



Honeyd Installation

In the previous chapters, you've learned about honeypots in general, and specifically which emulated services and ports you need to create an authentic-looking Windows honeypot. Starting in this chapter, you'll put those lessons to use by implementing Honeyd.

This chapter will introduce Honeyd and the functionality it offers. Then it will guide you through installing Honeyd and its related programs, with step-by-step instructions.

What Is Honeyd?

Honeyd stands for honeypot daemon. Honeyd (<http://www.honeyd.org>) is an open-source, low-interaction honeypot released by Dr. Niels Provos (provos@monkey.org) in April 2002 so he could study the methods and tactics used by malicious hackers. Dr. Provos is an experimental computer scientist who conducts research in steganography and network security. He is currently working for Google. A German native, he earned his Ph.D. at the University of Michigan, and he is an active member of The HoneyNet Project and other open-source projects.

Dr. Provos is particularly interested in analyzing hacker payloads. Most company networks are protected by firewalls. Whether or not the firewalls are successful in blocking an exploit attempt, they usually don't capture the actual exploit. Dr. Provos wanted to give malicious hackers a decoy place to attack, where he could observe the tricks and tools of their trade. Although Honeyd is a relatively small program, it filled a huge vacuum as a much-needed tool in the computer security community and quickly became the de facto honeypot.

Originally programmed for Unix and Linux systems, Honeyd was ported to the Windows environment by Michael Davis (mdavis@securityprofiling.com) of SecurityProfiling, Inc. (<http://www.securityprofiling.com>). Mr. Davis currently serves as the lead developer at SecurityProfiling, where he works on IDSs, with contributions to the Snort IDS project. He is also a member of The HoneyNet Project, where he develops data and network control mechanisms for Windows-based honeynets. Mr. Davis has ported Ngrep, Dsniff, Snort, Honeyd, libnides, and Sebek, and he is finishing up an ARPD port as this book goes to publication. If something is going to be ported from Unix to Windows in the intrusion detection or honeypot fields, there's a good chance Mr. Davis is doing the hard work. As is common among members of the Open Source community, both Dr. Niels Provos and Michael Davis are extremely friendly and eager to help others.

OPEN SOURCE IS FREE SOFTWARE

If you're not familiar with open-source software, it may be hard to believe that software can be free. Early on in the development of computers, most software was free to use. Today, proprietary, reimbursed software makes up the largest class of software. Proprietary software isn't necessarily all evil, as the profit motive ensures more software for us all to enjoy. Many open-source developers also offer commercial alternatives of their products, but with added enterprise features and phone support.

Open source is a step back to the days of freely available software. Open-source software may be released under different licensing terms known as Open Source Initiative, BSD, GNU, Free Software Foundation (also known as copyleft), shareware, freeware, or public domain. Here are the basic licensing terms of some of the different licensing agreements:

- *Freeware* is free to use or distribute. License terms vary for each product.
- *Public domain* software is free to use or distribute. It usually has no copyright and can even be reused in commercial programs and distributed for payment.
- *Shareware* is free to try, but you must buy it if you continue to use it after a set time period (usually 30 days). The creator retains the copyright. The software can be freely distributed (or there is a small distribution fee), but the program should not be modified in any way.
- *Open Source* can be freely copied, distributed, used, and reused in other programs. If any part of an open-source program is used in another program, the new program must follow the same Open Source rules. The code is copyrighted, and it should be referenced if used in another program.

Honeyd is released under an open-source agreement called the *4-clause BSD license* (http://www.wikipedia.org/wiki/BSD_License).

Why Use Honeyd?

As you'll learn soon, Honeyd has a multitude of options, but its plethora of configuration settings can be daunting for the first-time user. However, after you've used it a few times, you will understand the basics of how it operates, and find it fun and enjoyable.

I could have chosen an easier to configure honeypot to use in the next few chapters, but I chose Honeyd for the following reasons:

- Honeyd is the most popular honeypot in use today.
- It has more features and flexibility than most other honeypots.
- Installing and configuring Honeyd will increase your understanding of honeypots and how they function.
- Once you learn Honeyd, you will be able to optimally install and configure most other honeypots.

Honeyd's strength is its granularity and modular design. You can pick what you want it to do and when. Honeyd administrators modify its configuration as their knowledge matures or

their requirements change. Not all honeypots can grow, change, and scale as easily as Honeyd. Many honeypots are stuck mimicking the OSs they were coded to emulate. Other honeypots don't use emulation and let the host PC be directly attacked and probed.

If you would rather use a very easy to install honeypot, but without the flexibility of Honeyd, consider one of the Windows honeypot programs covered in Chapter 8.

Honeyd Features

As a low-interaction honeypot, Honeyd excels at mimicking OS IP stacks and offering up ports and services for remote hackers to probe. It can respond for any number of TCP and UDP ports, respond for one or more IP addresses, and be configured to emulate entire network topologies. On Windows systems, ports and services can be simple connections, or they can be represented by scripts and proxies. Honeyd supports a few different methods of logging activity, including text files and Syslog.

Note The current Windows version of Honeyd is a port of Honeyd Unix version 0.5. At the time of this book's release, the latest Honeyd version is 0.8b and is available at <http://www.honeyd.org>. The latest Unix version has not yet been ported to the Windows world. The Windows version lacks many features found in the Unix version, including subsystem support, asymmetric routing topologies, plug-ins, Generic Routing Encapsulation (GRE) tunneling, external machine integration, and Arpd.

IP Stack Emulation

Honeyd emulates and responds to ARP, ICMP, TCP, and UDP packets only. All other packet types are discarded. These are by far the most popular packet types, and they will be enough to fool most hackers.

Emulating the IP stack means reliably mimicking responses to network requests with the features specific to each OS. Each OS chooses different settings and values for different network packets. For instance, when you ping a computer from a Windows 2000 machine, Windows puts in a series of lowercase alphabet characters from *a* to *w* (and repeating if necessary) in the ping's payload. Other OSs use the whole alphabet or all visible characters in the ASCII character set. Honeyd keeps a separate state table for each honeypot, so settings remain predictable and reliable. The following are some of its other settings:

- IP fragmentation handling
- IP identification numbers
- IP TTL settings
- TCP window sizing
- TCP flags
- TCP network packet sequence numbering
- TCP timestamping

- UDP closed port responses
- ICMP responses

As we are all taught in TCP/IP packet classes, IP packets exist to route the upper-layer protocols—like TCP, UDP, and ICMP—from source to destination host addresses. The IP frame encapsulates the higher-layer protocol and contains its own header information. Upper-layer protocols are contained in the IP packet's data payload. When the IP frame is stripped off at the destination IP stack, the upper-layer protocol contains its own header and payload data. If you need more details, in Chapter 9, you'll find references to excellent information about TCP/IP network packet structure and settings.

Mimicking IP Information

Honeyd generates IP header information, including IP addresses, TTL information, and identification numbers. An IP packet can be broken down into smaller *fragments* if the data payload will not fit inside one packet. When fragments are received, the host collects the related fragments together and uses identifying information to determine how the smaller pieces should be reassembled back into the large data packet. A common hacker ploy (called a *frag* attack) involves sending malformed fragments that when reassembled deliver a malicious packet payload.

Each IP packet is given an *identification number* in its header, and the same identification number is used in related packet fragments that belong together. Honeyd adjusts the generation of the identification number. It can be zero, incremented by one, or a random number. You can tell Honeyd how to handle fragment packets. The default policy is to accept fragments and resolve any conflicting overlaps in favor of the older, original data.

Caution I have seen fragmented packets kill Honeyd, even when Honeyd has been set to drop fragmented packets. Honeyd recognizes the fragmented packets, logs them to the screen, and then exits. This bug has been corrected in the newer Unix versions, but not in the latest Win32 port.

Honeyd emulates the TTL IP packet setting inside virtual networks. The TTL setting is set on every IP packet and is decremented every time the packet is processed by a router (although not when processed by a switch or a bridge). This is intended to make sure that no IP packet ends up getting bounced continually around the Internet because of a routing mistake or packet malformation. If you create a virtual IP network with Honeyd, it will correctly decrement the TTL setting by one for each virtual router transverse. If the TTL reaches zero, Honeyd will correctly send an ICMP time exceeded message with the source IP address of the router that caused the TTL to reach zero.

Mimicking the TCP/IP Stack

In emulating OS TCP/IP stacks, Honeyd relies on the database files of Nmap and Xprobe2, which are probably the best tools for fingerprinting OSs and are used by many hackers. Because these tools specialize in only fingerprinting, they are stronger than anything Honeyd could do on its own. So, if Honeyd uses Nmap and Xprobe2 databases to manipulate and create traffic, it stands to reason it will do a fairly good job of fooling most hackers. And because Honeyd relies on the work of the other open-source tools, you can update Honeyd's emulation by updating the database files.

Note Even though the most popular TCP/IP characteristics are emulated, not all of them are. A hacker with intimate knowledge of what Honeyd does and doesn't emulate in the IP stack could easily fingerprint a Honeyd host. Thankfully, most hackers don't understand or check for Honeyd.

The configurations of many TCP protocol settings are based on Nmap's fingerprinting database, which is the `Nmap.prints` file in Honeyd. The fingerprinting database establishes the default characteristics for different OSs, with a focus on TCP packet responses. Listing 5-1 shows an example of an Nmap fingerprinting database entry for Windows 2000 Server with Service Pack 2 installed.

Listing 5-1. *Nmap Entry for Windows 2000 Server with Service Pack 2*

```
Fingerprint Windows 2000 server SP2
TSeq(Class=RI%gcd=<6%SI=<25224&>22C%IPID=I)
T1(DF=Y%W=5B4%ACK=S++%Flags=AS%Ops=MNNT)
T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=Y%DF=Y%W=5B4%ACK=S++%Flags=AS%Ops=MNNT)
T4(DF=N%W=0%ACK=0%Flags=R%Ops=)
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=0%Flags=R%Ops=)
T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=N)
```

Each item in the database entry tells Honeyd how it should react to particular packets and probes. The packet types and their descriptions are shown in Table 5-1. Network packet settings and flags will be discussed in Chapter 9. The Y and N answers in the Nmap database example shown here tell Honeyd whether or not a packet should be sent in response to a particular probe type or whether certain flags should be set.

Table 5-1. *TCP/IP Packet Types*

Packet Type	Description
TSeq	How to derive the TCP packet sequence numbers
T1	How to respond to a SYN packet sent to an open TCP port
T2	How to respond to a NULL packet sent to an open TCP port
T3	How to respond to a SYN, FIN, PSH, and URG packet sent to an open TCP port
T4	How to respond to an ACK packet sent to an open TCP port
T5	How to respond to a SYN packet sent to a closed TCP port
T6	How to respond to an ACK packet sent to a closed TCP port
T7	How to respond to a FIN, PSH, and URG packet sent to a closed TCP port
PU	How to respond to a probe sent to a closed UDP port

TCP Window Size

In Listing 5-1, you also see a `w=` parameter, which refers to the *TCP window size*. When two TCP hosts begin to talk, they need to negotiate an acceptable window size. The window size of a TCP packet is how much data can be sent to the receiver before it must transfer the data from its receive buffer to the waiting application. Every TCP packet acknowledged by the receiver contains an updated counter of how much buffer space is remaining before the data has to be transferred to the application. Once the buffer is full, the sender must wait until the receiver says it can receive data again. The number is expressed in a 2-byte hexadecimal value.

Most hosts begin with an initial window size, and then change it according to how much data the other communicating party indicates it can receive. Other hosts have a fixed value that cannot be changed. Usually, the receiver determines the maximum window size. In the example in Listing 5-1, the hexadecimal window size is 5B4, which equates to 1,460 decimal characters. Different OSs have different initial window sizes, and it is one of the ways to distinguish between different versions of Windows.

TCP Flags

Flag settings refer to six possible flags used in a TCP connection session. Flags are used by IP hosts to determine whether a TCP connection is starting, ending, ongoing, or other specific treatment. Table 5-2 provides brief descriptions of the different TCP flags.

Table 5-2. *TCP Flags*

Flag	Description
SYN	Synchronize is turned on when asking to establish a new connection.
ACK	Acknowledgment is used to accept a new connection and kept on while the session is active.
PSH	Push is used to tell hosts that packets have high-priority information.
URG	Urgent is used to indicate what data in the payload is considered urgent.
RST	Reset is used to close a session immediately.
FIN	Finish is used to close a session gracefully.

All TCP communication states are made up of one or more of these flags in particular combinations. All new TCP sessions begin with a three-packet negotiation sequence known as the TCP handshake. The TCP three-way handshake starts with the originating sender or requester sending a SYN packet to the remote host, asking if it can establish a TCP session. If accepted, the host sends back a SYN, ACK packet. This is the computer's way of saying, "Yes, I will accept your connection. Will you accept mine?" The original sender sends back an ACK packet to start up communications. Honeyd fully supports the TCP three-way handshake. Different flags are set (or not set) according to the OS's IP stack. The Nmap.prints database tells Honeyd which flags should be set and when for a particular OS emulation.

Note Although UDP is stateless (it doesn't have flags to determine its communication's state), Honeyd keeps track of UDP in a stateful way, so that it can track which Honeyd process is responsible for which UDP ports and responses.

Initial Sequence Number

The TSeq database entry tells Honeyd how to create an *initial sequence number* (ISN) for TCP connection establishment (SYN) sessions. Every TCP network packet contains a sequence number, which is used to keep track of data within a data stream. Both hosts on each side of the communication link pick a random ISN during the acknowledgment phase of the TCP handshake, and it is incremented during each successive related packet. Each communication partner will acknowledge the other's transmitted sequence number and send back what sequence number it expects in the next received packet. If the next packet arrives without the expected sequence number, the IP stack rejects the packet, or it waits for the correct sequence number to show up.

Man-in-the-middle attacks can succeed if the intruder can calculate and use the sequence numbers used by two communicating partners. If done correctly, an intruder can create a rogue packet and transmit it, and the receiver will accept it as valid. Early versions of many OSs, including Windows, incremented the sequence number by a fixed value that was not random at all. The lack of randomness made many OSs particularly susceptible to man-in-the-middle attacks. Today, most OSs go to great lengths to create randomly generated sequence numbers that cannot be easily predicted.

Note *Man-in-the-middle* refers to a classification of attacks that allow an unauthorized intruder to listen in on a privileged communication stream. If accomplished correctly, the intruder appears as the other legitimate party to each participating party. When a communication is sent between the two legitimate parties, it first travels to the intruder, who reads or manipulates the communication stream, and then sends it to the receiving party.

Still, there is no true randomness in the computer world, only approximations of randomness. Consequently, most “random” sequence number generations are still somewhat easy to predict. OS fingerprinting tools will capture or generate multiple packets from a computer and examine the changes in sequence number increments. Using a few math formulas, fingerprinting tools can often identify the particular OS that generated the sequence numbers. For any honeypot emulating IP stacks, this can be one of the most difficult personality traits that it can attempt to emulate. Do it poorly, and fingerprinting tools will become confused or identify the wrong system.

Timestamp

Honeyd also makes sure that a packet's *timestamp* is realistic for the OS it is mimicking. Every time a host creates a packet, it puts a timestamp in the packet. The timestamp is the current

time, reported as the number of milliseconds since midnight, measured in Coordinated Universal Time (UTC). (UTC is the successor to the Greenwich Mean Time, or GMT, standard.) If the receiver gets a packet with an unacceptable timestamp (the host timed out waiting for the packet to arrive), it drops the packet. Accordingly, Honeyd creates an acceptable timestamp value that depends on the OS's fingerprint database.

ICMP Behavior

Honeyd also mimics ICMP behavior, although emulation is limited to ICMP_ECHO (ping) replies and a few other messages (like ICMP time exceeded from a TTL timeout). By far, pinging hosts is the most common use of ICMP by an intruder.. Different OSs will send different, but predictable, types of headers and payload data in a ping request packet. When a remote computer pings Honeyd, it will respond with the appropriate request. If a UDP port is closed, Honeyd will send an ICMP port unreachable message.

An Xprobe2 database file, called Xprobe2.conf, determines how to respond to ICMP requests. It contains the ICMP behaviors for various OSs, and it is mapped to the appropriate Honeyd IP stack by the Nmap.assoc file.

ARP Proxying

ARP requests to virtual Honeyd addresses are responded to by Honeyd's host computer, not by Honeyd. This is known as *ARP proxying*. ARP broadcast queries for any IP addresses that Honeyd has reserved for itself are responded to by the host, which returns the MAC address of the host. This could be problematic if the hacker is located on the same network segment as Honeyd. If the hacker carefully analyzed the ARP responses or MAC addresses returned from the Honeyd host, he might find it suspicious that one MAC address belongs to more than one host IP address. Luckily, most hackers aren't analyzing ARP responses, and ARP responses aren't passed off the local segment. Hackers on remote segments always get (and expect to get) the MAC address of the gateway router, not the destination host. So, if the hacker is not located on the same segment as Honeyd, he will not notice anything particularly suspicious.

IP Addressing and Network Emulation

Honeyd can be assigned one or more IP addresses using Honeyd's bind command. With the help of the Arpd program (currently available only in the Unix version), Honeyd can respond for any unassigned IP address in your environment. IP addresses are bound to one or more OS personalities when defining templates, as described in the next section. Honeyd can emulate entire IP networks, composed of one or more subnets spread among many routers, each with its own latency and packet loss rate. The Windows version of Honeyd is limited to a *rooted tree* network topology model, where there is one entry point and one exit point, but most real networks follow this model anyway.

Note The newer Unix version of Honeyd supports asymmetric routes using trinary tree algorithms. It also lets you define GRE tunneling for layering one protocol over another.

When a packet heads for a destination IP address, it enters at the root (gateway) IP address and transverse networks and virtual routers until it reaches its final destination. Honeyd accumulates packet loss and latency (as defined in the Honeyd configuration file) from source to destination to determine whether a packet gets delivered (or dropped) and its delivery speed. Some honeypot administrators purposely slow down packets to and from the honeypot in order to slow down the hacker. This makes the job of the honeypot administrator a bit easier, because there is less information to deal with.

Network emulation is done well enough to fool traceroute and pathping (Windows 2000 and above) type utilities.

Tip You can get sophisticated enough with Honeyd's virtual networks to set up virtual router hosts matching your network layout. That way, when the hacker is mapping your network topology, you can offer up emulated router hosts for the hacker to attack.

Honeyd OS Personalities

To summarize, Honeyd can emulate IP packet information, sequence numbers, UDP packet headers, TCP packet headers, TCP flags, TCP window size, and ICMP responses. It can assist with ARP replies and represent one or more IP addresses, potentially making up entire virtual networks.

Honeyd refers to all these IP stack characteristic emulations as OS *personalities*. Honeyd's annotate command ties other handling characteristics, such as how Honeyd should handle fragments, to a particular personality. Before Honeyd sends any packet, it is analyzed and manipulated by the underlying personality to make sure it accurately mimics the forged OS. One instance of Honeyd can emulate one or more personalities. You can add your own custom emulations, but to find out what OSs Honeyd supports by default, open and view Honeyd's `Nmap.prints` file. It contains 17 different versions of Windows, from Windows 3.1 with Trumpet Winsock 2.0 to Windows XP and Windows Server 2003. Currently, Honeyd has the following Windows personalities defined:

- Windows 3.1 with Trumpet Winsock 2.0 revision B
- Windows for Workgroups 3.11 / TCP/IP-32 3.11b stack or Win98
- Windows NT4 / Win95 / Win98
- Microsoft Windows 95 4.00.950 B (IE 5 5.00 2314.1003)
- Windows 98SE + IE5.5sp1
- Windows NT 4 SP3
- Windows NT 4.0 Server SP5-SP6
- Windows NT 4.0 SP 6a + hotfixes
- Windows 98
- Windows 98 w/ Service Pack 1

- Windows NT 5 Beta2 or Beta3
- Microsoft Windows.NET Enterprise Server (build 3615 beta)
- Windows Millennium Edition v4.90.300
- Windows 2000 Professional (x86)
- Windows Me or Windows 2000 RC1 through final release
- Microsoft Windows 2000 Advanced Server
- Windows XP professional version 2002 on PC Intel processor
- Windows XP Build 2600
- Windows 2000 with SP2 and long fat pipe (RFC 1323)
- Windows 2000 Server SP2
- WinME
- Windows 2000 Professional, Build 2128
- Win XP Pro or Windows 2000 Pro SP2+
- Windows 2000 SP2
- Windows 2000/XP/ME
- Windows XP Pro
- Windows 2000 Professional RC1/W2K Advance Server Beta3
- Windows XP Professional RC1+ through final release

As you will no doubt note, there are many overlapping definitions. These were developed by different people and submitted during different time periods. Each has its own fine points and strengths, but you can pick any choice that is in the same class as the OS you are trying to emulate. For my honeypots, I pick a personality closest to the OS name I'm trying to emulate, rather than a general grouping. For example, I will choose Windows 2000 Server SP2 over Windows 2000/XP/ME when trying to emulate a Windows 2000 Server computer.

Note Interestingly, some OSs unintentionally introduce errors to the network packets they create. Honeyd faithfully reproduces the errors an IP stack would create, just as if it were the underlying OS.

With the IP stack emulation feature set alone, Honeyd is a formidable tool. But Honeyd is just getting started.

TCP/IP Port Emulation

Honeyd does an excellent job of mimicking different IP stack characteristics, and it has almost as much flexibility in emulating TCP and UDP ports. It can mimic any port number from 0 to 65,535, either TCP or UDP, and be characterized as simple, emulation service scripts, or proxy.

Simple Ports

A *simple port* is a rudimentary port-listening process. Simple ports will never do more than allow a remote connection to be established. No data can be sent back from Honeyd to the remote host, so simple ports are usually used to capture that a connection was made. Without a network sniffer, like Snort, running, Honeyd will log only summary information.

A simple port can be defined to respond differently depending on its status. Table 5-3 shows the different port states and their responses to connection attempts.

Table 5-3. *Honeyd Simple Port Behaviors*

Port Type	Setting	Behavior
TCP	Open	Default setting. Responds with SYN/ACK packet. The connection attempt is recorded on screen with a “Connection established...” message and in Honeyd’s log file. Remote hacker’s tool will hang if it is anything beyond a simple port scan.
	Block	No response. Defines the port as not existing, sometimes called <i>stealth</i> . In a production environment, this is often the result of a firewall, but is not a normal result of a system without a firewall. Nothing is recorded on screen for any probed blocked port, but the connection attempt is recorded in a log.
	Reset	Mimics an OS’s normal response to a nonexistent port. Sends a RST/ACK packet back, recorded on screen as a “Connection closed” message and in Honeyd’s log file.
UDP	Open	No response. Connection logged.
	Block	No response. Drop connection; not logged.
	Reset	Responds with ICMP port error message.
ICMP	Open	Default setting. ICMP reply sent.
	Block	Drop connection; no reply sent.

I usually set my default port action to be Reset or Block. When I want to capture every connection to the screen and to the log file, I use the Reset option. If I want to fake that my honeypot system is protected by a firewall, which is normal for a system attached to the Internet, I add a few blocked ports. Then my honeypot target seems a little protected, albeit weakly. Lastly, I put out a few open ports corresponding to the common ports that are normally listening on a Windows host connected to the Internet (listed in Chapter 3). If you set all of the ports to be open, hackers might be made suspicious by the number of open ports that seem to do nothing but accept connections.

For example, even though an Exchange Server will have ports 135 through 139 (RPC and NetBIOS) open, I’ll put a block on those ports. I’ll open ports 25 (SMTP), 80 (HTTP), 110

(POP), 113 (NNTP), and 143 (IMAP). This makes my honeypot seem to be protected, but leaves enough holes open to be attacked by the hacker. If you want to make the honeypot seem really weak, open ports 135 through 139.

Of course, simply offering open connections isn't enough to keep most hackers around. You need to go to the next level and offer some application responses.

Emulation Service Scripts

One of Honeyd's most flexible features is the ability to add emulation scripts that mimic application and network services. It is possible to mimic application services using just about any scripting language. For example, when a remote intruder connects to port 25 on your Exchange Server honeypot, you can offer up the same prompts that a real Exchange Server host would. If you are emulating an FTP service, you can prompt the remote intruder for names and passwords.

The more realistic you can make the service, the longer the hacker will stay around, and the more information you will be able to capture. For instance, by prompting your remote intruder for a login name and password, you can find out if she has privileged internal information. If the hacker just tries all the standard login names and passwords, then you know she is not a privileged insider. But if she is guessing only current, unique login names or using old passwords, then you know you've got bigger problems on your hands, because of the intruder's familiarity with your network.

You can create your own custom scripts, but Honeyd has dozen or so that you can download and use (<http://www.honeyd.org/contrib.php>). They include IIS emulators, FTP, POP, SMTP, and telnet. Most were written using Perl or Unix shell scripting, and they will need to be modified slightly to run on Windows systems. Chapter 7 will cover this topic in detail.

Proxy Services

Another Honeyd feature is the ability to proxy connections to other computers participating in the honeynet. This is often done to add realistic functionality to the server. Using this feature is fine, as long as you have strong data control and are capturing all the information between your honeypot and the external machine. Up until this feature, Honeyd offered very little to the hacker that could be used to compromise other systems. Once you proxy the hacker to a real computer, it is very difficult to keep a strong rein on data control.

A good example of proxying is using an external DNS server as a proxy host. If your honeypot is advertising DNS services, it will be quite perplexing if the hacker can't use the DNS server to resolve names. Writing an emulation script that would be able to resolve any name the hacker throws at it would be impossible. Instead, you can use an external DNS to respond to requests for the honeypot. Honeyd makes sure the response from the external proxy appears to be coming from the honeypot computer.

Caution Any links to external production systems off the honeynet should be evaluated carefully before using proxy services, because of the inherent additional risks involved.

HONEYD SUBSYSTEMS AND PLUG-INS FOR UNIX

Honeyd subsystems are not supported in the current Honeyd version 0.5 for Windows. Subsystems allow you to run real applications, like a fully functioning web server, on the Honeyd host. They run in the virtual IP address space and have the ability to initiate external connections. Service scripts are limited to passing text back to the hacker's screen. Scripts cannot initiate real connections, open new sockets, or bind new ports. You can try to emulate those services, but they will fail to convince the competent hacker.

For example, a subsystem is a great way to provide a working FTP service. When a hacker connects to TCP port 21, you can start an FTP service script that mimics the initial login screen and login attempts, and even lists files and directories. However, an FTP server will always need to open new ports and connections back to the client. In FTP active mode, the FTP server opens a new connection from its port 20 (the data port) to a new port (the client's original source port plus 1) on the client. In FTP passive mode, the client requests a new port on the FTP server, and if the FTP server doesn't comply, the request fails. Honeyd emulation scripts cannot reliably open new ports and connections on the fly.

In the FTP scenario, in order to open a new connection, Honeyd must be stopped, its configuration file must be updated, and then it must be restarted. As you might imagine, the hacker would have a problem accepting the Honeyd script as a real service with broken connections and delays. So, the Unix version of Honeyd will allow you to run a real application on the Honeyd host box. Subsystems can even be shared among different virtual hosts to increase Honeyd's performance in large, distributed rollouts with multiple Honeyd hosts.

Another great new feature in the Unix version is the ability to add in third-party *plug-ins*. Like any plug-in, Honeyd plug-ins are used to extend basic functionality. Unfortunately, plug-ins are a relatively new feature for Honeyd, and only a few of them exist. Perhaps, the most popular is Honeycomb (<http://www.cl.cam.ac.uk/~cpk25/honeycomb>), which automatically generates detection signatures for network intrusion devices. Honeycomb is an experimental tool for creating quick worm signatures on the fly.

In what I think is a deserved turn of events, Honeyd can even be used to make hackers attack themselves. It is possible, using a Honeyd variable called `$ipsrc`, to redirect (proxy) the attackers' commands back to themselves. A popular use of this tactic is to proxy attack attempts to port 22 (SSH) or port 135 (RPC) back to the hacker. This way, you can trick hackers into breaking into their own computers.

Templates: TCP/IP Port Setting Recommendations

On most Honeyd honeypots, every port has a default setting if not otherwise defined. In most instances, you will set a port's default behavior to the Block or Reset state. Then you will set an Open state on any ports that will mimic the Windows host you are trying to emulate. These ports were covered in Chapter 3. Lastly, consider configuring scripts and proxy ports for popular ports or for the ones on which you wish to allow remote intruders to interact. The more interaction the hacker has with your honeypot, the more useful information you will be able to capture.

A defined set of ports associated with an OS personality is collected in a Honeyd *template*. Personalities are matched to templates using Honeyd's `set` command (more on this in Chapter 6). One instance of Honeyd can run one or more templates, as illustrated in Figure 5-1. Templates

can be bound to one or more IP addresses, and you can assign one template (called the *default* template) to respond for all IP addresses that don't have a specific template assigned. You can also define several other system variables, such as system uptime, packet drop rate, and UID and GID numbers. The latter two parameters are relevant only in the Unix/Linux world.

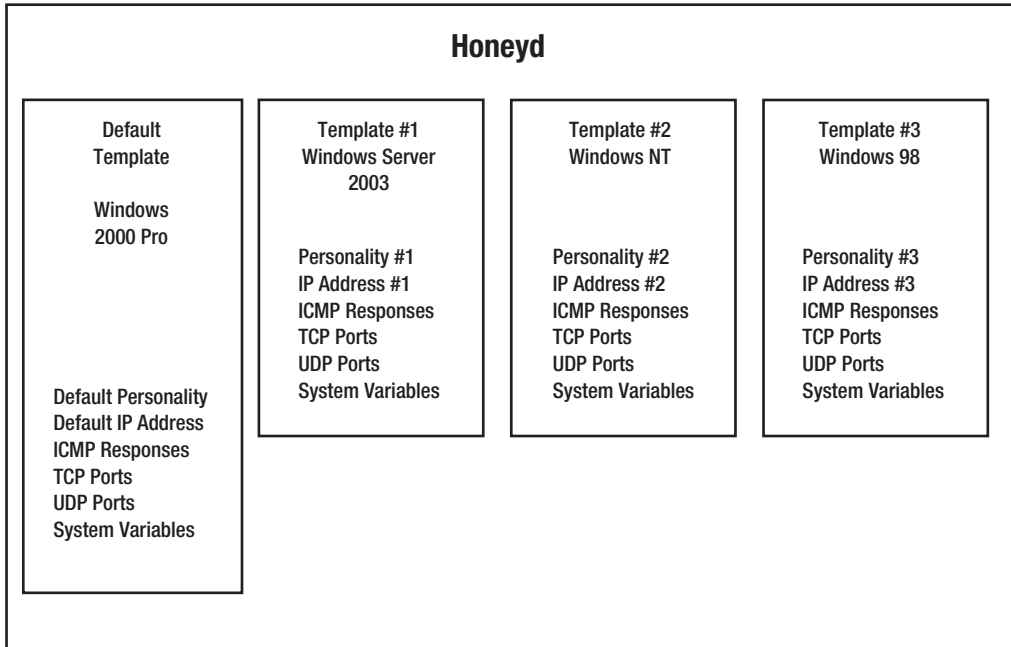


Figure 5-1. *Honeyd with multiple templates*

A single Honeyd host can emulate dozens to hundreds of different systems, each with its own IP address and behaviors. At last count, Honeyd is capable of mimicking the IP stack of more than 600 different OSs, including the most popular Windows, Unix, and Linux platforms and versions.

Honeyd Logging

Although limited, Honeyd gives you a few logging choices for noticed activity, including screen displays and Honeyd log files.

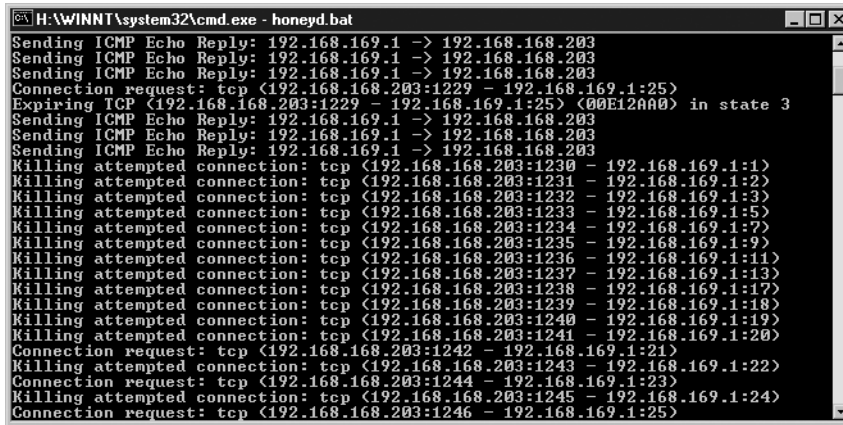
Note The Unix version of Honeyd can also be configured to send activity details to a Syslog facility, a standard log file format for recording activity and events. Syslog is nice because you can consolidate all your security logs in one place for analysis.

On-Screen Logging

If you use the `-d` command-line parameter when you start Honeyd, activity summaries are displayed on the screen, as shown in Figure 5-2. The information shown is rudimentary. Output fields include the following:

- Short activity description
- Protocol type (ICMP, TCP, or UDP)
- Source IP address
- Source port address
- Destination IP address
- Destination port address

If fragmented packets are detected, they will be noted on the screen, and the number of bytes in each packet will be displayed. Honeyd's default log file records slightly more information, but not much.



```
H:\WINNT\system32\cmd.exe - honeyd.bat
Sending ICMP Echo Reply: 192.168.169.1 -> 192.168.168.203
Sending ICMP Echo Reply: 192.168.169.1 -> 192.168.168.203
Sending ICMP Echo Reply: 192.168.169.1 -> 192.168.168.203
Connection request: tcp (192.168.168.203:1229 - 192.168.169.1:25)
Expiring TCP (192.168.168.203:1229 - 192.168.169.1:25) <00E12AA0> in state 3
Sending ICMP Echo Reply: 192.168.169.1 -> 192.168.168.203
Sending ICMP Echo Reply: 192.168.169.1 -> 192.168.168.203
Sending ICMP Echo Reply: 192.168.169.1 -> 192.168.168.203
Killing attempted connection: tcp (192.168.168.203:1230 - 192.168.169.1:1)
Killing attempted connection: tcp (192.168.168.203:1231 - 192.168.169.1:2)
Killing attempted connection: tcp (192.168.168.203:1232 - 192.168.169.1:3)
Killing attempted connection: tcp (192.168.168.203:1233 - 192.168.169.1:5)
Killing attempted connection: tcp (192.168.168.203:1234 - 192.168.169.1:7)
Killing attempted connection: tcp (192.168.168.203:1235 - 192.168.169.1:9)
Killing attempted connection: tcp (192.168.168.203:1236 - 192.168.169.1:11)
Killing attempted connection: tcp (192.168.168.203:1237 - 192.168.169.1:13)
Killing attempted connection: tcp (192.168.168.203:1238 - 192.168.169.1:17)
Killing attempted connection: tcp (192.168.168.203:1239 - 192.168.169.1:18)
Killing attempted connection: tcp (192.168.168.203:1240 - 192.168.169.1:19)
Killing attempted connection: tcp (192.168.168.203:1241 - 192.168.169.1:20)
Connection request: tcp (192.168.168.203:1242 - 192.168.169.1:21)
Killing attempted connection: tcp (192.168.168.203:1243 - 192.168.169.1:22)
Connection request: tcp (192.168.168.203:1244 - 192.168.169.1:23)
Killing attempted connection: tcp (192.168.168.203:1245 - 192.168.169.1:24)
Connection request: tcp (192.168.168.203:1246 - 192.168.169.1:25)
```

Figure 5-2. Honeyd screen activity summary example

Honeyd Log Files

Use the `-l` parameter on the command line preceding the log file name to enable Honeyd logging of basic activity information. The default log file name is `Honeyd.log`, but you can specify another name. None of Honeyd's logs contain packet capture information; they record only the bare connection basics. Listing 5-2 shows examples from the `Honeyd.log` file.

Listing 5-2. Honeyd.log File Entries

```
2003-09-30-15:24:41.0544 honeyd packet log started -----
2003-09-30-15:56:33.0387 udp(17) - 192.168.168.204 137 192.168.168.255 137: 78
2003-09-30-15:56:51.0913 udp(17) - 192.168.168.202 138 192.168.168.255 138: 235
```

```
2003-09-30-21:50:31.0776 tcp(6) - 192.168.168.203 1032 192.168.169.1 80: 48 S
2003-09-30-21:50:40.0789 tcp(6) - 192.168.168.203 1032 192.168.169.1 80: 48 S
2003-09-30-22:25:10.0686 icmp(1) - 192.168.169.1 192.168.168.203: 8(0): 60
2003-09-30-22:30:19.0440 icmp(1) - 192.168.169.1 192.168.168.160: 8(0): 84
2003-09-30-22:36:00.0961 honeyd packet log stopped -----
```

Default Honeyd logging includes the following fields (in order of their appearance):

- Date and time of logged event
- Protocol type (UDP, TCP, or ICMP)
- Source IP host address
- Source port number
- Destination IP host address
- Destination IP port number
- Total packet size (payload plus headers)
- Any TCP flags set (S=SYN, A=ACK, and so on)

It also includes Honeyd start and stop events. This can be handy when you are troubleshooting what events happened and when.

Honeyd must be exited normally in order to ensure all events are written to the log file. Honeyd will write events to the log as its memory space allocated to logging fills up, or when the program is exited normally (by pressing Ctrl-C). In the rare instances when Honeyd locks up or unexpectedly exits, all or some of the events may not be written to the log file.

Some honeypot administrators import Honeyd's log files to external databases, like SQL Server, MySQL, or Microsoft Access. I expect a future version to allow direct connections to back-end databases.

Logging is not Honeyd's strong point. The information it captures is barebones and has no built-in reporting mechanism. It is highly recommended that you also use a packet-capturing tool, like Ethereal or Snort, to capture all information headed to and from your honeypot. Chapter 10 will cover monitoring, logging, and reporting in more detail.

All in all, Honeyd is one of the most flexible defense tools you'll ever come across. It mimics IP stacks and TCP and UDP ports, and has the ability to emulate network or application services. The net effect of all this functionality is that, for a low-interaction honeypot, Honeyd does a lot. Now that you understand Honeyd's feature set, it's time to install it.

Honeyd Installation

Unfortunately, open-source tools with a multitude of functionality and flexibility are rarely easy to install. Honeyd is no exception. First-time honeypot administrators expecting the point-and-click GUI installations of most Windows programs will be disappointed. This section provides step-by-step instructions to guide you through the process, even if Honeyd is your first honeypot.

Actually, installing Honeyd by itself as a stand-alone product isn't that difficult. You download an executable, unzip it, configure a file or two, and it's up and running. But to get the fullest use of it, you need to install many other supporting files and programs and manually create moderately complex configuration files. The following steps summarize a typical Honeyd installation procedure:

1. Decide logistics.
2. Harden the host.
3. Install WinPcap.
4. Install Cygwin.
5. Install Honeyd.
6. Update the Nmap and Xprobe database files, if desired.
7. Download advanced scripts.
8. Install Snort.
9. Install Ethereal.

This section will discuss in detail all the steps needed to get the most out of Honeyd. The installation steps will assume that you have not already installed the software and that your primary system drive is C:. If drive C: is not your primary system drive, or if you want to install software to another drive, replace any reference to C: with the desired drive letter. If you have already installed a software component mentioned in the steps, it is up to you to decide whether to install it again or trust your current configuration.

Deciding Logistics

Like a carpenter that measures twice and cuts once, a good honeypot administrator does a lot of planning before configuring a honeypot. As Chapter 2 discussed, you first need to decide what you want to accomplish with your honeypot and which OSs you want to emulate. Do you want to emulate one Windows system or many? Do you want to emulate one IP host address or several? Which computer will you use to host Honeyd? Where will you place the host? What type of logging will you enable? How will you direct hostile traffic to your honeypot?

No matter where you decide to run Honeyd, it must be configured to run on its own virtual IP subnet. This is an important step that is often overlooked, or misunderstood, by first-time administrators. Honeyd must have its own IP subnet address space so that packets headed to or from it are not manipulated by the underlying Windows host's TCP/IP stack. If Honeyd were allowed to share the same IP address space as its host, the programming and packet-level driver tricks it performs would become more complex. And complexity is the antithesis of security and stability. You can use any IP address space that you like—public or private—as long as it is unique to Honeyd within your network.

Routing Problem

Giving Honeyd its own virtual IP address space adds an additional wrinkle to the setup. Since the new virtual subnet exists only in the memory space of the host computer, all remote computers will be unable to find it without routing assistance. If you have a router or firewall in front of your host, you will need to create a static mapping that routes packets headed to the honeypot through the host computer's adapter. You do not add this static route command on the Honeyd host computer; you add it on the router or firewall directing traffic to Honeyd. Static routing commands vary by router and firewall.

To add a static route to a multihomed (two or more network adapters) Windows computer, for example, use the following syntax:

```
route add -p <Honeyd network address> mask <subnet mask> <host adapter address>
```

For example, if Honeyd had a virtual IP address space of 192.168.169.0-192.168.169.255 with a subnet mask of 255.255.255.0, running on a host with IP address 192.168.168.200, you would need to make the following static route entry in a Windows multihomed computer acting as a router:

```
route add -p 192.168.169.0 mask 255.255.255.0 192.168.168.200
```

The `-p` parameter tells Windows to make the added route permanent (persistent across boots of the system).

Local Subnet Problem

Another common problem for new Honeyd administrators is that the host computer will redirect all network traffic from Honeyd to the gateway defined on the host computer's interface. This means if you contact Honeyd (for example, to do a ping test) from a computer on the same local subnet as the host computer, Honeyd will send the response back to the originator. But the host adapter will see the packet as arriving from a different subnet than its own and will forward Honeyd's response to the defined default gateway, which is usually a router or firewall. The originating host never gets a response back. This results in all connection attempts to Honeyd from computers on the same local subnet as the Honeyd host computer timing out. Hosts originating on the other side of the gateway or router will not experience this problem.

This can be a very frustrating issue when setting up Honeyd for the first time and trying to test it before making it available to the world. To fix this issue, you can do one of two things:

- Place another router in front of the Honeyd host computer, so that all other computers, local or not, are on another subnet. Reconfigure the host computer's gateway address to point to the new router. This will take all other computers off the local subnet. It is an optimal solution if you can get another router. The only downside is that adding yet another router in your network means you might need to make other static route adjustments on your other routing devices to account for the new router.
- For local testing purposes, you can temporarily configure the Honeyd host computer's gateway address to point to a local testing computer.

Most first-time honeypot administrators will run Honeyd on a host connected to a network segment on their firewall's DMZ or inside their network on the LAN. The majority of new Honeyd administrators start emulating one or just a few IP addresses, with a range of OS emulations. Advanced administrators set up a virtual honeynet emulating dozens of hosts stretched out over many subnets.

As long as your honeypot is contactable over the Internet, it will be visited. For this reason, your computer hosting Honeyd must be hardened.

Hardening the Host

Chapter 4 covered hardening the host OS. Here is a review of the most important hardening steps:

- The host should be in a location physically secure from unauthorized access.
- All patches and service packs should be applied.
- If installed, rename the administrator and guest accounts. Make sure the guest account is disabled.
- Secure the user accounts and limit them to only the ones that are necessary, and use complex passwords to protect remaining accounts.
- Use the NTFS file system to tighten file permissions.
- Uninstall unnecessary applications and services.
- Don't install Internet browsers, e-mail, word processors, or other high-risk applications.
- Maintain a clean copy of the system, in case it needs to be rebuilt.

Some honeypot administrators install a firewall in front of the host computer. This can be useful to block traffic to or from the host or Honeyd, depending on the needs of the administrator. Versions of Windows NT 4.0 and above have the ability to do IP filtering on the Windows IP stack, but this will have no effect on the traffic headed to Honeyd. When you enable the IP Filtering feature, Windows says you are enabling filtering for all adapters. But the packet-level capture driver, WinPcap (which you'll install in the next step), intercepts traffic before the Windows IP stack can manipulate it. Thus, the Windows IP Filtering option will have no effect on Honeyd. You can use this characteristic to your favor when hardening your host.

Note To enable IP Filtering, open the Control Panel and select Network Connections. Then right-click the active network connection, and choose Properties. On the General tab of the Properties dialog box, select Internet Protocol (TCP/IP), then choose Properties, then Advanced, then Options.

If the hacker compromises your host computer, consider all captured data suspect. Figure out how your host was compromised and close the hole.

Installing WinPcap

Honeyd, and many of its supporting programs, require the presence of WinPcap. WinPcap is a Win32 API for packet capturing at the driver level. It is needed to capture and inspect packets before Windows' own IP stack takes control. It is a port of libpcap, the versatile Unix API. Many programs take advantage of the WinPcap API. It can be installed on Windows computers with Windows 95 and above.

Note WinPcap's two main files are `Packet.dll` and `Wpcap.dll`.

To install WinPcap, go to <http://winpcap.polito.it> and download the latest version. (If you have an older version of WinPcap, uninstall it first.) It comes in a normal (auto-installer) package and a silent (transparent) package. Either install package works great.

To install WinPcap using the auto-installer package, follow these instructions:

1. Download the executable to your Honeyd host desktop and execute it. This will start the WinPcap Installation wizard.
2. In the WinPcap 3.0 Setup dialog box, click Next to continue.
3. In the WinPcap License Agreement dialog box, enable the check box to accept the terms of the WinPcap license. Click Next to continue. The installation will proceed rapidly and end quickly.
4. Click Next, and then click Finish to exit the WinPcap installation program.
5. Reboot your PC. Although this last step is not always necessary, it never hurts to do a reboot after a packet-level driver installation.

Note You should download and install WinPcap version 3.0 unless otherwise instructed by the product installation documentation. Newer versions of WinPcap are available, but they may cause problems.

You can confirm the successful installation of WinPcap under the Add/Remove Programs Control Panel applet, as shown in Figure 5-3. (Older versions of WinPcap installed a driver that could be seen in Network Neighborhood properties.)

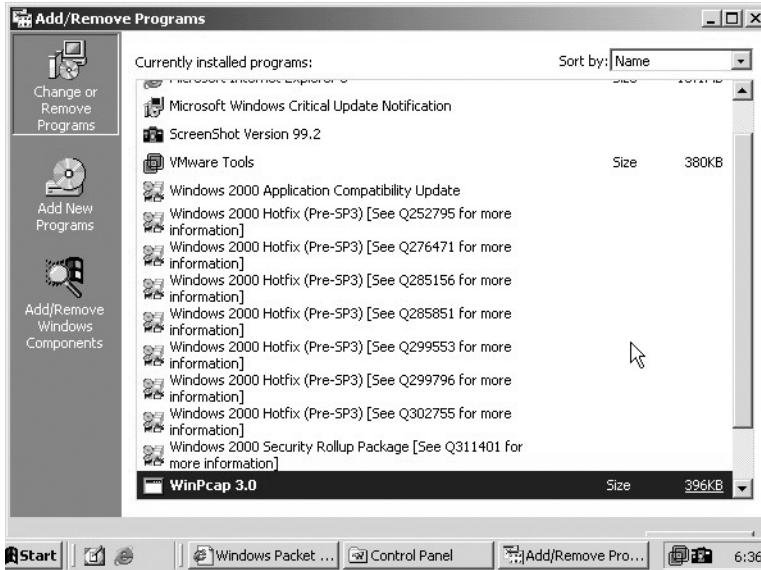
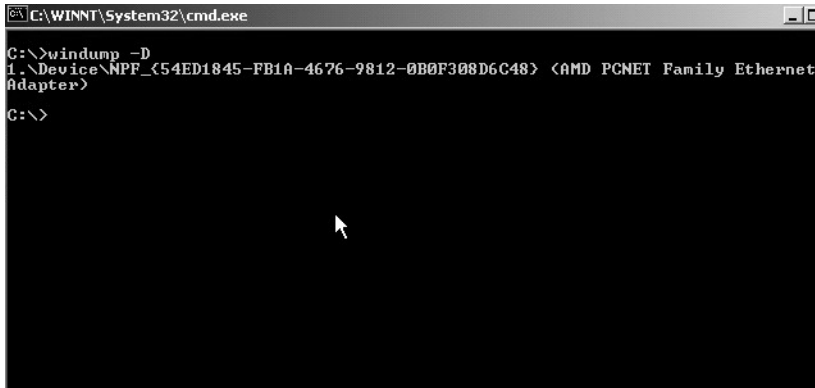


Figure 5-3. *Confirming WinPcap's successful installation in Add/Remove Programs*

There is another, more accurate, way to check for a correct installation. It's a good idea to do this extra step so that you don't need to worry about it later on if you're troubleshooting other installed components. For this method, you use WinDump, a Windows version of the Unix tcpdump utility, which captures and displays network packets.

1. Download Windump.exe from <http://windump.polito.it>. Place it in a location you can easily access from the DOS command prompt. I like to download it to C:\ so I can find it easily. (It does not have an installation routine and is executed directly at the DOS command prompt.)
2. Exit to the DOS command prompt.
3. Type **windump.exe -D** (make sure to type **-D**, not **-d**) and press the Enter key. If WinDump returns a number and the name of your interface card, along with some other less interesting information, as shown in Figure 5-4, then WinPcap is correctly installed. You can continue with the next software installation step. If WinDump does not work, try rebooting your computer (if you didn't after installing WinPcap) or troubleshooting the WinPcap installation using the documentation located on the web site (<http://winpcap.polito.it>).



```
C:\WINNT\System32\cmd.exe
C:\>windump -D
1.\Device\NPF_{54ED1845-FB1A-4676-9812-0B0F308D6C48} <AMD PCNET Family Ethernet
Adapter>
C:\>
```

Figure 5-4. *Windump.exe -D output example verifying a correctly installed WinPcap driver*

Caution Although WinPcap is known to work with most Ethernet network cards, there are a few known conflicts. WinPcap has been reported to conflict with other drivers that work at the packet level, including PGPnet and some personal firewalls. It also has been reported to cause connection problems on PPP links. WinPcap supports other types of network interface cards, like ATM, FDDI, and Token Ring, but it has not been tested widely to ensure that it works on all non-Ethernet cards.

Installing Cygwin

Cygwin is a Linux emulation environment for Windows. It consists of a large collection of Linux tools and utilities. Although Cygwin is an optional installation, it is essential for running Honeyd service scripts, since most scripts were created for Unix-like environments. Among other tools, it will install the shell scripting and Perl interpreters to run associated script files. The default installation contains over 2,000 files and over 500 subdirectories, and we need to add a few additional optional selections to get the Perl scripting engine and decompression utilities.

Follow these steps to install Cygwin:

1. The download size is roughly 26MB, so plan your download time accordingly. Go to <http://www.cygwin.com>. Click the small icon labeled Install Cygwin Now, located in the upper-right portion of the screen.
2. Save the Setup.exe file to your desktop and execute it.
3. In the Cygwin Setup dialog box, click Next to continue past the initial install screen.
4. A message may appear asking you to choose whether or not to disable your virus scanner. If so, disable your antivirus software and click Next.
5. In the Cygwin Setup – Choose Installation Type dialog box, choose the Install from Internet option, unless you have the Cygwin files installed elsewhere. Click Next.

6. In the Cygwin Setup – Choose Installation Directory dialog box, accept the default directory of C:\cygwin and the other defaults by clicking Next.
7. In the Cygwin Setup – Select Local Package Directory dialog box, click Next to accept the default location, or choose another location.
8. In the Cygwin Setup – Select Connection Type dialog box, click Next to accept the default, or choose the appropriate connection type.
9. In the Cygwin Setup – Choose Download Site(s) dialog box, choose a site to download from. Look through the available sites and choose one closest to you (or randomly pick one). Click Next to continue. You may need to do this step several times, choosing different download sites, to get the download started.
10. A Cygwin Setup – Select Packages dialog box will appear, asking you to select what Cygwin software packages to install, as shown in Figure 5-5. It has different install categories (Archive, Development, and so on). Select the Archive category and choose Install. This is to get Unix-style decompression programs, like zip file decompressors.

Caution For each category in the Cygwin Setup – Select Packages dialog box, you can click the Install action and choose to accept the Default(s), Install (all), Skip, Reinstall, or Uninstall. Be careful not to click Install in the top category, or it will install everything available with Cygwin—hundreds of megabytes.



Figure 5-5. *Cygwin Setup – Select Packages dialog box*

11. In the Cygwin Setup – Select Packages dialog box, select and expand (by clicking the + symbol) the Interpreters category. Select Perl-libwin32 and Python (scripting languages). Downloading just the binaries is fine; there's no need to download the source code.

12. In the Cygwin Setup – Select Packages dialog box, click Next. The files will download and install. You will see a percentage completion for each file download and for the whole installation. This step usually takes several minutes or longer.

Note The Cygwin download process can take over an hour if downloaded from a slow mirror site. If downloading is taking too long, cancel the process, and restart the downloading using a new mirror site.

13. In the Cygwin Setup – Create Icons dialog box, click Finish to finish. Click OK to acknowledge the end of the Cygwin setup program.
14. Go to the C:\cygwin\bin directory and verify that both the sh.exe and perl.exe files are located there. If not, repeat the previous steps.

Caution The Cygwin setup instructions change slightly every version, so the exact installation instructions may vary over time.

After the installation is complete, you need to add the C:\cygwin\bin and C:\Honeyd directories to the system PATH statement so its binaries can be accessed when needed. To do this on a Windows 2000 or above machine, follow these steps:

1. Go to the Control Panel and choose the System applet.
2. Click the Advanced tab and select Environment Variables.
3. Click the Edit button for the path system.
4. Go to the end of the current PATH statement (do not erase the current contents) and add the following text:

```
;c:\cygwin\bin;c:\Honeyd;
```
5. Click OK. Then click OK two more times to accept the changes and return to the main screen.
6. From the main screen, get to the DOS command prompt (choose Start ► Run, type **cmd**, and press Enter). Type **SET PATH** and press Enter. You should see C:\cygwin\bin and C:\Honeyd in the PATH statement.

Cygwin is an excellent learning environment for all things Unix and Linux. You can play with and install many utilities that have always been available only in the Unix world. Some honeypot users have become so excited by all the new tools and fun things to learn with Cygwin that they get distracted from their primary mission.

Note Alternately, many Perl programmers prefer ActiveState's (<http://www.activestate.com>) ActivePerl Perl engine over the one included with Cygwin. ActiveState was recently acquired by antivirus vendor Sophos. Both Cygwin's and Sophos's versions are free.

Installing Honeyd

Now you finally get to install Honeyd. Follow these steps:

1. Go to SecurityProfiling's web site (<http://securityprofiling.com/honeyd/honeyd.shtml>) and download the Honeyd binaries (honeyd-0.5-win32.zip) to your desktop.
2. Create a folder called C:\Honeyd.
3. Unzip the Honeyd binaries to the C:\Honeyd folder. Make sure to override the default subdirectory that the Honeyd binaries want to install to. Placing them in C:\Honeyd instead of C:\Honeyd-0.5 makes life a little easier.

You should have more than 70 files in C:\Honeyd and its six child folders. The Honeyd default directories are as follows:

- C:\Honeyd
- C:\Honeyd\compat
- C:\Honeyd\compat\sys
- C:\Honeyd\scripts
- C:\Honeyd\WIN32-Code
- C:\Honeyd\WIN32-Code\sys
- C:\Honeyd\WIN32-Prj

Honeyd is a command-line utility that you will be running from the DOS command prompt until you get proficient enough to trust using a batch file executed in Windows.

To test your installation, follow these steps:

1. Get to the DOS command prompt (choose Start ► Run, type **cmd**, and press Enter) and change to the C:\Honeyd folder.
2. Type **honeyd.exe -W** (case-sensitive). You should see output similar to the display from Windump.exe -D (see Figure 5-4).
3. To verify that SH is working, type **SH** and press Enter. You should be at a \$ prompt. Type **EXIT** and press Enter to exit.
4. To test Perl, change to C:\Cygwin\bin, type **Perl**, and press Enter. You should be at a blank line (this might appear locked up to you, because nothing is happening, but this is normal). Press Ctrl-C to exit.

If you saw anything other than what I've described here, you need to troubleshoot the previous installation steps.

Downloading Scripts

Honeyd's real power is its ability to emulate services using script files. Let's download some scripts to analyze and play with in Chapter 7.

1. Go to <http://www.honeyd.org/contrib.php> and download the different script files to the `\scripts` folder under `C:\Honeyd`.
2. From the DOS command prompt, switch to the `C:\Honeyd\scripts` directory. You will see that many of the script files are archived and must be uncompressed. Files ending in the extensions `.gz`, `.tgz`, and `.tar.gz` (called *tarballs*) are compressed with GNU zip (Gzip). Cygwin comes with a command-line version of Gzip. (You can find Gzip documentation at <http://www.gzip.org>.)
3. You can use a Windows utility or Gzip to uncompress the files. Archiving utilities—like Win-GZ (<http://www.crispen.org/src>), 7-Zip (<http://www.7-zip.org>), Power Archiver (<http://www.sfsu.edu/ftp/win/utills>), WinZip (<http://www.winzip.com>), and WinRAR (<http://www.rarlab.com>)—can uncompress tarballs with a friendly GUI. To use Gzip, type **gzip -d <filename>**. Gzip will automatically delete the compressed parent file. The uncompressed file could have a `.tar` extension. You can rename each file's `.tar` extension to `.pl` or `.sh`, depending on the scripting language it is written in.

Installing Snort

Snort is another essential, but optional, sidekick program for Honeyd. It can act as a packet sniffer, but, more importantly, it can serve as a network intrusion detection device. You can use Snort to monitor your honeypot links and have it alert you when it detects activity. It can use its rules to identify exploits, making your job a lot easier. It can also be used to replay attacks, if the packets were captured using a tcpdump-compatible utility (like Snort, Ethereal, or WinDump). Chapter 9 will cover Snort's configuration and use.

Follow these steps to install Snort:

1. Go to <http://www.snort.org/dl/binaries/win32> and download the latest version to your desktop.
2. Execute the Snort install program.
3. In the Snort Setup License Agreement dialog box, click the I Agree button to accept the Snort licensing agreement (after reading it, of course).

4. In the Snort Setup Installation Options dialog box, select the “I do not plan to log to a database” option or the “I am planning to log to one of the databases listed above” option. Click the Next button.
5. In the Snort Setup Choose Components and the Snort Setup License Agreement dialog boxes, click Next to accept the default install components.
6. In the Snort Setup Install Location dialog box, make sure the destination folder is C:\Snort and click the Install button.
7. In the Snort Setup Installation Complete dialog box, click the Close button. Click the OK button when it warns you to install WinPcap (it will do so whether or not you have WinPcap installed).

Installing Ethereal

Ethereal is an excellent open-source packet-capturing utility. It's relatively easy to install and use. Ethereal can be your best friend when trying to diagnose a honeypot runtime problem or when capturing malicious hacking packets. We will cover using Ethereal in Chapter 9.

To install Ethereal, follow these steps:

1. Download the Ethereal install executable from <http://www.ethereal.com/distribution/win32> and save it to your desktop.
2. Execute the Ethereal install program to start the installation wizard.
3. In the Ethereal Setup dialog box, click Next to continue.
4. In the Ethereal Setup License Agreement dialog box, click the I Agree button to accept the Ethereal licensing terms.
5. In the Ethereal Setup Choose Components dialog box, click the Next button to accept the default install components.
6. In the Ethereal Setup Install Location dialog box, click the Install button to accept the default install location, C:\Program Files\Ethereal.
7. Click the Close button when the installation completes.
8. Double-click the new Ethereal icon to test the installation. Choose Start in the Capture menu and click the OK button. Create some network packet activity (for example, start a web browser and surf to any Internet web site). Click the Stop button. Packet activity should populate your screen, similar to the example shown in Figure 5-6.

Well, we are finished installing Honeyd and all its support software. The hard part is over.

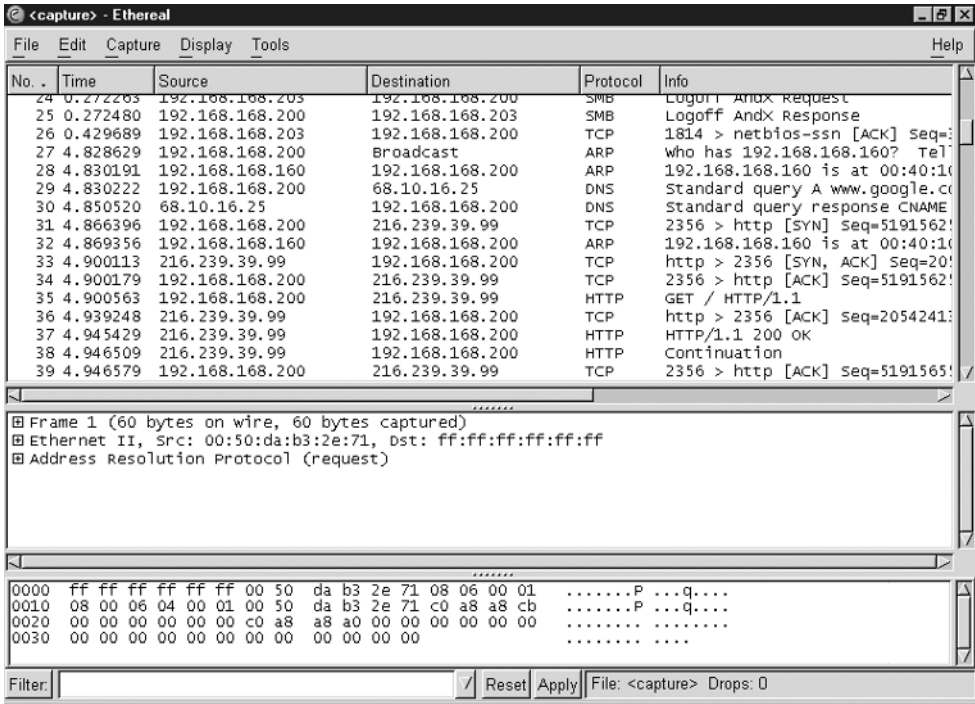


Figure 5-6. An *Ethereal* screen

Reviewing the Honeyd Directory Structure

During any of these installation steps, you can choose to install to a different drive letter and directories. Table 5-4 shows the directories I’ve recommended. If you deviated from these default drive letters or directories, replace your drive letter and directory any time the book references these subdirectory structures.

Table 5-4. *Recommended Honeyd Directories*

Directory	Contents Installed
C:\Honeyd	Honeyd honeypot software
C:\cygwin	Cygwin emulation program and related executables
C:\Snort	Snort Intrusion Detection System
C:\Program Files\Ethereal	Ethereal network traffic analyzer

Summary

This chapter first introduced Honeyd and summarized its low-interaction feature set. Honeyd is able to mimic the IP stack personality of hundreds of different OSs. Honeyd needs its own virtual IP network, and it can emulate one host or thousands of hosts. It can emulate any TCP or UDP port number with open, closed, or blocked states with simple port listeners. Port applications can be mimicked by installing service scripts and proxies (and subsystems in the Unix version).

Next, you installed Honeyd. That process actually involved many steps, including downloading and installing other support programs.

Now that you have Honeyd on your computer, your next step is to configure it. Chapter 6 discusses in detail how to configure and use Honeyd.