TOTALFÖRSVARETS
FORSKNINGSINSTITUT

FOI-R--1463--SE
December 2004
1650-1942

Technical report

Intrusion Analysis in Military Networks – An Introduction

Martin Karresand

Swedish Defence Research Agency
Command and Control Systems
Box 1165
SE-581 11 LINKÖPING
Sweden

FOI-R--1463--SE
December 2004
1650-1942

Technical report

Intrusion Analysis in Military Networks – An Introduction

Martin Karresand

Issuing organization Swedish Defence Research Agency Command and Control Systems Box 1165 SE-581 11 LINKÖPING Sweden	Report number, ISRN FOI-R--1463--SE	Report type Technical report
	Programme Areas C4ISTAR	
	Month year December 2004	Project no. E7091
	General Research Areas Commissioned Research	
	Subcategories C4I	
Author/s (editor/s) Martin Karresand	Project manager Mikael Wedlin	
	Approved by Johan Allgurén	
	Sponsoring agency Swedish Armed Forces	
	Scientifically and technically responsible	

Report title

Intrusion Analysis in Military Networks– An Introduction

Abstract

This report gives an introduction to the intrusion analysis field, specifically studying military networks. The report presents the results from the first of three planned years of research in the “Warfare in the IT domain” project. The state-of-the-art of some of the sub-fields, together forming the intrusion analysis field, are presented. A presentation of a practical implementation of a honeynet is included and two experiments are described. The honeynet has been used for studying the fundamentals of intrusion analysis, which are discussed in a separate chapter. There is also a chapter discussing some remaining issues to be solved in future work.

Keywords

Further bibliographic information	Language English
ISSN 1650-1942	Pages 35
Distribution By sendlist	Price Acc. to pricelist Security classification Unclassified

Utgivare Totalförsvarets forskningsinstitut Ledningssystem Box 1165 SE-581 11 LINKÖPING Sweden	Rapportnummer, ISRN FOI-R--1463--SE	Klassificering Teknisk rapport
	Forskningsområde Ledning, informationsteknik och sensorer	
	Månad, år December 2004	Projektnummer E7091
	Verksamhetsgren Uppdragsfinansierad verksamhet	
	Delområde Ledning med samband, telekom och IT-system	
Författare/redaktör Martin Karresand	Projektledare Mikael Wedlin	
	Godkänd av Johan Allgurén	
	Uppdragsgivare/kundbeteckning FM	
	Tekniskt och/eller vetenskapligt ansvarig	

Rapportens titel

Intrångsanalys i militära nätverk– en introduktion

Sammanfattning

Denna rapport ger en introduktion till intrångsanalysområdet med särskild inriktning på militära nätverk. Rapporten utgör milstolpe för det första av tre års arbete i projektet “Strid i IT-domänen”. I den finns en redogörelse för forskningsfronten inom några delområden, som tillsammans utgör intrångsanalysområdet. Vi presenterar också en implementation av ett honeynet, vilket vi använt för studier av grunderna i intrångsanalys. Delarna diskuteras sedan i ett separat kapitel. Rapporten innehåller även ett kapitel om kvarstående problem, som behöver lösas och därmed utgör grund för framtida arbete.

Nyckelord

Övriga bibliografiska uppgifter	Språk Engelska
ISSN 1650-1942	Antal sidor 35
Distribution Enligt missiv	Pris Enligt prislista Sekretess Öppen

Contents

1. Introduction

By knowing your enemy you have the upper hand and thus a key to victory. Therefore military computer networks are high value targets for elite hackers and foreign intelligence agencies. Huge sums are spent on developing the perfect intrusion detection or prevention system, but before that has been accomplished, if it ever will, we need to learn from those who actually succeed in penetrating our networks. If we study the structure and essence of intrusions, we master the foundation on which almost everything in computer security rest. Consequently the analysis of intrusions is the key that can help us understand and therefore also build the ultimate security management system.

The goal of a perfect security management system is however situated in a distant future. Before we can reach it there is a lot of hard work ahead. For example the problems concerning the detection of misuse and intrusions have to be solved. These problems are intricate and valid for all types of computer networks there are, regardless of whether they are for home users, commercial use or military purposes.

One of the more serious problems that is facing the current intrusion detection systems is encryption. Military networks are already using encryption to secure their communication and other networks are rapidly increasing their use of encryption techniques. This is of course a good thing regarding the confidentiality of the information sent, but from a intrusion detection point of view the encryption is having a negative impact. Currently much of the effort has been placed on developing better ways of performing network based intrusion detection. This is quite natural considering of the ability of early detection, ease of implementation, and fairly platform independent function of such systems.

The core of an NIDS (network based intrusion detection system) is the possibility to inspect the packets sent over a network. The more information available the more efficient and correct the detection. Hence when different parts of a packet, parts that approximately relate to different layers in the OSI stack, are encrypted the efficiency of a NIDS decreases. The very strong and thorough encryption policy of military networks might render existing NIDS systems almost useless.

The way to go is to look for signs of intrusions in the hosts themselves. By checking the behavior of the operating system, the applications, and the users, an intrusion detection equaling or even surpassing the NIDS in efficiency can be developed.

The idea of looking for signs of intrusions on the hosts themselves is not new, but it has been more or less neglected since the networking revolution began. HIDS (host based intrusion detection system) is the term used and such systems were really the first ones used to detect intrusions. The biggest advantage of the technique is that it can detect intrusions even in highly secure systems using encrypted communication channels. It is also easily adopted to distributed environments where the topology is constantly changing, such as peer to peer environments. Also other heterogeneous environments may benefit from using HIDS, for example when interoperability is required. Each node then can be using its own HIDS and yet cooperate with other nodes using other HIDS solutions, because their intra-communication can be implemented using standard COTS solutions.

But what to look for, what are the optimal set of attributes to use for detecting intrusions in a host computer? That is probably a really hard question to answer, since the perfect intrusion detection system has still not been invented. One possible way to find an answer might be to start from the foundation of a computer system, the hardware layer, and to look for signs left by intrusions on different layers all the way up to the user interface. That procedure is called intrusion analysis or computer forensics, depending on the circumstances, and is what this report is all about.

1.1 Background

This report is the report of the first year in the three year project named "Warfare in the IT-domain"[1], financed by the Swedish Armed Forces. The project aims at answering the following questions:

- How is a successful battle to be fought when using software based IT-weapons?
- What type of conventional weapons are needed in addition to the software based to win a battle in the IT domain?
- What types of IT related threats are there?
- What methods to use for IT surveillance and intrusion analysis?
- What is the current state-of-the-art within the area?

The main focus is on intrusion analysis and IT-surveillance.

1.2 Purpose

The purpose of this report is to work as a survey of the field of intrusion analysis and related fields. It is also meant to expose questions that need to be answered and in that way give a roadmap to future research within the field.

The report is tightly connected to the report "Intrusion Analysis in Military Networks – File Systems and Logging" [1]. They will, together with the other reports yet to be written in the project, form a suite covering different aspects of the intrusion analysis field, especially aimed at military networks.

[1]"Strid i IT-domänen" in Swedish.

1.3 Scope

The scope of the report is intrusion analysis and is limited to military networks. The differences between civil and military networks are not clearly defined however, and consequently the report might be applicable to any type of network intrusion analysis. It is a technical report presenting the current status of research in the field at the Swedish Defence Research Agency (FOI).

Intrusion analysis is closely related to the computer forensics field, but the report does not take any legal considerations into account. Nor does it specifically treat internal computer security policies.

The background survey covers what we felt were the main ideas and trends within each field in question. The selection was done by searching on Google for the words "computer forensics" and also "intrusion analysis". The same search was done on the ACM web site as well as on the LNCS (Springer Verlags) web site. All the papers found were then read and graded for perceived relevance to the report.

1.4 Structure

The report is structured into six chapters. The first chapter introduces the background and formalia concerning the report. Chapter 2 presents related work and theories in the intrusion analysis field. The following chapter, number 3, presents some practical experiences made during the work on building a honeynet for collecting research material. Chapter 4 discusses the different questions raised in the previous chapters. In chapter 5 some remaining research issues and unanswered questions are presented as well as future work related to the project. In the last chapter, number 6, the conclusions drawn from the material in the report can be found.

2. Related work

This chapter presents theories and papers from some related fields and the state-of-the-art in the intrusion analysis field. Each of the fields and their statuses are presented in separate sections.

The intrusion analysis field can really be seen as computer forensics without the legal requirements following by the criminal investigation connection. Computer forensics has been around as a profession almost since the first computers were invented, the intrusion analysis field on the other hand is much younger, it appeared when the trend of connecting computers into networks became popular. Thus by studying the computer forensics field the necessary theories for intrusion analysis are automatically included.

2.1 Computer forensics

The term computer forensics is really only one item in a tightly connected suite of terms. Some other, related terms that are used more or less interchangeably are; digital forensics, cyber forensics, network forensics are Internet forensics. They are all referring to the procedure of securing, collecting and interpreting evidence found in computer systems and networks. In this report the term *computer forensics* is used as a general term comprising all the others, unless otherwise specified.

2.1.1 Practical introduction Grundy has written an introduction to computer forensics on and with the help of Linux [2] focusing on the practical parts of the field. In the manual he presents Linux from the ground up and discusses installation as well as kernel, file system and boot sequence issues. About two thirds of the text is dedicated to the tools used and how to perform non-destructive collection of evidence. Most of the methods focus on how to do disk images and which tools to use when looking for anomalies and evidence.

2.1.2 Intrusion tracing Stallard and Levitt introduce an automated forensic analysis [3] called semantic integrity checking. They collect a base line of ordinary use and from there extract the data objects having redundancies that are necessary to get a secure system. The next step is to identify a possible victim of attack and look for contradictions in the system logs.

The authors use The Coroner's Toolkit [4] to collect data automatically. The data is then converted into XML format and used by JESS (The Rule System for the Java Platform), which is an expert system. JESS deduces asserted facts from the data and checks for inconsistencies. Conclusions are made concerning probable suspects (internal threats) or ways of entrance into the system (external threats).

Another approach [5] is to collect, at run time, configuration information. This information can then be used as a snap shot of a system at the time of a crash or intrusion. The information will provide an "as-is" state of the system, to be compared to the "should-be" configuration of the same system. By comparing the two configurations valuable forensic data can be extracted, possibly even giving a way of correlating data from distributed systems.

Computers that are used by an intruder to hide his or her tracks by logging in to several computers in sequence, jumping from one to another, are called *stepping stones*. When trying to backtrack such activities the forensic analyst often find her or himself stuck when the previous stepping stone used is to be found. In a paper by Carrier and Shields [6] a new protocol, STOP (Session Token Protocol), is introduced. The protocol can start collecting user and application data for a particular TCP connection. Special daemons running on each host in a system makes use of the IDENT protocol (Identification Protocol) [7] to be able to recursively trace connections.

King and Chen [8] propose a way, called Backtracker, of building dependency graphs from system calls and what objects they affect and then use the result to backtrack intrusions. The different objects used are files, file names and processes. The dependency graph shows how different objects are related and thus how the intrusion probably was executed. They have built a prototype currently implemented in a virtual machine environment, which makes it easier to log kernel events.

According to the authors there are three main ways of attacking Backtracker, generate large dependency graphs, attack the layers below Backtracker, and perform actions that are not monitored by the application.

Ning and Xu propose a way of learning attack strategies from correlated intrusion alerts by using attack strategy graphs [9]. By building such graphs for a number of attacks and then comparing them for isomorphy, similar strategies can be found. These strategies then can be used for improving for example responses to incidents. The strategies may also be generalized to different degrees depending on how they will be used.

The paper gives a detailed explanation of how the graphs are to be generated and also contain a set of experiments. The results of the experiments showed that the strategy learning method is depending on the underlying IDS and correlation engine. The authors explain that this limitation can be counteracted by using a heterogeneous set of IDSs.

2.1.3 Formal methods The research field needs to be standardized regarding methodology and procedures according to dos Reis and de Geus in a paper from 2002 [10]. They state that there are some differences between classical forensic disciplines and the new computer forensic field. The more important aspects are that the examinations are often located outside of a laboratory setting when doing computer forensic and that the new field is much more driven by the market and technology. Therefore the "computer forensic procedures and protocols should be written in a hierarchical manner so that good principles remain constant." [10, p. 9]

The authors state that some problems remain to be solved, among them the lack of tools and the problem of accessing encrypted data. They also mention the rapidly increasing amount of data to be checked for evidence due to the development in storage capacity.

Sremack presents a formalization of the computer forensic field in his master's thesis [11]. He first presents the current methodologies in computer forensics and also comparable approaches to data analysis in related fields. These different techniques are then used as a base from which he formulates his own computer forensics methodology, which is meant to fill in the gaps in the current methods.

His methodology is divided into four phases, macrocosm hypothesis, microcosm hypothesis, high level pattern matching, low level pattern matching. The macrocosm hypothesis is formed on a high, abstract level from previous knowledge of the system specification and the incident report. The microcosm hypothesis functions as a connection between the macrocosm hypothesis and the technical details and is meant to be used for example when evidence contradicts the macrocosm hypothesis. High level pattern matching is used to transform low level data into intermediate level evidence. The low level pattern matching is what traditionally is used in forensics to find patterns and anomalies in log data, etcetera.

Broucek and Turner present a taxonomy of computer forensics in a paper [12] from 2001. They also discuss similarities and differences between computer security and computer forensic. The taxonomy is a list of different fields involved in computer forensics taken from computer science, law, information systems, and social science. There is also a category they call "non-specific" [12, p. 8], which includes for example documentation and educational issues.

Regarding the similarities and differences they argue that the forensic field in computing is less mature, frequently performs examinations in the field instead of in a controlled laboratory environment, and also often involve non-technical experts. Also the question of how to react to a possible intrusion differs. In the computer security field disconnecting the affected service is often the good choice, while in the forensic field it might be better to let the intrusion proceed to enable more evidence to be gathered.

The practical aspects of computer forensics education and profession is discussed in an article [13] in IEEE Security & Privacy. The authors argue for the necessity of formally educated computer forensic professionals, different professional categories and what their educational requirements are. They also give proposals for a CNF (computer and network forensics) curriculum and discuss different approaches to the formation of academic level education in CNF. Such a program should have an extensive practical part involving learning-by-doing. To facilitate such training a well-equipped laboratory is needed, having a separate and dedicated network as to allow projects not suitable for public networks. The equipment should be heterogeneous and contain both workstations, servers and varying network equipment, such as routers, hubs, and switches. Also the software used should be heterogeneous and versions for Unix, Macintosh, and Windows should be available.

In an article in Communications of the ACM [14] the history of computer related forensics is reviewed. The author distinguishes between computer forensics, which he specifies as the well established methodology of law enforcement, and Internet forensics, which is the antipode to the black hat hacker community according to the author. The article also lists a number of tools that can be used for both hacking and Internet forensics. By giving this list the author argues for the similarities between the knowledge and techniques used by the hacker community and the knowledge and techniques needed to do Internet forensics.

2.1.4 Evasion techniques In [15] the author gives an introduction to the fields of computer forensics and anti-forensic methods, i.e. ways of degrading the effectiveness of computer forensics. He presents two main ways of performing anti-forensics, namely destroying information and hiding information, and different ways of achieving that. The text is an introduction to the field and consequently no detailed technical descriptions are given.

The previously mentioned paper by Johansson has its base in an article by grugq in Phrack [16]. It is a technical description of data destruction and data hiding. The article gives practical examples of how the TCT (The Coroners Toolkit) [4] from 2002 fails to implement the Berkeley FFS, or sometimes UFS (Fast File System) and the ext2fs (Second Extended File System) specification correctly. These file systems use *inodes* for pointing to disk blocks. Each file has a number of inodes keeping track of all blocks containing its data. The error makes it possible to hide any amount of information by using the first *inode*, the bad block inode. FFS has deprecated the use of that inode, but ext2fs uses it.

Because of the techniques used to delete files in ext2fs it is rather easy to recover them using TCT. The paper also presents a way of deleting files in a more thorough way by modifying the directory table and setting deleted inodes as virgin inodes in the inode table.

The techniques presented in the article are implemented in TDT (The Defiler's Toolkit), which is also included in the article in source code. The only way, according to grugq, to counter these techniques is to be pro-active and log all changes to the file system in an external, secure place. The article does not mention anything about if the techniques are applicable to the ext3fs (Third Extended File System), which basically is an ext2fs with a journal to record file system changes.

An follow-up article [17] by the same author presents a third method of hiding data from a forensic examiner. The method is called data contraception and aims at using only volatile memory, i.e. not writing anything to disk at the victim computer. This is done by using a common (vulnerable) utility as a server and then using it to inject and run code only in memory. Any non-standard tools used will run in the attacker's computer and consequently use its disk for writing. The requirement on the server utility is that it has the ability to act as either an "Inter Userland Device (IUD)– providing access to its own address space" [17, sect. 3] in RAM or an "Intra Userland Device (IUD) – providing access to another address space" [17, sect. 3], i.e. an external address space.

The following list specifies the main steps of performing a data contraception and is quoted from the article [17, sect. 6]:

- use an IUD to gain access to an address space
- upload the binary to execute into memory
- load the binary into an address space
- transfer control of execution to the binary

The author mentions three previous projects utilizing variants of the proposed methodology. These are MOSDEF, Core Impact, and ftrans, where the first two are commercial penetration testing tools and the last one is an anti-honeypot tool. He explains that Core Impact uses a technology called *syscall proxying*, which has some drawbacks, mainly referring to speed and problems with the fork() command.

MOSDEF was developed by Dave Aitel as an answer to the problems with Core Impact. It uses an Intra Userland Device and has a client equipped with a compiler, which gives the user the possibility to build programs on the attacked computer. There are however limits on the code that is possible to compile, it cannot be larger than a certain size and not too complex.

The third program is a pure anti-forensics tool built to operate in honey pots, where (almost) everything is logged. It uses a custom built Inter Userland Device and SSL. It also takes benefit of ul_exec() to execute a binary downloaded earlier. The binary is downloaded into the address space of the server and executed in a memory buffer.

2.1.5 Automatic signature creation Honeypots can be used to detect and analyze new kind of attacks. How this can be achieved is explained in a master's thesis by Patrick Diebold [18]. The thesis gives some background on general computer security and then presents the state-of-the-art within the honeypot and honeynet fields. The honeypots discussed are honeyd, Bait'n'Switch, the Intrusion Trap System, and Honey-Comb. The honeynets presented are generation 1 and 2 of Lance Spitzner's work, VMWare, and User Mode Linux. The author concludes that any of them can really be used for automatic detection and signature creation of new, unknown attacks.

The author suggests that a way to detect new attacks is to use an IPS (intrusion prevention system) to filter already known attacks. The IPS works in tandem with a firewall that blocks attacks emanating from the honeypot, i.e. protects Internet from attacks. The honeypot that acts as a bait should be a so called medium interaction honeypot, which by the author is defined as running real services that can be attacked, but in a sandboxed environment to make it secure. The next part in the proposed system is a logging facility that logs "as much as necessary and as little as possible" [18, p. 32]. There is also an IDS included to be used for tagging of successful attacks.

2.1.6 Data volatility The volatility of data in a computer can vary from nanoseconds to months and maybe years. The problem is discussed in a short paper on forensic analysis [19]. The author states that computers are often used in rather static routines, which results in only a small portion of the files being changed often. Remnants of deleted files can therefore exist for days or even weeks. But the chance of being able to recover material from magnetic disk is diminishing because of the development in hardware, were the bits are being packed more tightly for each new generation of disk drives.

As an experiment a dedicated mail, web, and ftp server was checked for remnants of deleted file time stamp attributes over a period of time and the author writes that after 20 days half of the deleted files attributes were overwritten and some remained for 100 days. When the same experiment was performed using a workstation the results fluctuated significantly, but there were still some time stamps left after 100 days.

Also the contents of a files remain for long periods even after the file has been deleted. In another experiment the author used half a dozen machines that were used daily, each night a digital hash was calculated for each disk block. The results showed that file contents was gradually overwritten, half of it remained after two to five weeks depending on how busy the machine was and the amount of free disk space.

The main memory, RAM, of a computer typically looses its content soon after the power is switched off, but some computers are always turned on and their main memory can almost permanently hold frequently used files. Also less frequently accessed files can still remain in main memory for hours, according to the author.

When a file has been deleted its contents is still remaining on disk. Building on that Park, Kim, and Noh [20] has developed a way of detecting executable ELF (Executable and Linking Format) files based on their structure. Their method can even handle fragmented files.

An executable ELF file consists of a header and then a varying amount of machine code. The machine code contains an operation and then some related bytes. The length of machine code instructions can differ, but the authors studied the statistical distribution of the distances between the operation codes and found that 5 bytes was a suitable value to use. They did also look at the ratio of instructions in an ELF file and concluded that a ratio of 40% is suitable to distinguish between ordinary binary files and ELF files. In experiments they reached a detection rate of 70% for detecting executable files, with 5% false alarms.

2.1.7 Logging A paper [21] by John Tan presents a description of the different logging facilities used in the main types of operating systems. He also discusses what, how and where forensic relevant activities should be logged. Tan writes that “What is not logged is lost” [21, p. 8]. He then admits that this can be unmanageable due to time and space constraints. Tan also states that centralized logging is the best choice, because then the logs are easily handled and backed up. They are also more secure than if they were stored at the possible victim of an intrusion.

The problem of logging is also discussed by Ben Laurie in an article. He states that the usual logging performed is of little use because it is centered on debugging [22, p. 54]. According to him something more useful for intrusion analysis would be to log the interaction with a software and to do it both before and after a request is acted upon. To make the logs useful they have to be possible to read automatically by text processing tools, for example *sed*, *awk*, *perl*, etcetera, i.e. only contain ASCII characters, have one line per logged event, and have unambiguous formatting and markup. He also comments on the security of logs. He concludes that preferably logs should be stored remotely on a secured host. The last remark in the article argues for planning for the inevitable, everybody involved in the computer society should be pro-active and plan ahead.

2.2 Incident handling

Incident handling is many times referred to as intrusion response. The field does however comprise other issues as well. Regarding how the response part is defined they can very well be isomorphic, but often incident handling is regarded as also including restoring a system to an non-compromised state. Amoroso does however include system restoration into the response process. He also deems it the most crucial part in some cases and picks an Internet service provider as an example [23, p. 189].

There is also a problem concerning the automation of response actions. Often a manual approach is out of the question because of the high frequency of alerts, it is impossible to cope with the workload for a human being. Currently the automatic responses used are often statically linked to a certain alert type and thus can possibly cause even more damage if the linking is erroneous. A solution to the problem is suggested by Toth and Kruegel [24]. They try to model a network (i.e. system) and the dependencies among its components. By coding the dependencies into a tree and then perform depth-first searches they can find the least costly response and also present different alternative responses to an administrator to choose from. The authors write that finding a global optimization can not be done in real-time, but that does not matter because there are no strict real-time demands on the responses taken, according to them.

3. Practical experience

To collect real data to be used for hands-on tests and evaluations of different tools a honeynet was built in the information warfare laboratory at FOI. The honeynet is run when specific experiments are set up and is not running all the time. Therefore it is not involved in any of the honeynet projects currently active on the Internet.

At the time of writing of this report two separate experiments has been run on the honeynet. The first experiment involved looking at the time to infection of three versions of unpatched and unprotected Windows hosts. The hosts were running Windows 98, Windows 2000, and Windows XP. The second experiment studied how poorly implemented SSH (Secure Shell) servers were hacked and also for what purposes.

3.1 Honeynet

The honeynet consists of a bridge working as a reversed firewall, i.e. letting everything in but almost nothing out, and up to nine honeypots in the form of real computers. To enable some interaction the bridge is set to throttle the outgoing traffic and only allow 20 kb/s of ICMP (Internet Control Message Protocol) traffic, restricted to *ping* and *traceroute*. There is however currently a rule set to deny all UDP traffic and only allow TCP traffic flows initiated from the outside. This setting was chosen to prevent the honyepots from being used as DoS (Denial of Service) zombies or stepping stones for hacking other computers outside the honeynet.

The bridge is accompanied by a logging host running *snort* and *tcpdump*. The logging host is invisible from the Internet and thus cannot be hacked. It will record every packet flowing through the bridge. In this way all non-encrypted traffic can be view in clear text with the help of for example *ethereal* [25].

The use of real computers as honeypots enables real hands-on training in all the steps involved in performing intrusion analysis. In that way it is possible to directly test different hypotheses made about the structures and procedures of intrusions. It also enables us to collect data to be used for evaluating different tools for intrusion analysis and forensics.

The next development of the honeynet computers will be to implement a kernel logging function to enable recording encrypted traffic in clear text. This will most probably be done using a special kernel module hooking the syscalls. It will also be possible to see commands entered in shells.

We will also change the rules of the reversed firewall to increase the level of interaction from the outside. By doing that we will have more possibilities of recording full-blown intrusions and also to collect source code from worms and other automatic tools. In the end this will give us constantly fresh material to further increase our intrusion analysis capabilities.

The data from the two experiments described in the following subsections was analyzed using ethereal for Windows running on an ordinary laptop. It was filtered in different ways to more easily reveal what was deemed the relevant parts of the traffic. The filtering was always done on the complete data set, which gave us the opportunity to follow the time line of the intrusions from the beginning.

3.1.1 Time to infection The experiment studying the time to infection for different versions of Windows used the first, i.e. unpatched, editions of Windows 98, Windows 2000, and Windows XP. All were installed with the default services and none of them were patched or secured in any way. This was done to make them as equal to standard home computers as possible. The matter of vulnerable home computers is further discussed in an article in Framsyn [26].

All of them were connected to the Internet at the same time. After approximately 30 seconds the computers running Windows 2000 and Windows XP were infected with RBOT [27] and Welchia [28]. The infection caused the two computers to start scanning each other as a step in the spreading of the worms. Because of the reversed firewall the worms did not manage to download any payload and thus could not perform any full infection of the computers. Hence there was only scanning traffic in the honeynet and consequently no more interesting traffic was collected.

The computer running Windows 98 remained untouched throughout the experiment. The probable reason for that was the lack of started and consequently exposed services in the default installation of Windows 98. However if the computer had been used in any way regarding Internet, for example reading mail, the probability of a successful attack had increased.

Several things remain to be done in this experiment. The disks of the computers need to be analyzed to see what traces the intrusions left behind. This might seem a trivial task to perform because we already know what hit the computers. Examining the disks will however be a good opportunity to train on a well documented practical example.

The experiment can also be run again using a different set of operating systems, or using the same systems with different patch levels. Another variation would be to use the computers for surfing the web, reading e-mail and other tasks that might be probable on a home computer. A interesting thing to test would be to connect to one of the peer-to-peer networks currently run on the Internet. Such an experiment would have to be properly prepared as not to violate any laws or copyrights.

3.1.2 Hacked SSH server We had noticed an increase in attack attempts at the SSH port (22) in our servers. Therefore we decided to set up a vulnerable SSH server and study how the attacks were made and also try to find out possible purposes for the attacks. To do this we used a RedHat 7.3 installation with the default firewall set to let only SSH traffic through.

After three days an attacker took our bait and performed a successful intrusion. The attack started with an automatic scan of the computer at several occasions. Each scan performed a login that was interrupted after a few steps. We could not read any of the encrypted packets sent during the hand-shaking procedure, but probably the tool performed password guesses. The tool also seem to have sent or recorded data regarding the address and other relevant information of our machine, because a few hours after the initial scans a new intrusion took place.

The second stage in the attack was most probably done manually, because we saw a mistyped command being issued and then corrected when we looked at the logfiles. The intruder set up an SSH session to our honeypot twice. The first session lasted for approximately 15 minutes. The second session lasts for about one minute.

The data collected during this experiment is interesting and will enable us to train on a higher level than with the data from the first experiment. In this case also the disk data might be of more interest because of the manual intrusion. There might be erased or otherwise manipulated log files, new files created, and other changes to the status of the honeypot.

4. Discussion

This chapter discusses some of the issues raised in the previous chapters. The main weight will be put on Chapter 2, related work.

4.1 Intrusion tracing

Intrusion tracing is really the core of intrusion analysis. By being able to follow the intruder and his or her steps through the system the intrusion can be analyzed. The idea to automate this process is good, but only if it is used as a help or support for a skilled analyst. If the method is trusted beyond its capacity it can be manipulated by a skilled hacker, which then can avoid detection, even if a human would have spotted the anomalies and unavoidable traces from the intrusion. A crime performed by a human must be responded to with equal wit and cunning. Consequently the computer based aids used should remain only tools, they should never be let to do judgments on their own.

Another issue is the privacy of the users in a system where an automatic tracing tool is used. As many of the papers point out, the best way of being able to perform a successful intrusion analysis is to be preemptive. This can be done by logging everything happening in the system, but then also the doings of the ordinary, good users of the system are recorded and possible to study. This is an ethical issue that has been debated for years and probably will continue to be. Thus every logged data item must be weighed morally to see if it really is necessary to log it, or if the cost of decreasing the privacy is to high a price to pay.

The quality of the information used for performing automatic tracing is also of uttermost importance. It can be degenerated in many ways, the first one that springs to mind is manipulation by the intruder. Every piece of data used for intrusion analysis must be regarded as possibly being contaminated. An automatic tool may support the analyst, but can certainly not be allowed to decide such things on its own.

4.2 Formal methods

As dos Reis and de Geus indicates in their paper [10] the problem of the ever increasing amount of data to be sifted through by an intrusion analyst is rapidly becoming overwhelming. We agree on that, but we also feel that the problem has to be solved once and for all. One possible way to go might be to find the optimal set of data to log, to find out what we really need to know to be able to spot an intruder. A comparison to the classical field of forensics is possible to make, they look for human traces, pieces of DNA that can be used to find the perpetrator. We need to find the same type of basic and almost ever-present piece of evidence, a type of DNA of the computer world.

One thing that might work against finding such evidence is encryption, a function that is not (yet) possible to do in the real world of DNA, at least not for the ordinary thief. Encryption that prevents intrusion analysis can be used by both the police and the criminals. When we encrypt our network traffic we at the same time diminish our possibility to recreate an intrusion, at least not if we do not have access to the encryption keys, which is really against the whole idea of encryption.

Criminals can use encryption to hide their traces, the intrusion analyst will still be able to see that something has happened, but not what. Finding encrypted information on a computer might be a good indicator of an intrusion, but what is gained on the detection side is lost on the analysis side of the spectrum.

The authors of the article say that because of the rapid development within the computer field, the procedures of intrusion analysis must be formulated in a hierarchical way to enable quick and easy changes. That sounds fair, but the procedures should really be the same and governed by sound scientific (or forensic) methodology. The techniques used should on the other hand be state-of-the-art, although they need to be checked and approved for correctness before use.

The same argument can be used for the other article [11] presenting a proposal for a formalization of the procedures of computer forensics. His methodology, which is divided into four phases, is really common sense. Probably it is already followed by every intrusion analyst, even if they do not know it. Hence the formalization might be good, but only useful for education in situations where the teacher is not himself familiar with forensics or a scientifically sound methodology.

The idea of a forensics education and profession presented in IEEE Security & Privacy [13] is really good. The authors argue for the formation of an academic level education and present a curriculum. We think a few things are missing in there, though. For example there is a need of non-technical aspects to be considered. First of all psychology is vital, there also need to be some kind of sociology studies included, where the students could look at the morals and behaviors of the hacker community. As a result the students would be able to produce simple attacker profiles, which would help them to later on plan and execute a live intrusion analysis.

4.3 Evasion techniques

The online hacker magazine Phrack has run two articles by grugq where he or she explains ways of evading intrusion analysis. The techniques mentioned in the first article [16] is defeated by logging changes to the file system, according to the author. Currently this is more or less exactly what is done, although the changes are not permanently logged. By using journaling file systems the changes made to the file system are possible to record more or less fully. The technique is there, it only has to be implemented. The penalty will be on the performance side, the time to write to disk can very well double.

In the second article [17] grugq presents a method to run code on a remote machine without leaving any traces. To be able to do this a vulnerable service on the victim computer must be at hand and that is where the weakness of the method is. It is rarely possible to get a shell on a computer without performing some kind of buffer overflow or other malicious action. By simply using ordinary computer security methodology, i.e. using firewalls, buffer overflow protection and regularly patching the software, attacks of this kind will be very hard to execute.

4.4 Automatic signature creation

The suggestion to use honeypots as means of automatic attack signature creation is not really feasible as we see it. First of all it is fairly easy to fingerprint, i.e. detect, a honeypot. In other words, the attacks collected are more or less only made by worms and other automatic tools. If a manual hacking attempt is made the hacker will most probably abort the attempt soon after he or she has detected the victim as being a honeypot. Should the intrusion proceed anyway there is two alternatives available, either the intruder is not skilled at hacking, or the traces left in the honeypot are decoys and only there to confuse the intrusion analyst. In either case the logged information is of little or no use.

The author writes that "as much as necessary and as little as possible" [18, p. 32] should be logged. He does not state what that is or how it should be measured. This is another of the key issues we need to find out, thus the author is right, but does not propose any solution.

The last thing we consider questionable is the filtering of known attacks. The idea might look good on paper, but in reality it is infeasible. Most of the intrusions are made using standard attacks, but put together in different combinations. By filtering known attacks the information leading to the use of a new technique might very well be lost.

4.5 Data volatility

The work done by Venema [19] on the persistence of data in different storage types is really interesting and thus has to be followed up. It is however somewhat sketchy on the RAM side, we want to see more research being done there.

Recreating executable ELF files by reassembling them only governed by the statistical measurements of the raw data is also something we would want to see more of. The results reported [20] are somewhat low to be really usable, but by continuing the work following the main idea, better results ought to be possible to achieve. This is an interesting and important part of the intrusion analysis field, in need of further attention.

4.6 Practical experience

The practical experiences gained by implementing our honeynet has been very valuable. It has allowed us to use real intrusions to train on and also as a testbed to develop logging facilities for syscalls and other kernel related functions.

Another advantage is that we have the possibility to extract different types of statistical data from the honeypots. The data has not yet been thoroughly evalauted, but will soon be.

The hacked SSH server showed us the difficulties of doing real life intrusion analysis. We thought we had logged enough data and yet quickly saw that we had missed valuable information. The experiment also gave us indications of what to look further into, for example how to defeat encrypted network traffic.

Other uses for the honeynet is for testing computer security tools, especially regarding host based intrusion detection. It may also be used to collect hacker tools, worms, and viruses.

5. Future work

In this chapter issues and questions left for future work are presented. As this is the first year report most of the work is still left to be done. We will not be able to cover all of the questions that are raised in this chapter during the two remaining years, but we have included them here both for others to be able to continue our work and as a guide for us.

5.1 Tools

One important issue left as future work is the question of surveying the tools used in the intrusion analysis field. As a start a brief survey of some of the web sites dedicated to forensics and intrusion analysis was made. There lists of tools were found, some of them seemed rather comprehensive.

At the forensix.org's web site there is a long list of different computer forensics related tools [29] and toolkits [30]. There are also a list of tools for integrity checking and management [31]. Another web site with a list of tools is l0t3k.org [32]. The contents are approximately the same, the tools are categorized differently. A third source for tools is [33] where Brian Carrier has collected links to different open source forensics tools.

More or less the same tools appear in all the lists and at several other web sites too. This might either indicate that the tools are good, or that they are the only tools available, or that the forensix.org lists are the sources to the other lists.

One thing that can be concluded from the lists of tools is that there is a large overlap between the set of tools used for hacking and the set of tools used for computer forensics. This was also mentioned in a paper by Berghel [14].

The tools found during a survey of the field will also have to be evaluated to see if and how they are suitable for a military environment. This will be a good opportunity to classify them and in that way see if there might be any types of tools missing.

Some of the types of tools we feel would be beneficial to a military intrusion analysis are presented in the list below. The list is not meant to be exhaustive, nor does it only contain tools strictly meant for intrusion analysis.

Log wiping, what tools are there and what can they do? We need to know how logs are wiped to be able to find out ways of restoring them. Log wiping can be extended to also include all types of file wiping software.

Disc wiping, the same questions as for log wiping applies to this category. The difference is that disc wiping may be done in a more permanent way, by even destroying a disk. How is the wiping affected by manipulation of the disk controller and its settings?

Swap browsing, the swap file may contain traces of the RAM, which is not structured in any particular way. Are there any tools that can be used for structuring the swap file of a computer and allow easy browsing of it?

Journal time lining, are there any tools that can create a time line of the disk access of a journaling file system, if the possibility to save the journal is implemented. What should be the requirements of such a tools?

File or data assembling, can that be achieved using only raw blocks? At least in the Linux world the file systems are implemented as to write data sequentially to disc blocks to decrease the time for disk access. Different file types have different structure, can a file be reassembled using only that information? How about slack space, are there ways of reassembling data from only parts of a disc block? Are even other layers or types of information, such as inodes, possible to reassemble with the help of block contents?

Disk imaging, there are tools to copy a disk sector for sector, but how do they do that? How do such tools handle unpartitioned space, "bad" sectors, etcetera? Are there tools for imaging non-standard hardware, PDA:s, mobile phones?

Volatile memory handling, different types of memory retain their contents for different periods of time. RAM is one of the more volatile types. What tools are there to save or browse the contents of RAM? What can be done for the contents of the registers in the CPU? Is performing controlled core dumps a suitable solution?

Last but not least, there are a lot of toolkits and suites put together. What are the requirements of the perfect intrusion analysis tool or toolkit? Can such a tool be created using already existing tools, or are there any new tools that have to be created?

5.2 Persistency of data

The issue of volatility was mentioned above, but needs to be further discussed. How much volatile information is there really and for how long does it remain? Examples of interesting types to look at are RAM, process information, core dumps, swap files, and cached information on both low and high level. Is for example the RAM and swap file emptied upon boot, or can information easily be extracted by rebooting into a very small footprint operating system and then accessing the ram that is not touched by the OS?

There are also different networking equipment that can be of interest, such as routers, switches, etcetera. The same questions as above are applicable to that equipment too.

5.3 Security requirements

Different types of system have different security requirements. One rather intuitive way is to divide the systems into two main types, namely civil and military systems. Their requirements differs a lot, even within the two system types. Sometimes the requirements of two system within the same group differs more than do two systems from different groups.

The main difference regarding the two groups is how the three classical pillars of computer security, confidentiality, integrity, and availability are prioritized. Military systems have a tendency to put more weight on the confidentiality than civil systems, but as mentioned before this can differ significantly within a group.

These requirements are rather important to how an intrusion analysis can and should be done. Therefore we have to find out more exactly what special requirements a military system has compared to a civil system.

5.4 Non-technical aspects

The report has dealt only with technical aspects of intrusion analysis, but if there were no users interacting with the systems, there would not be any security related problems to talk about. Thus we need to properly deal also with the softer side of computer security and make an investigation of how the users can be utilized by an attacker. This is often referred to as "social engineering" and is a skill that is very important during the initial steps of an intrusion.

Relating to the social engineering skills of a hacker the aspect of attacker profiling is at close hand. How do the psychological profile of the intruder affect the way the intrusion is performed? Can the profile of the intruder be extracted or detected during an intrusion, and if so, is it possible to do any kind of responsive actions to stop the intrusion and catch the intruder?

6. Conclusion

This chapter contains the conclusions drawn from the material presented in the report.

From the survey of related work we can conclude that there are active research being done in the field. There are however still some areas where more research is needed. These are for example:

- a continued work on forming a curriculum for higher education in the field,
- the automation of data mining in logs to work as an aid for the intrusion analyst,
- continuous studies of the state-of-the-art within the hacker community,
- research on the optimal set of attributes for detecting an intrusion, and
- measuring and documenting the volatility of data on different storage media.

The connection between the hacker community and the intrusion analysis field needs to be stressed more to get a good understanding of it throughout the field. An obvious proof of the connection is the tool set of both fields, where many of the tools used by hackers are also used by intrusion analysts.

Bibliography

[1] A. Vidström, M. Persson, and M. Karresand, "Intrusion analysis in military networks – file systems and logging," Dept. of Systems Development and IT Security, Command and Control Systems, Swedish Defence Research Agency, P.O. Box 1165, SE-581 11 Linköping, Sweden, Tech. Rep. FOI-R–1518–SE, Dec. 2004.

[2] B. Grundy, *The Law Enforcement and Forensic Examiner Introduction to Linux – A Beginner's Guide, ver. 2.0.5*, NASA Office of Inspector General, Computer Crimes Division, Code 190 Greenbelt Rd., Greenbelt, MD 20771, USA, Jan. 2004, http://www.linux-forensics.com/linuxintro-LEFE-2.0.5.pdf, last visited 2004-11-22.

[3] T. Stallard and K. Levitt, "Automated analysis for digital forensic science: Semantic integrity checking," in *19th Annual Computer Security Applications Conference.* Applied Computer Security Associates, Dec. 2003, http://seclab.cs.ucdavis.edu/papers/Stallard-ACSAC'03.pdf, last visited 2004-11-22.

[4] D. Farmer and W. Venema, "The coroner's toolkit," http://www.porcupine.org/forensics/tct.html, last visited 2004-11-22.

[5] D. Heimbigner, "Applications of configuration information to security," in *SCM*, B. Westfechtel and A. van der Hoek, Eds., vol. 2649, ICSE. Springer, 2003, pp. 259–266.

[6] B. Carrier and C. Shields, "The session token protocol for forensics and traceback," *ACM Transactions on Information and System Security*, vol. 7, no. 3, pp. 333–362, Aug. 2004.

[7] M. Johns, "Identification protocol, RFC 1413," 1993.

[8] S. King and P. Chen, "Backtracking intrusions," in *Proceedings of the nineteenth ACM symposium on Operating systems principles.* ACM Press, Oct. 2003, pp. 223–236.

[9] P. Ning and D. Xu, "Learning attack strategies from intrusion alerts," in *Proceedings of the 10th ACM conference on Computer and communications security.* Washington, D.C., USA: ACM Press, Oct. 2003, pp. 200–209.

[10] M. dos Reis and P. de Geus, "Standardization of computer forensic protocols and procedures," in *The FIRST'02 - 14th Annual Computer Security Incident Handling Conference*, vol. 1. The Forum of Incident Response and Security Teams (FIRST), June 2002, pp. 15–20.

[11] J. Sremack, "Formalizing computer forensic analysis: A proof-based methodology," Master's thesis, North Carolina State University, 2004, http://renoir.csc.ncsu.edu/Faculty/Vouk/Papers/Sremack/Sremack_MS_Thesis.pdf, last visited 2004-11-22.

[12] V. Broucek and P. Turner, "Forensic computing – developing a conceptual approach for an emerging academic discipline," in *5th Australian Security Research Symposium*, Perth, Australia, July 2001.

[13] A. Yasinsac, R. Erbacher, D. Marks, M. Pollitt, and P. Sommer, "Computer forensics education," *IEEE Security & Privacy*, vol. 1, no. 4, pp. 15–23, 2003.

[14] H. Berghel, "The discipline of internet forensics," *Communications of the ACM*, vol. 46, no. 8, pp. 15–20, Aug. 2003.

[15] C. Johansson, "Forensic and anti-forensic computing," Department of Software Engineering and Computer Science, Blekinge tekniska högskola, Karlskrona, Sweden, Tech. Rep., Dec. 2002, http://www.fukt.bth.se/~uncle/papers/forensics200212.pdf, last visited 2004-11-19.

[16] grugq, "Defeating forensic analysis on unix," *Phrack*, vol. 11, no. 59, July 2002, www.phrack.org/show.php?p=59&a=6, last visited 2004-11-19.

[17] ——, "Remote exec," *Phrack*, vol. 11, no. 62, July 2004, www.phrack.org/show.php?p=62&a=8, last visited 2004-11-19.

[18] P. Diebold, "Usage of honeypots for detection and analysis of unknown security attacks," Master's thesis, Technische Universität Berlin, 2004, http://user.cs.tu-berlin.de/~dakkonbb/honeypot/diploma.pdf, last visited 2004-11-22.

[19] W. Venema, "Forensic analysis," Conférences SPIRAL 2002, Nov. 2002, http://www.spiral.lu/SI/Event.nsf/0/9f98ab0e38c4c40fc1256c62004c3249/$FILE/021027_conf2002_Forensic%20Analysis_Wietse_JP.pdf, last visited 2004-11-19.

[20] J. Park, M. Kim, and B. Noh, "Detection techniques for ELF executable file using assembly instruction searching," in *Computational Science and Its Applications — ICCSA 2004: International Conference, Proceedings, Part I*, A. Laganà and et al., Eds. Heidelberg: Springer-Verlag, 2004, pp. 230–237.

[21] J. Tan, "Forensic readiness," White paper, stake, Inc., 196 Broadway, Cambridge, MA 02139, USA, July 2001, http://www.atstake.com/research/reports/acrobat/atstake_forensic_readiness.pdf, last visited 2004-11-22.

[22] B. Laurie, "Network forensics," *ACM Queue*, vol. 2, no. 4, pp. 50–56, June 2004.

[23] E. Amoroso, *Intrusion Detection – An Introduction to Internet Surveillance, Correlation , Trace Back, Traps, and Response*, 1st ed. Sparta, New Jersey, USA: Intrusion.Net Books, Jan. 1999.

[24] T. Toth and C. Kruegel, "Evalauting the impact of automated intrusion response mechanisms," in *18th Annual Computer Security Applications Conference*. IEEE, Dec. 2002, pp. 301–310.

[25] G. Combs, "Ethereal – the world's most popular network protocol analyzer," Oct. 2004, http://www.ethereal.com/, last visited 2004-11-29.

[26] M. Karresand and A. Vidström, "Datorn blev sjuk direkt," *Framsyn*, no. 4, pp. 32–33, 2004.

[27] B. Enconado, "Worm_rbot.ao," Sept. 2004, http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_RBOT.AO, last visited 2004-11-29.

[28] F. Perriot and D. Knowles, "W32.welchia.worm," July 2004, http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html, last visited 2004-11-29.

[29] "Computer forensics tools, software, utilities," Nov. 2004, http://www.forensix.org/tools, last visited 2004-11-30.

[30] "Computer forensics toolkits and suites," Nov. 2004, http://www.forensix.org/toolkits, last visited 2004-11-30.

[31] "Integrity management software," Nov. 2004, http://www.forensix.org/integrity-management, last visited 2004-11-30.

[32] "l0t3k.org," http://www.l0t3k.org/security/tools/forensic/, last visited 2004-12-02.

[33] B. Carrier, "Open source forensic tools," http://www.opensourceforensics.org/tools/index.html, last visited 2004-12-02.