White Paper: Honeypots

Reto Baumann, http://www.rbaumann.net
Christian Plattner, http://www.christianplattner.net

February 26, 2002

Contents

1 Abstract

A honeypot is used in the area of computer and Internet security. It is a resource which is intended to be attacked and compromised to gain more information about the attacker and the used tools. It can also be deployed to attract and divert an attacker from their real targets. One goal of this paper is to show the possibilities of honeypots and their use in a research as well as productive environment.

Compared to an intrusion detection system, honeypots have the big advantage that they do not generate false alerts as each observed traffic is suspicious, because no productive components are running on the system. This fact enables the system to log every byte that flows through the network to and from the honeypot, and to correlate this data with other sources to draw a picture of an attack and the attacker.

This whitepaper consists of two parts. The first part provides an overview and introduction to the different classes of honeypots. The second part presents the main concepts of honeypots in more detail.

The paper ends with a conclusion about the new technology of honeypots.

2 Introduction

Global communication is getting more important every day. At the same time, computer crimes are increasing. Countermeasures are developed to detect or prevent attacks - most of these measures are based on known facts, known attack patterns. As in the military, it is important to know, who your enemy is, what kind of strategy he uses, what tools he utilizes and what he is aiming for. Gathering this kind of information is not easy but important. By knowing attack strategies, countermeasures can be improved and vulnerabilities can be fixed. To gather as much information as possible is one main goal of a honeypot.

Generally, such information gathering should be done silently, without alarming an attacker. All the gathered information leads to an advantage on the defending side and can therefore be used on productive systems to prevent attacks.

A honeypot is primarily an instrument for information gathering and learning. Its primary purpose is not to be an ambush for the blackhat community to catch them in action and to press charges against them. The focus lies on a silent collection of as much information as possible about their attack patterns, used programs, purpose of attack and the blackhat community itself. All this information is used to learn more about the blackhat proceedings and motives, as well as their technical knowledge and abilities. This is just a primary purpose of a honeypot. There are a lot of other possibilities for a honeypot - divert hackers from productive systems or catch a hacker while conducting an attack are just two possible examples.

Honeypots are not the perfect solution for solving or

preventing computer crimes. Honeypots are hard to maintain and they need operators with good knowledge about operating systems and network security. In the right hands, a honeypot can be an effective tool for information gathering. In the wrong, unexperienced hands, a honeypot can become another infiltrated machine and an instrument for the blackhat community.

This white paper will introduce some basic terms as well as possibilities which can be used to implement a working honeypot.

3 Honeypots

This chapter provides a definition of what a honeypot is and looks closer at a honeypot's value. Furthermore, available products are shortly reviewed.

3.1 Honeypot Basics

The buzz word "Honeypot" is spooking around. Different vendors claim that they offer honeypot products, people are arguing about honeypots without having a clear image of what a honeypot is. To clarify this issue, a definition of what is meant when talking about honeypots is provided.

L. Spitzner[1] defines the term honeypot as follows:

> *A honeypot is a resource whose value is being in attacked or compromised. This means, that a honeypot is expected to get probed, attacked and potentially exploited. Honeypots do not fix anything. They provide us with additional, valuable information.*

In this paper, a slightly different definition is proposed:

> *A honeypot is a resource which pretends to be a real target. A honeypot is expected to be attacked or compromised. The main goals are the distraction of an attacker and the gain of information about an attack and the attacker.*

Honeypots do not help directly in increasing a computer network's security. On the contrary, they do attract intruders and can therefore attract some interest from the Blackhat community on the network where the honeypot is located.

3.2 Value of Honeypots

As mentioned, a honeypot can not be used to fix anything. It is even worse, a honeypot can attract more interest in a specific network than one would like. So what can a honeypot provide, what can it be used for?

There are two categories of honeypots[2] - *production honeypots* and *research honeypots*. A production honeypot is used to help migrate risk in an organization while the second category, research, is meant to gather as much information as possible. These honeypots do not add any security value to an organization, but they can help to understand the blackhat community and their attacks as well as to build some better defenses against security threats.

A honeypot is a resource which is intended to get compromised. Every traffic from and to a honeypot is suspicious because no productive systems are located on this resource. In general, every traffic from and to a honeypot is unauthorized activity. All data collected by a honeypot is therefore interesting data. A honeypot will in general not produce an awful lot of logs because no productive systems are running on that machine which makes analyzing this data much easier. Data collected by a honeypot is of high value and can lead to a better understanding and knowledge which in turn can help to increase overall network security. One can also argue that a honeypot can be used for prevention because it can deter attackers from attacking other systems by occupying them long enough and bind their resources. Against most attacks nowadays (which are based on automated scripts) a honeypot does not help deceiving individuals as there are no persons to deceive.

If a honeypot does not get attacked, it is worthless. Honeypots are normally located at a single point and the probability can be quite small that an attacker will "find" the honeypot. A honeypot does also introduce a certain risk - blackhats could get attracted to the whole network or a honeypot may get silently compromised.

3.3 Comparison of available Honeypots

This section provides a short overview of the available products and solutions[3]. Table 1 shows an aggregation of the most important factors.

	ManTrap	Specter	DTK	Custom
Involvement	high	low	middle	low - high
Expandable	√	-	√	√
Open Source	-	-	√	N/A
Freely available	-	-	√	√
Cost of ownership	high	low	middle	high
Logfile support	√	√	√	N/A
Notifications	√	√	√	N/A
# services	unl.	13	unl.	N/A
Configuration	diff	easy	mid	N/A
GUI	√	√	-	N/A

Table 1: Honeypot Comparison Table

Each available honeypot has different strengths. Specter is easy to install and even easier to run due to the nice graphical user interface. Unfortunately, its value is not very high, as no real operating system is provided. But this fact does also help in reducing the risk significantly.

ManTrap, DTK and custom built honeypots are highly customizable. Their value can be very high, as well as their risk. Therefore their cost of ownership, which represents the outlay of the routine maintenance, is higher.

[1] Lance Spitzner is member of a honeypot research group in the United States (http://project.honeynet.net).

[2] As defined by Marty Roesch, creator of Snort

[3] ManTrap (http://www.recourse.com), Specter (http://www.neoworx.com/products/specter/) and DTK (http://www.all.net/dtk)

ManTrap's main advantage over DTK and homegrown honeypots is the provided GUI. It is very comfortable to configure, analyze and manage this commercial solution.

4 Concepts

This chapter presents some basic honeypot and honeynet concepts. It is mainly a theoretical approach without looking at practical aspects. But most of the ideas can be directly used in a custom made honeypot.

4.1 Level of Involvement

One characteristic of a honeypot is its level of involvement. The level of involvement does measure the degree an attacker can interact with the operating system.

4.1.1 Low-Involvement Honeypot

A low-involvement honeypot typically only provides certain fake services. In a basic form, these services could be implemented by having a listener on a specific port. For example a simple *netcat -l -p 80 > /log/honeypot/port_80.log* could be used to listen on port 80 (HTTP) and log all incoming traffic to a logfile. In such a way, all incoming traffic can easily be recognized and stored. With such a simple solution it is not possible to catch communication of complex protocols. An initiated SMTP[4] handshake would not lead to much useful information since no answering service is responding.

On a low-involvement honeypot there is no real operating system that an attacker can operate on. This will minimize the risk significantly because the complexity of an operating system is eliminated. On the other hand, this is also a disadvantage. It is not possible to watch an attacker interacting with the operating system, which could be really interesting. A low-involvement honeypot is like a one-way connection. We only listen, but we do not ask questions ourselves. The role of this approach is very passive.

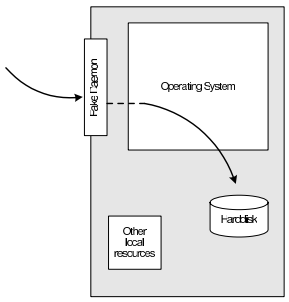

Figure 1: Low-involvement honeypot: A low-involvement honeypot does reduce risk to a minimum through minimizing interaction with the attacker

[4]Simple Mail Transfer Protocol - used to send and receive e-mails.

A low-involvement honeypot can be compared to an passive IDS[5], as both do not alter any traffic or interact with the attacker or the traffic flow. They are used to generate logs and alerts if incoming packets match certain patterns.

4.1.2 Mid-Involvement Honeypot

A mid-involvement honeypot provides more to interact with, but still does not provide a real underlaying operating system. The fake daemons are more sophisticated and have deeper knowledge about the specific services they provide. At the same moment, the risk increases. The probability that the attacker can find a security hole or a vulnerability is getting bigger because the complexity of the honeypot increases. A compromise of this system is still unlikely and certainly no goal as there are no security boundaries and logging mechanisms which are built for this kind of events.

Through the higher level of interaction, more complex attacks are possible and can therefore be logged and analyzed. The attacker gets a better illusion of a real operating system. He has more possibilities to interact and probe the system.

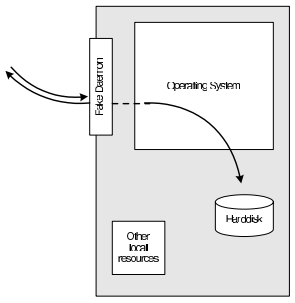

Figure 2: Mid-involvement honeypot: A mid-involvement honeypot does interact with the attacker in a minimal way

Developing a mid-involvement honeypot is complex and time consuming. Special care has to be taken for security checks as all developed fake daemons need to be as secure as possible. The developed versions should not suffer the same security holes as their real counterparts because this is the main reason to substitute these with fake variants. The knowledge for developing such a system is very high as each protocol and service has to be understood in detail.

4.1.3 High-Involvement Honeypot

A high-involvement honeypot has a real underlaying operating system. This leads to a much higher risk as the complexity increases rapidly. At the same time, the possibilities to gather information, the possible attacks as well as the attractiveness increase a lot. One goal of

[5]Intrusion Detection System

a hacker is to gain root and to have access to a machine, which is connected to the Internet 24/7. A high-involvement honeypot does offer such an environment. As soon as a hacker has gained access, his real work and therefore the interesting part begins.

Unfortunately the attacker has to compromise the system to get this level of freedom. He will then have root rights on the system and can do everything at any moment on the compromised system. As per se, this system is no longer secure. Even the whole machine can not be considered as secure. This does not matter if he is in a jail, a sandbox or a VMWare[6] box because there could be ways to get out of these software boundaries.

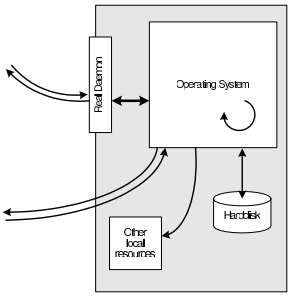

Figure 3: High-involvement honeypot: A high-involvement honeypot has great risk as the attacker can compromise the system and use all its resources.

A high-involvement honeypot is very time consuming. The system should be constantly under surveillance. A honeypot which is not under control is not of much help and can even become a danger or security hole itself. It is very important to limit a honeypot's access to the local intranet, as the honeypot can be used by the blackhats as if it was a real compromised system. Limiting outbound traffic is also an important point to consider, as the danger once a system is fully compromised can be reduced.

By providing a full operating system to the attacker, he has the possibilities to upload and install new files. This is where a high-involvement honeypot can show its strength, as all actions can be recorded and analyzed. Gathering new information about the blackhat community is one main goal of a high-involvement honeypot and legitimates the higher risk.

4.1.4 Overview

Each level of involvement has its advantages and disadvantages. Table 2 summarizes what has been seen so far.

Choosing the lowest as possible level of involvement should reduce the danger and risk as much as possible. The required maintenance time should also be considered when choosing a honeypot and its level of involvement.

[6] A tool to run multiple virtual machines on one physical system. See http://www.vmware.com for more information.

	Low	Mid	High
Degree of involvement	low	mid	high
Real operating system	-	-	✓
Risk	low	mid	high
Information gathering	connections	requests	all
Compromise wished	-	-	✓
Knowledge to run	low	low	high
Knowledge to develop	low	high	mid-high
Maintenance time	low	low	very high

Table 2: Overview of advantages and disadvantages of each level of involvement

4.2 Network Topologies and Honeynets

This chapter will discuss the placement of honeypots in a network as well as a special, more complex version of honeypots - a so called honeynet.

4.2.1 Honeypot Location

A honeypot does not need a certain surrounding environment as it is a standard server with no special needs. A honeypot can be placed anywhere a server could be placed. But certainly, some places are better for certain approaches as others.

A honeypot can be used on the Internet as well as the intranet, based on the needed service. Placing a honeypot on the intranet can be useful if the detection of some bad guys inside a private network is wished. It is especially important to set the internal thrust for a honeypot as low as possible as this system could be compromised, probably without immediate knowledge.

If the main concern is the Internet, a honeypot can be placed at two locations:

- In front of the firewall (Internet)
- DMZ[7]
- Behind the firewall (intranet)

Each approach has its advantages as well as disadvantages. Sometimes it is even impossible to choose freely as placing a server in front of a firewall is simply not possible or not wished.

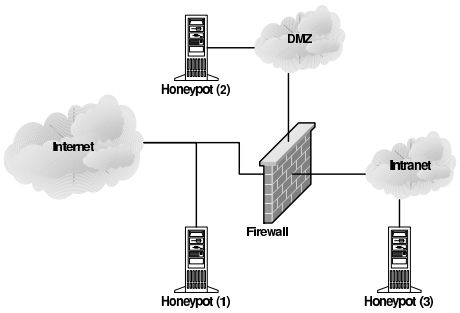

Figure 4: Placement of a honeypot

By placing the honeypot in front of a firewall (see figure 4 honeypot(1), the risk for the internal network does

[7] Demilitarized Zone, a network segment with is only partly accessible from the Internet.

not increase. The danger of having a compromised system behind the firewall is eliminated.

A honeypot will attract and generate a lot of unwished traffic like portscans or attack patterns. By placing a honeypot outside the firewall, such events do not get logged by the firewall and an internal IDS system will not generate alerts. Otherwise, a lot of alerts would be generated on the firewall or IDS.

Probably the biggest advantage is that the firewall or IDS, as well as any other resources, have not to be adjusted as the honeypot is outside the firewall and viewed as any other machine on the external network. Running a honeypot does therefore not increase the dangers for the internal network nor does it introduce new risks.

The disadvantage of placing a honeypot in front of the firewall is that internal attackers can not be located or trapped that easy, especially if the firewall limits outbound traffic and therefore limits the traffic to the honeypot. Placing a honeypot inside a DMZ (figure 4 honeypot(2)) seems a good solution as long as the other systems inside the DMZ can be secured against the honeypot. Most DMZs are not fully accessible as only needed services are allowed to pass the firewall. In such a case, placing the honeypot in front of the firewall should be favored as opening all corresponding ports on the firewall is too time consuming and risky.

A honeypot behind a firewall (figure 4 honeypot(3)) can introduce new security risks to the internal network, especially if the internal network is not secured against the honeypot through additional firewalls. This could be a special problem if the IPs are used for authentication. It is important to distinguish between a setup where the firewall enables access to the honeypot or where access from the Internet is denied. A honeypot does often provide a lot of services. Probably most of them are not used as exported services to the Internet and are therefore not forwarded to the honeypot by the firewall. By placing the honeypot behind a firewall, it is inevitable to adjust the firewall rules if access from the Internet should be permitted. The biggest problem arises as soon as the internal honeypot is compromised by an external attacker. He gains the possibility to access the internal network through the honeypot. This traffic will be unstopped by the firewall as it is regarded as traffic to the honeypot only, which in turn is granted. Securing an internal honeypot is therefore mandatory, especially if it is a high-involvement honeypot. With an internal honeypot it is also possible to detect a misconfigured firewall which forwards unwanted traffic from the Internet to the internal network. The main reason for placing a honeypot behind a firewall could be to detect internal attackers.

The best solution would be to run a honeypot in its own DMZ, therefore with a preliminary firewall. The firewall could be connected directly to the Internet or intranet, depending on the goal. This attempt enables tight control as well as a flexible environment with maximal security.

4.2.2 Honeynets

A honeypot is physically a single machine, probably running multiple virtual operating systems. Controlling outbound traffic is often not possible, as the traffic goes directly onto the network. In this case the only possibility to limit outbound traffic is to use a preliminary firewall. Such a more complex environment is often referenced as honeynet. A typical honeynet consists of multiple honeypots and a firewall (or firewalled-bridge) to limit and log network traffic. An IDS is often used to watch for potential attacks and decode and store network traffic on the preliminary system.

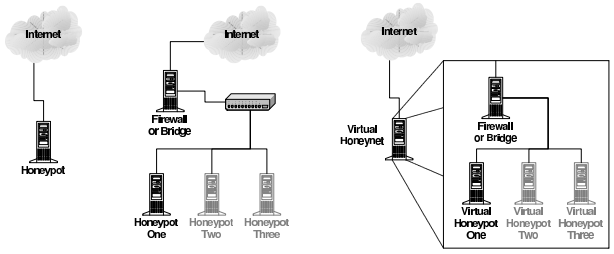

Figure 5: Different honeypot topologies: Simple honeypot, honeynet and a virtual honeynet

By placing a firewall in front of a honeypot (or multiple honeypots) the risk based on the honeypot can be reduced. It is possible to control the network flow, the inbound as well as the outbound connections. Logging of network traffic is much easier as this can be done on one centralized location for all honeypots. The captured data does not have to be placed on the honeypot itself and the risk of detecting this data by an attacker is eliminated.

By introducing new machines to the honeypot itself, more hardware is required. A solution with only one machine is thinkable. By using VMWare, setting up multiple virtual systems on one physical machine is possible. Through this attempt, it is even possible to place the firewall on the same machine as all virtual honeypots however the security of this solution isn't as good as having different physical machines. As soon as the honeynet is a virtual environment, the system could be compromised and the attacker could be able to break out of the virtual machines. Placing a bridge with firewall capabilities in front of a honeypot is much safer as the attacker can not see the bridge. Even attacking the bridge is not possible as the bridge has no IP address and therefore no attack point exists.

Introducing additional hardware also raises the complexity of the environment. Understanding networking and associated tools is important as long as the best security has to be provided.

4.3 Host based Information Gathering

This section will discuss possibilities that offer gain of information about ongoings on a honeypot by installing

information gathering mechanisms on the honeypot itself.

4.3.1 Basic Possibilities

Information gathering facilities can basically be grouped into two categories: facilities that generate streams of information (e.g. all keystrokes of an attacker on a honeypot) and facilities that offer the administrator to "peek" into the system and get information about a certain state of the honeypot (e.g. getting the current processor usage or a list of current processes).

Once again differentiating between systems using virtualization and systems that do not is necessary. If no virtualization is used, there is always the danger that an attacker can discover modifications of the system. He could be able to circumvent them or even make abusive use of them (e.g. by flooding a logging mechanism until it refuses to work properly). Therefore, special care has to be taken when modified elements are visible and also accessible by an attacker.

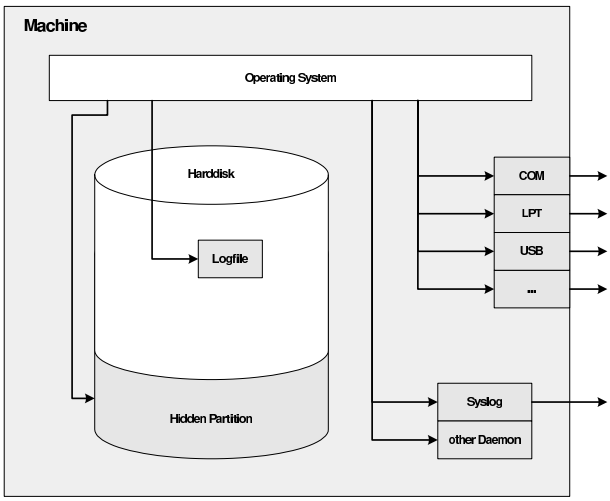

Figure 6: Different attempts of logging information on a honeypot

When using facilities that generate streams of data (and especially if no virtualization is used), the question arises on where to store this data. One possibility is to store this data on the honeypot itself, for instance on a hidden partition in raw format, where it can be studied later. The disadvantage is that this data can not immediately be studied by the administrator (unless one implements some sort of "peeking" mechanism to allow remote viewing of the logged data - but with every added modification to the system chances that an attacker discovers the existence of a honeypot increase). Another problem with locally logged data is that one could run out of reserved log space - resulting in a reduced or even non monitored system. The attacker also could get aware of this log area and try to manipulate it. It does not matter how the attacker manipulates the data, either by deleting or changing, the result in any case is a non-trustable logfile. As a conclusion, it can be stated that local stored log data cannot be trusted, it gives just hints about the possible actions of an attacker.

Information about an attacker which is stored in a safe (meaning the attacker cannot access it), remote place is much more trustable. Of course, the attacker has still the possibility to generate "false" log data (or to stop the log process at all) if he discovers the logging mechanism, but he will not be able to delete events if the data has once left the honeypot. The storage of continuous data to remote places can be accomplished by using serial cables, parallel cables (e.g. in the simplest form by using direct printer output, but a printed log file could be rather hard to analyze), USB or Firewire technology and network interfaces (for instance by sending a stream of UDP packets over an ethernet adaptor to a remote log machine). One could also think of special logging hardware, for instance a specialized PCI card, although such solutions are not readily available.

The implementation of "peeking" mechanisms into a honeypot is regarded as more difficult, since if no virtualization techniques are used and no special monitor hardware is available, some kind of two-way communication is needed. As carrier for the communication the same physical connection methods as stated above could be used.

4.3.2 Microsoft Windows

One could think that the large amount of observed attacks on systems running Microsoft Windows[8] operating systems makes them ideal for a honeypot (and especially for data gathering), but unfortunately the structure of these operating systems makes data gathering (at least host based) rather difficult.

Until today the source code of the Windows operating systems is not freely available, which means that changes to the operating system are very hard (if not impossible) to achieve. The modifications can therefore not be made in a transparent way, logging functionality must be integrated into user space programs which are visible to the attackers. The integration of data gathering mechanisms into loadable kernel drivers could possibly be a better solution.

Logging the list of running processes, periodically watching the event log and checking the integrity of the system files by using MD-5[9] sums seems to be the only relevant amongst the possible actions of a honeypot administrator.

4.3.3 UNIX derivatives

UNIX[10] derivated operating systems offer interesting opportunities for deploying data gathering mechanisms,

[8] Windows is a registered trademark of Microsoft corporation

[9] MD-5 is a message digest algorithm invented by Ronald L. Rivest, it is primarly used to verify data integrity. http://theory.lcs.mit.edu/~rivest/homepage.html

[10] A registered trademark of The Open Group

since all (or at least in many cases most) of their components are available as source code. The presence of available source code comes in very handy, offering the possibility to apply logging mechanisms to the source code, recompile the binaries and copy them to the honeypot. But one has to be careful with this method. Although modified binaries are rather hard to detect for an attacker, it is not impossible. There are ways an advanced attacker could detect changed (often called "patched") system binaries:

- **MD-5 Checksums:** If the attacker has a reference system which he can compare to the honeypot, then he is able to measure the difference between the reference system and the honeypot, for instance by calculating MD-5 checksums over all standard system binaries.
- **Library dependency checks:** Even if the attacker does not know how the exact structure of the original binaries looks, he can still use the unix "ldd" command to watch for strange shared library dependencies. If, for example, the unix binary "grep" (used for text searching) suddenly has a dependency on a function to communicate to the syslog daemon (a program to log system events), the attacker will get suspicious. A possible solution to this problem is the usage of statically linked libraries.
- **Trussing processes**: In UNIX operating systems, the super-user can supervise any process (especially the invoked system calls by a process) with the *truss* or *strace* command. If a binary like the "grep" command suddenly starts communicating with other processes (for example the syslog daemon), the attacker would know that something is "fishy" with the attacked system.

Most attackers install so-called "rootkits" upon the overtake of a machine. These rootkits often include precompiled system binaries, which are then copied over the existing binaries on the compromised system. In consequence of this, the achieved level of data gathering can be reduced or even be set to zero at all. There are basically two ways to circumvent that problem: on one hand as many system binaries as possible can be "patched" (in the hope that the attacker will not overwrite all patched binaries, therefore having at least *some* data about the attacker) or integrating the data gathering directly into the UNIX-kernel, where it is really hard to detect for an attacker.

Modifying UNIX kernels is not as easy as modifying UNIX system binaries, most notably because not all UNIX versions come along with a kernel in source code form. But when the source is available, this can be a very effective way to implement and hide data gathering measures. Of course, this effort can be undone by an advanced attacker by installing a fresh kernel.

Newer versions of UNIX systems use modular kernels, giving the user the possibility to add new functionality to the kernel during run-time. This can be a dangerous feature, since it allows an attacker to add special countermeasures directly into the kernel, e.g. a facility to hide installed files (or processes) by the attacker from other users.

4.4 Network based Information Gathering

Host based information gathering is always located at the host itself and is therefore vulnerable to detection and once detected it can also be disabled. Network based information gathering does not have to be located on the honeypot itself. It can also be implemented in an invisible way, as network traffic only gets analyzed but not manipulated. Network based information gathering is safer as it is harder to detect and quite impossible to disable (if securely implemented).

How can information be gathered?
Suppose we do run a honeynet as in Figure 7. In this scenario we have a preliminary firewall through which all traffic must go. Some information can be gathered by the firewall itself. The firewall can be configured in such a way, that certain packets or all packets are logged. An even better and more flexible solution would be to install an intrusion detection system. With the help of an IDS, detecting suspicious traffic is easy. Most IDS systems can be configured to log interesting sessions to separate files. Some protocols[11] can be decoded and logged in clear text which simplifies forensics a lot.

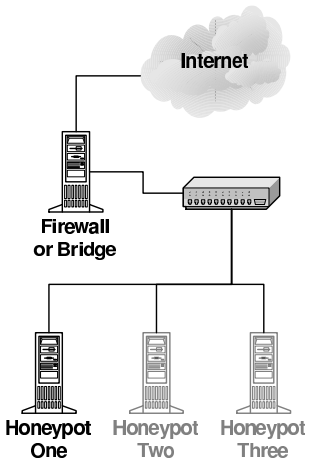

Figure 7: A simple honeynet with a preliminary firewall

By applying special filters, gathering of login and passwords for multiple services as FTP, POP3 and Telnet is easily achieved.

Logging all network traffic can also have a big advantage. By logging all traffic via tcpdump[12] for example, all TCP sessions can be reconstructed at a later time and analyzed in more detail. Analyzing important or interesting sequences can be done quite comfortably with

[11] As Telnet, FTP, SMTP or DNS queries
[12] Tool to collect and inspect network traffic.

Ethereal[13]. Anomalies in TCP/IP packets can be found and possible reasons may be found. It is also important to realize, that everything a hacker does on the compromised system has to go through the firewall and can therefore be logged. Some actions may be placed on the machine itself, but the command to start these actions have to be in the log somewhere. A complete logging can therefore be considered as a storybook of the attack.

As soon as the network traffic is encrypted, everything is getting much harder and complicated. There are multiple attempts of solving the problems with encrypted traffic. Some of them will be reviewed in detail in chapter 4.4.3.

4.4.1 Firewall

This chapter will have a look at the possibilities for network based information gathering at the firewall itself.

A firewall can be configured to log all traffic. This can be very useful as all packets are available at a later time for careful inspection. Logging the packets in a standard file format which is widely used may be a big advantage as multiple tools can be used to analyze and decode the recorded traffic. On Linux based systems, logging the traffic in a tcpdump compatible format seems a good solution as multiple tools for analyzing these streams are available.

A firewall can also be useful to trigger an alert as soon as a packet is destined for the honeynet. These alerts can be further refined to provide a sophisticated alerting system which can tell which service has been attacked.

A firewall is also a good place to run some statistics. All incoming packets can be counted as well as the outgoing ones. It could be interesting to see the emerging network traffic as an attack is under way.

4.4.2 IDS

A network intrusion detection system can be used to trigger alerts as soon as an attack takes place or has taken place. By configuring an IDS in a way that only alerts for running services are triggered a quite reliable warning system can be constructed. Having such a system is very versatile as watching the honeynet all the time is costly. Having an IDS helps minimizing the surveillance. To abandon surveillance completely would be a wrong approach. An IDS is based on signatures or anomalies. Both methods can fail to detect an attack. If the attack is very new there will not be any signature available. As long as there are no anomalies about the attack, an anomaly based IDS won't trigger any alerts. It could also be that an attacker can in what way ever guess a Telnet login and password in the first or second attempt. If the level for triggering an alert on the IDS is too high this login would pass the IDS undetected.

Some IDS also have the possibility to log separate TCP sessions to different files. By decoding them right away, analyzing these sessions is quite comfortable. By generating a file for each session, most interesting sessions can be found easily by looking at the file size as these sessions tend to be larger than an unsuccessful hacking attempt.

[13]Ethereal is a tool to analyze network traffic.

4.4.3 Encrypted Connections

Encrypted connections are considered to be safe and were introduced to stop people from intercepting and sniffing traffic. Unfortunately this is exactly what is intended to be done at a preliminary bridge and can not be done anymore with encrypted connections. By using encrypted connections, listening on the wire and logging is still possible but does not make much sense as the captured packets can not be interpreted as their payload is encrypted.

In most cases, a host based information gathering facility is present and can log all actions. However, as soon as the attacker detects the local logging mechanism he can shut down the required resources. It would be nice to have a secondary mechanism to be prepared for such an event.

Several approaches exist on solving the problem with encrypted connections.

- Man in the middle attack
- Brute force decryption
- Modified sshd daemons
- Customized kernel

Modifying the sshd daemon in a way that all decrypted data is logged to a file is certainly the simplest solution. However this attempt fails as soon as an attacker installs his own ssh daemon. Customizing the kernel seems far more advanced. It would be possible to modify the pty device to log all keystrokes to a file or to the serial port.

Man in the middle attacks are possible, but not as easy. Especially, a way has to be found to detect new ssh daemons listening on new ports and generate an intermediate. Brute force attacks on the encryption are not practicable as they need too much computing resources and time.

4.5 Active Information Gathering

Gathering information on a honeypot is mostly passive. Information is gathered out of the network stream or the bits and bytes on the machine itself. No information is retrieved by inquiring third parties for specific information about a certain identity.

But information gathering does not only have to be passive. It is possible to get more information about a person, an IP address or an attack by querying specific services or machines. This can be very powerful as valuable information can be found. However this attempt is also dangerous as the attacker could take notice and vanish.

The following services are available:

- whois
- fingerprinting network traffic
- portscan
- finger

Some of this methods can be detected by the attacker and are therefore a little bit more dangerous than others.

4.6 Dangers

Running a honeypot or honeynet is not something that should be underestimated - there are some dangers one must be aware of which basically are:

- Unnoticed takeover of the honeypot by an attacker
- Lost control over the honeypot installation
- Damage done to third parties

Unnoticed takeover of a honeypot is surely a bad thing which has to be avoided in any case - otherwise the benefits of a honeypot could be rather questionable. If there is the possibility an attacker can infiltrate the system without being noticed by the operator then there is obviously a flaw in the design of the honeypot monitoring.

The loss of control over the honeypot and the attacker is also a serious issue. A honeypot should be designed in a way that the operator can on one hand safely disrupt any communications from the honeypot with its environment and on the other hand do backups of system states at any time for later investigation. The operator should never rely on any machine correlated with the honeypot - any administrative action must be applicable even if the honeypot is under total control by an attacker. In this context one has to point out the possible dangers of virtual machines. It is never guaranteed that the virtualization software is perfect and the attacker has no way to break out from the virtual machine into the host operating system. The host operating system of a virtualized honeypot is therefore not trustworthy and relying on its proper functioning should be avoided.

Another aspect of loosing control is the deception of the operator by an attacker. If an attacker generates that much traffic and unfiltered log events that the operator is not able to keep track of all the ongoings, the attacker has good chances that the real purpose of his attack is never discovered.

Damage done to third parties can have a very high price. Legal consequences (above all, paying compensation) are not desirable. Assuring by all means that third parties are not caused any harm should have high priority. Being very careful with all the possible aspects of damage done to others is important. This does not only consist of direct attacks (like DDoS[14] attacks or takeovers of machines initiated from a honeypot), but also infringes of copyrights can be taken into this category. A honeypot which is used by an attacker as a MP3 server can lead to ungracious acquaintance with certain worldwide conglomerates.

Generally it can be said that the operator of a honeypot has a heavy responsibility. He must be very attentive, which is not easily achieved, particularly if a honeypot is running for a long time.

4.7 Protecting Third Parties

A goal of every honeypot is getting compromised. As soon as an attacker has invaded a system he can begin using the system for his own purposes. The actions which the attacker will take are unpredictable. Protecting third parties as well as own resources must be of high priority. Protecting own resources is normally easier to achieve, as it can be influenced by the placement of the honeypot itself. By having this system running on a non trusted network segment, the impact can be reduced.

Protecting third parties can be more difficult because a honeypot needs to interact with the global network to be attractive and to return some useful information. This fact alone should not lead to a totally open honeypot, as such a resource will be a powerful weapon in the hands of the blackhat community. Denying all outgoing traffic is also not a way to go as such a setup would not be of much interest for an attacker. Finding a good balance between these two extremes is difficult to achieve.

The obvious solution to manage outbound connections is to use a firewall (packet filter). This makes it possible to set certain limits for outbound connections.

- Allow only a certain amount of IP packets in a given time interval
- Allow only limited amount of TCP SYN packets in a given time interval
- Allow only limited simultaneous TCP connections
- Drop outgoing IP packets randomly

Implementing such firewall rules makes it possible to allow outbound traffic and at the same time to reduce the usefulness of the system for DoS attacks. It is also thinkable to deny outbound traffic to certain destination ports.

Another approach is to deploy an IDS based packet filtering software that makes it possible to drop packets which match specified signatures. The Hogwash[15] packet filter is an implementation of this concept, although it is normally used to filter inbound traffic.

The problem which could arise is the loss of attractiveness of the honeypot. An attacker will not be able to successfully launch any well known attacks against third parties, which could affect his behavior significantly.

4.8 Limiting Risk

As each honeypot introduces a certain risk it is desirable to reduce this risk as much as possible. First of all, the

[14] Distributed denial of service

[15] http://hogwash.sourceforge.net/

risk can effectively be reduced by choosing the honeypot with the lowest security risk which is still suitable. A high-involvement honeypot is not always needed. If a company internal attacker needs to be found who probes the internal network, a mid- or even low-involvement honeypot may be sufficient.

Sometimes a high-involvement honeypot is needed because of its flexibility and provided features. As shown, one of the best solutions to reduce the risk is to utilize a firewalled honeynet instead of a single honeypot.

Triggered alarms should be immediately sent to a honeypot administrator. IDS alerts which detect a possible root exploit should be sent to the responsible person as a pager notification so that immediate actions can take place and the surveillance can be tightened.

The possibility to shutdown a honeypot should always be present. As a last resort, shutting down a honeypot which is getting out of hand is an effective approach to stop all actions - attacks as well as information gathering. Having a honeynet makes it even more handy as all traffic can be blocked on the firewall. Inspecting and analyzing a honeypot is then without risk and enabling the whole system again can be done with a simple firewall rule configuration.

Minimizing the risk is possible but not always easy as additional hardware is needed or time expenses for the implementation increase. Reducing the risk is needed and should not be viewed as a minor matter.

4.9 Attractiveness

Being the owner of a honeypot can be an interesting experience, but what if the members of the blackhat community do not find their way to the honeypot or, even more dramatically, are not interested in the honeypot at all? In this section some possibilities to attract the normally unwanted will be discussed.

Normally a honeypot is put into one network segment. But if an administrator has more than one network segment at hand (for example a class B net and a class C net, or a collection of non-connected class C networks) then he can "widen" the "surface" of his honeypot installation. A straight way would be to put one or more honeypots into each segment, with the penalty of increased administrative and monetary expenses. A more advanced technique would be the installation of IP-tunnels from different networks to a central honeypot installation. There are some considerable advantages over a bunch of single honeypots. First, there is only one honeypot installation to administrate. Second, often no new machines must be installed in the monitored segments, since one can use existing computers to do the tunneling. Third, new networks can easily be integrated in the existent honeypot environment by just installing a new tunnel.

Another approach to lure attackers is the offering of interesting services on the honeypot. Of course the question arises, what an interesting service is or what it should look like. For instance, a so called "script-kiddie" will never bother about an open database port on a machine, the advanced attacker will notice that there is obviously a database running and that the machine could hold some interesting data about the company the network belongs to.

5 Outlook

Honeypots are a new field in the sector of network security. Currently there is a lot of ongoing research and discussions all around the world. Several companies have already launched commercial products. A comparison of available products showed that there are some usable low- to high-involvement honeypots on the market. In the sector of research honeypots, self-made solutions have to be developed as only these solutions can provide a certain amount of freedom and flexibility which is needed to cover a wide range of possible attacks and attackers. Each research honeypot normally has its own goals or different emphasis on the subject. Developing a self-made solution needs a good technical understanding as well as a time intensive development phase.

A honeypot is a valuable resource, especially to collect information about proceedings of attackers as well as their deployed tools. No other mechanism is comparable in the efficiency of a honeypot if gathering information is a primary goal, especially if the tools an attacker uses are of interest. But nevertheless, honeypots can not be considered as a standard product with a fixed place in every security aware environment as firewalls or intrusion detection systems are today. Installing and running a honeypot is not just a matter of "buy and go". The involved risk and need for tight supervision as well as time intensive analysis makes them difficult to use. Honeypots are in their's infancy and new ideas and technologies will surface in the next time. At the same time as honeypots are getting more advanced, hackers will also develop methods to detect such systems. A regular arms race could start between the good guys and the blackhat community.

References

[Amo99] Edward Amoroso. *Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Trace Back, Traps and Responses.* Intrusion NetBooks, 1999.

[Mem01] Project Honeynet Members. *Project Honeynet.* Oct 2001. http://project.honeynet.org.

[MS01] Patrick Mueller and Greg Shipley. *Dragon Claws it's way to the top.* Network Computing, Aug 2001. http://www.networkcomputing.com/1217/1217f2.html.

[MSK00] Stuart McClure, Joel Scambray, and George Kurtz. *Hacking Exposed 2nd Edition.* Computing McGraw-Hill, second edition, 2000.

[Nor99] Stephen Northcutt. *Network Intrusion Detection: An Analysis Handbook.* New Riders Publishing, 1999.

[Spi01] Lance Spitzner. *Honeypots - Definitions and Value of Honeypots.* Oct 2001. http://www.enteract.com/~lspitz/honeypot.html.