honeypots in the *green* zone is probably infeasible. If we follow *blue* zone procedures, our honeypot would need to mimic the subverted system potentially making us unable to restore the service functionality while we wait for the attacker. Setting up a generic honeypot in the *green* zone in commercial environments is also likely a waste of time as a knowledgeable intruder is bound to realize that the system has little purpose other than looking "interesting".

The true incentive for setting up honeypots, in *green* and *red* zones, is scientific research. We have discussed before that research and forensic analysis of new intrusion techniques are beyond the scope of intrusion detection efforts of small companies outside the information security field, and should not be practiced.

The priorities on incident response described in this section are summarized in Figure 3.11.

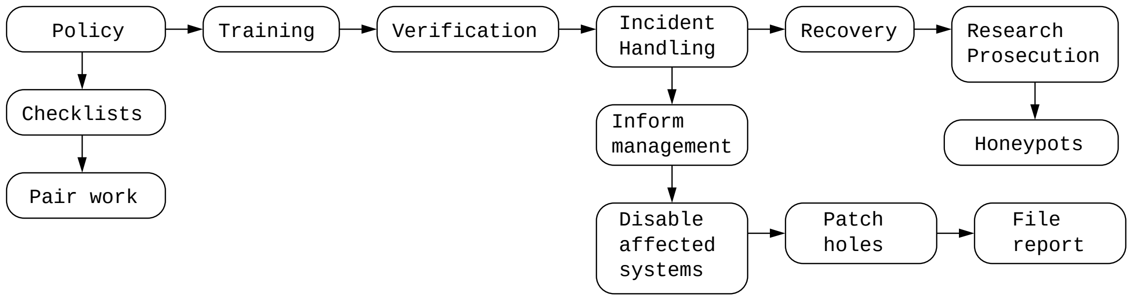

Figure 3.11: Summary of incident response priorities

3.3 Post deployment

There are three things we should consider after deploying the IDS: project continuity assurance, extended action with respect to the perpetrator, and involvement in a joint, organized effort such as GIAC or CERT[11]. Continuity planning should be part of the initial security infrastructure design and in our opinion is a priority post-requirement to IDS implementation. Extended actions and involvement in large-scale efforts should be seen as secondary issues only after the key functionality has been secured.

3.3.1 Continuity assurance

As with any process, we should designate a review period at which the feasibility and functionality must be re-evaluated, and resource budgeting must be performed again. As already discussed, implementation and running of an IDS is expensive, and we need a way to justify allocating the resources which could otherwise be used to improve some other part of the system. Our goal in continuity assurance is to be able to objectively say if the IDS brings any benefits, how much resources are needed to run it now and in the future, and convince the management of our case [33].

If during the re-evaluation period we detect an intrusion, using our IDS, it should be fairly straightforward to show that such system brings benefits. We can show management how we traced the attack, and how the IDS trail helped localize the damage. However, provided there were no intrusions, how can we estimate the benefits of the IDS? In these kinds of situations, it helps to run a system which correlates information. Our goal is to show that there were indeed no intrusions, and that our other security mechanisms (such as the firewall) are functioning properly. A simple case is our ability to compare events which occur outside our perimeter defense, and show the absence of these events inside the perimeter.

Our main tool for continuity assurance are reports generated during the *Reporting* phase as depicted in Figure 2.1. As the reports created by the IDS are rather technical and contain a lot of “noise” information, we should consider creating a summary reports which clearly shows activity, attack, and intrusion statistics – i.e. quantify IDS performance information.

It is important to keep in mind at this point, that despite our enthusiasm, as ana-

[11]GIAC (`http://www.giac.org`) runs `incidents.org` which collects attack and intrusion data from around the world. CERT groups, such as the CERT Coordination Center (CERT/CC) collect vulnerability and incident information, and issue advisories to help protect computer systems

lysts, for running the IDS, we should objectively evaluate the benefits and costs of our systems. The goal should not be to push our ideas through, but prove that IDS is needed, and is the right tool for minimizing the impact of computer related incidents. If we cannot provide conclusive evidence of IDS functionality in our reports, we should consider alternative methods of security engineering.

3.3.2 Tracking and prosecution

During a burglary, what we expect from our surveillance system is not only a trail of events, but help in identifying the intruder. What we need from the IDS is ability to track the intrusion, monitor the progress of events, gather information on the identity of the intruder, and finally, be able to provide authorities with the necessary data to prosecute.

Prosecution of intruders is a difficult and expensive process [29], and requires a considerable effort on behalf of the staff tracing the intrusion [33]. There are three areas on which we should concentrate: the event history, capturing the attacker, and trend analysis.

History Historical records relate directly to the *capture* phase portrayed in Figure 2.1. We have discussed the limitations put on our recording capabilities in Section 2.1.2. Because of the filtering performed by the later analysis stages, a good deal of information is lost. When extending the functionality of our IDS, we should look into increasing the recording capability of our system.

Costs related to event data storage for a small LAN are likely to grow linearly with the time as increases in network speeds are rare. By adding more disks, we can record more past events and store them for a longer time. We must consider two issues though: eventually, the database will reach the hardware expansion capability and require an upgrade, and the resource requirements to process the recorded events will grow exponentially [12] until we reach the event storage limit.

Capture and Prosecution The ultimate goal of the IDS is to bring the guilty to justice and receive compensation for damages. Typically, the IDS is used to provide evidence of the intrusion, and help disclose the identity of the intruder. As we can clearly see, this requires that our IDS records are accurate and that their integrity has been preserved after the incident.

[12]For each event n, we need to correlate $(n-1)$ events from the past. The time complexity is hence $O(n^2)$.

In terms of associated costs, the issue of integrity can only fully be assured by taking the system involved in the incident offline, and sealing it up as evidence. The process of documenting the procedures carried out during the incident handling must also be much more accurate than would otherwise be required for administrative purposes [29]. In the evidence gathering phase, costs caused by taking hardware offline can be seen as minimal. The time required by the administrators to extract, verify the evidence and track the attacker, as well as the followup documentation and forensics are much more expensive [14]. We must also note that despite considerable resources being expanded to track down and prosecute the intruder, there is no guarantee of conviction or return of costs.

Trend analysis More recently there has been talk of correlation and trend analysis in IDS. This area is still in its infancy, and lack of proper IDS make it difficult to implement by corporations not able to roll out their own systems [1]. The situation is not made any easier by the lack of implemented standards in alert reporting among various systems. Internet Engineering Task Force (IETF) ID working group has proposed a few drafts which would help correlate events between various systems, however in reality, many current implementations simply perform parsing on proprietary formats.

Some patterns are known already though. For instance, the publication of a patch for a given software is typically followed by a large increase in exploit attempts using that bug [29].

3.3.3 Getting involved

Regardless of most IDS products being commercial, a large part of the intrusion detection effort is done by the community. A good example are the Snort signatures [36] being developed by Snort users, the GIAC (SANS Institute), CERT and others. The community provides anything from analysis of new attacks, to advisories on security systems and updates to the IDSes allowing them to keep up to date.

The final step in the deployment of IDS is getting actively involved in some intrusion detection research or collective effort. Although some of this process might be undergone during forensic analysis of your attacked systems, getting involved beyond that is likely to become a near-full-time or full-time job for a security specialist. Although very noble and necessary, we recommend that this will only be attempted if the organization is big enough to support “charity” work in the field.

The situation is a bit different for companies which specialize in security consulting or providing security services to others (MSSP). In such cases, it should be considered that the company gets involved in intrusion detection research and development as part of their PR or marketing strategy for promotional purposes. The incentive in such cases being that contracts acquired in that way will cover the promotional expenses caused by the extra effort. Further benefits can be gained in terms of staff education, certification, and allowing them to keep up to date with the latest exploits, and “attack de jour”.

Chapter 4

Evaluation and Analysis

In this chapter we take some of the pre-IDS solutions suggested in Chapter 3, and match them against a full-blown IDS to see if we can achieve any benefits with evidently reduced effort.

Whenever possible, evaluations have been based on real network installations run by experienced system administrators. We believe this approach gives better results than simulated tests in laboratory environments. Feasibility of several tools have been tested in simulated conditions in a small laboratory network.

4.1 Evaluating *pre-IDS*

4.1.1 ARPWatch

ARPWatch[1] is an Ethernet monitoring program for keeping track of Ethernet/IP address pairings. Its main limitation as a security tool is that it must be run locally within the Ethernet segment hence requiring one server per segment on a multi-segment network.

Running ARPWatch

ARPWatch has very low requirements in terms of processing power, storage, and configuration. It is also fairly easy to install and use.

The following examples have been taken from a network where *ARPWatch* is run on a Netra X1 under Solaris 8. The server is connected to a switched network, and

[1] `http://www-nrg.ee.lbl.gov/`

```
Date: Tue, 9 Oct 2001 14:21:11 +0300 (EEST)
From: Arpwatch <arpwatch@company.com>
To: arpwatch@company.com
Subject: new station (costilla.comapny.com)

            hostname: costilla.company.com
          ip address: 10.1.1.167
    ethernet address: 0:1:3:88:3:eb
     ethernet vendor: 3COM CORPORATION
           timestamp: Tuesday, October 9, 2001 14:21:10 +0300
```

Figure 4.1: New station joins the network

```
Date: Tue, 25 Sep 2001 12:57:12 +0300 (EEST)
From: Arpwatch <arpwatch@company.com>
To: arpwatch@company.com
Subject: changed ethernet address (dyn-pc5.company.com)

            hostname: dyn-pc5.company.com
          ip address: 10.1.1.145
    ethernet address: 0:10:a4:a7:99:6
     ethernet vendor: XIRCOM
old ethernet address: 0:1:3:88:6f:d8
 old ethernet vendor: 3COM CORPORATION
           timestamp: Tuesday, September 25, 2001 12:57:12 +0300
  previous timestamp: Monday, September 10, 2001 14:59:04 +0300
               delta: 14 days
```

Figure 4.2: Change in IP/Ethernet address pairing

the switch port runs in normal mode meaning that ARPWatch is only listening to traffic directed to the server, and ARP broadcasts. Whenever an ARP event occurs, a mail message is generated and sent to the system administrators. There are twofold events:

1. An unknown hardware address has been detected: new station joined the network
2. The IP/Ethernet address pair has changed: DHCP assigned new address, network card has changed, etc.

Examples of posts generated by *ARPWatch* are shown in Figures 4.1 and 4.2[2].

[2]IP addresses and domain information have been modified for privacy reasons.

On the network in question *ARPWatch* generated 162 event alerts during about a 7 month period. Of course, initially, when the address database is empty, the number of mails will be quite high, but as all the machines get recorded, we should only receive notification of real events.

In this installation, further improvements have been made with several small scripts which queried the switch for the location of the given hardware address reported by *ARPWatch*. This allowed the administrators to pinpoint the physical location of the occupance of the event, and investigate what has happened. The scripts are conveniently integrated with IRM – an administrative ticketing system – allowing the system administrators to view reports with a web browser. Other automation features were not implemented or planned.

Evaluation

Overall, the administrators reported *ARPWatch* as a very useful tool for monitoring hardware-address changes in the local network, however use was more oriented towards network debugging than security. Low requirements of processing power and storage facilities make it possible to keep a "forever" audit history, which provides a weak audit trail in case of need of incident tracing (allows pinpointing the physical location of the machine which carried attacks and such).

No formal policy or procedures were instantiated for this tool. Overall effort of deployment was about one day which included creation of the switch query-scripts and some minor testing. Administrators also commented that since the presence of attacker machine on the network probably meant that they have somehow forced the physical defenses, *ARPWatch* does not really qualify in any way as a simple pre-IDS solution.

Possible applications in other environments include misuse detection. Malicious insiders may try to masquerade their identity by spoofing their IP address. Based on evidence, *ARPWatch* would provide clean audit information of such event, and in the above setup, even pinpoint the physical location from which the event occurred.

4.1.2 MRTG & Cricket

"The Multi Router Traffic Grapher (MRTG) is a tool to monitor the traffic load on network links. MRTG generates HTML pages containing graphical images which provide a LIVE visual representation of this traffic."[3]. It works by commu-

[3] `http://people.ee.ethz.ch/õetiker/webtools/mrtg/`

nicating, with SNMP, with network switches, and extracts traffic information from them. MRTG works on a multitude of platforms including popular UNICES and WindowsNT. It has been widely deployed to monitor traffic loads on networks, and system administrators are well familiar with it.

“Cricket is a tool that lets users visualize a set of measurements over time.”[3]. It was designed to help solve problems in using MRTG in large infrastructures, but essentially, it does very much the same thing.

Requirements and Installation

MRTG and Cricket are rather low in their requirements. For beginner users, Cricket recommends only using Cisco routers, series 2500, 3600, 7200 and 7500. Guided installation tours are available for both, so deployment should not cause problems.

Running statistical tools

We have evaluated installation of statistical tools in two environments in addition to the example given in Chapter 2.

Our first environment is a university laboratory where MRTG and Cricket are used side by side with Snort. The installation was part of the network design phase, and allows tracking traffic from the network and individual hosts.

The laboratory network runs 100Mbps Ethernet, with backbone running at 1Gbps. Network traffic monitoring has shown that typical network load is around 10% despite NFS mounts. Administrators have reported that clearly visible traffic profiles of daily usage and weekends can be observed.

Both MRTG and Cricket are used as a correlating help for detection from other sources. A number of incidents in the laboratory network has been investigated by monitoring traffic profiles from suspected hosts. Results have been positive in terms of accuracy, but required a certain delay in order to observe changes in traffic patterns.

A second scenario includes a much broader infrastructure, a network which connects Italian universities and public libraries. MRTG is used along with a number of custom wrappers which analyze traffic flows in network routers. The uniqueness of this installation is that it is the primary method of intrusion detection, and it is limited to processing traffic from a C-network block. Upon the detection of some anomaly, the administrators start a closer investigation.

The system is run by three persons: a system administrator who maintains the network, and two intrusion detection analysts. A rough estimate of 25-30% of daily time is spent browsing through traffic profiles looking for signs of anomalies.

A simple incident response procedure is installed, with pre-written report templates. Typically, the forensic, analysis, and reporting stage takes about 3-4 days for one of the analysts.

Our third example comes from Sweden, and is unique in a sense that MRTG is used by a service provider who also provides MSSP services. MRTG is configured to watch external links for traffic patterns.

As mentioned in Section 2.2.1, MRTG has proven a very successful tool in flagging down Denial of Service attacks. As the service provider is responsible for maintaining availability of their services at all times, statistical analysis provides an easy way to isolate availability related security problems.

Evaluation

In all three of the above cases, using statistical reporting tools have proved easy, beneficial, and applicable to intrusion detection, however in all three cases the systems are used in a different manner. We have hence deducted that statistical analysis is useful not only as a pre-IDS deployment, but also plays a part in advanced IDS. The trend seems to be that as network size and complexity grows, administrators abandon signature-based IDS systems and focus also on statistical profiles.

We should note however that statistical tools such as MRTG or Cricket are only useful in tracking “loud” incidents. This is typical of script kiddies, or attackers who are not concerned about being shut off. The resolution and accuracy of the above mentioned tools is not good enough to help detect low and slow attacks which an experienced attacker might carry out.

4.1.3 Proxies

Proxies are pass-through systems which may possibly cache data. In Section 3.2.2 we have suggested improvements to security by forcing traffic to and from the intranet to pass through proxies located in the DMZ. Our analysis revealed that although this does indeed improve security to a small degree, it does so by strengthening the perimeter defenses, and is not directly related to intrusion detection. Because of this, it falls outside the scope of this thesis.

4.1.4 Personal firewalls

A *Personal Firewall* (Personal FW) is a piece of software installed on a host computer which acts as a FW for only that machine, limiting network connectivity. Personal FWs often come with a nice graphical user interface to aid home users create FW rules more easily.

During recent years, personal FWs have greatly increased in importance, as more and more home users are connecting to the network using “always on” connections (xDSL, etc). In corporate environments, personal FWs find application on mobile computers which are frequently taken out of the perimeter defense. Some companies have realized the benefits of creating defenses around individual workstations, and have created distributed firewalls where the perimeter defense is enhanced by firewalls protecting each workstation individually.

Requirements and limitations

When using personal firewalls (or a distributed firewall system), the main problem is that we need to have a homeogenous workstation platform. If the environment is highly heterogenous, it might be difficult to find compatible system, and maintain a number of different configurations for each one.

The second major obstacle to proper functioning is of personal firewalls is the setup. It is quite imminent that the firewall rules be configured by a knowledgeable person who is familiar with the system. As now, not one, but a number of firewalls must be maintained, we must have have a good administrative back-end allowing administrators to remotely manage all the workstations. Letting users modify rulesets leads to an illusion of security as they are likely to create holes in the FW allowing intruders through.

Personal firewalls as intrusion detection tools

The functionality of personal firewalls as intrusion detection tools is depicted in Figure 4.3. An attack penetrating the perimeter defense will be aimed at some workstation on the intranet. A secondary firewall on the workstation will hopefully detect the attack, and flag an alarm.

The presented setup is functional, and carries an additional benefit of strengthening perimeter defenses. However, it is not without fault. If an intruder can penetrate the first perimeter defense and the personal firewall uses a similar system, a similar set of rules, or is vulnerable to the same problem which allowed

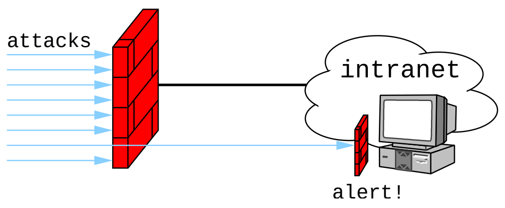

Figure 4.3: Personal firewall detecting a violation of perimeter defenses

bypassing the perimeter, the attack may still be successful. We therefore recommend that in using a different system for the main perimeter defense and personal firewalls.

In a fashion similar to an IDS the full benefit of personal (distributed) firewalls is achieved when we have a centralized system for reporting firewall events. The benefits are lost if users can close alert dialogues, or if the events are not reported anywhere (see also Section 4.3).

Finally, personal firewalls help prevent internal misuse. As discussed in Chapters 2 and 3.2.2, perimeter defenses are ineffective against attacks originating on the trusted network. With personal firewalls, the risk of insiders causing damage, or compromised machines being used to spread damage, is limited.

4.1.5 Sniffer detection

A number of tools have been developed for determining if a network interface is running in promiscuous mode. We have selected *antisniff* and *sentinel* which are publicly available free of charge.

Requirements and installation

Installation of both software on the test machine was quite trivial, and involved running the installation commands for an OpenBSD port. The hardware requirements seemed equally low, however for future use, the number of *bpf*[4] in the kernel could be increased for *antisniff*.

As per documentation the software is unable to detect special-purpose probes or hardware sniffers, and will most likely only work against typical computers which have interfaces in promiscuous mode. We consider this sufficient for our purposes,

[4]Berkeley Packet Filter – provides a raw interface to data link layers in a protocol-independent fashion on BSD derived systems. Source: `bpf(4)` man page, OpenBSD 3.0

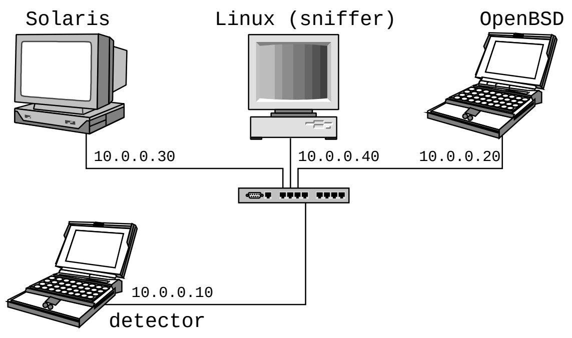

Figure 4.4: Lab setup for sniffer detection

as our goal is to detect subversion of intranet systems which have been made to work as intelligence gathering stations.

For these tests, we built a simplistic test network as portrayed in Figure 4.4.

The network comprised of four computers running various operating systems as listed below. They were connected through a 10Mbps hub, and no outside connections were provided.

- SparcStation LX machine with Solaris 2.6
- Debian Linux 2.2 (potato) on x86 architecture, this machine run an interface in promiscuous mode using *tcpdump*
- OpenBSD 3.1 on x86 architecture
- OpenBSD 3.1 on x86 architecture with a number of sniffer detection tools installed, working as the detection station

All of the machines run several services such as web or mail servers, however we purposefully did not include any traffic, as these kinds of tests would normally be run outside office hours in production networks.

Running sniffer detection tools

antisniff *Antisniff* has six different modes it uses to test for presence of promiscuous interfaces. Surprisingly, the most accurate results in this network were obtained with the *NT EtherPing* option, however some *ICMP time delta* test has also given promising results.

One of the big cons of *antisniff* was that it only takes a single IP address as argument for running the tests, and hence is not really usable for sweeping networks.

Since L0pth (authors of *antisniff*) has become Security Software Technologies[5], *antisniff* has been ported to Microsoft Windows and received a nice graphical user interface to aid in the detection of sniffers. The Windows version of *antisniff* has not been tested for this thesis.

Sentinel Sentinel offered three different modes for detection, two of which we used for testing: ARP test, and ICMP echo test. The options for providing a range of addresses to be checked are also very flexible ranging from a single address, to a list of addresses listed in a file, to scanning an entire C-class network.

The ARP -test detected two promiscuous mode machines `.20` and `.40` (see Figure 4.4). The ICMP echo reply test narrowed this to `.40` only. Altogether 11 IP addresses were used in the test, four of which actually existed on the network.

Sentinel worked quite fast, using about five to ten seconds per IP. This allows scanning of even larger networks in a reasonable amount of time.

Evaluation

Sentinel could well be used to detect sniffing systems on the network, however, some kind of tool to correlate the results from different tests would probably greatly improve the accuracy of results. It should most likely be run outside office hours as not to cause disturbances in the production network. *Antisniff* version tested during these experiments is rather a curiosity tool that administrators might want to run against a single host which is under a suspicion of sniffing. Better tools seem to exist though.

Based on these short test runs, we could definitely add promiscuous mode detectors to a pre-IDS toolbox. Although the detection of compromised systems would likely occur well after the actual intrusion, the ease of running and interpreting the results make this a viable method.

4.2 Evaluating NIDS – Snort

In this section we look at what full-scale NIDS have to offer. We have selected Snort (`www.snort.org`) as it is a free software, and hence easy to obtain. Snort

[5] `http://www.securitysoftwaretech.com/`

is highly popular, and available for a multitude of platforms with one of the largest, and most enthusiastically maintained rules[6] databases. A good indication of how much Snort influences current signature-based NIDS development is that Snort and Dragon rules are used by other vendors to update their systems [43].

Requirements

Snort runs on a variety of platforms, including most UNICES, Windows, and others. The requirements are mostly dictated by the following three factors:

1. The speed of the network, in corporate LANs, typically 100Mbps
2. The number of signatures to be processed
3. Event history

The first two dictate requirements for the network interfaces and processor speed. The third one dictates speed and size of the storage the system must have available.

In single-sensor environments, or even in dual-sensor scenarios, it is typically sufficient to utilize a modern PC with sufficient disk space to store several days worth of logs.

Our tests were run on two systems: a Pentium MMX laptop, and a Pentium III desktop machine. The operating system used in both cases was OpenBSD (3.0 and 3.1).

Installation

Installation in itself is not difficult. In most instances it amounts to either installing a pre-compiled package, or running the `make` installation. The difficult part is tuning the system for a particular network and purpose.

A rough estimate for installation and configuration of the underlying operating platform and Snort (in single-sensor/console configuration) is two working days.

In our analysis we used two Snort installation. The first is a trial-run of an intrusion detection system in a small company. Snort is installed on an OpenBSD system and combines the console. SnortSnarf is used to parse logs which are saved and maintained on the same system.

[6]Snort uses the term *rule* to mean a signature because it not only includes the pattern to match, but also additional information such as action to be taken.

The second installation in in a larger, university network where Snort runs in proxy mode between the network and the firewall.

Running Snort

The first step after installation was to determine what typically happens on the network. This helps eliminate small glitches which might have previously gone unnoticed, and which might be causing some of the rules to trigger alerts.

In scenario number one, Snort logs are inspected about once a month by a well experienced security expert. Since the network is thought to have strong perimeter defenses and well established usage policy, the threat of penetration is seen as very low. Snort is used mostly to keep a record in case an intrusion is noticed by other means.

No automation has been built into the system. There is also only low degree of help expected from the system in case an intrusion is noticed. The loging facility records IPs and port numbers, but little more information about events. This allows the system to run unattended for long periods of time between inspections.

This trial run has been running for about six months at the time of this writing, and no intrusions were noticed or inspected using it. No reports were presented to management, and no incident handling plan has been deployed.

In the second scenario, Snort is used more actively to monitor activities on the network. It has been configured as a proxy, with all traffic passing through it to allow the administrators to see intrusions from the outside, and misuse by legitimate users on the network.

Very little trimming has been done to the Snort rule set as the university network allows a good degree of freedom for people to install software, services, and machines. SnortSnarf is also used as the interface for viewing log data.

Because in this configuration, the system is rather large, a statistical tool, Cricket, is used to double-check suspicious activity which triggers alarms (Section 4.1.2).

Two part-time administrators monitor the system on daily basis. The intention is to find problems and patch holes. There is no intent to pursue the guilty parties beyond a notice to their service providers or equivalent.

Evaluation

If used actively, and possibly by correlating data with other systems, a full-blown NIDS provides most of the information that all the other tools we presented do and some that the other systems can't provide. The information is also much broader, noting intrusions "as they happen", not after-the-fact detection. However, as pointed by the contrast in required supervision work, an IDS requires substantial human resources to provide benefits.

IDS might also be considerably more useful in environments where the perimeter defenses either are not very strong, or cannot be made tight enough to provide a high level of security.

4.3 What else?

What did we miss? In Section 2.6 we have mentioned that an alternative approach to carrying out intrusion detection is to utilize host-based systems. Probably one of the basic things we should be doing is to consider existing data to look for signs of intrusions. Every system which is connected to a network provides some kind of logging. We should have at least firewall logs and systems logs from individual computers. Inspection of those logs can often provide some insightful information as to what is happening. We have found that too often, system logs are not inspected at all.

Additionally, we quickly evaluated tools which allow monitoring of network connections in real-time (or from a tcpdump). The tools were *ettercap* and *sniffit*. We have found that once an intrusion is detected, these tools could be useful to tracing attacker actions and gathering evidence. The downside is that the intrusion must have already happened, and that the compromised system should be kept on the network allowing the attacker to perform their doings.

Finally, if our networks are hightly Microsoft Windows -oriented, we should not forget anti-virus (AV) software. Although AV software doesn't really belong to the intrusion detection toolkit, it is used to limit the spread of malicious code. It can also act as a type of HIDS by detecting presence of trojans or remote-administration toolkits.

Chapter 5

Conclusions

Despite apparent benefits, intrusion detection systems are still quite rare as security measures in small company networks. The main reasons are the high cost of maintenance and difficulties in evaluating performance. In this thesis we investigated what are the main sources of the costs, and how to prioritize deployment to achieve best results using the minimum of resources.

We have divided the intrusion detection process into five stages, and discussed priorities in each one based on factors such as goals set for the intrusion detection system, available resources, and network design. We have also identified some common causes of problems in corporate networks, and suggested a *pre-IDS* security measures which could be used to combat a number of those issues.

The results from the *pre-IDS* stage evaluation are mixed. Some small tools are well suited to perform a certain amount of intrusion detection, while others should be seen more as providing network maturity which allows efficient deployment of NIDS. Specifically, personal and distributed firewalls (Section 4.1.4) can be used to both strengthen the perimeter defense, as well as act as a low-tech intrusion detection system with limited capabilities. Furthermore, tools such as *ARPWatch* and *sentinel* (Section 4.1.5) are useful in monitoring behavior of computers behind the security perimeter.

In NIDS deployment several points of notice, which aid prioritization, have been observed:

- Detection of events of interest should be measured in terms of both speed and accuracy.
- Solutions for small networks do not necessarily scale to large infrastructures.

- In incident response, priorities should be focused on automation of human-performed tasks.
- And finally, intrusion detection is not a silver bullet, and is only viable if the rest of the network and security infrastructures have reached maturity.

Raising of false alerts is clearly going to lower the effectiveness of the IDS, and our priority during the detection phase should be to correlate events from a number of systems to increase both speed and accuracy of detection. An effective way of correlation is to use a signature-based NIDS in combination with a statistical profiler.

By studying deployed systems, we have noticed that under limited resources, the usability of signature-based systems decreases with network size. As networks grow, the emphasis is shifted from popular signature-based systems to statistical analysis.

Machine automation of response, although cost-effective, has highly limited applications. The burden of incident response is still carried by security administrators. In order to simplify the process, and decrease time needed to analyze alerts and perform recovery, administrators should be equiped with policies, checklists, and guidelines allowing them to work methodologically. This is especially crucial if the goal of the incident response is to pursue legal action against the attacker.

Finally, intrusion detection systems are not an answer to all network security problems. They require a certain level of maturity, and are only effective if monitored and maintained. Too often, existing facilities are not used to full potential, and the faults are being covered by IDS deployment.

Bibliography

[1] Focus-ids: Ids correlation. http://www.securityfocus.com, Mar-Apr 2002.

[2] Focus-ids: Statistical anomaly analysis. http://www.securityfocus.com, Jan-Mar 2002.

[3] J. R. Allen. Driving by the rear-view mirror: Managing a network with cricket. In *1st Conference on Network Administration*. WebTV Networks, April 1999.

[4] E. Amoroso. *Fundamentals of Computer Security Technology*. Prentice Hall, Englewood Cliffs, NJ., 1994.

[5] R. Bace. An introduction to intrusion detection assessment. Technical report, Infidel Inc., April 1999.

[6] S. M. Bellovin. Fragroute vs. snort: the tempest in a teacup. Bugtraq mailing list, April 2002. In reply to Darren Reed regarding sensor positioning in IDS.

[7] S. Berinato. Finally, a real return on security spending. CIO Magazine, February 2002. http://www.cio.com/archive/021502/security_content.html.

[8] CERT. Cert advisory ca-2001-26 nimda worm. Technical Report CA-2001-26, CERT, September 2001. http://www.cert.org/advisories/CA-2001-26.html.

[9] D. B. Chapman and E. D. Zwicky. *Building Internet Firewalls*. O'Reilly and Associates, Inc., September 1995.

[10] D. E. Commer. *Internetworking with TCP/IP*, volume 1. Prentice Hall, 3 edition, 1995.

[11] D. Curry, H. Debar, and M. Lynch. Intrusion detection message exchange format. Technical report, IETF, December 2001. Expires June 27th, 2002.

[12] R. Danyliw. Analysis console for intrusion databases. http://www.andrew.cmu.edu/ rdanyliw/snort/snortacid.html, April 2002.

[13] Demarc Security. Demarc puresecure. http://www.demarc.com, April 2002. Snort frontend.

[14] D. Dittrich. Honeypot project: The forensic challenge. http://project.honeypot.org/challenge, January 2001.

[15] M. Dobrucki. Tamperproof systems. Nixu Ltd, internal training material, January 2001. Limited availability.

[16] K. K. Frederick. Network intrusion detection signatures, part 3. In *SFOnline*. SecurityFocus, February 2002.

[17] R. Graham. Sniffing faq, v.0.3.3. http://www.robertgraham.com/pubs/sniffing-faq.html, September 2000.

[18] T. Grenman and L. Pesonen. Social engineering. In *Hakkeri proceedings*, 2000. Available at special request only.

[19] L. Heberlein, K. Levitt, and B. Mukherjee. A method to detect intrusive activity in a networked environment. In *Proceedings of the 14th National Computer Security Conference*, October 1991. Washington DC.

[20] P. Innella, O. McMillan, and D. Trout. Managing intrusion detection systems in large organizations, part i. In *SFOnline*. SecurityFocus, April 2002.

[21] P. Innella, O. McMillan, and D. Trout. Managing intrusion detection systems in large organizations, part ii. In *SFOnline*. SecurityFocus, April 2002.

[22] J. R. Ivar Jacobson, Grady Booch. *Unified Software Development Process*. Addison Wesley, 1999.

[23] R. Kalinen. Discussions on intrusion detection systems. No written documentation.

[24] H. Koukkula. Tik-110.451 tietoturvallisuuden kehittämisprosessi. TML/HUT course, 1999. Course material may no longer be available.

[25] J. Lasser. Implementing snort in a production environment. *;login:*, 26(7), 2001.

[26] T. Lunt, R. Jagannathan, R. Lee, S. Listgarten, D. Edwards, and J. Ford. Ides: The enhanced prototype, a real-time intrusion-detection expert system. Technical report, SRI International, October 1988.

[27] Merriam-Webster, Inc. Merriam-webster online, September 2001. http://www.webster.com.

[28] H. F. T. Micki Krause, editor. *Handbook of Information Security Management*. CRC Press LLC, 1999.

[29] J. Mäkynen. What to do after being hacked. Presented at the 4th Nordu Conference, session 2, February 2002.

[30] Network ICE Corporation. Protocol analysis vs. pattern matching. Technical report, Network ICE Corp., 2000.

[31] S. Northcutt. Ids signatures and analysis, part 1&2. SANS Practicals, February 2001.

[32] S. Northcutt. *Intrusion Detection FAQ*. SANS, 1.52 edition, November 2001. http://www.sans.org/newlook/resources/IDFAQ/network_based.htm.

[33] S. Northcutt and J. Novak. *Network Intrusion Detection: An Analyst's Handbook*. New Riders, 2 edition, September 2000.

[34] T. H. Ptacek and T. N. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, "XXX", Suite 330, 1201 5th Street S.W, Calgary, Alberta, Canada, T2R-0Y6, 1998.

[35] O. Rintamäki. Kyselytutkimus tietoturvasta: Inhimilliset virheet ja virukset suurin uhka. *Turvallisuus*, 4:22–26, 2001.

[36] M. Roesch. Snort, lightweight network intrusion detection tool, December 2001. http://www.snort.org.

[37] S. C. Sanchez. Ids "zone" theory diagram. http://infosec.gungadin.com, July 2000. referred 2.1.2002.

[38] R. Seppänen. Tietoturvaan syydetään rahaa heikoin tuloksin. *ITviikko*, (6), February 2002. Research by the Economic Intelligence Unit.

[39] G. Shipley and P. Mueller. Dragon claws its way to the top. *Network Computing*, August 2001. http://www.networkcomputing.com/1217/1217f2.html.

[40] S. Smaha. Haystack: An intrusion detection system. In *Proceedings of the IEEE Fourth Aerospace Computer Security Applications Conference*. IEEE, December 1998. Orlando, Florida.

[41] L. Sundström. Tietoturvatarkastuksen automatisointi. Work in progress, May 2002.

[42] B. Tomhave. How does an ids help to save money. http://www.securityfocus.com, April 2002.

[43] Various. Focus-ids mailing list, May 2002.

[44] J. Wanderley. Network intrusion detection system on mass parallel processing architecture. *Phrack*, 0x0b(0x39), August 2001.

[45] M. Ward. Employees seen as computer saboteurs. BBC News Online, April 2002.