Active Honeypots

Master Thesis in Computer Science

submitted by

Dieter Joho
Zurich, Switzerland
Register No. 98–913–890

Supervisor:
Dr. Reinhard Riedl

Department of Information Technology
University of Zurich, Switzerland

December 16, 2004

Contents

Abstract

Honeypots, systems to lure and research attackers, are subject to intensive research for quite some time. Whereas those systems led to significant progress in IT security, this technology is far from being mature yet.

Based on a honeypot cost-benefit analysis it is pointed out, why this technology did not make it to become an essential part of todays IT infrastructure yet.

Active honeypots are systems, that optimize the collection of informations about the proceedings and the tools used by attackers, on the basis of adaptive generated targets. This should propagate the use of deception strategies and bring more insights about the proceeding of attackers in return. The deployment of active honeypots implies difficulties and raises questions about the feasibility of such systems.

A feasibility study of active honeypots outlines the potential of this research domain and future developments in the field of active defense systems.

Chapter 1

Introduction

This master thesis is an *active honeypot* feasibility study.

1.1 Motivation

A honeypot is a deception system that allures attackers with the actual intention to observe them. Honeypots have no productive value. Thus, there is no reason for anyone to interact with a honeypot at all. Therefore, every connection attempt to such a system is most likely a probe, scan or attack. An attacker who breaks into a honeypot is comprehensively monitored. A honeypot is a facility to learn the tools and tactics intruders use.

The primary goal of this master thesis was the development of an active honeypot system. An active honeypot is an advanced honeypot that autonomously reacts to security incidents. It generates adaptive targets which are suitable to the ambitions of the intruder. By challenging attackers, more information about their proceeding and their tools can be gained than with traditional honeypots.

During the work on this master thesis, it turned out that the state-of-the-art in honeypot research was insufficient to achieve the original goal. Therefore the goal had to be adapted towards an active honeypot feasibility study. It covers the state-of-the-art in honeypot technology and current projects in automated attack response. The problems that have to be solved in order to proceed in active honeypot research are brought up as well as proposals on the further proceeding in active honeypot research.

This master thesis primarily addresses people who are involved in honeypot research. It establishes a basis for further decisions on how to proceed in active honeypot research. Particularly conclusions are important for decision-makers who are interested to become engaged in active honeypot technology.

The feasibility study is methodically based on an analysis of existing literature, interviews with security professionals and experiments with traditional honeypots.

1.2 Structure

This thesis is divided into eight chapters.

1. Introduction
2. Intrusion Detection Systems
3. Introduction to Honeypots
4. Honeypot Cost-Benefit Analysis
5. Active Honeypots
6. Problems of Active Honeypots
7. Active Honeypot Feasibility
8. Summary, Conclusions and Future Work

Chapter two gives an overview of intrusion detection systems. It covers the goal of intrusion detection and presents todays concepts and tools. The state-of-the-art in intrusion detection and shortcomings of such systems are presented. The chapter closes with thoughts about the future of intrusion detection systems.

Chapter three is about honeypots and starts with an in-depth definition of that term. It gives an overview of the different types of honeypots and their preferred field of application. The value of this kind of deception system is presented and discussed. The section that compares intrusion detection and honeypot systems is particularly useful for security professionals who are interested in improving their IT security infrastructure.

Chapter four of this master thesis takes a closer look at the costs and the benefits of honeypots. It starts with a discussion about the evaluation of security investments and takes a closer look at the return on security investments. The benefits of honeypots are outlined by examples of possible uses. The main part of this chapter analyzes the costs of honeypot systems. Based on the results from that cost-benefit analysis, conclusions about the future are drawn. This chapter addresses decision-makers who are intending to use honeypots. It gives an overview of possible uses and helps to estimate the costs and benefits.

Chapter five starts by defining the term *active honeypot*. The section about the motivation for such advanced deception systems is particularly important for people who are involved in IT security research. A generic concept for such deception systems is presented. This chapter is looking for application scenarios and examples in theory and practice, to get an overview of the current state of active honeypot technology. It ends with a proposal of an architecture for an active honeypot system.

Chapter six illustrates the drawbacks of active honeypot technology. Attacker identification and classification, automated incident handling and automated attack response are challenges that are not yet solved. How attackers defeat honeypot systems is described as well as how to protect these deception systems. These problems allow to make predictions about the further development of active honeypot technology.

Chapter seven is the actual active honeypot feasibility study. It takes a closer look at the feasibility of adaptive target generation, automated incident handling and automated attack response. All these subjects are investigated with respect to technical, economical and legal feasibility. Educated speculations about future research in active honeypots are made on the basis of the subjects mentioned.

Chapter eight is a summary of this master thesis. It includes a definition of the term active honeypot as used in this thesis and describes the methodical approach. The feasibility of adaptive target generation, automated incident handling and automated attack response are summed up. Examples of more simple active components, than the ones from chapter five, are presented. The main difficulties of this work are described. Prerequisites for Future research close this last chapter. This part of the master thesis addresses security professionals and decision-makers who want to get a quick overview of active honeypots, its main problems and future work.

Chapter 2

Intrusion Detection Systems

2.1 Introduction

Computer security is a topic that has grown rapidly over the past few years. Through the growing number of computers connected to the Internet, new benefits as well as new threats have arisen. Systems and networks are attacked on a daily basis from everywhere on the world. Prevention tools like firewalls cannot protect networks on their own anymore. Attackers get smarter and they are able to overcome such prevention systems.

To get a picture on what is going on inside computer networks, intrusion detection systems (IDS) grew in popularity. Nowadays, they build an important element of most security infrastructures.

This chapter gives an overview of intrusion detection systems. It covers the goal of intrusion detection and presents todays concepts and tools. The state-of-the-art in intrusion detection and shortcomings of such systems are presented. The chapter closes with thoughts about the future of intrusion detection systems.

2.2 The Goal of Intrusion Detection

Intrusion detection is the art of detecting inappropriate, incorrect or anomalous activity in a networked computer environment. It is the process of identifying a weakness before it results in some kind of failure. This needs to be fast and reliable to stop unauthorized activity before it can do any harm.

A definition of the Term *Intrusion Detection System* can be found on Wikipedia[1]:

"An Intrusion Detection System (IDS) is a tool used to detect attempted attacks or intrusions by crackers or automated attack tools,

[1] http://wikipedia.org

by identifying security breaches such as incoming shell code, viruses, malware or Trojan horses transmitted via computer system or network."

An intrusion detection system is an active process or device that analyzes system and network activity for suspicious behavior. This can either be unauthorized use of IT resources or even malicious activity. The way an IDS detects suspicious behavior varies widely. It informs the system administrator if a successful attack occurred or if an attack is in progress. A good IDS informs the system administrator while the attack is still going on and presents some kind of diagnostic. This can either be information about the attacker himself or the target of the attack. Some intrusion detection systems are even able to suggest some counter measures. The ultimate aim of any IDS is to catch perpetrators red-handed, before they can do real damage to the computer infrastructure.

Intrusion detection systems protect computer networks from attacks, compromises and misuse. Further they monitor network activity, audit network and system configurations for vulnerabilities, analyze data integrity and more.

2.3 Intrusion Detection System Architecture

This section presents a general architecture of an intrusion detection system. It covers the functional components that are needed to build an IDS as well as some administration components that are useful to create large-scale intrusion detection systems.

2.3.1 Functional Components

An intrusion detection system consists of three main components according to Christian Goetz [Goe]:

Data Collector: A data collector gathers all sort of information about any activity on the network or the system. This component uses all sort of logs to collect information about the system state, resource usage, network activity or system calls. It is important that all these informations will be presented to the following components without loss.

Data Analyzer: A data analyzer processes the information given by the data collector in order to detect suspicious activity. There are different ways of how to process the collected data to detect attackers as described in the section about detection techniques.

Result Display: A result display component will work up the information given by the former components and present the result. In case of an attack this component generates an error message to inform the system administrator or launches some intrusion response.

A system that combines all of these three components is called a *sensor*.

2.3.2 Administration Components

A good intrusion detection system consists of several sensors that are placed on important devices or nodes to collect as much information as possible. Therefore it is useful to have additional components, that allow to control and configure all these sensors. This components are called *Sensor Configuration Management* and *Event Notification*. Sensor configuration management does the remote control and configuration of each sensor. It is also responsible for the permanent saving of the configuration data. To facilitate an effective response to detected incidents, the event notification is very important. A component that combines the components mentioned above is called a *Management Station*.

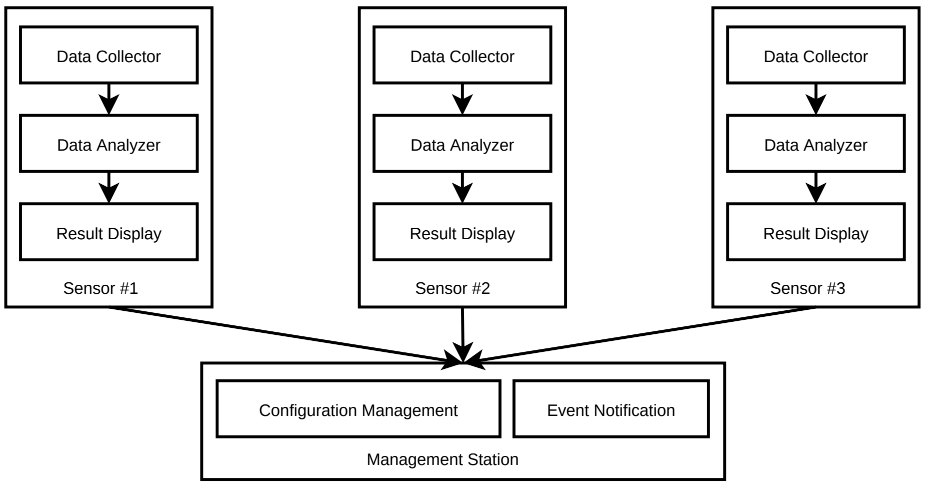


Figure 2.1: Intrusion Detection System Architecture

A good intrusion detection system combines all these components as shown in Figure 2.1 .

2.4 IDS Components

Although there are several different types of intrusion detection systems, do they all contain roughly the same components. A selection of these components will be presented in this section.

2.4.1 Vulnerability Scanner

A vulnerability scanner is a type of computer program specifically designed to search a given target (piece of software, computer, network, etc.) for weaknesses. The scanner systematically engages the target in an attempt to assess where the target is vulnerable. The program can be used either prophylactically (to find holes and plug them before they are exploited) or maliciously (to find holes and exploit them). The most effective protection from attacks is always to eliminate all security holes and vulnerabilities before they get exploited by real attackers.

The information gathered from the scanners can also be used for later event correlation.

Nmap: Nmap[2] is a free port scanning software distributed by Insecure.Org. It is designed to detect open ports on a target computer, determine which services are running on those ports, and infer which operating system the computer is running (this is also known as fingerprinting). It has become one of the de-facto tools in any network administrator's toolbox, and is used for penetration testing and general computer security. Like most tools used in computer security, nmap is useful to both system administrators and attackers. System administrators can use it to test for possible unauthorized servers in the workplace, whereas attackers use it to probe a potential target.

Nessus: Nessus[3] is a comprehensive open-source vulnerability scanning program. It consists of nessusd, the nessus daemon, which does the scanning, and nessus, the client, which presents the results to the user. It begins by running nmap to see which ports are open on the target and then tries various exploits on the open ports. It presents a summary of all tests.

2.4.2 System Integrity Verifier (SIV)

A system integrity verifier is a tool that reacts to modifications of system components. It includes a database containing the important files of the system and their attributes. Attackers often replace programs to install back-doors or to change the behavior of existing commands to hide themselves. The SIV compares the information from the database against the files on the system. If the size, checksum or other attributes of system files do not match to the database, it generates a notification message for the system administrator.

System integrity verifiers are very reliable. However, the configuration can be long-winded and has to be adapted each time the system configuration changes. These tools can be of high value to clean up a compromised system without the need of reinstalling it. Some well-known and free system integrity verifiers are:

AIDE: The *Advanced Intrusion Detection Environment (AIDE)*[4] is a very small utility that creates a database from the regular expression rules that can be found in the configuration files. The usual file attributes can be checked. It has several message digest algorithms to check the integrity of a file like md5, sha1, etc.

Tripwire: Tripwire[5] is like AIDE a policy driven file system integrity checking tool that allows system administrators to verify the integrity of their data.

Integrit: Integrit[6] is a more simple alternative to file integrity verification programs like AIDE or Tripwire. It compares a snapshot of the most essential parts of the computer with the system.

[2]http://www.insecure.org/nmap/
[3]http://www.nessus.org
[4]http://sourceforge.net/projects/aide
[5]http://sourceforge.net/projects/tripwire/
[6]http://sourceforge.net/projects/integrit/

2.4.3 Log File Monitor (LFM)

All systems and services generate files that contain messages about what was happening. This so called log files are often hard to read. Particularly if the system or service is heavily used, the log files can contain thousands of lines of text. This makes it nearly impossible to extract useful information. Log file monitors search for patterns in these log files that portends to suspicious activity. Further they extract important data and generate human readable output, such as reports, charts, diagrams etc.

MIDAS: MIDAS[7] is a network monitoring and intrusion detection server. It uses a distributed client/server model that scales to very large networks.

2.5 Types of Intrusion Detection Systems

The first solutions that were called *Intrusion Detection Systems* could not do more than log simple procedures on systems like logins etc. In the meantime the range of functions has widened more and more. IDS can be distinguished in four categories:

- Intrusion Detection Systems that recognize security holes and attack possibilities in advance on network level.
- Intrusion Detection Systems that recognize security holes and attack possibilities in advance on system level.
- Intrusion Detection Systems that recognize attacks in real-time on network level.
- Intrusion Detection Systems that recognize attacks in real-time on system or application level.

2.5.1 Network Based Intrusion Detection System (NIDS)

A network based intrusion detection system is used to detect suspicious traffic on the network. This is done by scanning all packets that travel through the wires. The NIDS often contains a database of known attack signatures. If a packet matches some of these signatures, it gets classified as suspicious. Any suspicious packets are logged and enriched with data about the severity of that packet. If the severity level gets high enough, a warning will be generated. Suspicious activity may be scans on unused ports.

As most networks are switched today, it is very important to place the NIDS at the proper location to be able to check all packets. Therefore it will commonly be placed in front of the switch that serves the network segment, that has to be monitored.

Network based intrusion detection systems use a *Sniffer* to eavesdrop on the network. Such a sniffer interface has to be integrated in a network segment. To read all data that travels through the wire, the interface has to operate in

[7]http://sourceforge.net/projects/midas-nms/

the *promiscuous mode*. This means that it receives also packets that are not addressed to itself. To protect the interface from attacks, this device can operate in *stealth mode*. In that case there is no network protocol bound to the interface and it cannot be addressed by attackers.

SNORT: Snort[8] is a sniffer which can be used as a network intrusion detection system. It features rule based logging and can perform content searching and matching and detects a variety of other attacks.

Firestorm: Firestorm[9] is an extremely high performance network intrusion detection system (NIDS). At the moment it is just a sensor but plans are to include real support for analysis, reporting, remote console and on-the-fly sensor configuration.

2.5.2 Network Node Based Intrusion Detection System (NNIDS)

In contrast to a network based intrusion detection system, that is attached to network wires, is a network node based intrusion detection system installed on the target system itself. It analyzes the whole network traffic to the system. The NNIDS analyzes also operating system events. Some of the network node based intrusion detection systems are able to block network packets if they detect an attack.

In switched networked environments it is often impossible to place the intrusion detection system before the switch. In this case it is not possible to observe the whole network segment with one intrusion detection system. This is a reason to use a node based intrusion detection system that is able to collect additional information.

There are some negative aspects that have to be considered. It is impossible to run a NNIDS in stealth mode. It needs to be installed on a production system. This means that the NNIDS tees resources from the production system it observes. Most people do not like to touch production systems.

2.5.3 Host Based Intrusion Detection System (HIDS)

A host based intrusion detection system consults several types of system log files and compares them against an internal database of common signatures for known attacks or unusual activity. If, for example, a user logs in from an unusual account, the HIDS detects that and generates a warning.

This type of IDS gets often replaced by NNIDS because HIDS do not watch network traffic and are therefore not able to detect port scans.

2.5.4 Intrusion Prevention System

All the intrusion detection systems above do not really provide security on their own. They are able to detect attacks, but they are not able to react to them.

[8]http://sourceforge.net/projects/snort/
[9]http://sourceforge.net/projects/firestorm-ids/

While they alert the system administrator of suspicious activities, much time elapses until this person is able to notice the alert and to react.

An intrusion prevention system is capable of immediate reaction on incidents by dropping suspicious network packets. This ability is a big advantage but it forces the administrator to take much more care in configuring such a system. Otherwise it is able to drop valid packets and lock out normal users. Currently there are not many possible reactions to incidents. However, if it will be possible to monitor and control applications and processes on the system in the future, a greater variety of reaction mechanisms than simple packet dropping will provide an additional level of security [Str].

2.5.5 Inline Intrusion Detection System

An inline intrusion detection system is some sort of a transparent gateway. This means that the user will not take notice of such a system. It has two network interfaces where packets are routed through. Every packet is analyzed and if it appears to be suspicious, it can be dropped.

This type of IDS can be used to actively protect important network segments where security has higher importance than availability.

2.5.6 Application Intrusion Detection System

Unlike network based IDS, allows an application intrusion detection system to observe activity on the application level. It is able to filter packets for an application like HTTP from the data flow. The application intrusion detection system analyzes the filtered packets and if they contain invalid or dangerous load, they are dropped. This type of IDS is not yet in widespread use because it operates only on a minority of applications and protocols.

2.5.7 Honeypots

The purpose of intrusion detection systems is, as the name implies, to detect intruders. Honeypots are able to do the same, but they also go one step further. A honeypot is a deception device that lures attackers. It is a system that is located inside a computer network and has no productive value. It just waits to become attacked. Because this system has no productive value, all connections to it are most likely scans, probes or attacks.

Attackers who operate on a honeypot are heavily monitored. This allows to retrace their activity and to learn from them. The gained knowledge can be used to harden the production systems.

2.6 Detection Techniques

The last section was mainly on how the different IDS systems collect data. This section now covers different methods of how to analyze the audit data. This step is very important to detect attacks. There are two different methods, one is signature based and the other is anomaly based.

2.6.1 Signature Based Intrusion Detection

Usually an IDS is looking for predefined signatures of bad events in the network traffic. The IDS contains a set of rules that have to be checked against the packets passing the sensors. This type of IDS is easy to implement and has a low false positive rate [Spi03c] [ACM01].

Signature based IDS work like virus detectors, which have a set of patterns from known viruses. Files are compared to the patterns and if there is a match, the checked file is classified as infected. The IDS must contain attack descriptions in order to detect offenses. Such a description can be a simple pattern that matches some network packet or a complex state machine or neural network that maps multiple sensor outputs to abstract attack representations.

Unfortunately, the rule set has to be in an ongoing adaption. Every new attack needs a new rule to become detected. Like most virus detectors, a signature based IDS cannot handle threats that never occurred before. This means that such an IDS is always one step behind the attacker. It also suffers from false alarms when signatures match intrusive and non intrusive sensor outputs at the same time.

2.6.2 Anomaly Based Intrusion Detection

In addition to the signature based analysis, some IDS also perform an anomaly or statistical analysis to detect malicious traffic. In this case, unusual or abnormal traffic is considered to be an attack. Symptoms of abnormal traffic can be an enormous amount of open TCP connections, ICMP packets or packets with invalid source- or destination addresses [NP04].

Such an IDS has to be trained to gain knowledge on how normal network traffic looks like. Therefore, such a system performs a training phase. During this training step the network data flow will be translated into meta data. The system collects network addresses, used ports, flags, packet sizes or timeouts. The last step is to define some thresholds for the values the IDS collected before.

When put into productive use, it compares the actual traffic with patterns it generated. If the used bandwidth changes or the number of open TCP connections exceeds the defined threshold, a warning will be generated. Such systems are useful in detecting users who access services they are not allowed to or in detecting services that should not be active in a network. An anomaly based IDS strengths are the detection of novel attacks.

If the system gets trained while attacks are launched, it treats the attacks as normal traffic. Unfortunately, this type of system has to be adapted to any change in network traffic. A new service on the network can lead to a huge amount of false alarms. An attack that fits into normal network traffic remains undetected.

2.7 IDS from the Present to the Future

This section covers the state-of-the-art in intrusion detection and how the future of these systems might look like.

2.7.1 State-of-the-Art

The earliest components of IDS can be traced back over 20 years. The first programs behaved like system integrity verifiers and belonged to the host based IDS. They reviewed system logs and searched for malicious signatures. In the last few years, network based IDS evolved more and more. In todays computer environments, where systems are connected through networks, it is vital to detect malicious traffic.

A few years ago a perimeter defense like a well configured packet filter was sufficient to protect the network from the Internet. However, the present state of perimeter security apparently is a subject of debate. Brian Laing, chief technology officer at *Blade Software*[10] told in an interview:

> *"The perimeter is becoming so wide and so much access is being allowed through it. In essence, it's rapidly disappearing."*

This statement seems a bit pessimistic. However, the fact that most organizations use intrusion detection systems to improve perimeter security, shows that the demand is rising.

The main problem today is, that the access points to the internal network have changed. Virtual private networks (VPN) are common in most IT-infrastructures. This is one reason why the IDS are transferred from perimeter defense to the internal network. They observe no longer the traffic that comes from the Internet, but all data flows inside the internal network. To achieve a complete control over the internal network, many sensors have to be placed on important nodes. Sensors need maintenance to provide maximum security and this is the main problem of todays infrastructure. Maintenance is expensive and even though many organizations use intrusion detection systems, only a minority of them spends enough resources on maintenance and data evaluation. It often happens that the huge number of alerts from the IDS exceeds the capacity of the security personal for in-depth incident analysis. As consequence many attacks remain undetected.

2.7.2 Shortcomings of Todays IDS

Although intrusion detection systems provide an additional level of security, most of them suffer from shortcomings as described by Lance Spitzner in *Honeypots: Simple, Cost-Effective Detection* [Spi03c].

False Positives

If the NIDS generates alerts where there is no threat, this is called a false positive. On large networks with huge amount of traffic, there are usually many

[10]http://www.blade-software.com/index.html

suspicious packets. If the NIDS is configured too restrictive, there will be many warnings where there is no attack. This makes it laborious for the system administrator to separate false alarms from real attacks.

False Negatives

On the opposite, where the NIDS fails to detect attacks, no alert is generated. This is called a false negative and often happens when new kind of attacks appear. A proper configured anomaly based IDS may detect such attacks. However, any IDS needs a continuous maintenance to provide a valuable service.

Switched Networks

Switched networks spread very fast in the past for performance reasons. This generated new problems for NIDS. Hubs route incoming traffic to all machines. Therefore one sensor in the network is able to analyze the whole network traffic. Switches route incoming traffic directly to the destination machine. This creates a problem for NIDS because the sensor will miss some packets.

Large Scale Network Environments

NIDS in large network environments tend to generate a large amount of data. This makes it very difficult to review the generated log files. As a result, attacks remain undetected because there are not enough resources to analyze the traffic.

High Network Traffic

All packets that pass a sensor have to be analyzed to get a continuous report. On fast gigabit networks, this can lead to problems, because many NIDS are not able to handle more than 200Mbit/s of data. If some packets bypass the sensor unchecked, the NIDS is not able to detect attacks or it generates false positives.

Encrypted Traffic

Today it is more and more common to encrypt network traffic for security reasons. NIDS have limited capabilities in analyzing encrypted packets and are therefore not able to decide if a packet is malicious or not. Especially the IPv6 standard, which includes encryption, renders many of todays NIDS useless.

Complexity

The inset of intrusion detection systems increases the complexity of the network setup. There is the hazard that instead of giving another level of security, new threats through misconfiguration of these systems arise. [LLO+03]

2.7.3 The Future of Intrusion Detection Systems

Switched networks will continue to spread in the future. Thus more sensors are needed in the network to analyze the whole traffic.

As network traffic increases rapidly, NIDS need to handle a larger amount of traffic. Such systems already exist, but they cost much more than todays IDS generally do.

Slow attacks, that send their malicious packets over a large period of time, are also hard to detect. Some projects as Spade[11] try to detect strange packets and group them using sophisticated statistical analysis, which makes them able to detect slow port scans.

As Matthew Tanase writes in his paper about the future of IDS [Tan01], the future lies in data correlation. Future IDS will examine input from different sources. This concept allows to place sensors on various nodes on the network. Sensors on hosts can also filter encrypted traffic. The administration of IDS will become simpler because the clients report the generated data automatically to a centralized monitor. This monitor uses statistical analysis and predictive artificial intelligence to detect abnormal events.

Future IDS will interconnect different security components on the network. Log files from the firewall (packet filter, IDS, operating systems, honeypots, etc.) will be collected. This allows a much deeper analysis, reduces false positives and results in improved attack detection. The IDS' resistance against attacks directed to themselves will grow massively.

The main goal of future IDS development is still to offer an automated IDS system that needs as few maintenance as possible at an affordable price.

[11]http://sourceforge.net/projects/snortspade/

Chapter 3

Introduction to Honeypots

3.1 Introduction

This chapter is about honeypots and starts with an in-depth definition of that term. It gives an overview of the different types of honeypots and their preferred field of application. The value of this kind of deception system is presented and discussed. The section that compares intrusion detection and honeypot systems is particularly useful for security professionals who are interested in improving their IT security infrastructure.

3.2 Definition of Honeypot

A *honey pot* in the kitchen allures flies. This is exactly what a honeypot in a computer network does. It allures attackers with the intention to observe them. Lance Spitzner defines the term honeypot as follows [Spi03d]:

> *"A honeypot is a security resource whose value lies in being probed, attacked or compromised."*

The concept of a honeypot is simple. It is a resource that has no productive value. There is absolutely no reason for anyone to interact with a honeypot. Thus, any attempt to communicate with the system is most likely a probe, scan or attack. Conversely, if the honeypot initiates any outbound connections, the system has probably been compromised. [Spi03h]

It does not matter what this resource really is, but it matters that the resource's value lies in being probed and attacked. A honeypot has to be attacked to gather any information. An intrusion detection system that generates no warnings may be an indicator of normal network activity. However a honeypot that does not get attacked is worthless.

A honeypot is a deception tool. It deceives attackers by pretending to be an important host containing interesting and valuable information or services. The *honey* on such a system consists of different ingredients. This can be an interesting system name, such as *accounting*, a large number of user accounts, a huge amount of data, etc. A honeypot does not run any production service and it is often hidden in the network topology. Therefore every connection to it is an evidence of an attacker or at least an indication of a misconfiguration of the network.

New attacks cannot be averted by a honeypot. However it is possible to detect them. An attack launched at a honeypot can be a frustrating experience for the intruder. An attacker who realizes that there are deception tools on the network might choose another target to attack. The time an attacker wastes on the honeypot can be used by the system administrator to secure the production systems. However the major task of a honeypot is to monitor all activity of the intruder on the system to learn about his course of action and the tools he uses.

The first honeypot, although it was not called that way, was maintained by Bill Cheswick. On 7 January 1991 an intruder, exploiting a sendmail bug, broke into an Internet Gateway at the AT&T Bell Laboratories. The staff led this attacker for several months on a chase in order to trace his location and learn his techniques. There is a paper about the whole attack. It covers the attackers success and disappointments, the bait and traps used to lure and detect him and the chroot *'Jail'* that was built to watch his activities. [Che]

3.3 Types of Honeypots

There are several different types of honeypots and similar devices that vary in their field of application, strength or weakness. The main distinction between different types of honeypots is made on the level of interaction they provide to the attacker. Which type of honeypot to use depends on the aim to achieve. Table 3.1 shows the trade offs between different types of honeypots in four categories [Spi03d]. These categories are:

Installation and Configuration: This category measures the time and effort in installing the honeypot. The more functionality a honeypot provides, the more complex is it to setup and operate.

Deployment and Maintenance: This category is similar to the one above. The more options and services a honeypot provides, the more time and resources are needed to deploy and maintain the system.

Information Gathering: The more interaction a honeypot allows, the more information can be gained.

Level of Risk: A higher level of interaction means more complexity and results in bigger risk. A honeypot that provides a maximum of interaction to an attacker, can be misused by the intruder to attack other systems.

3.3.1 Low-Interaction Honeypot

A low-interaction honeypot provides, as the term describes, limited interaction between the attacker and the honeypot. The simple design makes it simple to install, configure, deploy and maintain. A low-interaction honeypot is no real operating system. It is merely a program that emulates services and logs any

Level of Interaction	**Installation and Configuration**	**Deployment and Maintenance**	**Information Gathering**	**Level of Risk**
Low	Easy	Easy	Limited	Low
Medium	Considerable	Considerable	Variably	Medium
High	Extensive	Extensive	Extensive	High

Table 3.1: Types of honeypots and their trade offs

connections to them.

This type of honeypot has a very low risk level, because low-interaction stands here also for low-risk. In return there is the disadvantage that the information gathered by the honeypot is limited. Most systems are not able to log more than:

- Time and date of the attack.
- Source IP address and source port of the connection.
- Destination IP address and destination port of the connection.

As the IP protocol allows manipulation of the packet header, the only reliable information about the attacker is time and date of his connection attempt.

The main goal of low-interaction honeypots is to detect unauthorized connection attempts. It is useful for organizations that have no operational experience with honeypots at all. It allows them to get to know that technology, so that they can move on to high-interaction honeypots to improve attack detection later.

Examples of low-interaction honeypot software are Specter[1] and Honeyd[2].

Honeyd

Honeyd is a low-interaction or production honeypot for UNIX systems. It is primarily used for detecting suspicious connections. Honeyd is a program that is installed on a regular operating system. It can monitor millions of IP addresses and is therefore first choice to monitor whole subnets or extremely large networks. Niels Provos from the University of Michigan started this open-source project in 2002.

Honeyd is installed on a fully patched UNIX operating system. This eliminates the risk, that the honeypot itself can be exploited. Honeyd does no more than simulate operating systems and simple services on given IP addresses [Spi03f]. These addresses are often called *nonexistent addresses*, because this are addresses that are not used by real systems and therefore can be redirected to the honeypot. A router or an arp daemon[3] is needed to reroute the traffic

[1]http://www.specter.com
[2]http://www.honeyd.org
[3]http://www.citi.umich.edu/u/provos/honeyd/arpd-0.2.tar.gz

from the observed IP addresses to the network interface of the honeypot. Any connection attempt is logged by honeyd.

If honeyd receives a connection attempt to a port on such a nonexistent address it emulates the service specified. Once the emulated service is started, it interacts with the attacker and logs his activity. The emulated services are realized by shell or Perl scripts attached to a specific port. This scripts can log additional information about the attacker to get more in-depth information. After the attacker closes the connection, the emulated service exits and is not running any longer. This is an extremely efficient and resource saving method. Figure 3.1 shows an example of a honeyd deployment.

By collecting the traffic to whole subnets, honeyd is able to observe thousands of addresses at the same time. This enables the honeypot to detect any worm scanning for SQL servers, vulnerable remote procedure calls (RPC) services or an employee scanning for open file shares. It is able to simulate more than 450 different operating systems at the same time. By simulating the operating system on application and IP-stack level, scanning tools like *Nmap*[4] may be tricked.

The main disadvantage of honeyd is the fact, that it only provides simulated systems. Advanced attackers notice immediately that they are not operating on a real operating system. This honeypot is therefore not able to gather in-depth information about the intruder. The main field of application is to check if the internal network is proper configured and that there is no malware that tries to spread.

The modularity of honeyd was the main reason to use it as a starting point for this master thesis. It allows to attach scripts to simulate new services that monitor any activity of the attacker.

3.3.2 Medium-Interaction Honeypot

A medium-interaction honeypot offers more ways to interact to attackers than low-interaction honeypots do. A connection to an (emulated) service on a low-interaction honeypot will be closed by the system after presenting some banner. An emulated service on a medium-interaction honeypot may respond to commands from the attacker with some bogus information.

To limit the risk for low-interaction honeypots, the attacker can only use emulated services. Medium-interaction honeypots do the same, but they often allow the use of jail or chroot. This are UNIX functions, that allow the administrator to create some virtual operating system inside a real one. The attacker then connects to an environment that behaves like a real operating system, but is controlled and heavily monitored from the underlying operating system.

Running a medium-interaction honeypot is time and resource consuming. There are only a few tools that can be used out of the box. Most of them belong

[4]http://www.nmap.org

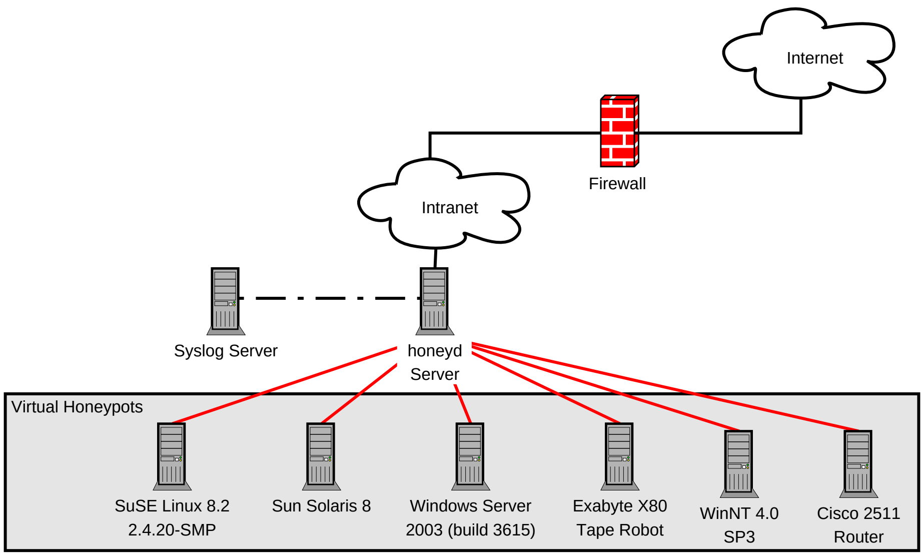


Figure 3.1: General honeyd deployment

to the group of *homemade honeypots*. This are systems created by individuals to suit a specific need.

3.3.3 High-Interaction Honeypot

High-interaction honeypots can do anything, that medium-interaction honeypots can do and much more. They are real systems that capture network traffic. Attackers who break into a high-level honeypot operate on a real system. This has the advantage that much more information can be gained from the attacker. More information about the attacker means better analysis of the attack. The main purpose of high-interaction honeypots is to learn from the attacker and to do some actual research.

It is common practice of attackers to misuse captured systems as file or Internet relay chat (IRC) servers. The administrator then is able to listen to the IRC sessions to learn from the conversations the intruder conducts. With the information gathered production systems can be hardened. [Pro03b]

There is one big disadvantage with high-interaction honeypots. As the attacker acts on a real system, he is able to misuse that system for further attacks on production systems. A firewall that blocks all traffic from the honeypot might solve that problem, but some risk remains.

Examples of high-interaction honeypots are Honeynet[5] and Symantec Decoy Server[6] formerly known as ManTrap.

[5] http://www.honeynet.org/
[6] http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=157

Honeynet

A *Honeynet* belongs to the class of high-interaction honeypots [LLO+03]. It is not a single product, but composed of multiple technologies and products. A honeynet is an architecture [Shu04]. It consists of several machines that look like production systems. The goal is to create a highly controlled environment in which everything is monitored and logged. They offer a maximum of interaction to the attacker. Therefore it is important to protect the real production machines from the honeynet by an access control device like a packet filter.

The concept of a honeynet was first published in August 1999. In June 2000 a nonprofit research group of security professionals formed the *Honeynet Project*[7] [Spi03d]. The main goal of this project is to do research *Blackhats*. The term blackhat describes those people who subvert computer security without authorization. The mission statement of the Honeynet Project is:

> *"... To research the tools, tactics, and motives of the blackhat community and share the lessons learned."*

Honeynets do not emulate systems or services. They are a group of computers with standard operating systems on a network, designed to be attacked. A honeynet can capture everything an attacker does on any of the deception systems. A lot more information can be gained if the information from several systems can be collected and analyzed. This allows to correlate the data and to get more in-depth information.

A honeynet consists of several systems that build a subnet. Devices like routers can also be part of that honeynet and gather useful information. While connections from outside the honeynet are most likely probes, scans or attacks, connections from inside the honeynet indicate a break-in into one of the systems [Pro03c].

To deploy a honeynet is not that easy as it seems. It takes a lot of work, time and money to set up such a honeypot system. The more systems that are involved in the honeynet, the more information can be collected and the more information has to be analyzed. It is hard to keep control of all the different systems and to realize whenever a system is cracked. Attackers can cause real damage if the honeynet is not monitored with caution. Therefore the administrator has to take care to secure the honeynet from other parts of the network. He has to prevent or limit outbound connections to prevent intruders from attacking other systems.

GenI Honeynet GenI (1st generation) honeynets were developed in 1999. The purpose of GenI honeynets was to capture the maximum amount of attacker activity and give the intruders a feeling of a real network. This type of honeynet is not effective in capturing advanced attackers, because it can be easily detected. The problem is the layer 3 firewall in front of the honeypots. This device works as a *Network Address Translation (NAT)* gateway and controls all

[7]http://project.honeynet.org

inbound and outbound connections. All inbound connections are allowed, but the outbound connections are limited to protect other non-honeypot systems. Figure 3.2 shows a GenI honeynet.

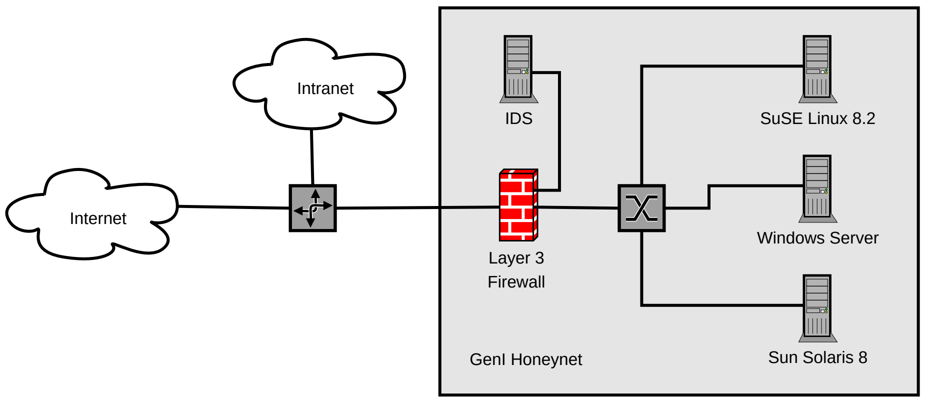


Figure 3.2: GenI Honeynet

GenII Honeynet GenII (2nd generation) honeynets were developed in 2002 by the Honeynet Project [Pro03b]. Figure 3.3 illustrates the architecture of such a honeynet. They did this after identifying the problems and issues in GenI architecture. In the GenII honeynet, the firewall is a layer 2 device that makes it much harder to detect. It works in bridge mode and controls all inbound and outbound traffic as in GenI architecture. The GenII honeynet contains an *Intrusion Prevention System (IPS)* rather than just an intrusion detection system. The IPS has the ability to block and modify malicious traffic. An attacker who runs an exploit against a production system can be blocked by the intrusion prevention system.

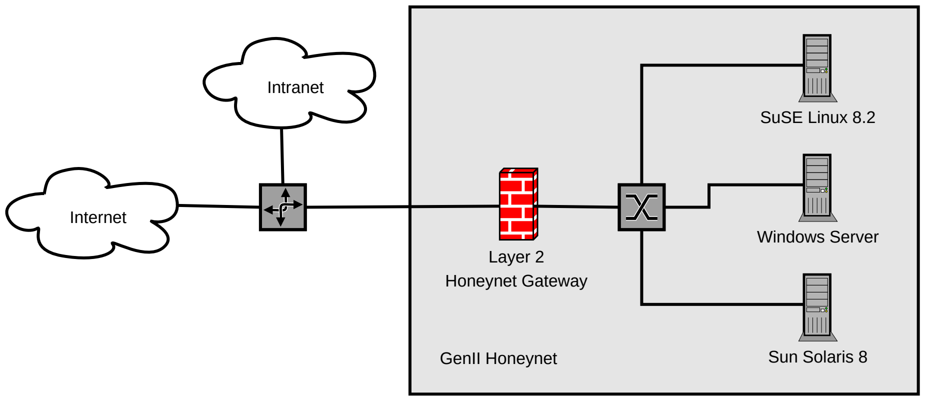


Figure 3.3: GenII Honeynet

Sebek - Data Capturing Tool The true value of honeynets lies in the huge amount and quality of information they capture. The Honeynet Project can be taken as an example of how to detect attacks never seen before by maintaining such honeynets. They developed *Sebek*[8], a data capturing tool. The sebek clients are installed on every honeypot inside the honeynet. To prevent attackers from replacing the data capturing tool to blot out their traces and to circumvent encryption, the data is captured on system level by a hidden kernel module. If a file is copied to the system, the kernel module creates a copy and exports it to a remote system. It also supports keystroke logging and much more. The remote system that collects all data is the sebek server. This tool can also be used to evaluate the information gathered. Figure 3.4 shows a typical sebek deployment.

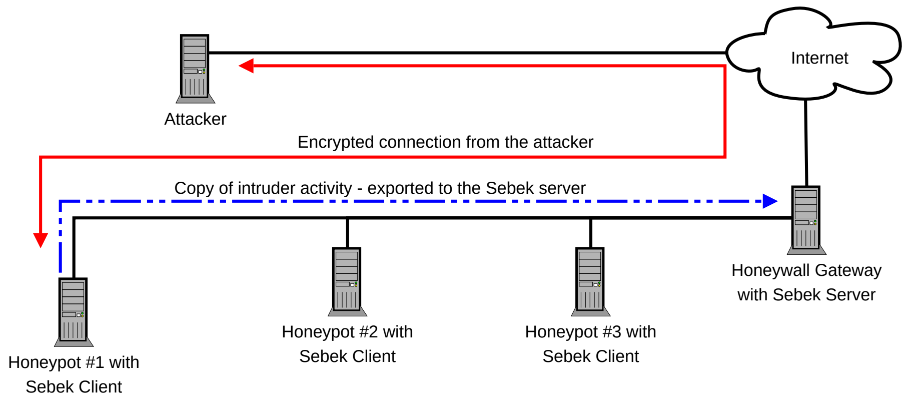

Internet
Attacker
Encrypted connection from the attacker
Copy of intruder activity - exported to the Sebek server
Honeywall Gateway with Sebek Server
Honeypot #1 with Sebek Client
Honeypot #2 with Sebek Client
Honeypot #3 with Sebek Client

Figure 3.4: General Sebek deployment. The intruder activity is duplicated by the Sebek client module on the honeypot and exported to the LAN (hidden to the attacker). The Honeywall Gateway passively captures the exported data.

3.3.4 Honeytoken

A simple form of honeypot is a honeytoken described by Lance Spitzner in *Honeytokens: The Other Honeypot* [Spi03e]. A token in this context means a file, a network port or even some real world artifact like a medical record. Therefore a honeypot has not to stringently be a computer system. It does not matter what it is, but it has to be some item no one should interact with. Honeytokens are used to detect unauthorized access to resources. A hospital, for example, generates a honeytoken by inserting the medical record of a fictive person named *'john doe'* into the database. If this record is accessed, there should be assumed that someone is collecting data he is not allowed.

3.4 Deployment Strategies

Many honeypots do their job well, if they are deployed as single systems inside a network. Such deployments need a minimal administration effort. To build up

[8]http://www.honeynet.org/tools/sebek

an effective honeypot environment, it is necessary to have a computer security policy. This policy defines what the threats are, where the important systems are located, what the main goals for attackers might be, how the systems will be protected and what to do when an incident occurs. [Tec]

This section contains some deployment strategies that can be used together with other security devices like firewalls and IDS to protect the network.

3.4.1 Minefield

If the honeypots are installed among production servers, it is called a *minefield deployment*. The honeypots are distributed across the whole production network. To lure attackers, they often mirror some of the real server data.

Most attacks do not just target a single system. Especially worms and netbots scan whole networks for vulnerable machines. If the honeypots are distributed among production servers, there is a high chance to detect all attacks. They provide information about the attacked service and the changes made to the compromised system.

This deployment strategy takes a big security effort. Compromised systems are often used for further attacks. It requires a great effort to secure every single honeypot from attacking other targets.

3.4.2 Shield

Important servers need special security measures. In a *shield deployment*, each honeypot is paired with a server it is protecting. While normal traffic to the server is not affected, a firewall/router redirects all suspicious traffic to the honeypot.

Every connection to a mail server that does not target the specific SMTP port is suspicious and can be redirected reliably to the honeypot. However the honeypot is not able to protect this server from attacks to the SMTP port, because this kind of traffic will be classified as *normal*.

3.4.3 Honeynet

A honeynet is a type of honeypot as well as a deployment strategy. In a honeynet, the honeypots build an own network segment with fake services and traffic to lure attackers and to learn from them. The goal of this sort of deployment has rather the function of detection than prevention or protection [Pro03c].

3.5 Value of Honeypots

To examine the value of honeypots, they have to be separated into two categories - *production honeypots* and *research honeypots* [Spi03b].

3.5.1 Production Honeypot

A production honeypot is a system that reduces the risk in a networked environment. This is realized by maintaining a system that monitors unauthorized traffic but does not provide any attack point to the intruder. A production honeypot is therefore mostly a low-interaction honeypot.

The value of a production honeypot can be demonstrated by using *Bruce Schneiers* security concept of *Prevention, Detection and Response* [Sch01].

Prevention

The term prevention means to keep attackers out. Honeypot resources do that by keeping the attackers busy. As long as an attacker wastes his time on a honeypot, he cannot attack production systems. Attackers do not like to be under surveillance. They are often scared off, if they are aware of the presence of deception systems.

Automated attacks, such as worms or auto-rooters, begin their attack by random scans of entire networks looking for vulnerable systems. If they find any, these automated tools will capture these systems. Honeypots are able to slow down the scans or even to stop them. They do this by using several TCP tricks. By using a windows size of zero, the attacker can be kept in a holding pattern [Spi03b]. This is a trick that cannot be done on production systems, because this would slow down the network traffic. However in a non-productive environment like a honeypot, this tactic is able to prevent worms from spreading.

Although there are some arguments that justify honeypots as prevention tools, there might be better ways to increase prevention. A well defined and asserted security policy might provide a better prevention at a lower cost.

Detection

The main goal of a production honeypot is to detect unauthorized activity. Whereas an intrusion detection system has to be configured properly to detect suspicious traffic, a production honeypot needs no long winded configuration just to detect unauthorized traffic. It does not produce any false alarms. If such a honeypot receives normal traffic, then there has to be some network misconfiguration. This kind of system is very good at detecting new kinds of attacks, that have never been seen before.

If an attacker evades the honeypot and assaults a productive system, the honeypot is not able to detect anything. Therefore it is important to fill the honeypot with as much *honey* as possible, to attract attackers.

Response

To evaluate an incident it is necessary to take the compromised system offline for a precise investigation. Production systems have to run all day long and do not allow an in-depth examination. They also pollute the gathered information from the attacker with normal activity. This makes it hard to reproduce the incident.

After an incident, the honeypot can be taken offline without caution. It can be stripped down for further investigations. This allows to get maximum information from the data recorded.

One of the areas the *Honeynet Project* intends to research is *Early Warning and Prediction*. Their intent is to give more value to the data, that is gathered by honeynets, by predicting future attacks. [Pro01]

3.5.2 Research Honeypot

The lack of information about attackers, their proceedings and the tools they use is a big problem for the the IT industry. It is difficult to protect systems if the enemy is not known.

If a system gets compromised, the blackhats often leave the tools they use on the machine. This allows security professionals to examine them just like archaeologists do with ancient weapons.

A research honeypot goes one step further. It allows to follow an attack step by step from the beginning to the end. A research honeypot is often a regular computer system. It is a real system and does not just emulate some services. The blackhat should have the possibility to interact with the honeypot just as with every other operating system on a production machine. This kind of honeypot has to pretend to be very interesting to the attacker. To get as much information as possible from the attacker, the honeypot logs all events from the system, the installed applications or the keystrokes to a remote system. This prevents the blackhat from covering his tracks.

Research honeypots are able to capture automated threats like worms or auto-rooters. They capture these tools for examination of their dangerous payload. Furthermore, it is possible to capture the communication between blackhats. They often start Internet relay chat (IRC) sessions from captured systems to talk to other members of the blackhat community.

Research honeypots do not reduce the risk to an organization. Quite in contrary, they might increase the risk if not properly operated and secured [BP02]. However they are an essential tool to learn from the blackhats and to improve prevention, detection and response to incidents. The *Honeynet Project* is one example of a community using research honeypots to improve the knowledge of computer incidents.

3.5.3 Advantages

A honeypot does not provide additional security on its own. It even attracts blackhats to itself or the network it resides in. Nevertheless does this technology provide some advantages. Four of them will be presented according to the book *Honeypots: Tracking Hackers* [Spi03d].

Data Value

Every computer infrastructure contains a lot of data sources that collect huge amount of data on what is going on. Every service and every machine has its own log files to report any type of event. Most organizations are not able to derive information from that data, because they are overwhelmed by an incredible number of reported events. Honeypots, in contrary, collect very little data. They start logging if a scan or attack is in progress and they stop it if the attacker leaves the system. There is no need to separate normal and dangerous traffic, because the honeypot receives no normal or productive traffic. Every connection to this system is an evidence of an attack.

Resources

As honeypots monitor little activity, they do not have the problem of resource exhaustion. Normal traffic that consists of huge amount of network packets bypasses the honeypot. The honeypot system itself rarely has to handle more than a few attackers at a time. Moreover, it might be an additional advantage of a honeypot to slow down the attacks by minimal resources.

Simplicity

A honeypot does not need complex configurations. Just to drop a system somewhere on the network might be enough to capture some suspicious activity. The simplicity of the honeypot concept is the reason for its reliability.

Return on Investment

Investments into security technologies can become victims of their own success. Firewalls or encryption might reduce the risk of an attack. An organization that is well protected by such expensive technologies for years might perceive that there is no threat anymore.
Honeypots demonstrate their own value by capturing unauthorized activity. They can be used to justify other investments into security technologies.

3.5.4 Honeypot vs. IDS

Honeypots have the ability to overcome some of the disadvantages of intrusion detection systems as written by Lance Spitzner in *Honeypots: Definitions and Value of Honeypots* [Spi03b].

Capacity

High traffic networks tend to overstrain the logging capacity of intrusion detection systems. IDS for gigabit networks increase expenses in time, money and hardware. In contrast to IDS, honeypots only need minimal resources, because normal traffic bypasses these systems.

Information content

Intrusion detection systems in large networks suffer from the high amount of traffic. They have to inspect thousands of connections a day to find a few suspicious packets. There is a high chance to miss them or to generate a big number of false alarms. Honeypots in contrary just have to handle traffic directed to themselves. All traffic to the honeypot is suspicious. This fact lowers the chance of false alarms.

New kind of attacks

It is difficult for intrusion detection systems to detect new kind of attacks. They often need descriptions to identify attacks. All traffic to a honeypot seems malicious or dangerous and there is a high chance to detect attacks that never occurred before.

Insider detection

A big challenge for intrusion detection systems is to detect attacks from insiders. Any connection from one machine to another inside the network is most likely caused by regular usage. However connections from machines inside the network to the honeypot are very suspicious and might be an evidence of a regular user who exceeds his competences.

Encrypted traffic

Intrusion detection systems that eavesdrop on the network wire are not able to monitor encrypted traffic. As the honeypot is the endpoint of the encrypted connection, encrypted traffic to honeypot systems is easy to analyze.

3.5.5 Disadvantages

There is no ultimate security solution in reality. Therefore even honeypots suffer from some disadvantages [Spi03d].

Narrow Field of View

A disadvantage of honeypots is the fact that they have only a limited view. While an IDS analyzes the data of a whole network, the honeypot only captures connections to itself. If attackers evade the honeypot and attack directly production systems, they will not be detected.

Fingerprinting

Fingerprinting means that attackers are able to identify a honeypot on the basis of some characteristics. At the moment the attacker is aware of the presence of a honeypot, he might use it for an attack or just leave it and probably turn toward production systems.

Risk

Honeypots might introduce an additional risk to the existing infrastructure. A successfully identified and attacked honeypot can be misused for further malicious activity. By maintaining and observing the honeypot, the risk can be minimized, but never totally eliminated.

3.6 Conclusion

The term honeypot has no exact definition. It describes a wide range of deception tools, that are used to lure attackers. Opposite to a packet filter, a honeypot does not provide security on its own. It has to be integrated into a security concept to harden the IT infrastructure.

Especially low-interaction honeypots might be interesting for organizations that have large and important networks to maintain. This type of honeypot is not only useful in capturing attackers, but also in monitoring the network. It allows to detect misconfigurations, intruders or crooked users.

High-interaction honeypots are not yet widespread. A reason might be the lack of know-how in most organizations to setup and maintain such complex systems. It seems difficult for IT coordinators to justify additional security investments. In contrast to packet filters or anti-virus software, which have propagated into almost every organizations network, high-interaction honeypots do still remain in universities or other research environments.

Honeypot technology is still in its infancy. Most of the tools that exist at the moment are complicated to deploy and to maintain. There exist not much automatism that could ease the honeypot usage. However to spread the use of honeypot systems, their handling has to be improved.

Although honeypot technology is very young, there already exist tools to defeat honeypots. The blackhat community is active in building lists of signatures that indicate the presence of a honeypot. Honeypot Hunter[9] is a tool, designed for spammers, that checks if open proxies are honeypots. It exist already instructions on how to detect the Sebek Win32 client[10] that is used on honeypots to capture data from the intruder. Furthermore, all sorts of virtual honeypots, that are built with tools like *VMWare*[11], can easily be detected.

An often described advantage of high-interaction honeypots is, that blackhats conduct conversations over Internet relay chat (IRC) servers, they installed on captured systems. Their conversation over this server can be eavesdropped to learn from them. If the attackers have conversations in foreign languages, this advantage disappears. IRC conversations are often held in Russian, Romanian or other languages and it is not likely that the system administrator of the

[9]http://www.send-safe.com/honeypot-hunter.php
[10]http://www.security.org.sg/vuln/sebek215.html
[11]http://www.vmware.com

honeypot is able to understand all of them.

Unfortunately, all the endeavor is useless if the organizations do not increase their interest in IT security. To convince the management of investments, it is necessary to present some numbers. Therefore the next chapter covers the costs and the benefits of honeypots.

Chapter 4

Honeypot Cost-Benefit Analysis

4.1 Introduction

Honeypots are often called *simple and cost-effective* solutions [Spi03c]. In contrast to intrusion detection systems, honeypots have merely to handle a few connections a day. The low load leads to very low hardware expenses. In fact, most honeypot solutions run on old, discarded hardware. The honeypot software itself is not that expensive. There are a lot of excellent open-source tools that can be used for free. As opposed to intrusion detection systems does a honeypot not need a long winded configuration. If this type of security device is so auspicious, why are honeypots used by such a small number of organizations to protect their networks?

This chapter takes a closer look at the costs and the benefits of honeypots. It starts with a discussion about the evaluation of security investments and takes a closer look at the return on security investments. The benefits of honeypots are outlined by examples of possible uses. The main part of this chapter analyzes the costs of honeypot systems. Based on the results from that cost-benefit analysis, conclusions about the future are drawn. This chapter addresses decision-makers who are intending to use honeypots. It gives an overview of possible uses and helps to estimate the costs and benefits.

4.2 Return on Security Investment

The growing number of incidents and their rapid development in intelligence are the main reason for any organization to increase their IT security budget. IT security is an ongoing process that must never stop to prevent from the serious consequences of successful attacks.

A major problem of the security department is to justify their expenditures to the management. The investments into security can be represented in money that was paid for new security devices, services and employees. However, the advantages of these expenditures are not so easy to determine. Investments into security solutions seem to be successfully, if the organization does not detect any successful attacks. The argument *"we detected no successful attacks"* does not mean that there were no successful attacks. Virtually no one is designing, performing or publishing effectiveness trials on security widgets. If necessary widgets were pharmaceuticals, not one of them would come close to achieve *Federal Drug Administration (FDA)* approval. [Bla]

For years security professionals treated security as an absolute requirement. This means that security had to be implemented as good as possible. In some cases it makes sense to pay for *'perfect'* security. Particularly if it prevents the organization from incidents that throw them out of business. Many dangers menace todays IT infrastructure that will not endanger an organization as a whole, but may nevertheless cause a monetary loss. Therefore cost effectiveness is a big issue for todays security professionals. To accommodate this development, security providers defined the *Return on Security Investment (ROSI).*

4.2.1 Definition of Return on Security Investment

A simple definition for return on security investment might be:

> RETURN ON SECURITY INVESTMENT = [(CHANGE IN REVENUE) + (COST SAVINGS)] / (INVESTMENT) [Bla]

It is important that all variables are well defined to guarantee the seriousness of such a calculation. Otherwise to present the return on security investment would sound like:

> *Investments into IT security = $150'000.- versus "we believe we have not been attacked".* [Bei03]

This statement exactly depicts the problem of an IT manager who has to justify the security budget against the management. This situation leads to the fact, that it is necessary to have well defined methods to determine the ROSI. A well defined method is based on exactly defined variables:

Investment: The investment stands for the purchase cost of the security device. This is the money spend to buy and install an IDS or a honeypot. Probably other costs have to be considered too. There might be administrative expenses, personnel training and impacts on the existing infrastructure.

Change in revenue: The change in revenue describes the amount of additional earnings due to the new security device. There may be new things an organization may do, because of the additional security widget. A firewall, for example, allows an organization to connect their computers to the Internet and provide additional services to their customers. For a security provider like an anti-virus software developer, a research honeypot may provide information about new malware that spreads on the Internet. It is important to pay attention to not add risks that were already taken. If the organization already provided services to the Internet, then the installed firewall will not generate any change in revenue, although it reduces some risks.

Cost savings: The security department will only buy a new security device if it prevents the organization from a quantifiable threat. This is a big problem for IT related investments. The installation of fire extinguishers inside a building leads to a decreased fire insurance fee. The fire risk is quantifiable today, because over a period of many years losses from fire have been systematically documented and analyzed. Unfortunately, this is not the case with IT security risks. There are currently no detailed information or statistics about the possible loss from IT related incidents. This might perhaps change in the future.

4.2.2 Decisions based on ROSI

Return on security investment has been discussed for years by academics and security professionals. The result was a lot of rhetoric like:

> *"A company needs to assess the investment versus the chance of something occurring multiplied by the severity of the problem."* [Kar]

This conclusions are obvious, but the term *'assess'* delves into soft claims that cannot be quantified. It is sure that security incidents may cause a *'loss of brand'* or a *'decrease in market value'*. However it is almost impossible to establish causation between financial impact and the event.

Return on security should enable decision making when intangibles are a factor. Cost-Benefit analyzes are very difficult if not impossible to make, because risks are often not all valued in the same currency.

A long time ago society decided that preventing fires was more important than to keep information about the causes and frequencies of such incidents secret. In IT security there exists just the opposite situation. Any company that detects any IT security incident tries to keep it secret, because it fears to loose customers if it is known that it was successfully attacked. This makes it impossible to gather detailed information or statistics to quantify the risks. This attitude gets obvious if one is looking for information about security incidents. There are only a few detailed incident analyzes. All of them come from universities or non-profit organizations that do not have to fear a loss in market value.

Return on security investment (ROSI) will not become a serious decision calculation tool, until all organizations are willing to share all information about their security incidents.

4.2.3 Alternatives to ROSI

Return on security investment is in its infancy. In the future many question marks that are included in such calculations may disappear, but right now it is necessary to have other, perhaps more qualitative methods to justify the acquisition of new security devices.

Most organizations have no clue how much money they have lost because of IT related incidents. If they do not know how much money such an incident costs, they are not able to decide how to allocate their resources. This organizations will remain vulnerable to the growing number of attacks and will spend more and more money on incident recovery.

If the benefits of such investments cannot be designated straight, it might be wiser to take a look at the costs and the consequences that might arise from incidents.

4.2.4 Security Incident Costs

Most organizations with an IT infrastructure have suffered from the consequences of a security incident in the past. They lost some research data or user information and had to reinstall some of their systems. Many of them are not able to express their loss in money as a consequence of this incident. There are several reasons for this. The main problem seems to be that organizations simply do not have enough personnel to maintain their logs of the incident. Often changes in personnel do not meet the requirement that security is an ongoing process. The employees who analyze the incident need to have the qualifications to detect a break-in. This exceeds the skills of most system administrators.

Valid information is the basis of every calculation. To get the necessary information to calculate the loss caused by IT related incidents, David A. Dittrich comes up with the following questions [Dit02]:

- Who worked on responding to and investigating the incident?
- How many hours did each of them spend?
- How many people were prevented from working because of the incident?
- How much productive time did each of them lose?
- How much do you pay each of those people to work for you?
- How much overhead do you pay (insurance, sick leave, etc.) for your employees?

These questions may help to analyze the costs of an incident. However, they are not so easy to answer as it seems in the first place. It is often not possible to determine how many people were prevented from working. Especially at universities, where most public accessible incident analyzes come from, it is hard to determine how many users suffered from the incident. A student does not get paid and therefore it might not result in any direct costs to the university, if the students loose productive time. However, in a more accurate contemplation, it is a fact that the students choose a university that provides them a good and functioning infrastructure. Therefore it is necessary to assign costs to every user, whether he gets paid or not.

The first step was to find out if some incident cost analysis models do exist. There are only a few of them that are based on serious procedures. The most important ones are presented in the next section.

I-CAMP I

The first *Incident Cost Analysis and Modeling Project (I-CAMP I)* was founded by the Chief Information Officer of the Committee for Institutional Cooperation (CIC) in 1997. Representatives of the Big Ten Universities[1] examined systematically the real costs of security incidents at their universities. The goal of the study was to develop a cost-analysis model for IT related incidents. They defined the term incident as:

> *Any event that takes place through, on, or constituting information technology resources requiring a staff member or administrator to investigate and/or take action to reestablish, maintain, or protect the resources, services, or data of the community or of its individual members.* [RDFZ97]

In this study 30 IT-related incidents were examined and the researchers found out that [RDFZ97]:

- 210 employees were involved in incident investigation/resolution.
- 9‘078 employee hours were devoted to incident investigation/resolution.
- 270‘805 computer/network users were affected by the incidents.
- The calculated costs for these 30 incidents totaled $1‘015‘810.

I-CAMP II

The goal of the second *Incident Cost Analysis and Modeling Project (I-CAMP II)* was to redefine the cost-analysis to allow managers to evaluate organizational risks and costs. The first part was about the incident cost analysis where the second part treats the frequency of such incidents. The main reason for the correlation of cost and frequency was, that many sorts of incidents do not entail great costs, but due to their absolute frequency, a vast amount of expenditures might add up. [RCH99]

As a result, the study presents some specific recommendations for future research and best practices on how to handle IT related incidents. It contains a questionnaire and calculation templates (like in Figure 4.1) to allow an easy but reliable cost estimation. The ICAMP-II study used the following type of analysis.

- Persons affected by the incident were identified, and the amount of time spent/lost due to the incident was reported.
- Staff/Faculty/Student employee time cost was calculated by dividing the individual’s wage rate by 52 weeks and 40 hours per week to come up with an hourly rate. The wage rate is then multiplied by the hours logged, and varied by +/- 15%.

[1]University of Illinois, Indiana University, University of Iowa, University of Michigan, Michigan State University, University of Minnesota, The Ohio State University, Pennsylvania State University, Purdue University, University of Wisconsin

- A benefit rate of 28% is added (an average of the institutions in the study) to come up with a dollar loss per individual.
- The total of all individuals' time, plus incidental expenses (e.g., hardware stolen/damaged, phone calls to other sites, etc.), is then calculated using a simple spread-sheet approach.

The I-CAMP models have been successfully used in several intrusion cases. They prove that fair and accurate damage estimates can be produced, if the parties involved are willing to do their work seriously [RCH00].

Unquantifiable Issues

Both I-CAMP studies tried to quantify the monetary loss caused by the incident. Unfortunately, there are some issues that remain unquantifiable:

Reputation: Users who are prevented from doing their work because of a system failure, blame the organization that maintains the infrastructure. It is most likely that they will switch to more reliable service providers for their next projects.

Privacy Invasion: It is hard to predict the costs, which originate from stolen passwords and other sensitive data.

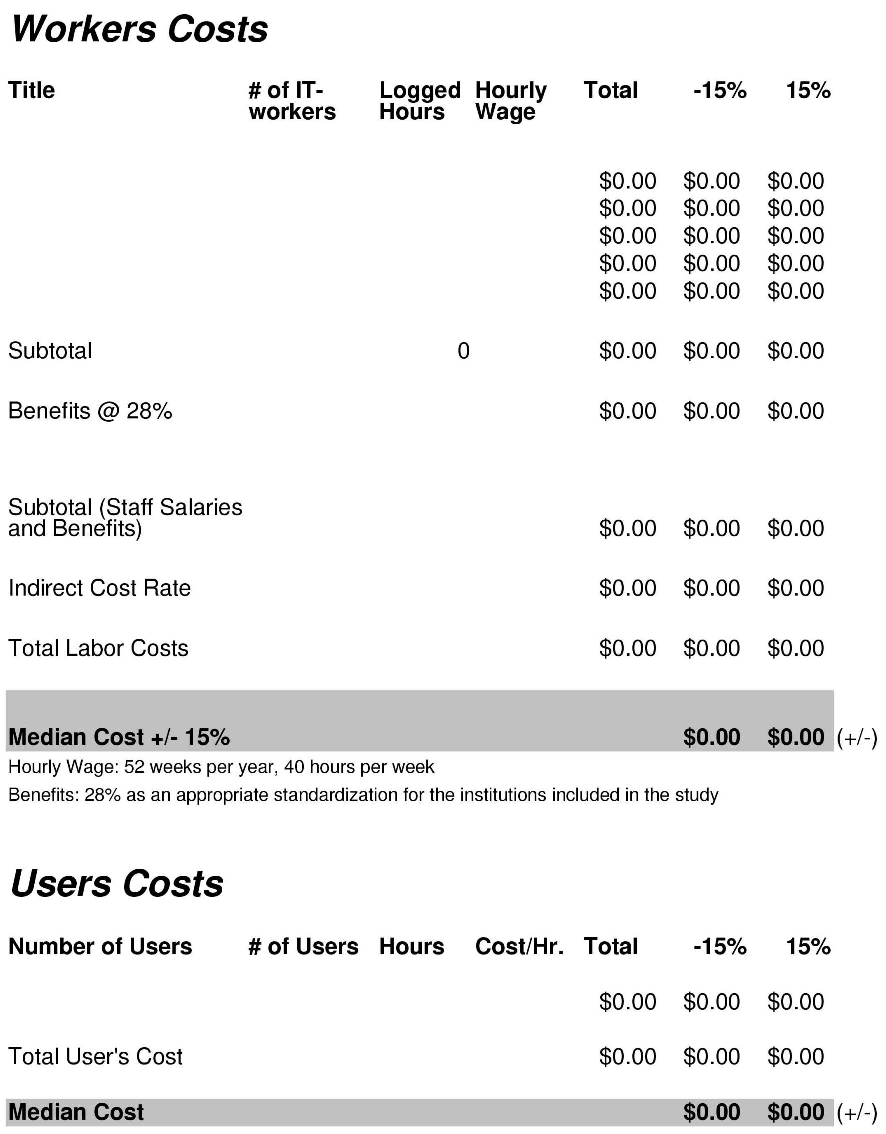
True loss to employees: To calculate the loss to employees, only their wages are included into the calculation. The costs from stress or frustration are ignored.

Performance Issue: Due some mailbombs or other attacks, the resources are prevented from doing the regular work. As a consequence, an incident might result in a delay in mail delivery, etc.

Network Reliability: Employees begin to save their data on local hard disk due to the network failure during an incident. This increases the chance that they might loose their data in future, because local disks will not be backed up.

4.3 Honeypot Benefits

The term *Honeypot* is a buzzword that is known to most security professionals. Many authors wrote papers about the installation, the deployment and the application of honeypots. However, to convince the management to allocate money for such a security device it is necessary to take a closer look at the information that can be gained from a honeypot and the benefits that arise from such deception systems. It is not reasonable to give a general overview of the information that can be gained with a honeypot. The reason is, that there is no general honeypot. *Honeypot* is a term for a wide range of deception systems. There are many different honeypot setups that collect different information. Therefore the following sections cover some example setups and the information that can be gained from them.



Workers Costs

Title	# of IT-workers	Logged Hours	Hourly Wage	Total	-15%	15%	
				$0.00	$0.00	$0.00	
				$0.00	$0.00	$0.00	
				$0.00	$0.00	$0.00	
				$0.00	$0.00	$0.00	
				$0.00	$0.00	$0.00	
Subtotal		0		$0.00	$0.00	$0.00	
Benefits @ 28%				$0.00	$0.00	$0.00	
Subtotal (Staff Salaries and Benefits)				$0.00	$0.00	$0.00	
Indirect Cost Rate				$0.00	$0.00	$0.00	
Total Labor Costs				$0.00	$0.00	$0.00	
Median Cost +/- 15%					**$0.00**	**$0.00**	(+/-)

Hourly Wage: 52 weeks per year, 40 hours per week

Benefits: 28% as an appropriate standardization for the institutions included in the study

Users Costs

Number of Users	# of Users	Hours	Cost/Hr.	Total	-15%	15%	
				$0.00	$0.00	$0.00	
Total User's Cost				$0.00	$0.00	$0.00	
Median Cost					**$0.00**	**$0.00**	(+/-)

Figure 4.1: ICAMP Cost Calculation Template

4.3.1 Network Observation

Particularly large networks tend to suffer from misconfigurations or crooked users. To guarantee the proper functionality of the network, it is essential to have an entrapment device to detect malicious behavior.

A simple low-interaction honeypot setup, as shown in Figure **??** , provides the administrator with information that helps to maintain the network. A low-interaction honeypot does not collect in-depth information. It merely collects the connection attempts to the honeypot. A log file from such a honeypot might look like in Table 4.1. This log file is designed to be processed by scripts or tools. The gained information consist of [Spi03f]:

Date & Time: The date and the time of the connection attempt are very useful to correlate the single log entries.

Protocol: The next entry denotes the used protocol. This can be TCP, UDP or ICMP.

Source: The IP address and the port from where the connection was established indicates the next two columns. This information is very useful to detect the source of the suspicious connection. However, it has to be considered, that the attacker might have faked this information.

Destination: The fifth column shows the destination address and port. This should normally be an address of one of the virtual honeypots.

Transferred Bytes: The far right column shows the number of total bytes transferred.

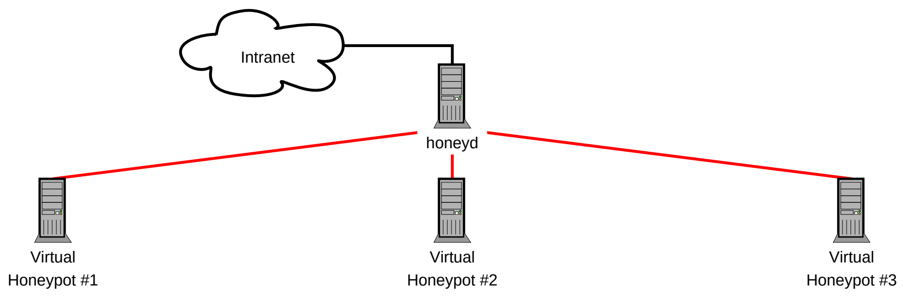


Figure 4.2: Simple low-interaction honeypot setup with honeyd to monitor suspicious network activity

The honeyd logs are very effective in telling what is going on. It shows what activity is occurring inside the network [Spi03g]. The honeypot theoretically should detect no connections at all. However, in reality there is always traffic to the deception system. Often there are some services that cause broadcast traffic. There might also be some users who scan the network for open and unprotected file shares. These users are most likely to probe also the honeypot. If there is any server or router misconfiguration, this type of honeypot will detect some of the

```
2004-09-13-22:55:58.0187 tcp(6) - 192.168.1.1 59480 192.168.1.10 80: 40 A
2004-09-13-22:55:59.0030 tcp(6) - 192.168.1.1 59457 192.168.1.10 180: 44 S
2004-09-13-22:55:59.0030 tcp(6) - 192.168.1.1 59457 192.168.1.10 273: 44 S
2004-09-13-22:55:59.0030 tcp(6) - 192.168.1.1 59457 192.168.1.10 443: 44 S
2004-09-13-22:55:59.0031 tcp(6) - 192.168.1.1 59457 192.168.1.10 385: 44 S
2004-09-13-22:55:59.0031 tcp(6) - 192.168.1.1 59457 192.168.1.10 17: 44 S
2004-09-13-22:55:59.0031 tcp(6) - 192.168.1.1 59457 192.168.1.10 5232: 44 S
2004-09-13-22:55:59.0031 tcp(6) - 192.168.1.1 59457 192.168.1.10 818: 44 S
```

Table 4.1: Example of a honeyd log file

misrouted traffic. This helps to find the error in the network. Systems that are infected with malware like worms or viruses often generate some random traffic. Sooner or later they will try to open connections to the honeypot. Connections to ports where services are located that had some security flaws in the past are very suspicious. The log file helps to detect the system where the suspicious packets come from. To identify the system that behaves in a suspicious way is the first step in detecting any malware or attacker.

Network Observation Experiment

A sample honeyd setup at the Functional Genomics Center Zurich (FGCZ) gave an overview of the threats that are located inside a university network.

The network of the University of Zurich has a packet filter that protects it against the Internet. Unfortunately, there are no firewalls to separate the constituent institutes inside the network.

The goal of this experiment was to reveal the dangers that threat the FGCZ network. The used honeypot was a honeyd, that was listening to one address inside the FGCZ subnet. It simulated a windows 2000 server.

The honeypot that was up and running for a month detected about 400 connections. Most of them were connections to port 445 and 135, where the windows file sharing and RPC services listen. This services have suffered from many security flaws in the past. It must be obvious that a system that probes the honeyd more than twenty times for port 135 is infected with the W32.Blaster worm[2]. If the same system opens many connections to various ports on the honeypot over a short period of time, this indicates a port scan that is looking for open ports on the honeypot.

The result of that experiment shows that, although there came no suspicious traffic from within the FGCZ network, a lot of suspicious packets travel inside the university network. This honeypot setup was able to identify virus infected machines.

This type of honeypot does not provide in-depth information about the attacker. The honeypot log file contains no information about the tools the at-

[2]http://www.cert.org/advisories/CA-2003-20.html

tacker used or about his intentions. However, it provides some basic information and allows to make conclusions about the state of the network. A honeyd that just listens on the ports and logs all connection attempts is easy to install and maintain and provides valuable information.

4.3.2 Fighting Spammers

Spam is probably the most annoying cyber-pollution that targets all people who use email service. The word spam stands for *Unsolicited Bulk E-mail (UBE)* and *Unsolicited Commercial E-mail (UCE)*. This includes all sort of emails that are sent by professional spammers that are paid to spread the messages to random recipients.

Spam is not classified as an attack, but the huge amount of spam that fills up the email accounts, causes a great monetary loss. Organizations have to take measures to prevent spam from spreading and people waste their time in handling these unrequested messages.

Honeypots are useful in fighting spammers as described by Lance Spitzner and Laurent Oudot in their papers *Fighting Spammers With Honeypots* [Spi03a] [Oud03b]. Spammers take big effort to do their job anonymously. They use third party hardware to send the spam. Needless to say that the affected organizations do not want to pay for the bandwidth and other resources, used by spammers. Honeypots are an instrument for these organizations to defeat the spammers.

Fight Harvesting

Spammers begin their work by harvesting valid email addresses. They do this for example by using bots, that go through web pages and collect all email addresses they can find. Honeypots can help in providing them fake email addresses that reveal the identity of the spammer. If a spambot processes a fake web page, the script that generates the page will also generate some fake email addresses. These generated addresses contain information about the spammer, like the source IP address of the spambot, the time of the harvest, etc. Such an address would look like *102.168.1.13_11-11-2004_spam@gugus.com*. If the organization (gugus.com) receives spam, the receiver's address contains the information that the spammer used a spambot that came from the IP 102.168.1.13 and collected the address on the 11.11.2004. Unfortunately, this tactic only works with real dumb spambots.

Protect Open Proxies

Spammers either directly connect to a remote mail server or bounce through open proxies. An open proxy is a service opened to the world, that handles almost any kind of request. It allows everybody to remain anonymous while crawling the net. Spammers who send their emails via several proxies stay anonymous. Figure 4.3 shows the path of the email and the IP addresses that remain in the log files of the used proxies.

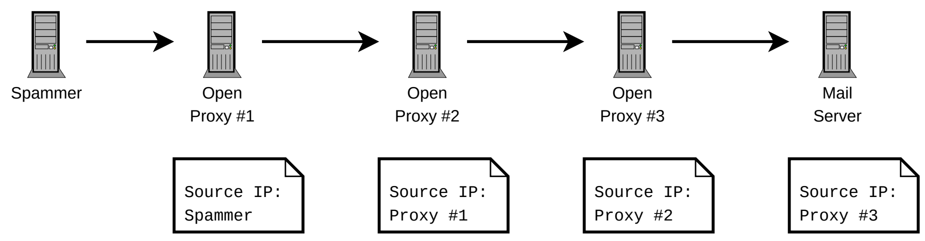


Figure 4.3: Spammers hide their identity by using open proxies

To disclose the identity of the spammers, it is necessary to set up a honeypot that runs a fake open proxy service. Some known proxy services and the ports where they listen are:

- socks proxy server on port 1080
- squid proxy server on port 3128
- web caching service on port 8080

A simple setup for a honeypot proxy would be to install honeyd. It has to be configured to accept connections on the ports listed above. Honeyd then logs all source IP addresses that open a connection to the proxy server.

Even though honeyd is a good choice for such a task, there are more specialized tools. *Bubblegum Proxypot*[3] is a small tool to fool spammers by simulating an open proxy. There are scripts that do some statistical analysis of the proxypot log files to find the identity of the spammer. Skilled spammers do not send thousands of emails without checking that the proxy is working properly. The proxypot therefore provides mechanisms to also lure skilled spammers. It can be configured to forward the first emails from a specific address, that are most likely test messages, and block the following.

This kind of honeypot collects IP addresses of spammers or at least addresses of other open proxies. The emails that arrive at the honeypot can also be saved. However, such deception systems do not only succeed in detecting spammers, they also slow down or block their activity by simulating but avoiding the sending of real spam.

Protect Open Relays

Open relays are *Mail Transfer Agents (MTA)* that accept third-party relays of email messages even though they are not destined for their own domain. This open relays are mail transfer agents that are poorly configured. Spammers use this servers to route huge amounts of unsolicited emails. Organizations that relay spam, unwillingly or not, are most likely to be blacklisted on international lists (Real-time Blackhole List (RBL), etc.). An Internet Service Provider (ISP) that gets blacklisted because of a misconfiguration in his MTA will loose clients

[3]http://www.proxypot.org

and money.

Honeypots can help to fight spammers using open relays by running modified mail transfer agents. This mail transfer agents queue every incoming mail without sending one out. The goal of such honeypots is to waste resources of spammers and to collect information about them. This information can be used to report spam abuses to official spam classifiers that update their blacklist.

4.3.3 Detect Malware

Any software that is developed for the purpose of doing harm to a computer system is called malware. This includes viruses, worms, Trojan horses, backdoors, spyware and more. New types of such pieces of software are developed every day. Some of them spread fast on the Internet and can cause much damage. This section describes a honeypot architecture that catches such malware and presents a honeyd setup to catch Internet worms.

Catching Malware

New and modified versions of malware appear every day on the Internet. For security professionals, maintainers of intrusion detection system signature lists or anti-virus software developers it is important to catch such malware. This allows them to build new signatures that is able to detect the malicious software and to harden their systems.

Figure 4.4 illustrates a honeypot architecture to detect new kinds of malware. If new malware reaches the honeypot, it installs itself on that system and launches a new process. This allows it to spread or do any harm to the system. To detect the infection, a *System Integrity Verifier* is running on the honeypot. As soon as it detects any changes to system files or process lists, it raises an alarm. The *Management Console* logs, that a new attack was started. The *Intrusion Detection System* scans all traffic to and from the honeypot. As soon as the honeypot establishes connections to the Internet, it logs that event to the management console. This is most likely a sign for malware that tries to spread itself. To prevent production systems, a packet filter blocks all outgoing traffic. The management correlates all logged events to find out, which binaries are responsible for the outgoing traffic. It is wise to install an *Anti-virus Software* on the honeypot, to catch only new sorts of malware.

This kind of honeypot is of great value for anti-virus software developers. This technique enables them to detect and examine new malware. As sooner as they detect new malware, as faster are they able to update the signatures of their virus scanners and provide an excellent and secure service to the customer. To install their own anti-virus software on the honeypot prevents already known malware from infecting the honeypot.

Catching Worms

Internet worms like MS-Blaster, Slammer, Code Red or Nimda have wreaked havoc in the past. This pieces of software do the following actions to spread

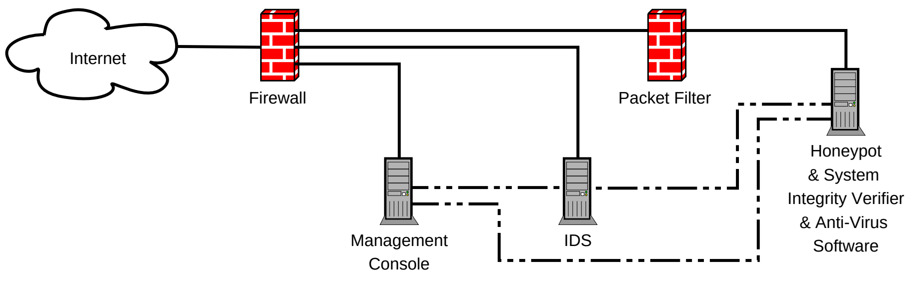


Figure 4.4: Honeypot architecture to detect malware

themselves:

Infection: Infect a target by exploiting a vulnerability.

Payload: Launch malicious actions on the local infected target.

Propagation: Use the infected target as means for external propagation.

Laurent Oudot describes in his paper *Fighting Internet Worms with honeypots* [Oud03a] how to catch the MS-Blaster worm. He deploys a honeyd honeypot that is listening on port 135. The honeyd configuration file can be seen below:

```
create default set default personality "Windows XP Pro"
add default tcp port 135 open
add default tcp port 4444 "/bin/sh scripts/WormCatcher.sh $ipsrc $ipdst"
set default tcp action block
set default udp action block
```

The MS-Blaster worm exploits a bug in the Microsoft RPC DOM service to access the target machine. From there it downloads the msblaster.exe and installs it on the target machine. It uses the cmd.exe to create a hidden remote shell process that is listening on TCP port 4444, allowing an attacker to issue remote commands on the machine. If the attacking system opens a connection to port 4444 of the target machine, the msblaster.exe gets caught by the WormCatcher.sh script:

```
!#/bin/sh
# Creation of a directory for every contaminated host
# attacking the honeypot, in order to archive different binaries
mkdir /tmp/$1-$2
# Download of the worm through TFTP in this directory
# (specific behaviour for MSBlast)
cd /tmp/$1-$2/
tftp $1 <<EOF
get msblast.exe
quit
EOF
```

This allows to catch the MS-Blaster worm for later investigation without endanger a Windows system. Unfortunately, this process does only work for worms that are already known.

4.3.4 Intelligence Gathering

There are many different honeypot products and setups, that all have their own purposes. The main object of honeypots is though to get to know the proceeding of intruders and the tools they use. There are a lot of questions that have to be answered to get to know the attacker.

- Who is the attacker?
- From which system does the attacker launch his attack from?
- Which tools does the attacker use?
- What purpose does this attack have?
- What skill level does the attacker have?
- What information or data did the attacker collect?
- How did he break into the system?
- ...

Many of these questions can easily be answered by using a honeypot. A sniffer running in front of the honeypot captures all traffic to and from the deception system. This discloses the IP address of the attacker, which services he connects to and more.

The difficulty is to gather the information unbeknown to the attacker. This can be realized by kernel modules that capture all events on system level. This proceeding allows to capture process events or kernel activity. At this level even encryption does not hide the attackers activities, because all keystrokes and actions are captured. This technique allows to track the intruders activity on the system step by step. The log file contains the order and the time of all commands that were executed.

Attackers often change, replace or add files to the captured system. They modify configuration files to change services to behave the way they like. To cover their tracks, attackers often replace important system programs. On UNIX systems it is common to replace the *ls* command, that does directory content listing, if the attacker wants to hide some files he uploaded to the captured system. He also replaces the *ps* command, that does a snapshot of the current processes, to conceal services he started. Therefore it is always useful to have some system integrity verifier that detects all changes to files on the system. This allows to capture the tools the attacker used to control the honeypot.

A high-interaction honeypot should be able to answer most of the above questions. The log of the honeypot contains information about the origin of the attack. At least the IP address of the system the attacker (mis-)used is known. The honeypot can give information about which files were accessed and which commands were executed. Therefore it is possible to estimate which information or data were captured by the attacker.

Some of the above questions cannot be answered clearly. The purpose of the attack or the skill level of the attacker have to be guessed. Actually there are no serious proceedings how to define the skill level of the attacker. Research in this field of application is very difficult, because there is almost no data available to examine. Organizations that have been attacked, often conceal the attack and the information they gained, because they fear to loose reputation or customers if it gets known that their security was to weak.

4.4 Honeypot Costs

This section is about the costs of honeypot installations. People who do honeypot research or sell honeypot products often claim that such devices are cheap and cost-effective. If this would be true, then many more organizations would have honeypots to protect their networks. This section analyzes the cost components of a honeypot installation. The fact that a deception system has to be integrated in the existing security concept makes it impossible to present some absolute calculations. Nonetheless it might be useful to analyze the expenses of a honeypot installation.

4.4.1 Deployment Costs

The total deployment costs include the purchasing, installing and configuring of the honeypot. It seems difficult to provide a total price in dollar, because the available products differ wide in price and service. However, the goal of this section is more to give an overview of the components of the deployment costs, than to provide some inaccurate numbers.

Honeypot Purchase and Installation Costs

To purchase a honeypot does not mean just to buy some software package. It should also be considered, that the honeypot has to run on hardware. Fortunately the system requirements for most honeypot software are modest. It is therefore sufficient to run the honeypot on discarded hardware that cannot be used anymore to run production services.

Most honeypot deployments need some additional devices to protect the other systems from attacks, that originate from the compromised honeypot. To separate the honeypot from production systems, additional devices like packet filter, switch or router are needed. To prevent attackers from manipulating the log files on the honeypot, it is vitally to have an additional computer that runs a syslog server where all honeypots can log the collected activities. Figure 4.5 shows a sample setup that contains the additional devices that are needed for a safe honeypot setup.

There are many different honeypot software packages. Both proprietary and open-source honeypots are available. Proprietary software may be easier to configure. The honeypot, the management server, analyzing tools and support are provided by the same manufacturer. Such a complete solution with one license costs about $800.-. Just for a single honeypot the license fee is a negligible

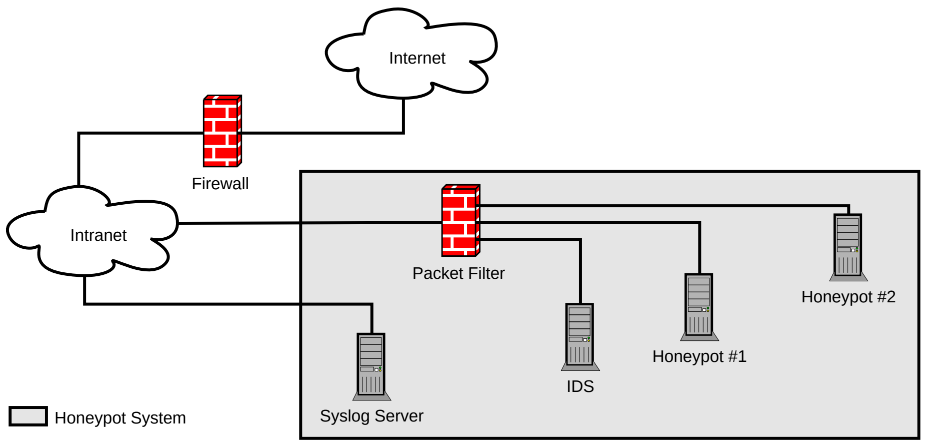


Figure 4.5: Honeypot deployment containing additional devices that are needed to protect the production network

factor, whereas in a honeynet deployment with many installations it might be a reason to use some open-source software. Open-source packages in contrary can be downloaded for free. If it is required to change the software to fit to special needs, an open-source honeypot might be the only choice. To deploy a whole honeypot setup, different open-source products may be combined.

As there is no general honeypot setup that is useful for every organization, there are two sample setups. One is a simple and cheap setup with basic components. This can be used by organizations that are not willing to invest a lot in additional IT security concepts, but like to be informed about the suspicious activity inside their network. The other setup is a complex one, this allows to collect more in-depth information about potential attackers.

Simple Deployment Example

Figure 4.6 shows an example of a simple honeypot deployment. A honeypot is located inside the internal network. Any connections to that system are logged to a remote server to prevent an intruder from modifying the log files. To prevent a successful intruder from attacking production systems inside the internal network, a packet filter has to be installed before the honeypot. This packet filter drops all connections from the honeypot to secure the other systems.

Complex Deployment Example

An example of a complex honeypot deployment is illustrated in Figure 4.7 . Several honeypots are installed on different locations inside the network. This setup collects information both from the *Demilitarized Zone (DMZ)* and the internal network. If an organizations provides services to the Internet, it is important to know what kind of dangers threaten these services. A setup with multiple honeypots allows to correlate the data captured. This procedure enables to detect automated attacks that scan the whole network for vulnerable

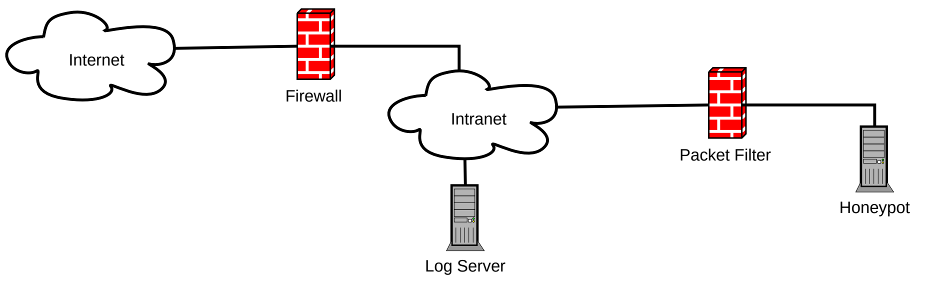


Figure 4.6: Example of a simple honeypot deployment

systems. It is important to have a management station that operates all honeypots to facilitate the configuration of these deception systems. The dotted lines represent the separate management network.

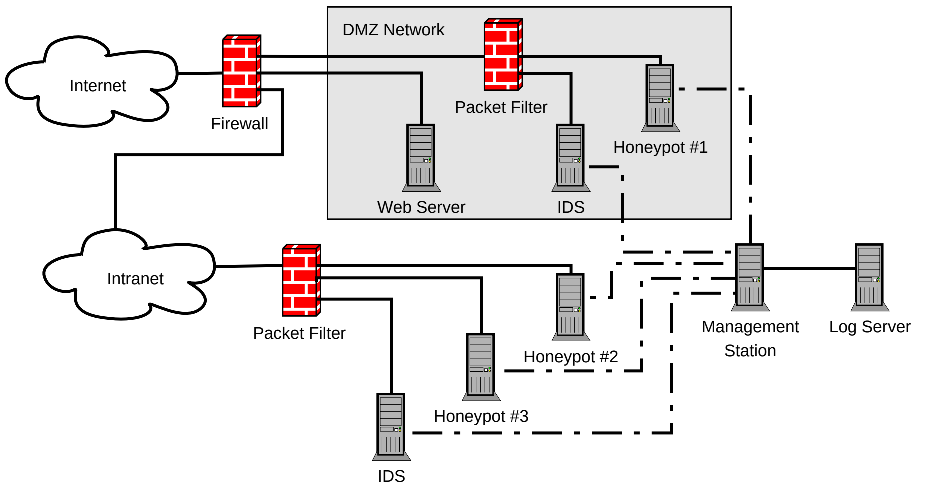


Figure 4.7: Example of a complex honeypot deployment

4.4.2 Maintenance Costs

If the project is well specified, the costs to purchase and install a honeypot can be calculated, whereas the maintenance costs can hardly be estimated. They loom during the time when the honeypot is running.

Maintenance costs consist of different components. One component is the observation of the honeypot. This designates the time where an administrator is checking the proper functionality of the system. Another component is the effort that has to be made to reinstall or clean up the system after a compromise. The incident analysis is the third and most important component.

Honeypot Observation Costs

Administrators should always keep an eye on the honeypot to notice any incident. This process is very important, because every incident that remains undetected is a security risk. An attacker may misuse the honeypot to attack other systems. At the time, he is aware of the presence of a fake system, an intruder will most likely try to blot out all evidence of himself. In most cases an attacker will delete all files on the system. Even though the honeypot might log all events to a remote server, important evidence files might be destroyed.

A low-interaction honeypot like *honeyd* might be controlled on a weekly basis. This type of honeypot simulates various systems on unused IP addresses and logs any connection attempts to them. There exist different tools that help to analyze the log files of this honeypot. One of them is *honeydsum*[4]. This tool takes the log file of *honeyd* and generates an abstract. This allows an administrator to comprehend with minimal effort, what happened during a certain time period on the honeypot.

A high-interaction honeypot needs much more effort to be observed. From this type of deception system originates a high risk for the network. An attacker who managed to break into a high-interaction honeypot, operates on a real operating system. He is able to do a lot of damage from there. Therefore it is very important to notice the incident while the attacker is still operating on the system. Many honeypot solutions include some notification mechanism to alert the administrator of an incident. As a consequence an administrator might be alerted dozens of times a day. He then always has to check what happens on the system, to be able to disconnect the honeypot from the network before the intruder is able to wipe out his tracks.

Honeypot Recovery Costs

Attackers often install back-doors after they break into a system. This allows them to enter this system in the future more easily. Therefore it is very important to recover the honeypot after an incident. One way to do so, is to reinstall the whole system. This guarantees that all back-doors from the previous attacker are removed. This proceeding may be a secure, but expensive procedure. Another solution may be to delete all tracks from the attacker. Sometimes it is appropriate and cheap just to delete the compromised user account. However if the attacker installed additional software, the proper functionality of the honeypot cannot be guaranteed anymore. Just to undo the changes, the attacker made to the system, can never eliminate all risks.

To recover the honeypot system more convenient, it is a good idea to use virtual honeypots. This can be achieved by using software, like *VMWare*[5], that helps to emulate several virtual systems on a real one. Just a few commands or mouse clicks after an incident, the virtual honeypot is already up and running. Unfortunately, this kind of honeypot system can easily be detected. There exist

[4]http://www.honeynet.org.br/tools/
[5]http://www.vmware.com

several tools that allow attackers to detect, if they operate on an emulated system.

Incident Analysis Costs

To do an appropriate incident investigation is an expensive process. Especially skilled attackers try to hide themselves, what makes it difficult to expose their activity. [Pro00a]

On January 15, 2001 the *Honeynet Project* launched the *Forensic Challenge*[6]. They placed an image of a compromised system on their website. The goal was to allow incident handlers around the world to all look at the same data and to see what information about the attack can be gained. The compromised system was a default Red Hat Linux 6.2 Server installation. Additional data from the *Snort* intrusion detection system was also provided. The given information corresponds to the data that gets collected by a high-interaction honeypot. All submissions had to include an assembly of the time spent on this challenge. Figure 4.8 shows the costs of this incident, calculated with the template provided by the ICAMP study. The hourly wage of US$33.65 for security incident handlers was given by the Honeynet Project.

The most astonishing fact was how time consuming this investigation was. Dave Dittrich, coordinator of the Forensic Challenge, wrote on the website:

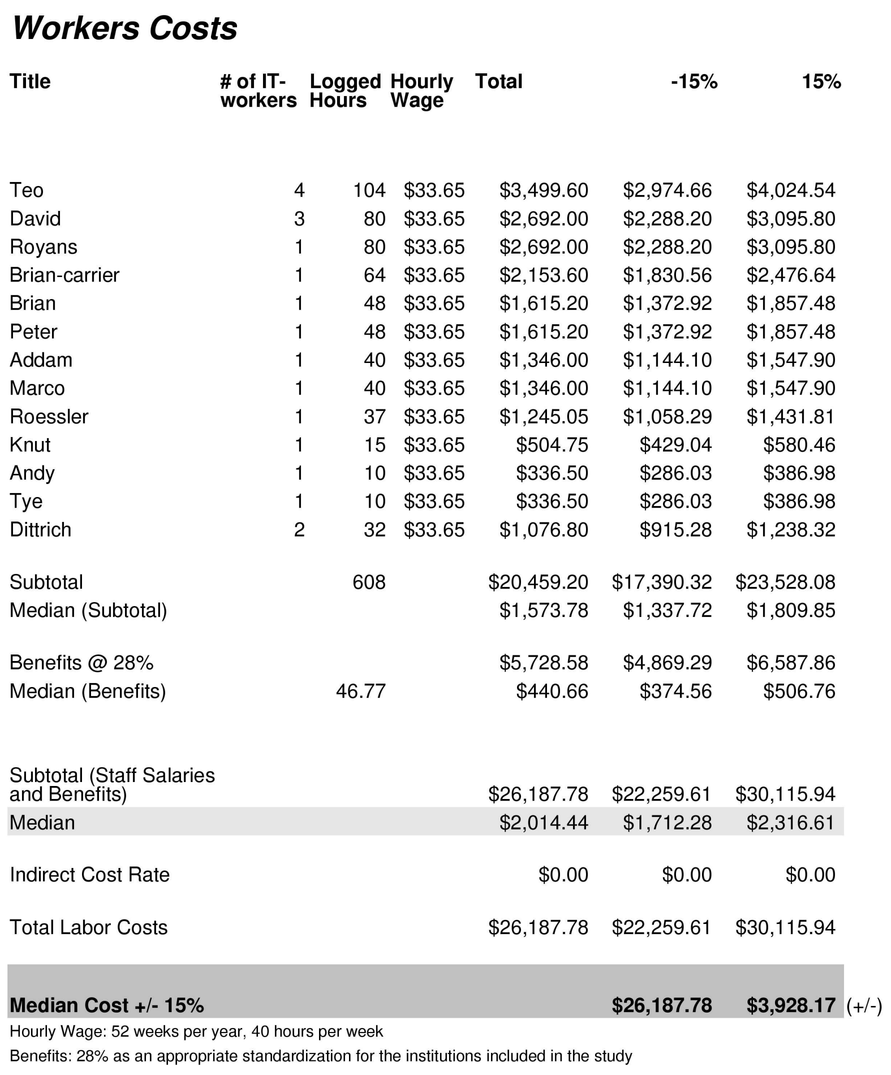
> *"Many entrants (and some who contacted us who couldn't make the deadline) had no idea how much time this analysis would take, and it took a lot . . . Most finished when they ran out of time, not when they felt they were done."*

The results from the participants show, that they took very thorough and professional efforts to analyze the compromised system. Although the teams did a great job, every entrant found at least one thing that the others did not. This proves that in computer forensic there is no perfect incident analysis, because time is a limiting factor.

The entrants spent an average time of about 46 hours per person. They spent a whole working week to find out what the intruder did in about half an hour. Using a standard mid range annual salary of US$70,000, the cleanup costs amount about US$2,000. The hourly wage of US$33.65 that was calculated from the annual salary seems to be a very low estimation. Several other studies calculated an hourly wage for an incident investigator of about US$70 this would lead to average cleanup costs of more than US$4,000. It has to be considered that this are the investigation costs for just one single incident. There are organizations that detect dozens of similar events in a month.

All thirteen teams that participated in that challenge did together a really in-depth analysis of this incident. It is probably the most precise investigation that ever was published. The total incident investigation costs sum up to US$26,000. The huge amount of money that is needed to do an in-depth incident investigation is the main reason why honeypots are not yet widely used. It

[6]http://www.honeynet.org/challenge



Workers Costs

Title	# of IT-workers	Logged Hours	Hourly Wage	Total	-15%	15%
Teo	4	104	$33.65	$3,499.60	$2,974.66	$4,024.54
David	3	80	$33.65	$2,692.00	$2,288.20	$3,095.80
Royans	1	80	$33.65	$2,692.00	$2,288.20	$3,095.80
Brian-carrier	1	64	$33.65	$2,153.60	$1,830.56	$2,476.64
Brian	1	48	$33.65	$1,615.20	$1,372.92	$1,857.48
Peter	1	48	$33.65	$1,615.20	$1,372.92	$1,857.48
Addam	1	40	$33.65	$1,346.00	$1,144.10	$1,547.90
Marco	1	40	$33.65	$1,346.00	$1,144.10	$1,547.90
Roessler	1	37	$33.65	$1,245.05	$1,058.29	$1,431.81
Knut	1	15	$33.65	$504.75	$429.04	$580.46
Andy	1	10	$33.65	$336.50	$286.03	$386.98
Tye	1	10	$33.65	$336.50	$286.03	$386.98
Dittrich	2	32	$33.65	$1,076.80	$915.28	$1,238.32
Subtotal		608		$20,459.20	$17,390.32	$23,528.08
Median (Subtotal)				$1,573.78	$1,337.72	$1,809.85
Benefits @ 28%				$5,728.58	$4,869.29	$6,587.86
Median (Benefits)		46.77		$440.66	$374.56	$506.76
Subtotal (Staff Salaries and Benefits)				$26,187.78	$22,259.61	$30,115.94
Median				$2,014.44	$1,712.28	$2,316.61
Indirect Cost Rate				$0.00	$0.00	$0.00
Total Labor Costs				$26,187.78	$22,259.61	$30,115.94
Median Cost +/- 15%					**$26,187.78**	**$3,928.17** (+/-)

Hourly Wage: 52 weeks per year, 40 hours per week

Benefits: 28% as an appropriate standardization for the institutions included in the study

Figure 4.8: Incident analysis costs of the Honeynet Challenge calculated with the template provided by the ICAMP model

is impossible to justify such huge investments in a time where the budgets for the IT are cut. The benefits from honeypots depend on the maintenance. This are expenses that have to be made every month and perhaps will not amortize at all.

4.5 Risks

Honeypots belong to network security devices like intrusion detection systems or packet filter. They can increase the security level by luring and confusing potential attackers.

High-interaction honeypots gather the most information from an attacker, but they suffer from the high risk of being misused as a starting point for attacks on production systems. This risk can be minimized by observing these systems cautiously. An intrusion detection system, that is placed before the honeypot, generates an alarm, whenever connections to the deception system are opened. This eliminates the chance of an intruder to break into the honeypot unseen. It has to be considered that there can be dozens of connections a day and to check every single one of them would be very time consuming.

4.6 Conclusions

The cost-benefit evaluation of honeypots makes it necessary to separate them in high- and low-interaction honeypots.

Low-interaction honeypots are useful for every organization which has to maintain a computer network. It helps them to monitor the network status and to guarantee the functionality of their services. A low-interaction honeypot deployment detects primarily suspicious network connections. It does not matter if those connections originate from attackers who scan systems for vulnerabilities, malware which tries to spread, network misconfigurations or crooked users who are looking for unprotected resources. All these threats have to be eliminated to provide an efficient infrastructure. It is possible to deploy a low-interaction honeypot at low costs. There are excellent open-source tools, as honeyd, which can be used out-of-the-box with minimal configuration effort. A system administrator with average computer or networking skills is able to configure and maintain such a system. There are excellent tools that process the log files of these honeypots and generate human readable output as statistics and diagrams. This allows to get a picture of the state of the network within minutes. Low-interaction honeypots are cost-effective in observing networks and are first choice if an organization plans to get involved in honeypot technology.

High-interaction honeypots address a more specific audience as low-interaction honeypots do. This kind of honeypot is primarily used to learn from attackers. It is useful to get to know the proceeding of attackers and the tools they use. A high-interaction honeypot can be used to detect and identify malware that spreads on the Internet. The deployment costs of such systems are negligible, because the incident handling costs has the lion's share of the total costs. This costs cannot be reduced without a negative impact on the benefits of those systems. Another drawback of high-interaction honeypots is, that the costs cannot be calculated in advance. The costs depend on the number of incidents and evolve during the use of the system. As high-interaction honeypots mostly are real computer systems, they increase the risk of the whole IT infrastructure if they are not maintained conscientiously. High-interaction honeypots address companies and organizations that are involved in IT security research as *Computer Emergency Response Teams (CERT)*, *Anti-virus Software Developers* or the *Honeynet Project*.

Chapter 5

Active Honeypots

5.1 Introduction

This chapter starts by defining the term *active honeypot*. The section about the motivation for such advanced deception systems is particularly important for people who are involved in IT security research. A generic concept for such deception systems is presented. This chapter is looking for application scenarios and examples in theory and practice, to get an overview of the current state of active honeypot technology. It ends with a proposal of an architecture for an active honeypot system.

5.2 Definition

The term *active honeypot* describes an advanced honeypot system. In contrary to traditional honeypots that undergo passively all attack attempts, active honeypot systems actively react to them.

The active part of this honeypot system generates an adaptive target that suits the attacker. An active honeypot system gathers foremost as much information about a potential attacker as possible. It generates a suitable target on the basis of the gained information. This generated target is tailored to the needs and abilities of the attacker. This facilitates to get more in-depth information from intruders.

5.3 Motivation for Active Honeypots

Attackers are getting numerous and more intelligent. They cause increasing damage. IT security suffers from the lack of knowledge about the dangers that threat todays networks. It is vital for the security community to get more information about attackers, their proceeding and the tools they use.

Traditional honeypots are excellent tools to learn from the blackhat community. Unfortunately, those systems suffer from various disadvantages. Traditional honeypots have a limited view. If an attacker is not interested in the honeypot system, he chooses a more valuable system for his attack. A honeypot that does not become attacked, gathers no information and is therefore totally useless.

To improve honeypot technology, the deception system has to become more interesting to an attacker. This lowers the chance that he simply evades this system. The fact, that there is no general type of attacker, but a whole bunch of individuals with totally different intentions, leads to the conclusion that a next generation honeypot system has to be able to simulate different targets. So that each attacker finds his favored target on the honeypot.

Active honeypots induce attackers to spend more time on the honeypot. As long as they stay on the honeypot, the system is able to gather information about their activity. As Ralf Hofstetter points out in his diploma thesis [Hof04], an active honeypot is able to give the system administrator more time to react to an incident. The more information can be gained from an attacker, the better the future protection from this threat will be.

5.4 Generic Concept for Active Honeypots

This section presents a generic concept for active honeypot. It starts with the goals of such deception systems. The automation of the incident handling will certainly boost honeypot diffusion.

5.4.1 The Goal of Active Honeypots

A honeypot must be attacked to collect any information. To be an attractive target for potential intruders, it should contain a lot of *'honey'*. This means, that it must contain a lot of interesting information for the attacker or interesting vulnerabilities. A blackhat who attacks the honeypot should *'stick to the honey'*. As long as an attacker stays on the honeypot, as long does he provide insights into his work.

The main goal of active honeypots is to advance in blackhat research. More precise information about security incidents and attackers are needed to improve IT security. An active honeypot is able to challenge attackers by generating adaptive targets. Challenged attackers will spend more effort and time on the honeypot and reveal their knowledge.

5.4.2 Automated Incident Handling

Incident handling is a very complex process that takes a lot of effort. As the *Forensic Challenge* of the *Honeynet Project* illustrates, an in-depth incident handling requires a lot of manpower and money. To automate this process will save a lot of resources, if it can be done in an adequate manner.

Particularly human attackers show a wide range of different behaviors. It seems not at all simple to automate incident handling. Fortunately, many of them lead their assault the same way. They use the same tools and proceed in a similar fashion.

An active honeypot that handles an incident on its own will be able to develop patterns on how to react to an incident. If such patterns exist, they can be used to build up an effective defense against attackers.

Automated incident handling consists of two important steps. The first one is the adaptive target generation and the second one is the automated attack response. Adaptive target generation builds the main part of this master thesis. Automated attack response is all the same presented, due to the potential of this concept.

5.4.3 Adaptive Target Generation

A honeypot can be described as a target. It is a target for attackers which is waiting to become assaulted. It is not at all simple to configure the honeypot in a way that it attracts interesting attackers. A honeypot with a standard configuration may detect attacks, but they are certainly not the most interesting ones. Just to catch the same attackers every day does not contribute to the blackhat research. It is important to catch attackers who are skilled and who use new tools ore new proceedings to capture a computer system.

There is not a general attacker who threats computer networks. In contrast, there are a lot of totally different individuals with different intentions. Some are looking for valuable data, whereas others are looking for resources they can use for free. This are just two categories of attackers. It is impossible to create one honeypot that suits to all potential intruders. It might be possible to create one system that contains all sorts of vulnerabilities, but this system will look so suspicious to blackhats, that it will certainly not be able to lure advanced attackers. A solution to this problem consists of a system that generates the targets on the fly, while an attacker assaults.

Foremost it is necessary to know the attacker. This is a real problem for active honeypots. They need to gather as much information about the attacker as possible, before he succeeds to break into the system. Information that can be gained during the attacker probes the system are: Time and date, source IP address, target port or service, etc.

The active honeypot system puts all this information chunks about the attacker together to get a picture of his intentions. Based on this evaluation, the system chooses a suitable target and simulates this target for the attacker. Unfortunately, the amount of information, that can be gained before the attacker has achieved to break into the system, is limited. Therefore it is really important to have some classification scheme that is able to identify a single attacker on a few criteria that can be gathered beforehand.

During the attack onto the simulated system, as much information as possible has to be gathered. Every single keystroke can be recorded. To advance in blackhat research, it is important to automatically evaluate the whole incident afterwards. To appraise how well the generated target suited to the attacker helps to improve the active honeypot technology.

5.4.4 Automated Attack Response

Security professionals and system administrators spend a lot of their time, just by responding to security incidents. They reinstall compromised systems or track down attackers. Many of these activities are time consuming and annoying. To improve the benefits and decrease the costs of honeypots, an active honeypot system automates the attack response.

Counter Intelligence

Attackers leave a lot of tracks on the target system. Unfortunately, this information can not be used to generate a target, because it has to be built before the attacker has achieved to break into the system. To improve the attacker identification process it is necessary to get more information about him, while he probes the target system. This is achieved by using counter intelligence.

Some tools do not only passively gather data of the attackers activity on the honeypot. They also actively gain information on any system that connects to it. This feature is useful to get real-time knowledge about who is attacking. Many information can only be obtained during the attack is in progress. The honeypot does some port scanning on the attacker, tries to grab a telnet banner or fingering the attackers system. There are a lot of information that can be gained by actively gather information: [Spi03d]

- Finger: Finger the remote system for user information.
- Tracer: Finger systems which are remotely connected to the system where the attack originates from.
- Portscan: The honeypot does a port scan on the attacking system.
- Whois: Whois lookup on the remote system.
- DNS: Resolve the name of the attacking IP address.
- Telnet/FTP/SMTP Banner: Connect to the specific ports on the attacking system and obtain the banners.
- HTTP Server Header: Connect to the HTTP port on the remote host and issue the HEAD command.
- HTTP Document: Connect to the HTTP port on the remote system and issue the GET command.
- Traceroute: Traceroute to the remote host using ICMP.

All the information that can be gained by using this tools improves the attacker identification and therefore the target generation process as well.

Counter Measures

The incident handling process also includes counter measures. System administrators counteract attackers by isolating the infected systems to stop the attack. Active honeypots are able to stop attacks on their own to protect the production network.

The following list presents a selection of possible countermeasures:

- Cutting off network segments to protect sensitive network parts from the attacker.
- Isolation of captured hosts. This is done by closing ports on switches.
- Banning chosen network flows by inserting filtering rules on remote devices like routers or firewalls.
- Feeding a kind of active realtime blackhole list (RBL) dedicated to worms. This allows to block IP addresses of malicious hosts.

Counter Offense

Countermeasures might allow to protect sensitive network segments, but they are not able to stop the worm problem. Active honeypots are able to eliminate the worm problem as described in *Fighting Internet Worms With Honeypots* [Oud03a].

Performing a counter offensive sounds quite simple in theory. A worm has infected host A and is trying to propagate itself to host B, which is the active honeypot. It can be assumed that host A is infected with that worm, due to the technical vulnerability in that host. If the vulnerability was not removed by the worm's payload and host A is still accessible, the honeypot (host B) can launch a counter attack to host A. By abusing the same vulnerability on host A, which was used by the worm, host B takes control of host A. If the honeypot succeeds in capturing host A, it can kill the running worm process, clean host A and harden its security.

To highlight the applicability of that theory, there is a honeyd configuration that stems the MS-Blaster worm. The honeyd configuration file looks like this:

```
create default
set default personality "Windows XP Pro"
add default tcp port 135 open
add default tcp port 4444 "/bin/sh scripts/strikeback.sh $ipsrc"
set default tcp action block
set default udp action block
```

The TCP port 135 of the honeypot remains open and accepts remote RPC requests. MS-Blaster connects to this port to abuse the DCOM vulnerability. A script attached to the TCP port 4444 launches the counterstrike. The strikeback.sh script looks like this:

```
!#/bin/sh
# Launches a DCOM exploit toward the infected attacking host
# and then run cleaning commands in the remote DOS shell obtained
./dcom_exploit -d $1 << EOF
REM Executes the following orders on the host :
REM 1) Kill the running process MSBlast.exe
taskkill /f /im msblast.exe /t
REM 2) Eliminate the binary of the worm
del /f %SystemRoot%\system32\msblast.exe
REM 3) Clean the registry
echo Regedit4 > c: \cleanerMSB.reg
echo [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
     >> c:\cleanerMSB.reg
echo "auto windows update" = "REM msblast.exe" >> c: \cleanerMSB.reg
regedit /s c: \cleanerMSB.reg
del /f c:\cleanerMSB.reg
REM N) Specific actions to update the Windows host could be added here
REM N+1) Reboot the host
shutdown -r -f -t 0 exit
EOF
```

This script kills the MS-Blaster process, removes the worm binary and cleans the registry. To improve this script, it should also contain some code to harden the system.

5.5 State-of-the-Art

Honeypots are not yet widespread. Although there was a honeypot hype some years ago, it has become more quiet nowadays. There are several reasons, why honeypots did not yet succeed to become an omnipresent security device.

The first reason is, that honeypot technology is still in its infancy. There are several implementations of honeypot tools. However, many of them are still in development state and need a lot of knowledge in terms of setup and administration.

The fact that honeypots lure attackers, leads to some averseness to this technology. Nobody wants to attract attackers, if he has to secure a production environment.

It is also very difficult to get the resources for a serious honeypot infrastructure. A honeypot needs hardware, software, additional IP addresses and a lot of administration time to keep it save and to evaluate the collected data. Most of todays IT budgets have no room for investments in such infrastructure.

Although there are some concepts for active honeypots available, none of them covers adaptive target generation. They cover counter intelligence or counter measures. There are honeypots that actively react to security incidents, but none of them made it beyond test setups. There are no known productive installations of active honeypots.

It is hard to get information about active honeypot systems at the moment. There are nor publications nor tools available that would conform to the idea of adaptive target generation. Active honeypot technology is in pre-development state. This diploma thesis proposes thus some application scenarios and an architecture for such a system.

5.6 Application Scenarios for Active Honeypots

This section presents several scenarios for active honeypots and highlights how they provide additional security to the IT infrastructure.

5.6.1 Research Environment

The major field of application for active honeypots is definitely the research environment. IT security research suffers from the lack of know how about blackhats. They need additional tools to advance in their research.

Honeypots do not need high-end hard- and software. This seems to be a perfect prerequisite to become an important research topic. Particularly in academic environments where manpower is cheap and network resources are available.

Incident Database

An active honeypot that generates adaptive targets and handles incidents on his own can be used to build an attack database. An active honeypot, which is waiting for attackers, has to be deployed. This device stores all collected information to a database together with information about the target it simulated for a certain attacker. All the information from this database is useful for further research in IT security.

Target Evaluation

This scenario goes one step ahead. There are no serious statistics about what attackers are really looking for. To secure production systems more effectively, it is vital to know the intentions of the attackers. To advance in blackhat research, it is necessary to generate statistics about the preferred targets of attackers. An active honeypot does not only generate adaptive targets, it evaluates how well the generated target suited to the intentions of the attacker as well. The time period an attacker spends on a simulated target and the amount of activity gives a qualitative estimation about how well the target suited to the attacker. This helps to get to know whether attackers might be more interested in the data on the system or the resources.

Risk Research

Most organizations which use IT do not know what dangers they face. This makes it difficult to protect their networks. Active honeypots which handle the whole incident automatically can be used to build a database about attacked targets. This helps to generate statistics about systems that are endangered. Particularly IT security companies might be interested in knowing what dangers are threatening the networks they have to secure. This allows to build a more specific defense. Organizations which are aware of the threat from the Internet are willing to spend more money on prevention.

5.6.2 Production Environment

Active honeypots are in a pre-development state. This prevents the industry from investing into such a technology. If honeypot technology becomes mature, it has certainly the potential to do a good job in production environments.

Distract Attackers

The legal situation does not allow to efficiently pursue attackers today. If they operate from foreign countries, there is low chance to bring them to trial for their activity. Therefore it is not the main purpose of the industry to catch attackers. They are more interested in confusing them. Attackers who are confronted with adaptive generated targets are not able to distinct between deception and production systems. This will most likely distract them and they will look for other, probably easier targets.

5.7 Examples of Active Honeypots in Theory and Practice

At present there are no active honeypots in use. At least are there no publications about such setups.

There is a Korean paper *'Design of Active HoneyPot System'* [KKLM04]. This paper describes the implementation of a virtual emulation service that leads the intruder to a honeypot. The gained information on the access paths and the skills of the intruder allow to generate a policy to protect the other systems from new attacks.

As illustrated in Figure 5.1 , this system is structured in four components. A firewall, a network intrusion detection system, a honeypot and a system management server. If the firewall detects any packets from abnormal users, the packets are redirected to the honeypot. All activity from the intruder on the honeypot will be logged on the management server. This server will then actively launch some counter measures. The management server is capable of generating new firewall rules or intrusion detection patterns and update the other devices.

This active honeypot system enhances the security of the infrastructure by reacting not only to already known, but also to new attacks.

Especially in environments where the availability of services is very important, it is dangerous to generate the firewall rules automatically. This can lead to a lock out of normal users who require the production services. This fact often gives the reason for not to use such active systems.

Even though the system described is called an active honeypot, it is all the same not that type of active honeypot, this diploma thesis is about. The

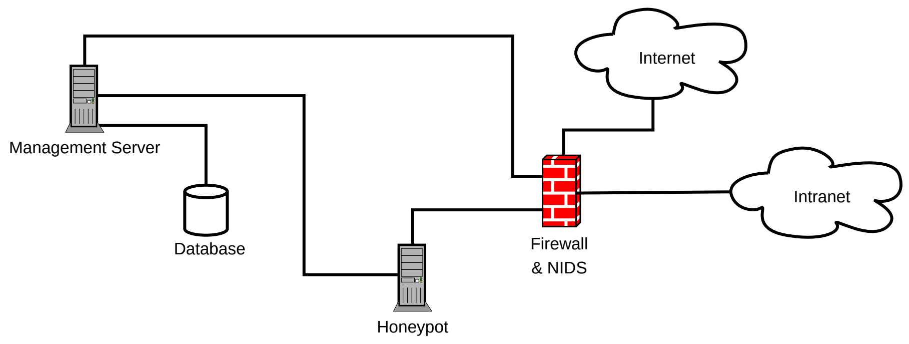


Figure 5.1: Architecture of the Korean active honeypot system.

active component, in the Korean paper, is responsible for the redirection of the malicious user and the rewriting of the firewall rules. The target system is a traditional honeypot system with all its disadvantages.

5.8 Active Honeypot Architecture

This section proposes a general architecture for an active honeypot. It is based on the introduced Korean paper *'Design of Active Honeypot System'* [KKLM04]. The *'active'* part of the system described in this paper does merely concern the identification of malicious users and their redirection to a honeypot. Nevertheless, this paper is a good starting point to build a system which generates adaptive targets based on the attackers behavior.

A general architecture for an active honeypot as illustrated in Figure 5.2 consists of the following components:

Honeypot Server: This is not a real honeypot. It is just the server that generates virtual honeypots, the so called targets.

Management Server: The management server is the heart of the active honeypot setup. This system gathers all data from the honeypot. It processes the collected data and generates adaptive targets in real-time, based on the information that is stored in the database.

Database: The database is important, because it holds all kinds of information like the collected data from the honeypot or the firewall.

5.8.1 Database

The database is the memory of the active honeypot. It contains all logged data from the virtual honeypots.

An active honeypot which does incident handling on its own will be able to handle several incidents at the same time. Therefore it is important to store all

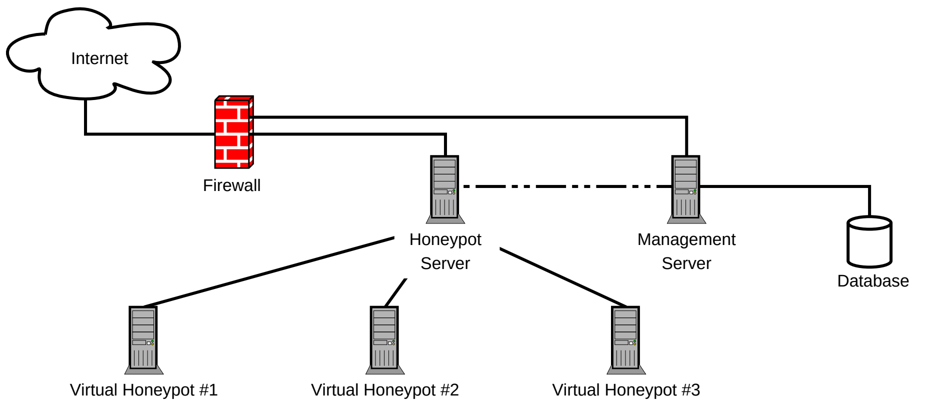


Figure 5.2: General architecture for an active honeypot.

gathered data in a database. An active honeypot gathers about the same data as any high-interaction honeypot does. That includes:

- Time & Date of the attack.
- Source IP address of the sent network packets.
- Payload of the sent network packets.
- Keystrokes that were sent to the honeypot.
- Changes that were made to the file system.
- ...

This is not a complete list, but it gives an impression of the database content. During the development of such an active honeypot system, it is most likely that this database will also change. Unless the problem of attacker identification is properly solved, it is most likely that the attributes of this database will change often.

5.8.2 Management Server

This type of active honeypot needs a controller that can handle the whole incident management and builds thus the active component. This task is accomplished by the management server. The incident handling is a very complex process that has to be separated into different tasks.

Data Collection

Proper decision making is only possible, if it is based on an outright data base. Therefore it is important to collect as much data as possible from the attacker to generate a target that fits. The virtual honeypot systems logs the attackers activity to the management server. This device stores the complete data in the database. As there is no publication about what attributes are needed to identify an attacker, this task has to be accomplished with caution.

Information Processing

The data itself gives no clue about the intentions of the attacker. It is necessary to enrich the collected data with generated meta data. The whole information processing is realized on the management server.

Thus it is important to extract meta data from the database. For example, extracted information can be:

- Skill level of the attacker.
- Intentions of the attacker.
- Quality of the simulated target. Did the attacker spend a lot of time on the fake target?
- ...

As the extracted information depends heavily on the collected data, there is no complete specification of this process.

Reaction Evaluation

Based on the gathered data and the extracted information, the management server chooses a target for every single attacker.

All the information from the database are needed to generate a target that might suit to an attacker. It is very important to evaluate the quality of the target after the intruder leaves the system. On the basis of the time he spent on the honeypot or the activity of the attacker, it is possible to make conclusions about how well the target lured the attacker. This information is useful to improve the active honeypot.

Target Generation

The management server creates targets based on information sent from the honeypot.

Target creation requires a lot of information. The target, for example, has to be bound to a specific IP address. Furthermore, it is necessary to specify how the target might look like. There are several possibilities how to generate a target. It is possible to define some components in advance that can be put together to build a virtual honeypot. An other approach would be to define complete virtual honeypot setups and just attach the one that suits best to the IP address, the attacker is connected to.

It does not matter, not to know which targets might be suitable to the attackers. The evaluation process afterward gives feedback to improve the targets or develop new ones. The target creation process will evolve during the test phase with active honeypot systems.

5.8.3 Honeypot Server

The honeypot server starts a virtual honeypot for every attacker. This virtual system exhibits a single target. For simplicity, it might be appropriate to use honeyd as a honeypot server. The management server changes its configuration file and adds a new virtual host for every attacker. Although this does not allow high-level interaction to the attacker, this approach will help to get to know the concept of adaptive target generation.

5.9 Conclusions

As outlined in this chapter, an active honeypot which creates adaptive targets does entail a lot advantages. A honeypot that creates targets that suit to the abilities of the attackers will certainly distract them. If the target challenges the attacker, he will be busy and stay longer on the deception system. This allows to get more in-depth information about his activities and the tools he uses. To advance in blackhat research, it is important to get to know what targets attackers prefer. Although there are no serious statistics about their favored targets, it seems that they are just looking for resources. To capture a system with high-bandwidth or a lot of disk space seems to be the preferred aim of blackhats.

Unfortunately, there are currently no active honeypot systems available. The presented architecture gives an overview about how such a system might look like. It is not an exact definition. The reason is that it needs a lot more experiments to determine exactly which information can be gathered or how a target can be built.

This chapter also covered counter measures, counter intelligence and counter attack. Although there are systems which are able to perform such tasks already, they are not in widespread use. The reason is that it is dangerous to perform counter attacks. Any damage that is caused by the honeypot will most likely have consequences. No one will risk a law enforcement and therefore these techniques are treated with caution.

An active honeypot is a sensitive device. It needs an in-depth planning to place such a setup in the existing infrastructure. Just to install and look what might happen can be very dangerous. However, with an appropriate effort an active honeypot should bring some additional security to the IT infrastructure.

Chapter 6

Problems of Active Honeypots

6.1 Introduction

This chapter illustrates the drawbacks of active honeypot technology. Attacker identification and classification, automated incident handling and automated attack response are challenges that are not yet solved. How attackers defeat honeypot systems is described as well as how to protect these deception systems. These problems allow to make predictions about the further development of active honeypot technology.

6.2 Automated Incident Handling

As the chapter about the *Cost-Benefit of Honeypots* illustrates, is incident handling a really complex and resource consuming process. The given example shows, that 40 working hours were needed to find out what an attacker did in half an hour. This is a 80:1 ratio. The fact that computer forensic is so laborious is the main challenge for the implementation of an active honeypot. The automation of such a complex process like incident handling holds many problems.

An active honeypot generates adaptive targets on the fly, while the intruder is acting on the deception system. An intruder might spend single sessions from some hours to a few minutes on the honeypot. Therefore the time between the information collecting and the reaction must not be more than a few seconds.

The short time period which is available to generate a target for the attacker is a reason to automate this process. Human operators would have no chance to analyze and react to an incident in that short time period. Unfortunately, the automation of the target generation is not at all trivial. Especially if the generated targets have to be solid enough to trick advanced attackers. The complexity of this process is a reason why the problem of adaptive target generation is not yet solved.

6.3 Attacker Identification

Active honeypots improve cyber defense by actively react to security incidents. To react to incidents, it is vital to know the attacker. The ability to link attackers with attacks allows to develop effective counter measures.

Whereas the methods for identifying perpetrators of crimes in the law enforcement context are well developed, similar capabilities do not currently exist for IT security incidents. The ability to accurately identify and locate attackers is still missing.

Traditional honeypots do not need an attacker identification process. Whereas hundreds of potential attackers might probe a system every day, only a minority of them will ever achieve to break-in. It is unlikely that several attackers are going to capture a system at the same time. Active honeypots generate a target based on information about the attacker. Therefore it is essential to separate all potential attackers who are probing the system. Without a profile of the attacker, it is not possible to generate a suitable target. Therefore it is essential to identify all potential attackers who are probing a system.

The attacker identification is a vital condition to react to incidents. This section describes a general attack procedure and presents several methods for attacker identification.

6.3.1 Attack Procedures

A general attack procedure of an intruder who is interested in the resources or the data of a target system will most likely consist of the following steps.

1. The attacker probes the target system for security flaws. This step starts most likely with a port scan.

2. If some security flaws were found, the attacker exploits the security breach to get access to the machine.

3. Attackers are not willing to share their captured systems to other blackhats. Therefore they will fix the security holes.

4. To reenter the compromised system later, they need a convenient but hidden entrance. This entrance is created by installing a backdoor.

5. Attackers misuse the captured system over a long time. It is not necessarily the case that they perform their whole attack in one go. From published incident analysis, it is known that blackhats do often reenter captured machines days or weeks after the first compromise for further activities.

To get a complete overview of the proceeding of intruders, it is important to assign the single activities to the different attackers. If it is known which activities were done by whom and in which order, it will be possible to build a system that is able to generate targets to lure and challenge blackhats.

6.3.2 IP Address Based Identification

As long as the intruder connects from the same system, he can be identified by his IP address. All incoming connections to the honeypot are grouped using the source IP address of the incoming network packets. To analyze one particular attacker, all connections from that specific IP address have to be analyzed.

The problem with IP address based identification is, that intruders often attack from many different systems to blur their traces. Intruders might also modify the headers of the packets and write random source IP addresses into them to lure the honeypot system. This proceeding renders IP based identification useless. [Gra00]

6.3.3 File Based Identification

Another way to identify attackers is file based identification. Lance Spitzner gives in his book [Spi03d] a brief description about identification through file recovery. Configuration files can tell a lot about the identity of an attacker, about his motives and where he is coming from. Attackers do often install services on captured machines and have therefore to write or rewrite configuration files.

It is very difficult to get such information from a running honeypot. There is a big chance that the attacker might notice that the files have been accessed or copied from the system. Especially on active honeypots, where attacker identification has to be performed in realtime. That is why file based identification is not suitable for such deception systems.

6.4 Attacker Classification

An ideal active honeypot system generates one specific target for every attacker. In reality, this method is an over-the-top objective. Particularly because honeypot technology is still in its infancy. Whereas there might be thousands of unique attackers with totally different intentions, it is appropriate to class them in a few categories. To limit the number of adequate targets, will reduce complexity and ease the development of such a deception systems.

An attacker classification can be based on several characteristics. These characteristics should be measurable and extractable from the data gathered from the honeypot system. Examples of such characteristics could be source IP address, established connections, target system, used tools, etc.

6.4.1 Automated vs. Human Attacker

One type of classification that seems to be applicable, is the separation of human and automated attackers.

Automated Attackers

To the category of automated attackers belong all tools that are able to probe and compromise computer systems on their own. Some of them are also able to launch new attack cycles themselves. In the last few years these tools experienced a fast development in intelligence and number. They mostly target known security holes. While they were first scanning and then attacking in the early stage, todays automated attack tools are able to do both tasks in the same time. Automated attacks result mostly from machines that are infected with worms or viruses. This attacks are not at all subtle. They scan whole networks for a specific security flaw. It does not matter what kind of services or data is located on the target machine. The only aim for this tools is to infect as much systems as possible, to control resources - mostly network bandwidth. This allows to launch vast distributed denial of service (DDOS) attacks to bring down web services as http or dns.

Human Attackers

Human attackers do not have the same goal as automated attackers have. Although they are also interested in controlling as many systems as possible, they prefer a more subtle proceeding in capturing systems. In contrast to automated attackers do the human ones fear to become uncovered. Whereas automated attackers assault every available system, choose human attackers the perfect target. They are looking for valuable data, computing power, disk space or network bandwidth. This allows them to sell sensitive data or use expensive computing power for free. They also like to use third party disk space to share prohibited data and misuse foreign network bandwidth to launch denial of service (DOS) attacks.

Humans Automate their Attacks

It is not at all simple to separate human attackers from the automated ones. Often the human attacker starts his attack with some tools that probe subnets for known security flaws. If there are any, these tools are also able to compromise the target system on their own. From there on, the intruder itself logs onto the compromised machine and does everything he likes. An example for such an attack are the IRC bots that roam around the Internet. This tools compromise vulnerable systems and establish connections to chat servers. Human attackers are then able to control the attacked systems through these IRC connections. Until the human attacker logs into the compromised system himself, it is not possible to distinguish between human and automated attacker.

6.4.2 Skill Level Classification

Another applicable classification scheme is based on the skill level of the attacker. Ralf Hofstetter differences in his diploma thesis [Hof04] three types of skill levels (low-level, medium-level and high-level attacker). This paper distinguishes only two types of skill levels, *Script Kiddies* and *Advanced Blackhats*. The reason for this decision is the point that the latter categories are much more easier to separate.

Script Kiddies

Script Kiddies or *Low-level Attackers* are persons who have no advanced computer or networking skills. However, they are able to use scripts and programs, developed by others, to exploit certain security holes. The scripts and programs they use allow them to launch attacks toward computer systems. Examples for programs they use are WinNuke[1] and Back Orifice[2]. [Spi00a] [Spi00b] [Spi00c]

The scene of the script kiddies had an enormous growth over the past few years. There are groups of organized script kiddies who share knowledge and programs to attack new systems and exploit new security holes.

It has been shown, that often boredom, curiosity or just 'playing war' are the impulsion of that people.

Advanced Blackhats

Advanced Blackhats or *High-level Attackers* are skilled programmers who exploit vulnerabilities in computer systems. The procedure of attacking a system is far more elaborate than that from script kiddies. Advanced blackhats can be distinguished from script kiddies by the fact that they attack only chosen targets and act much more subtle. As of the wider knowledge, it does not make them less dangerous. Attackers often utilize buffer- or heap overflows to get access to systems.

In contrast to the script kiddies tend advanced blackhats to make little 'noise'. They are good at covering their tracks. Therefore it is difficult to get information about attacks of advanced blackhats. Many organizations will not even notice that hey have been attacked.

Advanced blackhats are usually financially or nationally motivated. This means that they are state- or company-sponsored spies.

Handy Targets

The differentiation between script kiddies and advanced blackhats is necessary to present them targets that fit to their totally different attack strategies.

Script kiddies who use automated scripts to attack computer systems are quite easily to detect. If they launch attacks that probe whole networks, every intrusion detection system alarm should go off. There exist collections of known attack signatures for the tools they use. The deception has not to be really sophisticated. Attackers of that type tend to be less advanced, but they are so numerous that they are not less dangerous. To attract script kiddies and learn new attack tools, the best solution is to simulate a huge network with many machines. These type of attacker will most likely rush up to that honeynet.

[1]http://www.users.nac.net/splat/winnuke/
[2]http://www.cultdeadcow.com/tools/bo.html

Advanced blackhats evaluate the best target and launch attacks that fit precisely. They are difficult to trick. Advanced blackhats will most likely notice, if they are not operating on a production system. Therefore it might be best to place a copy of a production system as a high-interaction honeypot on the network. This will trick these attackers and the compromised system can be examined after the incident.

6.5 Automated Attack Response

The *'active'* in *active honeypot* stands for an automated response to attackers. This means that the systems reacts to connections on the basis of the gained information from the incoming network packets. There are many different possible reactions. Not all of them fit to every attack or every kind of system. It is necessary to have some policies that define attack situations and the response that suits. This is important because system administrators do not have enough time to react to every malicious connection that occurs.

There are several problems with active responses to incidents. First, the honeypot alerts the attacker by actively querying his system. An attacker who is watching his own system detects all scans from the honeypot. This decreases the value of the honeypot.

A more serious problem is the potential damage that can be caused by active response mechanisms. Since attackers often misuse third-party systems as a starting point for their attacks, innocent people may be harmed. In this case the honeypot owner may be liable for all damages that were caused by the deception system. Especially if the attackers intrusion attempt originates from a computer in a hospital or military network it can be really problematic. A response, like port scanning, from an active honeypot may damage or take down the remote system.

Another concern is that the attackers system may have counter measures of its own. Both systems may be programmed to counterscan any attacker. The attacker scans the honeypot, which counterscans the attacker. The attackers system does itself a counterscan towards the honeypot. This results in a loop of scans.

Whereas counter intelligence might be justifiable are counter offenses definitely not allowed. Counter offense is a technique that can only be used in the own network. However, they are also in a known environment extremely dangerous. Counter measures, as cutting off network segments, are not applicable if there are services that have to run all day long.

It lies in the hands of the administrator of the honeypot to balance the value of active intelligence measures versus the involved risks.

6.6 Hunting Down Honeypots

Most honeypot systems, whether they are low- or high-interaction ones, share a common trait - their value diminishes upon detection. If an attacker identifies a system as a honeypot he either leaves it or feeds it with bogus information. Therefore it is very important to hide the true identity of such a deception system. [Spi04b]

6.6.1 System Load Monitoring

Honeypots pretend to be valuable production systems. They must contain a lot of *honey*. That honey can be a system name, valuable data, many user accounts, a high system load or many services. It is no problem to place some fake research or accounting data on the honeypot. A small script is able to create thousands of fake user accounts. This steps turn a simple honeypot into a system that looks important to an attacker.

The low amount of captured data gives a honeypot a competitive edge over an intrusion detection system. Whereas an IDS may register thousands of connections a day, a honeypot may only detect a few. This facilitates the investigation of the recorded activities. A more or less intelligent intruder will become suspicious, if a system, that seems to contain valuable data or many user accounts, has a very low system load. A computer with thousands of user accounts that registers no logins or a system with terrabites of research data that logs no data transfer is most likely a deception system.

To solve this issue the honeypot has to generate some traffic on its own, without stripping bandwidth from the production systems. Honeynets build often an own network segment. If they generate traffic inside that subnet, they will not disturb production systems. The generated traffic can be filtered from the log files of the honeypots to keep the honeypot monitoring simple.

6.6.2 Red Queen Effect

Low-interaction honeypots, that just do network observation, collect enough information from the intruder before they are detected. The value of such systems lies in the monitoring of all connection attempts and therefore they do not need a great effort to hide their real identity. Whereas high-interaction honeypots, that should keep intruders busy as long as possible, render useless after detection. An attacker, that is aware of the presence of a honeypot, is most likely to wipe his tracks and leave the system. Skilled attackers are able to detect any type of honeypot. It is merely a matter of time, how long the deception system can monitor the intruders' activity before it gets discovered.

The *'Red Queen Effect'* is an evolutionary theory. It explains that all individuals have to adapt quickly to the environment, just to survive. The same theory applies for honeypots. As fast as honeypots developed during the past few years, as fast developed techniques to defeat them. Honeypot technology is in an arms race. The blackhat community has a great interest in detecting honeypots, because they fear to be monitored. Although they like to share their tactics and tools among themselves they are afraid to be called to account for their actions. Organizations and governments take great effort to enact laws against cyber crime.

6.6.3 Honeypot Detection Tools

As fast as honeypot technologies develop, as fast increases the number of detection tools. This is software, used by the blackhat community, to check if they probe, attack or operate on a honeypot system.

Honeypot Hunter

One of the more unique examples of a honeypot detection tool is the commercial software *Honeypot Hunter*[3]. This tool is used by spammers to detect fake proxies and spam-catching honeypots. This is achieved by opening a fake mail server on port 25 (SMTP). Honeypot Hunter connects back to that local mail server through the proxy. If the proxy reports that the connection is up, but the mail server receives no connection, the proxy is identified as a honeypot. [OH04b]

Sebek Identification

There exist also noncommercial code which is used to detect high-interaction honeypots like *Sebek.c*, *sebek-find.c* and *unsebek.c*[4]. Sebek is a client/server based application that is used to capture the intruders activity on the honeypot (client) on kernel level and sends it to the server. This hidden remote logging produces an overhead. If an intruder on the honeypot reads one single byte, Sebek has to transfer nearly 100 bytes of data. The generation of many read calls will lead to a congested network and finally dropped packets. If Sebek is installed on a system the average round-trip will grew for example from 0.7 milliseconds to over 4800 milliseconds.

VMWare detection

A lot of tools have been developed to detect virtual honeypots[5] and papers have been published that identify potential issues[6]. *VMWare* is a widespread commercial software for virtual machines that allows to launch multiple instances of different operating systems on a single piece of hardware. The VMWare software maps the physical resources of the underlying operating system to resources of the virtual machines. This allows to simulate whole honeynets on one real machine. [Cla01] It can often be guessed that a system is running on top of VMWare by looking at the MAC addresses of its network interfaces. MAC addresses that are assigned to *VMWare Inc.* are: [OH04a]

```
00-05-69-xx-xx-xx
00-0C-29-xx-xx-xx
00-50-56-xx-xx-xx
```

[3]http://www.send-safe.com/honeypot-hunter.php
[4]http://www.phrack.org/fakes/p62/p62-0x07.txt
[5]http://seclists.org/lists/honeypots/2002/Oct-Dec/0029.html
[6]http://phrack.nl/phrack62

This does not inevitably mean that it is a honeypot, but this might be enough to make an attacker suspicious. Beside from that are most blackhats interested in resources as computing power or network bandwidth and do not want to operate on virtual systems.

6.6.4 Avoiding Detection

The best approach to hide the identity of a honeypot is probably customization. Many tools that reveal the presence of honeypots or virtual emulation services rely on known signatures. These signatures can be directories or binaries at certain locations in the file system, processes with known names, special user accounts and so on.

By customizing the honeypot, these tools will most probably fail in detecting the identity of the honeypot system. To rename binary files or to install the honeypot or VMWare software in another directory than the default one, might be enough to deceive many of these tools. The more the default behavior of the honeypot is modified, the potentially more difficult will it become for attackers to identify it. Advanced users should definitely also modify the source code of the honeypot software as to change how network packets are created. [Spi04b]

6.7 Conclusions

The need for timely and unequivocal identification of attackers is essential for an active honeypot system. An attacker who wants to reconnect to a target should find the same system that he left some hours ago. If the active honeypot system is not able to identify an attacker unequivocally, it is impossible to generate a consistent target. As a consequence, the attacker will notice the strange behavior and leave the system. Unfortunately, the technical basis for such identification has not received much attention to date from the research and development community. In addition, there may be some complicating factors for the implementation of identification in deception systems. However, until research and development resources are committed to investigation of the relevant issues, active honeypots will not be able to generate consistent, adaptive targets. If organizations and companies begin to share incident related information, it will become possible to advance in attacker identification and classification. At the moment there is just not enough information freely available to develop serious identification methods or classification schemes. The *Honeynet Project* is an example of what can be achieved by cooperation in this field. This prove the remarkable results that evolve from their research.

Although the topic of automated attack response is widely discussed, this will never become a part of a serious IT security infrastructure. The main reason is based on the fact that automated attack response can cause much damage to innocent parties. The consequences of such sweeping blows will most probably cause too much damage. Blackhats are very good at hiding themselves in other organizations networks. In order to counter the attacks of intruders it is inevitable to identify them beforehand. This will not be possible in the near future.

As the chapter about the costs and the benefits of honeypots demonstrated, is it too difficult and expensive for most organizations to do an in-depth incident investigation. Therefore it is necessary to automate the incident handling. Advances in that field will most likely boost honeypot diffusion. This task is very complex, what makes it hard to solve. All the same is the main difficulty, that there is not enough information about the blackhat community to automate incident handling.

Every honeypot software that is used out of the box, suffers from a high chance of being identified as a deception system. If the honeypot itself has just the function of a *burglar alarm*, that does not really diminish the functionality. If the deception device is a high-interaction honeypot that should keep attackers busy as long as possible, it is inevitable to do some customization. The effectiveness of honeypots only exists as long as potential attackers are not aware of the presence of such deception systems. Whereas some people argument that honeypot technology must remain secret in order to be effective [Cor03], is it important for this technology to have an open discourse in order to maturate and become an inherent part of IT security solutions.An active honeypot system has to be very flexible. This cannot be achieved with a real honeypot system, because changes to real systems take an immense effort. Therefore the usage of virtual honeypots seems to be the only solution. Using virtual honeypots or using technologies to generate adaptive targets at all, increases the chance that the identity of the honeypot gets revealed sooner. In generally it can be said that any effort to keep attackers longer on the honeypot might result in the revealing of the true identity of the honeypot. Fact is, that the best honeypot is a real production system.

There are many problems related to the usage of active honeypot systems. Most of them are based on the fact that this technology is still in its infancy. It necessary to develop another attitude towards security incidents to push the development of security devices like honeypots. For the IT security it will become vital to conduct an open discussion and to share all information about threats, incidents and attackers.

Chapter 7

Active Honeypot Feasibility

7.1 Introduction

During the work on this master thesis, it turned out that the state-of-the-art in honeypot research was insufficient to experiment with that technology. Therefore the goal had to be adapted towards an active honeypot feasibility study.

This chapter is the actual active honeypot feasibility study. It takes a closer look at the feasibility of adaptive target generation, automated incident handling and automated attack response. All these subjects are investigated with respect to technical, economical and legal feasibility. Educated speculations about future research in active honeypots are made on the basis of the subjects mentioned.

7.2 Automated Incident Handling

Automated incident handling is the main purpose of an active honeypot. It includes adaptive target generation and automated attack response. The automation of the incident handling process will certainly decrease the honeypot operation costs and therefore increase its value.

This section takes a closer look at the feasibility of automated incident handling. It is divided into examination of high- and low-interaction honeypots, to take their varying complexity into consideration.

7.2.1 Technical Feasibility

The technical feasibility is given, as long as it is practicable with todays or future technology.

Low-Interaction Incident Handling

Automated low-interaction incident handling is far more easy to realize, due to the limited information collected by the honeypot. There is very limited, but well defined information about the attacker. The exact definition of the collected information and the applicable reactions is an important condition to automate the incident handling process. Hence there are limited ways to handle an incident on low-interaction level, there are already solutions, which do that in an effective way. Present low-interaction honeypots, like honeyd, provide enough functionality to implement automated incident handling. This leads to the conclusion that automated low-interaction incident handling is possible and does also already exist.

High-Interaction Incident Handling

It requires a lot of knowledge to operate a high-interaction honeypot. In contrast to low-interaction honeypots, the data that can be collected on high-interaction ones is just roughly defined. It depends on the simulated system. Whereas a simulated file server collects data about file access, a simulated Internet gateway does collect data about network connections. This results in a lack of standardized procedures to handle an incident.

Human attackers show a boundless range of behaviors. There is no complete definition of human attackers' activity on the honeypot. Based on the capability of humans to adapt and adapt, there will probably never be one.

The lack of a defined set of information, that can be gained from the honeypot and the extensive behavior of human attackers impedes the development of a general solution to incident handling. Every single high-interaction honeypot needs his own incident investigation process for every single attacker. Due to the high complexity of this challenge, there is no usable solution yet.

7.2.2 Economical Feasibility

Incident handling is a very expensive step from the economical point of view. It takes big effort to do such an investigation carefully. Nevertheless incident handling is a very important process to move ahead in IT security. This section takes a closer look at the expenses and savings from the implementation of an automated incident handling process.

Low-Interaction Incident Handling

Technology and knowledge of low-interaction incident handling do already exist. Developers can use current tools to build a honeypot that performs an automated incident handling on low-interaction level. There are already some rudimentary solutions to that problem. From the economical point of view, there is no doubt that an automated low-interaction handling reduces the honeypot maintenance costs and therefore the overall security expenditures as in chapter 4.3.1.

High-Interaction Incident Handling

As there is no experience in automated high-interaction incident handling, it would take huge effort to develop such a process. Most likely, the lack of knowledge in this field does not allow to start the development right away. The slow propagation of honeypot technology and the low public attention to that technology in general, will most probably not justify the huge expenses to start a system with automated incident handling from scratch.

7.2.3 Legal Feasibility

As the incident handling is not only allowed, but also a must, there is no problem with automated incident handling in general.

7.3 Adaptive Target Generation

Although adaptive target generation is part of the automated incident handling process, it is nevertheless important to take a closer look at the applicability of this step. This process is all the same an essential condition for automated incident handling or automated attack response.

7.3.1 Technical Feasibility

A system that generates adaptive targets tends to become very complex. The interaction with human attackers complicates the whole process of target generation, because human beings show a wide range of different and often unpredictable behaviors. The complexity of a system that generates a unique target for every single attacker would certainly exceed todays technological potential. This statement is based on the fact, that todays network technology does not allow to unequivocally separate the attackers. A solution to this problem is the classification of the attackers. This allows to reduce the complexity of the system. Unfortunately, classification requires in first place the identification of the attacker.

The technical equipment needed to get information from the attacker is already available. There is a wide range of devices and tools that log and process data on what is going on in a networked environment. Routers, switches, packet filters, intrusion detection systems and operating systems themselves log all kind of data on what is happening. This data can be processed to extract valuable information. There are current projects that analyze the collected data and draw conclusions about the attacker and his intentions. *Counterpane*[1] is just one example of a company that provides such services. This demonstrates that it is possible to identify and classify attackers based on their behavior and proceeding. Even if all these projects cannot do it without any human interaction, they are a good starting point for future research.

The main problem is therefore not just the identification and classification, but the automation of these steps. An adaptive target has to be generated within seconds or minutes to present a consistent system to the attacker. The short time period between the scan and the compromise of the system, makes it impossible to have any human interaction in the process of attacker classification and target generation. There are no publications available about such a fully automated attacker identification and classification process. The fact that there are no known research projects either which address this problem, leads to the assumption that there is no solution to that problem at the moment. Artificial intelligence could possibly solve the problem of handling human interaction on its own. Neural networks might be able to interact with humans in an adequate manner. Unfortunately, this field of research is still in its infancy.

An additional problem arises due to the fact, that the identification of the attacker has to be finished before he breaks into the system. Identification and classification of attackers have to be done before the target generation process can start. Before the attacker operates on the generated target, very few information about him is available. It is doubtable, if it is possible to identify an attacker due to the lack of valuable information about him.

7.3.2 Economical Feasibility

Due to problems, related to the adaptive target generation, which are not yet solved, it is questionable if such a system would justify upcoming investments.

[1]http://www.counterpane.com

7.3.3 Legal Feasibility

Legal problems of honeypots arise due to the liability, if an attacker misuses the honeypot to attack a third party host [Spi04a]. The placement of a honeypot inside the own network is harmless. As the legal feasibility for honeypots, which are targets, is given, there is no problem with adaptive target generation.

7.4 Automated Attack Response

To automatically respond to an attack is one of the main goals of security professionals. The ability to *strike back* would give many afflicted security professionals and system administrators a certain satisfaction.

7.4.1 Technical Feasibility

There are many different types of responses to an attack. The rewriting of firewall rules allows to drop malicious network packets. A counter scan gives more in-depth information about the attacker and a counter offense might push him back.

From the technical point of view, an automated attack response is feasible as demonstrated in chapter 5.4.4. The problem arises from the consequences of such responses. Particularly if the attacker operates from a system that also has an automated attack response, both systems will most likely end up in a vicious circle. Counter scan will follow counter scan until one of the systems breaks down.

7.4.2 Legal Feasibility

Counter measures and counter offenses in particular are not harmless at all from a legal point of view.

Although a port scan is not against the law, it can cause a lot of troubles. Attackers often operate from third party systems to hide their identity. Any countermeasures that target the attacker himself are most likely to hit a third party. If they detect scans or other counter measure from the honeypot, they will most likely react to that *offense*. If the attacker operates from government or a military systems in particular, there is a high chance that the honeypot owner will be the target of a criminal prosecution.

Therefore automated attack response has to be treated carefully. As long as it targets the own network, there is no problem with that. However, any third party that gets hit by a sweeping blow, that was originally directed towards an attacker operating from their facilities, will most certainly show some kind of reaction. This will have legal consequences or at least result in bad reputation. Therefore it is not likely that automated attack response will become widespread in the near future.

7.5 Conclusions

Although active honeypots have a lot of advantages that will bring additional security into existing IT infrastructures, is it not likely that this technology will diffuse into most networks within the next few years.

It needs a huge effort to overcome the current drawbacks of this technology. It is most likely that it will not be possible to solve the problem of attacker identification in an adequate manner, because todays network technology does not allow to do so.

The fact, that an active honeypot is not feasible at the moment does not imply, that there should not be ongoing research endeavor. The drawbacks of active honeypots are at the same time difficulties that other parties have to struggle with. To call attackers to account, the identification process has to be done beforehand. To improve IT security, statistics of the IT incident costs must be made available. The difficulties of active honeypots will become solved. It is merely a matter of time, until the technology and the knowhow to develop such systems are available.

Chapter 8

Summary, Conclusions and Future Work

8.1 Introduction

This chapter is a summary of this master thesis. It includes a definition of the term active honeypot as used in this thesis and describes the methodical approach. The feasibility of adaptive target generation, automated incident handling and automated attack response are summed up. Examples of more simple active components, than the ones from chapter five, are presented. The main difficulties of this work are described. Prerequisites for Future research close this last chapter. This part of the master thesis addresses security professionals and decision-makers who want to get a quick overview of active honeypots, its main problems and future work.

8.2 Definition of Active Honeypot

The term *active honeypot* describes an advanced honeypot system. Unlike traditional honeypots, that undergo passively all attack attempts, active honeypot systems actively react to them.

The active part in this type of honeypot system generates an adaptive target that fits the attacker. An active honeypot system foremost gathers as much information about a potential attacker as possible. On the basis of the information gained, it generates a suitable target. This adaptive generated target is tailored to the needs and abilities of the attacker. This facilitates to get more in-depth information from intruders.

8.3 Methodical Approach

The feasibility study is methodically based on an analysis of existing literature, interviews with security professionals and experiments with traditional honeypots.

8.4 Feasibility

This feasibility study is based on the three main components of an active honeypot:

- Automated incident handling

- Adaptive target generation
- Automated attack response

8.4.1 Automated Incident Handling

The automation of the incident handling process is necessary to reduce the immense honeypot maintenance costs. It is also a prerequisite to advance in blackhat research.

Automatic incident handling on a low-interaction honeypot can be done with justifiable effort. There are currently a lot of tools that allow incident handling with minimal human interaction. Although there is currently no tool available that does the whole incident handling on its own, it is merely a matter of time until an active low-interaction honeypot improves network security.

Whereas the challenge of an automated low-interaction handling process is mostly solved, the automation of high-interaction incident handling faces a lot more obstacles. Due to the limited interaction and information that a low-interaction honeypot provides, its incident handling process is well defined. High-interaction honeypots provide a high level of interaction to attackers. Human attackers show a boundless range of behaviors. This makes it impossible to define a general incident handling process. Every single attack has to be managed on its own. Observing the adaptability of mankind, there will probably never be a general and therefore automated incident handling.

8.4.2 Adaptive Target Generation

A system that generates adaptive targets tends to become overly complex. The interaction with human attackers in particular complicates the whole process of target generation. To get a grip on that challenge it is essential to identify attackers and to group them. This allows to define several targets that suit a certain group of attackers.

Unfortunately, there is no indication that the classification of attackers can be done in an adequate manner. Alone the fact, that the target generation process has to be done before the attacker achieves to break into the target system, makes the automated attacker classification impossible. Maybe future network technologies move ahead in this field, but right now there is no solution to that problem.

8.4.3 Automated Attack Response

The ability to *strike back* would give many security professionals and system administrators afflicted a certain satisfaction. There are several possibilities to respond to attacks. Counter scans give more information about the attacker. The lock of certain ports on a switch separates endangered network segments and a counter offense may eliminate the attacker. There are a lot of tools available to automate this process. Unfortunately, any automated attack response resembles a sweeping blow, because attackers hide themselves within the facilities of third parties.

The drawback of automated attack response is therefore not the technical, but the legal feasibility. Every probe or attack on a foreign host will certainly have legal consequences or at least result in a bad reputation. Therefore, the automation of the attack response is not applicable. It needs an administrator who weighs up the counter measure and the possible consequences.

8.5 Types of Active Honeypots

The definition for the term *active honeypot* that is used in this master thesis is obviously not the sole definition. There are many other technical approaches on how the active component of a honeypot might look like.

8.5.1 System Snapshot

Honeypots can gather a lot of useful data. Network connections, transferred data to and from the honeypot and keystrokes that were pressed are some examples of information that can be gained by a honeypot. There is a lot more information about the attacker available. Unfortunately, blackhats usually are very carefully. They wipe out their tracks before they leave a captured system. Therefore it is useful to have a snapshot of the system before the attacker leaves. The active component inside a honeypot could take a system snapshot on its own. This has to happen at defined moments. If the blackhat installs new attack tools on the system or stores data for further distribution, a trigger activates the generation of a system snapshot. The generated snapshot can be carefully analyzed without ruffle at a later time to get more in-depth information about the attacker. As this investigation does not take place on the honeypot itself, the attacker will not become suspicious.

8.5.2 Rewriting Firewall Rules

Whereas a honeypot might currently not be able to identify an attacker, it is all the same able to identify malicious connections. If a connection to a honeypot is classified as malicious, the active component rewrites the firewall rules to drop all suspicious connections. An attacker is therefore no longer able to attack other systems inside the network from the same host.

Although this proceeding sounds simple it is not harmless at all. The automatic rewriting of firewall rules can lead to a situation where normal traffic gets blocked too. If not carefully implemented and observed, connections from normal users may be blocked. A company that offers web services cannot afford to block paying customers.

8.6 Problems of this Work

This section covers problems which came up when writing this master thesis. Some of these problems have a rather subjective character.

8.6.1 Information Vacuum

If a building catches fire, insurance and police force will make an in-depth investigation to find the reason for the combustion. This investigation process is responsible for the huge amount of information about fire incidents that exists today. IT incidents in contrary are still threaten stepmotherly. A company that suffered from a break-in into their IT infrastructure will conceal the whole incident, since they fear to loose customers or market value. In contrast to real burglaries, IT security incidents seem to be a sign of incompetence.

The major problem of this work was to find reliable and serious information about security incidents. Particularly during the interviews with different security professionals, I was often told not to publish the interview. This secretiveness prevents the

IT industry from developing a good working incident management. Every company or organization has to create their own incident handling policy and they are based only on the information they have about their own incidents. I am sure that most IT incidents have no consequences, nor for the attacker, nor for the IT security department. It is hard to improve security, if you do not learn from mistakes.

There are no publications about *active honeypots* available. Probably there is some research or literature about active honeypots. Unfortunately, this information is not publicly available.

To move on in honeypot- and especially in active honeypot research, it is vital to have more information about incidents and the blackhat community. Every public incident available I found, was published by universities. This shows that these institutions do accurate research in IT security.

8.6.2 Honeypot Acceptance

The main problem of honeypot technology is the lack of acceptance as it turned out during my work on this topic. Although most security professionals I spoke to knew about honeypots, only a few of them maintained such a system. Again only few of them were able to maintain their honeypot in a manner, that it produced results, which could be used to improve the security of their network environment. The absence of honeypot installations might lead to the assumption that this technology needs more time to evolve. It is questionable, if it gets the time it needs to mature. It is written in the *IEEE Security&Privacy issue vol.2, nbr.6 of Nov/Dec 2004*:

> *"... A final farewell - Regular readers of The Honeynet Files department will notice its absence in upcoming issues. The department has been discontinued to allocate additional space [it takes about 2 pages] to content of interest a wider audience ... "*

. Honeypots and particularly active honeypots need more time to develop. This technology has huge advantages and is able to improve IT security.

8.7 Further Research Prerequisites

Nowadays honeypots with generated adaptive targets are merely some generic concept. A lot of surrounding work is needed, before active honeypots can be built and used in real life. Particularly the problems that were mentioned in the last chapter need to be solved first in order to proceed. Future work that has to be done includes:

Low-interaction active honeypots: Whereas this paper was about active honeypots that belong to the class of high-interaction honeypots, there are also low-interaction active honeypots. Due to the limited complexity of these systems, is it interesting to implement an *active* component. The automation of low-interaction incident handling will give new insights in incident handling.

Incident statistics: In order to go ahead in honeypot technology, it is necessary to get more information about the blackhat community and about the consequences of IT incidents. Although there are several loose statistics about the consequences of IT incidents it is necessary to bring them together and correlate them to get an overview about the current IT security situation.

Attacker identification: The identification of the attacker is an unsolved problem. Although there were some incidents which led to the disclosure of the attackers' identity, there are no general concepts about how to do it.

Target identification: A complete simulation of targets on a single honeypot is unfeasible and too expensive in terms of man-hours. A more straight forward way would be to install a honeynet with single honeypots that simulate different targets. To run such a system for a long time will result in information about what targets the blackhat community is looking for.

Incident handling concept: To ease the use of honeypots at all, a concept on how to perform an in-depth incident investigation would be a great help for many security professionals and administrators. This could lead to an increase in interest towards deception systems.

Conceal deception systems: The blackhat community takes big effort in developing tools and tactics to identify and defeat deception systems. On examining their proceedings, the honeypots can be hardened to maintain the quality of the results from deception systems.

Acknowledgments

I want to thank my Supervisor Dr. Reinhard Riedl for giving me the opportunity to write this master thesis. I am thankful for the freedom he granted me and all the valuable input and suggestions.

I am grateful for the suggestions that were brought up during the interviews with the following people:

- Dr. Rolf Oppliger, eSECURITY
- Maria von Kaenel, SWITCH
- Urs Zumstein, GENESIS COMMUNICATION
- Paul Beck, Informatikdienste der Universität Zürich
- Soenke Gold, Credit Suisse Zurich

Thanks to Dr. Etzard Stolte and the Functional Genomics Center Zurich for providing infrastructure for the experiments with honeypots.

Thank you Tamara and Rolf for proof-reading this thesis.

Task Description

Aktive Honeypots

AUFGABENBESCHREIBUNG

1. Kurze Einführung in die allgemeine Thematik IDS (2/15)
   - Ziele und Aufgaben, Realisierungskonzepte, Architekturen, State-ofthe-Art und aktuelle Herausforderungen
   - Rolle, State-of-the-Art und aktuelle Bedeutung von Honeypots und Motivation für aktive Honeypots
2. Identifikation und Diskussion der wesentlichen Forschungsfragen fr̈ aktive Honeypots (4/15)
   - Generisches Grundkonzept für aktive Honeypots und Überblick über den State-of-the-Art in der Theorie und in der Praxis
   - Überblick über realistische, generische Einsatzszenarien und vertiefte Diskussion von Stärken, Schwächen und Herausforderungen des Einsatzes von Honeypots im jeweiligen Szenario
   - Diskussion von fiktiven und/oder realen Praxisbeispielen
   - Überblick über die zentralen Probleme, Teilformalisierung der Probleme und Vergleich mit bekannten Problemen in anderen Gebieten unter anderem für die Identifikation von Angreifern, die Klassifikation von Angreifern, das Design von Reaktionsmustern für bestimmte Angreifer / Angriffstypen, das Design eines teilautomatisierten Prozesses für das Vorfallsmanagement und die Anpassung der Reaktionsmuster, Bewertungskonzept für die Nützlichkeit eines Honeypots, etc.
   - Diskussion der besonderen Herausforderungen der einzelnen Probleme und nutzbarer Lösungsansätze, respektive vorhandener Lösungen
3. Konzeptioneller Entwurf von Teillösungen (4/15 + fakultativ)
   - Selektion von Teilproblemen, Identifikation der Defizite bisheriger Lösungen (soweit überhaupt vorhanden) und Beschreibung der Anforderungen an die Lösungen
   - Konzeptioneller Lösungsentwurf und teilformale Dokumentation
   - Validierung der Lösungen, vertiefte Diskussion ihrer Eigenschaften und Diskussion der Bedeutung der Resultate für den State-of-the-Art
4. Durchführung von praktischen Experimenten (1/3 + fakultativ)
   - Definition der Forschungsziele, die mit den praktischen Experimenten verfolgt werden sollen

- Entwurf, Durchführung, Dokumentation, Evaluation und Interpretation der Experimente
- Schlussfolgerung für die Theorie und Identifikation zukünftiger Forschungsfragen und Forschungsvorhaben

Hinweise:
Die Arbeit sollte unter anderem auf den Diplomarbeiten von Jean-Luc Besson und Ralf Hofstetter aufbauen.

Mit freundlichen Grüssen

(Reinhard Riedl)

Bibliography

[ACM01] Julia Allen, Alan Christie, and John McHugh. Intrusion detection: Implementation and operational issues. *Software Technology Support Center*, 2001. http://www.stsc.hill.af.mil/crosstalk/2001/01/mchugh.html, accessed August 16, 2004.

[Bei03] Marcel Beil. Return on security investments - noch nie war er so wertvoll. *PSI Index 2003, Portraets Schweizer Internet- und Software-Dienstleister*, February 2003.

[Bes03] Jean-Luc Besson. Next generation intrusion detection and prevention for complex environments. Master's thesis, University of Zurich, December 2003.

[Bla] Bob Blakley. Return on security investment: An imprecise but necessary calculation.

[BP02] Reto Baumann and Christian Plattner. White paper: Honeypots, March 2002.

[Che] Bill Cheswick. An evening with bedferd in which a cracker is lured, endured, and studied.

[Chu04] Anton Chuvakin. Issues discovering compromised machines. *Security Focus*, Infocus 1808, October 2004. http://www.securityfocus.com/infocus/1808, accessed December 8, 2004.

[Cla01] Michael Clark. Virtual honeynets. *Security Focus*, Infocus 1506, November 2001. http://www.securityfocus.com/infocus/1506, accessed December 1, 2004.

[Cla03] Heather Clancy. Dipping into honeypots, November 2003.

[Cor03] Joseph Corey. Local honeypot identification. *Phrack Inc.*, 0x0b, 2003. http://www.phrack.org/fakes/p62/p62-0x07.txt, accessed November 16, 2004.

[Dit02] David A. Dittrich. Developing an effective incident cost analysis mechanism. *Infocus 1592*, June 2002. http://www.securityfocus.com/infocus/1592, accessed October 29, 2004.

[Goe] Christian Goetz. Intrusion detection systeme im vergleich.

[Gra00] Robert Graham. Faq: Network intrusion detection systems, March 2000.

[GSS03] Simson Garfinkel, Gene Spafford, and Alan Schwartz. *Practical Unix and Internet Security.* O'Reilly and Associates, 1005 Gravenstein Highway North, Sebastopol, CA 95472, February 2003.

[Hof04] Ralf Hofstetter. Honeypots mit dynamisch erzeugten angriffszielen zur taeuschung von potentiellen angreifern. Master's thesis, University of Zurich, February 2004.

[Kar] Eric Karofsky. Executive summary: Insight into return on security investment.

[KKLM04] Miyoung Kim, Misun Kim, Hyewon K. Lee, and Youngsong Mun. Design of active honeypot system. In *Lecture Notes in Computer Science*, volume 3043. Springer-Verlag Berlin Heidelberg, 2004.

[LLO$^+$03] John Levine, Richard LaBella, Henry Owen, Didier Contis, and Brian Culver. The use of honeynets to detect exploited systems across large enterprise networks, June 2003.

[Mar01] Wiliam W. Martin. Honey pots and honey nets - security through deception. *SANS Institute*, May 2001. http://www.sans.org/rr/papers/4/41.pdf, accessed August 16, 2004.

[NP04] NWC and PM. Anomalie-detection-geraete. *Network Computing*, April 2004.

[OH04a] Laurent Oudot and Thorston Hoz. Defeating honeypots: Network issues, part1. *Security Focus*, Infocus 1803, September 2004. http://www.securityfocus.com/infocus/1803, accessed December 1, 2004.

[OH04b] Laurent Oudot and Thorston Hoz. Defeating honeypots: Network issues, part2. *Security Focus*, Infocus 1805, October 2004. http://www.securityfocus.com/infocus/1805, accessed December 1, 2004.

[Oud03a] Laurent Oudot. Fighting internet worms with honeypots. *Security Focus*, Infocus 1740, October 2003. http://www.securityfocus.com/infocus/1740, accessed August 16, 2004.

[Oud03b] Laurent Oudot. Fighting spammers with honeypots: Part 2. *Security Focus*, Infocus 1748, November 2003. http://www.securityfocus.com/infocus/1748, accessed August 16, 2004.

[Pro00a] Honeynet Project. Know your enemy: A forensic analysis, May 2000.

[Pro00b] Honeynet Project. Know your enemy: Motives, June 2000.

[Pro01] Honeynet Project. Know your enemy: Statistics, July 2001.

[Pro03a] Honeynet Project. Know your enemy: Defining virtual honeynets, January 2003.

[Pro03b] Honeynet Project. Know your enemy: Genii honeynets. easier to deploy, harder to detect, safer to maintain., November 2003. http://www.honeynet.org/papers/gen2, accessed November 16, 2004.

[Pro03c] Honeynet Project. Know your enemy: Honeynets. what a honeynet is, its value, how it works, and risk/issues involved., November 2003. http://www.honeynet.org/papers/honeynet, accessed November 16, 2004.

[RCH99] Virginia Rezmierski, Adriana Carroll, and Jamie Hine. Incident cost analysis and modeling project, i-camp ii. Technical report, University of Michigan, 1999.

[RCH00] Virginia Rezmierski, Adriana Carroll, and Jamie Hine. A study on incident costs and frequencies. *;login: The Magazine of USENIX and SAGE*, August 2000.

[RDFZ97] Virginia Rezmierski, Stephen Deering, Amy Fazio, and Scott Ziobro. Incident cost analysis and modeling project. Technical report, University of Michigan, 1997.

[Sch01] Bruce Schneier. *Secrets and Lies: Digital Security in a Networked World.* John Wiley and Sons, Inc., 605 Third Avenue, New York, NY 10158-0012, 2001.

[Shu04] Faiz Ahmad Shuja. Honeynet, March 2004. http://www.honeynet.org.pk/honeynet/, accessed November 16, 2004.

[Spi] Lance Spitzner. L. spitzners network whitepapers. http://www.enteract.com/ lspitz/, accessed August 16, 2004.

[Spi00a] Lance Spitzner. Know your enemy i: The tools and methodologies of the script kiddie, February 2000.

[Spi00b] Lance Spitzner. Know your enemy ii, February 2000.

[Spi00c] Lance Spitzner. Know your enemy iii, February 2000.

[Spi03a] Lance Spitzner. Fighting spammers with honeypots: Part 1. *Security Focus*, Infocus 1747, November 2003. http://www.securityfocus.com/infocus/1747, accessed August 16, 2004.

[Spi03b] Lance Spitzner. Honeypots: Definitions and value of honeypots. *Tracking Hackers*, May 2003. http://www.tracking-hackers.com/papers/honeypots.html, accessed August 16, 2004.

[Spi03c] Lance Spitzner. Honeypots: Simple, cost-effective detection. *Security Focus*, Infocus 1690, April 2003. http://www.securityfocus.com/infocus/1690; accessed August 16, 2004.

[Spi03d] Lance Spitzner. *Honeypots: Tracking Hackers.* Addison Wesley, 75 Arlington Street, Boston, MA 02116, October 2003.

[Spi03e] Lance Spitzner. Honeytokens: The other honeypot. *Security Focus*, Infocus 1713, July 2003. http://www.securityfocus.com/infocus/1713, accessed August 16, 2004.

[Spi03f] Lance Spitzner. Open source honeypots: Learning with honeyd. *Security Focus*, Infocus 1659, January 2003. http://www.securityfocus.com/infocus/1659, accessed August 16, 2004.

[Spi03g] Lance Spitzner. Open source honeypots, part two: Deploying honeyd in the wild. *Security Focus*, Infocus 1675, March 2003. http://www.securityfocus.com/infocus/1675, accessed August 16, 2004.

[Spi03h] Lance Spitzner. Strategies and issues: Honeypots - sticking it to hackers. *Networkmagazine*, April 2003. http://www.networkmagazine.com, accessed August 16, 2004.

[Spi03i] Lance Spitzner. The value of honeypots. *informIT*, 10. January 2003, January 2003. http://www.informit.com/articles/article.asp?p=30489&seqNum=1, accessed November 23, 2004.

[Spi04a] Lance Spitzner. Honeypots: Are they illegal? *Security Focus*, Infocus 1703, January 2004. http://www.securityfocus.com/infocus/1703, accessed December 16, 2004.

[Spi04b] Lance Spitzner. Problems and challenges with honeypots. *Security Focus*, Infocus 1757, January 2004. http://www.securityfocus.com/infocus/1757, accessed August 16, 2004.

[Str] Stefan Strobel. Der wandel von intrusion detection zu intrusion prevention.

[Tan01] Matthew Tanase. The future of ids. *Security Focus*, Infocus 1518, December 2001. http://www.securityfocus.com/infocus/1518; accessed August 16, 2004.

[Tec] Recourse Technologies. The evolution of deception technologies as a means for network defense.

[ZCC00] Elizabeth D. Zwicky, Simon Cooper, and D. Brent Chapman. *Building Internet Firewalls*. O'Reilly and Associates, 101 Morris Street, Sebastopol, CA 95472, June 2000.