Interested in learning more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

Hands in the Honeypot

A honeypot is a program, machine, or system put on a network as bait for attackers. The idea is to deceive the attacker by making the honeypot seem like a legitimate system. A honeynet is a network of honeypots set up to imitate a real network. Honeynets can be configured in both production and research environments. A research honeynet studies the tactics and methods of attackers. A production honeynet is set up to mimic the production network of the organization. Honeypots return highly valuable data that is much eas...

Copyright SANS Institute
Author Retains Full Rights

AD

GIAC Security Essentials Certification (GSEC)
Practical Assignment
Version 1.4b

Hands in the Honeypot

Kecia Gubbels

November 3, 2002

Abstract

A honeypot is a program, machine, or system put on a network as bait for attackers. The idea is to deceive the attacker by making the honeypot seem like a legitimate system. A honeynet is a network of honeypots set up to imitate a real network. Honeynets can be configured in both production and research environments. A research honeynet studies the tactics and methods of attackers. A production honeynet is set up to mimic the production network of the organization. This type of honeynet is useful to expose the organizations current vulnerabilities. Honeypots return highly valuable data that is much easier to interpret than that of an IDS (Intrusion Detection System). The information gathered from honeypots can be used to better prepare system administrators for attacks.

This paper focuses on the description and analysis of honeypots as well as how and where they are used. I describe the process of setting up and running a honeypot. Commands and associated output is provided to demonstrate how one would configure and install a honeypot. I set up two honeypots in an air-gapped security lab to test their effectiveness. I used the Nmap vulnerability scanner to test each of the honeypots in terms of their ability to emulate various operating systems and services. I also describe any potential problems that I encountered during my testing. This paper also takes a look into the mind of the enemy. Recommendations for honeynets are provided.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Introduction

“In warfare, information is power. The better you understand your enemy, the more able you are to defeat him.” While this quote may sound like something out of Sun Tzu's **The Art of War**, it actually comes from The Honeynet Project. The Honeynet Project is a non-profit research organization dedicated to learning the tools, tactics, and motives of the blackhat community and sharing the lessons learned.

The Honeynet Project was started for the purpose of recording the actions of attackers. The results were quite surprising. According to the findings of the Honeynet Project, a random computer on the Internet is scanned dozens of times a day. The time before someone successfully hacks a default install of RedHat 6.2 server is **less than seventy-two hours**. A home computer with Windows 98 and file sharing enabled was hacked five times in four days. The fastest time for a server to be hacked is **fifteen minutes** after it was plugged into the network. The findings of the project have made the security community stand up and take notice. Attackers will not go away on their own: We need to research their tactics, motives, and tools to protect us and our organizations.

If you are a security professional you can see how this applies to attackers and other network intruders. The more you know about your enemy and their method of attack, the better you are able to defend against him. Honeynets and honeypots are one way that we, as professionals, can gain insight into the world of our attackers.

Honeypots

A honeypot is a program, machine, or system put on a network as bait for attackers. The idea is to deceive the attacker by making the honeypot seem like a legitimate system. Honeypots are typically virtual machines that emulate real machines by feigning running services and open ports, services which one might find on a typical machine on a network. These running services are meant to attract the attention of attackers so that they spend valuable time and resources will be used to try to exploit the machine while the attacker is being monitored and recorded by the honeypot.

Honeypots contain enough interesting information to attract a hacker to a site, but yet they do not have a company’s crucial data available. The main purpose of a honeypot is to imitate a network while attackers attempt to log onto it; meanwhile, the honeypot captures details of the attacker’s attacks, such as who they are, what they want, what their skill level is, and what types of tools they use. The information gained after the honeypot is attacked can be used to learn about vulnerabilities of the current network so as to improve it for the future. The amount of information that can be uncovered about their techniques depends on the number of times the attacker attempts to log on.[1]

[1] Cassi, 2002

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Honeypots are **designed** to be broken into for two reasons. The first is to find information regarding weak areas of a system. The administrator can learn how the system was broken into by observing approaches used by attackers. The second reason is to collect information necessary to aid in the prosecution of attackers. Honeypots purposely leave a noticeable gap in the system for intruders to go through in order for other areas of the system to look more secure. Therefore, the honeypot protects the rest of the system by attracting attention to itself.

The purpose of the honeypot will help decide how the information is used. Honeypots can capture information that can be used to infer not only how but who attacked it, while at the same time protecting the real network. This information could also be used as evidence to prosecute the attacker.

Figure 1 graphically illustrates an example of a Honeynet.

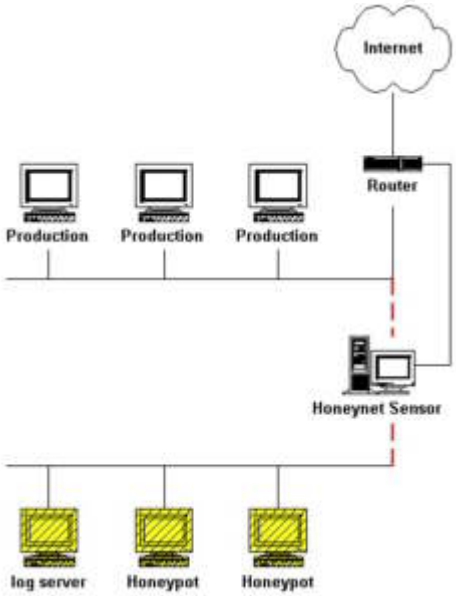

Figure 1 Honeynet Illustrated [adapted from (Spitzner, p. 258)].

Where Honeypots Are Used

Honeypots are currently being used in both production and research environments. Research honeypots study the way attacker's progress and establish their lines of attack. These types of honeypots are more focused on researching the attacker by using a number of different configurations to lure them in. Production honeypots on the other hand tend to mirror the production network of the company in order to expose current vulnerabilities of the network. Production honeypots can significantly reduce the risk of intrusion by uncovering these vulnerabilities and alert administrators of attacks.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Depending on the reason, an effectively arranged honeypot may provide a variety of the following functions[1]:

1. Provide alerts to attacks in progress
2. Leave attackers out in the open and isolated from a production network
3. Provide real time monitoring of the attack
4. Supply information on how the attacker breaks in
5. Give evidence for prosecuting the attacker

Honeypots should not provide attackers with these functions:

1. Access to real data
2. Administrative controls to the production network
3. Legitimate users
4. Legitimate network traffic
5. Control over other devices attached to a production network

Social Engineering

Social Engineering describes a non-technical kind of intrusion which relies heavily on human interaction and often involves tricking other people to break normal security procedures. Social engineering is a way to sweeten the honeypot.[2] The more your honeynet looks like your organization, the more realistic the environment. This will help the attacker believe your honeynet. A good way to do this is to add user accounts and subscribe them to mailing lists, create e-mail to and from the users, and give them documents and directories. A honeypot that remains un-attacked is worthless.

There are many ways to use and build honeypots. Most honeypots are set up as default systems, that is, there is no customization of scripts that define the honeypot. These will definitely attract attackers, but the lessons learned may be limited. The more an attacker probes and/or connects to a honeypot, the greater the damage they can do, and consequently the more information can be collected. Low interaction honeypots have an easy installation but have a limited amount of information that can be acquired from the attacker.

High interaction honeypots, that is, those that attract a great deal of attention from attackers, are extremely risky; although, much more information can be obtained. There are multiple honeypot solutions; none of these solutions are better than the others, it depends on the amount of risk you want to take, and the amount of information you want to acquire from the attacker.

A typical attacker will try to cover their tracks so it is important to collect data from as many places as possible. The system logs of a machine will be the first thing revised by the attacker to hide the attack. Therefore, these files should be copied to a remote

[1] McMullen, 2001
[2] Honeynet Project, 2002

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

machine so they can be compared later. An administrator must be careful in doing this because any connection to a remote machine may be discovered by the attacker. On the other hand, intrusion detection systems and packet sniffers are undetectable by attackers and are another great way to monitor their moves. With these types of devices, each keystroke made by the attacker is traced.

Compare and Contrast Honeypots – LaBrea & Honeyd

To demonstrate the installation and running of a honeypot I chose to install two UNIX-based versions: LaBrea and Honeyd, each of which is described in more detail below. Because of concerns about the safety of the systems, specifically, the fact that both capture unused IPs and supposedly relinquish them when needed, the honeypots were installed in an air-gapped security laboratory environment, i.e., with no access to the outside Internet. The installation machine used was an IBM E-pro with a Pentium III processor and 128 MB of RAM running Red Hat Linux 7.3.

I simulated how an attacker might probe a system by using the Nmap vulnerability scanner. Nmap ("Network Mapper") is an open source utility for network exploration or security auditing created by Fyodor (www.insecure.org). Nmap uses raw IP packets to determine such things as: what hosts are up, what services they offer, the operating system, what filters are in use on a packet filter, and so on.[1]

Figure 2 illustrates a sample listing of some of the flags available to Nmap that was employed in this research.

-v	Verbose – Highly recommended for interactive use
-r	Randomize – This will randomize the order in which the target ports are scanned
-R	Shows and resolves all hosts (even those that are down)
-p	Specify a range of ports to scan
-sS	TCP SYN stealth port scan (best all around TCP scan)
-sU	UDP port scan
-O	Uses TCP/IP fingerprinting to guess the operating system

Figure 2 Nmap Illustrated [adapted from (Nmap)].

LaBrea

LaBrea:The Tarpit was the first honeypot installed and tested. This unix-based program creates a tarpit or “sticky honeypot” that takes unused IP addresses on your network and creates virtual machines that answer to connection attempts. The reason LaBrea is sticky is that it answer those connection attempts in a way that sometimes causes the machine at the other end to get stuck. The way LaBrea works is by watching ARP requests and replies. The forged replies sent by the honeypot are crafted to maintain an open connection with the attacker, slowing down or even stopping the automated attack.[2]

[1] www.insecure.org
[2] Spitzner, p44

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

The version of LaBrea downloaded was labrea-2.4b3-1.i386.rpm. The download location is at http://www.hackbusters.net/LaBrea.html. Libnet, a dependency needed by LaBrea to run, was also downloaded and installed. Libnet is a high-level toolkit that allows the construction and injection of network packets. You can find libnet (libnet.tar.gz version 1.1.0) from http://www.packetfactory.net/projects/libnet/.

Running LaBrea

Once LaBrea was running with the default scripts and I used nmap to scan the network using the following command:

#nmap -sS -p 1-500 100.x.x.0/24

The -sS flag indicates that Nmap should perform a TCP SYN scan (half-open scan); the -p flag indicates the ports to scan (here ports 1-500 only); and 100.x.x.0/24 indicates the IP range to scan, here given in CIDR notation (which would be machines from 100.x.x.0 to 100.x.x.255).

After the scan I examined the log files. Figure 3 is from the log file /var/log/messages on the honeypot machine. It shows the attack machine scanning my honeypot and which port. Another important note is that Nmap can be set to scan ports randomly. If a sequential scan is used, that seems too obvious and it may be caught by an Intrusion Detection System, some of which expect scans to be in sequential order.

Figure 3 illustrates a sample of my /var/log/messages file on the LaBrea honeypot.

```
Oct 30 16:17:19 localhost /usr/sbin/labrea: Initial Connect (tarpitting): 100.x.x.54 1069 -> 100.x.x.1 7003 *
Oct 30 16:17:19 localhost /usr/sbin/labrea: Initial Connect (tarpitting): 100.x.x.54 1070 -> 100.x.x.1 1001
Oct 30 16:17:19 localhost /usr/sbin/labrea: Initial Connect (tarpitting): 100.x.x.54 1071 -> 100.x.x.1 6142 *
Oct 30 16:17:19 localhost /usr/sbin/labrea: Initial Connect (tarpitting): 100.x.x.54 1072 -> 100.x.x.1 94
Oct 30 16:17:19 localhost /usr/sbin/labrea: Initial Connect (tarpitting): 100.x.x.54 1073 -> 100.x.x.1 1414 *
Oct 30 16:17:19 localhost /usr/sbin/labrea: Initial Connect (tarpitting): 100.x.x.54 1074 -> 100.x.x.1 27665
Oct 30 16:17:19 localhost /usr/sbin/labrea: Initial Connect (tarpitting): 100.x.x.54 1075 -> 100.x.x.1 1021 *
Oct 30 16:17:19 localhost /usr/sbin/labrea: Initial Connect (tarpitting): 100.x.x.54 1076 -> 100.x.x.1 834
Oct 30 16:17:19 localhost /usr/sbin/labrea: Initial Connect (tarpitting): 100.x.x.54 1077 -> 100.x.x.1 35 *
Oct 30 16:17:19 localhost /usr/sbin/labrea: Initial Connect (tarpitting): 100.x.x.54 1078 -> 100.x.x.1 1453
Oct 30 16:17:19 localhost /usr/sbin/labrea: Initial Connect (tarpitting): 100.x.x.54 1079 -> 100.x.x.1 1412 *
Oct 30 16:17:19 localhost /usr/sbin/labrea: Initial Connect (tarpitting): 100.x.x.54 1080 -> 100.x.x.1 839
Oct 30 16:17:19 localhost /usr/sbin/labrea: Initial Connect (tarpitting): 100.x.x.54 1081 -> 100.x.x.1 17007 *
Oct 30 16:17:19 localhost /usr/sbin/labrea: Initial Connect (tarpitting): 100.x.x.54 1082 -> 100.x.x.1 2046
```

This is what the systems administrator would see if an attacker scanned the network. If you look at the sample Nmap scan and results in Figure 4, you will see that a virtual machine appears on the free IP address. Every port appears to be open which seems a bit obvious that it is a honeypot. (NOTE: of course LaBrea was run with default scripts. Later, I describe how customizing scripts can make for a much more realistic honeypot.) This can throw the attacker off because they are not sure which vulnerability on the machine to attack.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Figure 4 illustrates an Nmap scan of my network containing the LaBrea Honeypot.

```
Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Host  (100.x.x.1) appears to be up ... good.
Initiating Connect() Scan against  (100.x.x.1)
Adding open port 1408/tcp
Adding open port 792/tcp
Adding open port 2027/tcp
Adding open port 1517/tcp
Adding open port 1422/tcp
Adding open port 2013/tcp
Adding open port 1103/tcp
Adding open port 662/tcp
Adding open port 926/tcp
Adding open port 1383/tcp
Adding open port 774/tcp
Adding open port 508/tcp
Adding open port 303/tcp

The Connect() Scan took 15 seconds to scan 1554 ports.
Interesting ports on  (100.x.x.1):
Port      State      Service
1/tcp     open       tcpmux
2/tcp     open       compressnet
3/tcp     open       compressnet
4/tcp     open       unknown
5/tcp     open       rje
6/tcp     open       unknown
7/tcp     open       echo
8/tcp     open       unknown
9/tcp     open       discard
10/tcp    open       unknown
11/tcp    open       systat
12/tcp    open       unknown
13/tcp    open       daytime
14/tcp    open       unknown

Nmap run completed -- 1 IP address (1 host up) scanned in 27 seconds
```

Honeyd

The second honeypot chosen to install was Honeyd. Honeyd is an open source honeypot solution designed for Unix Systems. It is developed and maintained by Niels Provos of the University of Michigan and was first released in April 2002. I found this to be a much more powerful honeypot than LaBrea, for several reasons, one of which the ease with which it can be configured.

Honeyd is used to detect attacks or unauthorized activity. It does this by emulating the services of many operating systems and is easily customized. It can listen on any port you want, with as little or as much emulation as you choose. Similar to LaBrea, Honeyd assumes the identity of any IP address that does not have a valid system. As with any honeypot, Honeyd has both advantages and disadvantages. Figure 5 lists these according to Spitzner in his book Honeypots Tracking Hackers.

© SANS Institute 2002, Author retains full rights.

Figure 5 advantages and disadvantages of Honeyd.

Advantages of Honeyd	Disadvantages of Honeyd
Can monitor any UDP or TCP port and entire networks.	As a low-interaction solution, it cannot provide real operating solutions for attackers to interact with.
As an OpenSource solution, it is free and will develop quickly with the input and development of others in the security community.	As an OpenSource solution, it provides no formal support for maintenance and troubleshooting.
Resists fingerprinting efforts by emulating operating systems at IP stack level as well as the application level.	No built-in mechanism for alerting, nor any mechanism for capturing extensive information.

Figure 5 Honeyd adapted from (Spitzner, p. 166).

I downloaded honeyd-0.3.tar.gz released on 7-30-02. You can download Honeyd from http://www.citi.umich.edu/u/provos/honeyd/. There are also several dependencies that you must be sure you have, they include: libevent (an asynchronous event library), libdnet (the [not so] dumb network library), and libpcap, a packet capture library.

Libdnet provides a simplified, portable interface to several low-level networking routines such as: network address manipulation, kernel arp cache and route table lookup and manipulation, network firewalling (IP filter, ipfw, ipchains, pf, ...), network interface lookup and manipulation, and raw IP packet and Ethernet frame transmission.[1] Libpcap provides implementation-independent access to the underlying packet capture facility provided by the operating system. Libpcap is the library used to grab packets from the network card.[2]

The systems administrators must direct network traffic towards the honeypot. Several means are mentioned on the Honeyd website, and I chose to use arpd, an arp (address resolution protocol) daemon. I used the following command to run arpd:

```
./arpd -i eth0 100.x.x.100/30
```

This command specifies that aprd should listen on the ethernet interface eth0 and listen for MAC addresses mapped to the IP range 100.x.x.100-100.x.x.103. Later you will see that I created four separate personalities and associated each of these with one of these IPs.

Running Honeyd

Once Honeyd is installed you must carefully configure it using the config.sample file. An example is provided inside the file, but I highly recommend creating your own because by default the only machine configured is a single machine running AIX.

[1] Libdnet, http://libdnet.sourceforge.net/
[2] Libpcap, http://www.tcpdump.org/

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Honeyd comes with several dozen 'personalities', that is, different machines which may be emulated. Honeyd uses the very same database of signatures that Nmap uses and replies to nmap probes based on the emulated operating systems. What this allows one to do is to provide a better simulation of an actual network, which is likely to include several different types of operating systems, each of which runs different services.

I browsed the nmap.personalities file which contains the OS personalities and decided to emulate a small network composed of the following: an AIX, FreeBSD, Windows 98, and Windows NT machine. You can be specific, right down to the version and service pack level. Figure 6 contains a copy of the configuration file that illustrates each of the personalities and associated running services I emulated.

Figure 6 Honeyd configuration file.

```
# Example of some simple host templates and their bindings

annotate "AIX 4.0 - 4.2" fragment old
create template1
set template1 personality "AIX 4.0 - 4.2"
add template1 tcp port 80 "sh scripts/web.sh"
add template1 tcp port 22 "sh scripts/test.sh $ipsrc $dport"
add template1 tcp port 23 proxy $ipsrc:23
set template1 default tcp action reset
bind 100.x.x.100 template1

annotate "FreeBSD 2.2.1 - 4.1" fragment old
create template2
set template2 personality "FreeBSD 2.2.1 - 4.1"
add template2 tcp port 23 proxy $ipsrc:23
add template2 tcp port 53 proxy $ipsrc:53
add template2 tcp port 110 "sh scripts/pop3.sh"
set template2 default tcp action reset
bind 100.x.x.101 template2

annotate "Windows 98 w/ Service Pack 1" fragment old
create template3
set template3 personality "Windows 98 w/ Service Pack 1"
add template3 tcp port 80 "sh scripts/web.sh"
add template3 tcp port 21 "sh scripts/ftp.sh $ipsrc $dport"
set template3 default tcp action reset
bind 100.x.x.102 template3

annotate "Windows NT 4 SP3" fragment old
create template4
set template4 personality "Windows NT 4 SP3"
add template4 tcp port 25 "sh scripts/smtp.sh"
add template4 tcp port 21 "sh scripts/ftp.sh $ipsrc $dport"
set template4 default tcp action reset
bind 100.x.x.103 template4
```

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Figure 7 contains the FreeBSD personality from my config.sample file. The first line tells which operating system we wish to emulate. Here we used “FreeBSD 2.2.1 – 4.1”. I was sure to list the personality exactly as specified in the nmap.personalities file, as I found through trial and error. The “create template” line is used to associate a template to each personality you create. You then add to your template a list of ports you wish to provide as services. In figure 7 you see I have chosen to use port 23 (Telnet), port 53 (Domain Name Server DNS), and port 110 (POP3). You can see the pop3 service uses a script called pop3.sh, which provides emulated services of pop3. Honeyd also offers several other scripts that can be used to make the services even more realistic. The default TCP action is reset, causing connection attempts to be reset (the default behavior of most machines). The final line binds template2 to the specific IP address on my network of 100.x.x.101. You can bind your template to multiple addresses if you so choose.

Figure 7 FreeBSD personality from my config.sample file.

```
annotate "FreeBSD 2.2.1 - 4.1" fragment old
create template2
set template2 personality "FreeBSD 2.2.1 - 4.1"
add template2 tcp port 23 proxy $ipsrc:23
add template2 tcp port 53 proxy $ipsrc:53
add template2 tcp port 110 "sh scripts/pop3.sh"
set template2 default tcp action reset
bind 100.x.x.101 template2
```

The scripts used to emulate certain services can be downloaded from the website under the contributions section at http://www.citi.umich.edu/u/provos/honeyd/contrib.html. I used: ftp.sh, pop3.sh, web.sh, and smtp.sh. In using these scripts, your virtual machine becomes even more realistic.

Be forewarned against blindly running these scripts. Using scripts without looking at the code is not recommended, as I found out by chance. For example, I used the smtp.sh script for my Windows NT machine. I attempted to telnet to port 25 (smtp) to see how the script would handle this scenario. Figure 8 shows my results:

Figure 8 Scan of header file results.

```
 [root@localhost root]# telnet 100.x.x.103 25
Trying 100.x.x.103...
Connected to 100.x.x.103.
Escape character is '^]'.
220 localhost.localdomain.localdomain ESMTP Sendmail 8.12.2/8.12.2/SuSE Linux 0.6
```

As you can see, the header came back that the system being run was SuSE Linux running sendmail. The reason this is the case is that the smtp.sh script specifies that this is the text to return upon an attempted connection. And of course, an attacker would realize that a Windows NT 4.0 machine would not be running sendmail, a UNIX-only smtp service. This is just one justification for reading the code before implementing it. Figure 9 shows a segment of the smtp.sh code.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Figure 9 Segment from smtp.sh.

```
set -x -vDATE=`date`
host=`hostname`
domain=`dnsdomainname`
log=/tmp/honeyd/smtp-$1.log
MAILFROM="err"
EHELO="no"
RCPTTO="err"
echo "$DATE: SMTP started from $1 Port $2" >> $log
echo -e "220 $host.$domain ESMTP Sendmail 8.12.2/8.12.2/SuSE Linux 0.6; $DATE\r"
```

Once I had prepared the configuration file, I scanned my honeypot.

Figure 10 Nmap command.

```
[root@localhost root]# nmap -sS -p 1-100 100.x.x.100/30 –O
```

With these flags:
-sS TCP SYN stealth scan
-p scan ports in this range: 1-100
-O operating system

I choose to scan only ports 1-100, as the ports I specified fell into that range. I used CIDR (**C**lassless **I**nter-**D**omain **R**outing) notation to scan only 100 – 103, since this range contains all the bound IP addresses. This is accomplished using 100.x.x.100/30. As you can see from my scan results in Figure11, my configuration file worked. I found all the ports open that I specified and the operating systems were also guessed.

Figure 11 Nmap scan results.

```
[root@localhost root]# nmap -sS -p 1-100 100.x.x.100/30 -O

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Interesting ports on  (100.x.x.100):
(The 97 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
23/tcp     open        telnet
80/tcp     open        http

Remote operating system guess: AIX 4.0 - 4.2

Interesting ports on  (100.x.x.101):
(The 98 ports scanned but not shown below are in state: closed)
Port       State       Service
23/tcp     open        telnet
53/tcp     open        domain

Remote operating system guess: FreeBSD 2.2.1 - 4.1
```

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

```
Uptime 8.018 days (since Thu Oct 24 12:18:18 2002)
Interesting ports on  (100.x.x.102):
(The 98 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
80/tcp     open        http

Remote operating system guess: Windows 98 w/ Service Pack 1

Interesting ports on  (100.x.x.103):
(The 98 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
25/tcp     open        smtp

Remote operating system guess: Windows NT 4 SP3

Nmap run completed -- 4 IP addresses (4 hosts up) scanned in 12 seconds
```

Since Honeyd was installed in a laboratory environment, there was no concern that the honeypot would use my free production IP addresses. If I was to install this in my production environment, I would not want to take all my unused IP addresses. Instead what I choose to do was to pass a range of IP addresses to Honeyd to use. I did this by changing arpd (Address Resolution Protocol daemon). I found arpd on the Honeyd page at http://www.citi.umich.edu/u/provos/honeyd/ to download and configure. Once I had arpd running, I passed the parameters in Figure 12. Once again I used CIDR notation to choose addresses 100 – 103.

Figure 12 Arpd parameters.

```
./arpd –i eth0 100.x.x.100/30

        With this flag:
        -i interface
```

Advantages and Disadvantages of Honeynets and Honeypots

There are many advantages and disadvantages of honeypots. The most evident advantage is the simplicity of a honeypot. They do not include any actual production services, so no one in the organization will need to access the honeypot. For this reason, any connection to the honeypot is likely a scan or attack. Also, any information sent from the honeypot is a sign that the honeypot has been taken over.

Unlike intrusion detection systems, which only identify when and how an attacker got into the network, a honeypot is a distraction as well. In addition, honeypots are not concerned with overload of network traffic or discerning between legitimate and illegitimate packets, as other intrusion detection systems may be. Honeypots only pay attention to the traffic that comes to them. The data collected from honeypots is smaller but nonetheless, of high importance. Every time the honeypot is attacked, the production network becomes more protected.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

A honeypot can educate the administrator about the event of an intrusion. If a honeypot was hacked, the administrator can sharpen up their response skills to a hack before their real system is confronted.[1] The honeypot can also be used as an early warning system, alerting administrators of any hostile intent before the real system is compromised. Honeypots are very simple. They are not used by anyone on the production network; therefore, the only action on a honeypot is likely an intruder. This makes it easy for the administrator to deal with the information.

One disadvantage of the honeypot is encouraging an aggressive atmosphere. You want your honeypot to be attacked. Another disadvantage is that some honeypots are unable to capture information about attacker's motives, habits, and actions.[2] If a honeypot was successfully broken into, the attacker can use this to begin other attacks either against part of your real system or a third party's system.[3] Honeypots can also add risk to a network. If an attacker successfully breaks into a honeypot, they can use this in order to jump from system to system if not properly self-contained. If this occurs, the administrator of the honeypot will be responsible for the intrusion and destruction of an organization, not the attacker.

If the honeypot is attacked, the administrator should prevent major changes to the honeypot, because any noticeable changes will make the attacker more suspicious. If the attacker does figure out the system is a honeypot, he can share this with other attackers, and the system will become worthless. Another possibility may be that the attacker will become upset to figure out he was wasting his time, and become more aggressive in order to break into the actual production system.

The Enemy

In learning about the enemy, we focus on the threat, the tactics, the tools, and the motives they use. In the threat, one common methodology here is that of the script kiddie. This is someone who is looking for an easy in to gain control to as many systems as possible. The way they do this is by using a small number of exploits and being patient in finding someone with the vulnerability. Some of these tools are designed to leave backdoors for the attacker to use later on. The threat here lies in the fact that they are random attacks. You could have a machine up for only days that no one knows about and be scanned. You may believe that no one knows about your system, so you are safe. You may also believe that you have the most secure system possible. Well, all it takes is one mistake and you will be found. Another threat is that the tools for this attack are widely distributed. This means anyone can use them. This makes the unprotected system easy to exploit and easy to kill.[4]

Over the past several years of research, attackers have shown a pattern of focusing on specific vulnerabilities. A typical attacker uses automated scanning tools to find and

[1] Kotry, 2002
[2] Seifried, 2002
[3] Kotry, 2002
[4] Spitzner, 2002

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

identify targets, storing them for later use. Normally an attacker will store information about a system's IP address, operating system, exposed services, and applications offered by the system. Based on this information, an attacker can determine if a system is vulnerable to attack. Additional scans may be performed to determine exact versions of an operating system or application are being used. One variant of the auto-scanning tools is an "autorooter". An "autorooter" is a scanning tool which attempts to exploit a particular vulnerability whenever it is discovered, without human interaction. Exploited machines are then logged into a database, for later retrieval and use by the attacker.

Generally, attack tools are hard to develop. Very few attackers possess the knowledge necessary to create the tools. However, once the tools are written, they can be used by anyone with access to the tool. Attackers often set up websites and IRC (Internet Relay Chat) channels to distribute their tools, using them to not only to distribute, but also to educate others on the use of their tools. Sometimes these sites are set up on compromised machines without the system administrator's knowledge.

We have a lot to learn about the attackers and how to better capture and analyze their activities. Some of the tactics used in the past to capture the attackers involved setting up default installations of our commonly used systems either alone or within a Honeynet. The data that moved in and out of them was considered suspicious. This information was then captured and studied. The attackers who found this information would use mainly script kiddie tactics to find and exploit vulnerabilities in the system. The tools they use are usually scripted. Although we have gathered much information through the use of these honeypots, we must develop more sophisticated honeynets if we hope to continue to attract and learn more about these attackers.

One of the goals for the future of the honeypot is to deploy more and more honeynets around the world. The greater number of honeypots there are the quicker new trends will be identified. You can also gather more data to confirm a common trend. The newer areas for the honeynets may also attract a different group of attackers. Depending on the type of system or site attacked, we may see new tools. Yet another advantage of distributed honeynets would be to find vulnerabilities in zones of trust. In businesses, many site-to-site relationships are established. It would also be possible to see the repercussions of this trust. All of these are reasons why distributed honeynets can be an advantage to many people and organizations.

Honeypot Recommendations

The idea of a honeypot is simple but sometimes it may be difficult and time consuming to configure. It may even be especially complicated to set one up on a production network because the honeypot must remain isolated from the rest of the production network. Some companies may be better off using their resources to concentrate on tighter security rather than implementing deception devices such as honeypots. If an organization does decide to implement a honeypot, it should not be the only means of intrusion detection. Good security practices such as controlled physical access, limited

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

services running, strict password rules, and expiring passwords are the only chance of keeping attackers out.

In conclusion, a honeypot does not address a specific problem; it is a tool that contributes to the overall security architecture. A honeynet is nothing more than a highly controlled network with production systems (honeypots) inside. The systems are set up much like a typical organization so the problems the attackers expose are true to the environment. Honeypots return highly valuable data that is much easier to interpret. This makes analysis and reaction time much quicker. The lesson to learn from this is that you can not hide. Even if you have a very secure organization, all it takes is one mistake. The information we gather from a honeynet can be shared to help others avoid making the same mistakes. The key is to continually adapt to the changing enemy.

© SANS Institute 2002, Author retains full rights.

References

Cassi, Richard et al. "System Administration, Networking and Security." Sans Institute. (2002) URL: http://www.sans.org/newlook/resources/IDFAQ/honeypot.htm. (29 July 2002).

Cohen, Fred et al. "A Framework for Deception." 13 July 2001. URL: http://heat.ca.sandia.gov/papers/Framework/Framework.html. (1 Nov 2002).

Green, David. "From Honeypots to a Web of SIN – Building the World-Wide Information System." URL: http://www.csu.edu.au/special/conference/apwww95/papers95/dgreen/dgreen.html. (23 July 2002).

Honeynet Project. Know Your Enemy. Addison Wesley, 2002.

Kidman, Angus. "Like hackers to a honeypot." 13 Aug 2002. URL: http://www.smh.com.au/articles/2002/08/10/1028158034383.html. (15 Oct 2002).

"Know Your Enemy." The Honeynet Project. 21 July 2000 URL: http://project.honeynet.org/papers/enemy. (10 August 2002).

Kotry, Hisham. "Building a Virtual Honeynet." 17 July 2002. URL: http://www.linuxsecurity.com/feature_stories/feature_story-100.html. (29 July 2002).

"Libdnet." URL: http://libdnet.sourceforge.net/. (1 Nov. 2002).

"Libpcap." 13 May 2002. URL: http://www.tcpdump.org/. (1 Nov 2002).

McMullen, John F. "Enhance Intrusion Detection with a Honeypot." Techrepublic Online. 12 Apr 2001. URL: http://builder.com.com/article.jhtml?id=r00220010412mul01.htm&_requestid=166143. (7 Aug 2002).

"Nmap." 10 Aug 2002. URL: http://www.insecure.org/nmap/. (25 Oct. 2002).

Schwartz, Matthew. "Networks Use 'Honeypots' To Catch an Online Thief." 4 Apr 2001. URL: http://www.cnn.com/2001/TECH/internet/04/04/trap.a.thief.idg/. (23 July 2002).

Seifried, Kurt. "Honeypotting with VMware – Basics." 15 Feb 2002. URL: http://www.seifried.org/security/ids/20020107-honeypot-vmware-basics.html. (26July 2002).

Spitzner, Lance. "Building a Honeypot." 20 Mar 2002 URL:

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

http://rootprompt.org/article.php3?article=210. (15 Oct 2002).

Spitzner, Lance. "Honeypots. Definitions and Value of Honeypots." 17 May 2002 URL: http://www.enteract.com/~lspitz/honeypot.html. (23 July 2002).

Spitzner, Lance. Honeypots Tracking Hackers. Addison Wesley, 2003.

© SANS Institute 2002, Author retains full rights.

Last Updated: May 21st, 2013

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Malaysia @ MCMC 2013	Cyberjaya, MY	Jun 03, 2013 - Jun 08, 2013	Live Event
SANS Pen Test Berlin 2013	Berlin, DE	Jun 03, 2013 - Jun 08, 2013	Live Event
Industial Control Systems Security Training - Houston	Houston, TXUS	Jun 10, 2013 - Jun 15, 2013	Live Event
Security Impact of IPv6 Summit 2013	Washington, DCUS	Jun 14, 2013 - Jun 16, 2013	Live Event
SANSFIRE 2013	Washington, DCUS	Jun 14, 2013 - Jun 22, 2013	Live Event
SANS Canberra 2013	Canberra, AU	Jul 01, 2013 - Jul 13, 2013	Live Event
Digital Forensics & Incident Response Summit 2013	Austin, TXUS	Jul 09, 2013 - Jul 16, 2013	Live Event
SANS London Summer 2013	London, GB	Jul 09, 2013 - Jul 16, 2013	Live Event
SANS Rocky Mountain 2013	Denver, COUS	Jul 14, 2013 - Jul 20, 2013	Live Event
SANS Mumbai 2013	Mumbai, IN	Jul 22, 2013 - Jul 27, 2013	Live Event
SEC528 SANS Training Program for the CompTIA® New Advanced Security Practitioner (CASP) Certification	Washington, DCUS	Jul 22, 2013 - Jul 26, 2013	Live Event
SANS San Francisco 2013	San Francisco, CAUS	Jul 29, 2013 - Aug 03, 2013	Live Event
SANS SEC 560: Network Penetration Testing @ Bangalore 2013	Bangalore, IN	Aug 05, 2013 - Aug 10, 2013	Live Event
SANS Boston 2013	Boston, MAUS	Aug 05, 2013 - Aug 10, 2013	Live Event
Critical Security Controls Summit	Washington, DCUS	Aug 12, 2013 - Aug 18, 2013	Live Event
Industrial Control Systems Security Training - DC	Washington, DCUS	Aug 12, 2013 - Aug 16, 2013	Live Event
SANS Thailand 2013	Bangkok, TH	Aug 19, 2013 - Aug 31, 2013	Live Event
SANS Virginia Beach 2013	Virginia Beach, VAUS	Aug 19, 2013 - Aug 30, 2013	Live Event
Mobile Device Security Summit 2013	OnlineCAUS	May 30, 2013 - Jun 06, 2013	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced