HONEYPOT SECURITY

February 2008

TABLE OF CONTENTS

SUMMARY

A honeypot is a deception trap, designed to entice an attacker into attempting to compromise the information systems in an organisation. If deployed correctly, a honeypot can serve as an early-warning and advanced security surveillance tool, minimising the risks from attacks on IT systems and networks. Honeypots can also analyse the ways in which attackers try to compromise an information system, providing valuable insight into potential system loopholes.

This article provides a brief explanation of honeypots, and how they can be deployed to enhance organisational and enterprise security across critical systems and networks.

I. WHAT IS A HONEYPOT?

According to Lance Spitzner, founder of the Honeynet Project, a honeypot is a system designed to learn how "black-hats" probe for and exploit weaknesses in an IT system[1]. It can also be defined as "*an information system resource whose value lies in unauthorised or illicit use of that resource*"[2]. In other words, a honeypot is a decoy, put out on a network as bait to lure attackers. Honeypots are typically virtual machines, designed to emulate real machines, feigning or creating the appearance of running full services and applications, with open ports that might be found on a typical system or server on a network.

A honeypot works by fooling attackers into believing it is a legitimate system; they attack the system without knowing that they are being observed covertly. When an attacker attempts to compromise a honeypot, attack-related information, such as the IP address of the attacker, will be collected. This activity done by the attacker provides valuable information and analysis on attacking techniques, allowing system administrators to "trace back" to the source of attack if required.

Honeypots can be used for production or research purposes. A production honeypot is used for risk mitigation. Most production honeypots are emulations of specific operating systems or services. They help to protect a network and systems against attacks generated by automated tools used to randomly look for and take over vulnerable systems. By running a production honeypot, the scanning process from these attack tools can be

[1] http://rootprompt.org/article.php3?article=210

[2] http://www.spitzner.net/honeypots.html

slowed right down, thereby wasting their time. Some production honeypots can even shut down attacks altogether by, for example, sending the attackers an acknowledgement packet with a window size of zero. This puts the attack into a "wait" status in which it could only send data when the window size increases[3]. In this way, production honeypots are often used as reconnaissance or deterrence tools.

Research honeypots are real operating systems and services that attackers can interact with, and therefore involve higher risk. They collect extensive information and intelligence on new attack techniques and methods, and hence provide a more accurate picture of the types of attacks being perpetrated. They also provide improved attack prevention, detection and reaction information, drawn from the log files and other information captured in the process. In general, honeypot research institutions such as universities and military departments will run research honeypots to gather intelligence on new attack methods. Some of the research results are published for the benefit of the whole community.

CLASSIFICATION OF HONEYPOTS

Honeypots can be classified into two general categories: low-interaction honeypots that are often used for production purposes, and high interaction honeypots that are used for research purposes.

Low-interaction Honeypots

[3] http://safari.oreilly.com/0321321286/ch09lev1sec12

Low-interaction honeypots work by emulating certain services and operating systems and have limited interaction. The attacker's activities are limited to the level of emulation provided by the honeypot. For example, an emulated FTP service listening on a particular port may only emulate an FTP login, or it may further support a variety of additional FTP commands.

The advantages of low-interaction honeypots are that they are simple and easy to deploy and maintain. In addition, the limited emulation available and/or allowed on low-interaction honeypots reduces the potential risks brought about using them in the field. However, with low-interaction honeypots, only limited information can be obtained, and it is possible that experienced attackers will easily recognise a honeypot when they come across one.

Example: Façades

A façade is a software emulation of a target service or application that provides a false image of a target host. When a façade is probed or attacked, it gathers information about the attacker. Some façades only provide partial application-level behaviour (e.g. banner presentation), while others will actually simulate the target service down to the network stack behaviour. The value of a façade is defined primarily by what systems and applications it can simulate, and how easy it is to deploy and administer.

Façades offer simple, easy deployment as they often require minimal installation effort and equipment, and they can emulate a large variety of systems. Since they are not real systems, they do not have any real vulnerabilities themselves, and cannot be used as a jumping-off point by attackers. However, because they provide only basic information about a potential threat, they are typically used by small to medium-sized enterprises, or by large enterprises in conjunction with other security technology.

High-interaction Honeypots

High-interaction honeypots are more complex, as they involve real operating systems and applications. For example, a real FTP server will be built if the aim is to collect information about attacks on a particular FTP server or service.

By giving attackers real systems to interact with, no restrictions are imposed on attack behaviour, and this allows administrators to capture extensive details about the full extent of an attacker's methods. However, it is not impossible that attackers might take over a high-interaction honeypot system and use it as a stepping-stone to attack other systems within the organisation. Therefore, sufficient protection measures need to be implemented accordingly. In the worst case, the network connection to the honeypot may need to be disconnected to prevent attackers from further penetrating the network and machines beyond the honeypot system itself.

Example one: Sacrificial Lambs

A sacrificial lamb is a system intentionally left vulnerable to attack. The administrator will examine the honeypot periodically to determine if it has been compromised, and if so, what was done to it. Additional data, such as a detailed trace of commands sent to the honeypot, can be collected by a network sniffer deployed near the honeypot. However, the honeypots themselves are "live" and thus present a possible jumping-off point for an attacker. Additional deployment considerations must be made in order to isolate and control the honeypot, such as by means of firewalls or other network control devices, or by completely disconnecting the honeypot from the internal network.

Because sacrificial lambs are themselves real systems, all results generated are exactly as they would be for a real system. However, sacrificial lambs require considerable administrative overhead, such as the installation of a full operating system, and manual

application configuration or system hardening. The analysis is also conducted manually and may require additional tools. They also require additional deployment considerations as explained above, and will likely require a dedicated security expert to manage, support, and to analyse the resulting data from the honeypot system.

Example two: Instrumented Systems

An instrumented system honeypot is an off-the-shelf system with an installed operating system and kernel level modification to provide information, containment, or control. The operating system and kernel have been modified by professional security engineers, unlike the sacrificial lamb model. After modifying the operating system and kernel, they will leave the system running in the network as a real target. Instrumented systems combine the strengths of both sacrificial lambs and façades. Like the sacrificial lamb system, they provide a complete copy of a real system, ready for attackers to compromise, while at the same time (like façades) they are easily accessible and difficult to evade. Furthermore, the operating system and kernel in these systems have been modified to prevent attackers from using them as a stepping-stone for further attacks on other parts of the network.

Example three: Spam Honeypots

Honeypot technology is also used for studying spam and email harvesting activities. Honeypots have been deployed to study how spammers detect open mail relays. Machines run as simulated mail servers, proxies and web servers. Spam email is received and analysed to ascertain the reasons why they were received[4]. In addition, an email trap can be set up, using an email address dedicated to just receiving spam emails.

[4] http://www.honeyd.org/spam.php

II. HONEYPOT DEPLOYMENT STRATEGIES

To maximise the strengths of honeypots, and minimise the risks involved, deployment should be carefully planned. The following is a set of common honeypot deployment strategies:

1. Install honeypots alongside regular production servers. The honeypot will likely need to mirror some real data and services from the production servers in order to attract attackers. The security of the honeypot can be loosened slightly so as to increase its chance of being compromised. The honeypot can then collect attack-related information. However, if a successful attack takes place on the honeypot within the network, that compromised honeypot machine might be used to scan for other potential targets in the network. This is the main drawback of installing honeypots within the production system. In other honeypot deployment methods, (some of which are outlined below) this would not happen, as the whole honeynet can itself be a fictitious network.

2. Pair each server with a honeypot, and direct suspicious traffic destined for the server to the honeypot. For instance, traffic at TCP port 80 can be directed to a web server IP address as normal, while all other traffic to the web server will be directed towards the honeypot. To camouflage the honeypot, a certain amount of data, such as the website contents of a web server, may need to be replicated on the honeypot.

3. Build a honeynet, which is a network of honeypots that imitate and replicate an actual or fictitious network. This will appear to attackers as if many different types of applications are available on several different platforms. A honeynet offers an early warning system against attacks and provides an excellent way to analyse and understand an attacker's intention, by looking at what kind of machines and

services have been attacked, and what type of attacks have been conducted. The Honeynet Project[5] is an excellent example of a research honeynet.

[5] http://www.honeynet.org/

III. EXAMPLES OF HONEYPOT SYSTEMS

Examples of freeware honeypots include:

1. Deception Toolkit[6]: DTK was the first Open Source honeypot released in 1997. It is a collection of Perl scripts and C source code that emulates a variety of listening services. Its primary purpose is to deceive human attackers.
2. LaBrea[7]: This is designed to slow down or stop attacks by acting as a sticky honeypot to detect and trap worms and other malicious codes. It can run on Windows or Unix.
3. Honeywall CDROM[8]: The Honeywall CDROM is a bootable CD with a collection of open source software. It makes honeynet deployments simple and effective by automating the process of deploying a honeynet gateway known as a Honeywall. It can capture, control and analyse all inbound and outbound honeynet activity.
4. Honeyd[9]: This is a powerful, low-interaction Open Source honeypot, and can be run on both UNIX-like and Windows platforms. It can monitor unused IPs, simulate operating systems at the TCP/IP stack level, simulate thousands of virtual hosts at the same time, and monitor all UDP and TCP based ports.

---

[6] http://www.all.net/dtk/index.html

[7] http://labrea.sourceforge.net/labrea-info.html

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

[9] http://www.honeyd.org/

5. Honeytrap[10] : This is a low-interactive honeypot developed to observe attacks against network services. It helps administrators to collect information regarding known or unknown network-based attacks.
6. HoneyC[11]: This is an example of a client honeypot that initiates connections to a server, aiming to find malicious servers on a network. It aims to identify malicious web servers by using emulated clients that are able to solicit the type of response from a server that is necessary for analysis of malicious content.
7. HoneyMole [12] : This is a tool for the deployment of honeypot farms, or distributed honeypots, and transport network traffic to a central honeypot point where data collection and analysis can be undertaken.

In the corporate environment, the following commercial products are available:

1. Symantec Decoy Server: This is a "honeypot" intrusion detection system (IDS) that detects, contains and monitors unauthorised access and system misuse in real time[13].
2. Specter[14]: This is a smart honeypot-based intrusion detection system. It can emulate 14 different operating systems, monitor up to 14 different network services and traps, and has a variety of configuration and notification features.

---

[10] http://honeytrap.mwcollect.org/

[11] https://www.client-honeynet.org/honeyc.html

[12] http://www.honeynet.org.pt/index.php/HoneyMole

[13] http://www.symantec.com/business/support/documentation.jsp?language=english&view=manuals&pid=51899

[14] http://www.specter.com/default50.htm

IV. CONCLUSION

Honeypots have their advantages and disadvantages. They are clearly a useful tool for luring and trapping attackers, capturing information and generating alerts when someone is interacting with them. The activities of attackers provides valuable information for analysing their attacking techniques and methods. Because honeypots only capture and archive data and requests coming in to them, they do not add extra burden to existing network bandwidth.

However, honeypots do have their drawbacks. Because they only track and capture activity that directly interacts with them, they cannot detect attacks against other systems in the network. Furthermore, deploying honeypots without enough planning and consideration may introduce more risks to an existing network, because honeypots are designed to be exploited, and there is always a risk of them being taken over by attackers, using them as a stepping-stone to gain entry to other systems within the network. This is perhaps the most controversial drawback of honeypots.