Honeypots for Windows

ROGER A. GRIMES

Apress®

Honeypots for Windows

Copyright © 2005 by Roger A. Grimes

ISBN (pbk): 1-59059-335-9

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Jim Sumser
Technical Reviewers: Alexzander Nepomnjashiy, Jacco Tunnissen
Editorial Board: Steve Anglin, Dan Appleman, Ewan Buckingham, Gary Cornell, Tony Davis, Jason Gilmore, Chris Mills, Dominic Shakeshaft, Jim Sumser
Assistant Publisher: Grace Wong
Project Manager: Sofia Marchant
Copy Manager: Nicole LeClerc
Copy Editor: Marilyn Smith
Production Manager: Kari Brooks-Copony
Production Editor: Kelly Winquist
Compositors: Kinetic Publishing Services, LLC; Dina Quan
Proofreader: Katie Stence
Indexer: Carol Burbo
Artist: Kinetic Publishing Services, LLC; Dina Quan
Cover Designer: Kurt Krames
Manufacturing Manager: Tom Debolski

Distributed to the book trade in the United States by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013, and outside the United States by Springer-Verlag GmbH & Co. KG, Tiergartenstr. 17, 69112 Heidelberg, Germany.

In the United States: phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders@springer-ny.com, or visit http://www.springer-ny.com. Outside the United States: fax +49 6221 345229, e-mail orders@springer.de, or visit http://www.springer.de.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

The source code for this book is available to readers at http://www.apress.com in the Downloads section. You will need to answer questions pertaining to this book in order to successfully download the code.

CHAPTER 11

Honeypot Data Analysis

To most administrators, analyzing the collected data is the best reason for running and managing a honeypot. The honeypot is a means to an end. Each forensic analysis is its own “police investigation,” with puzzles and clues. Depending on how you conduct your analysis, you can either look like a seasoned pro or a bumbling Inspector Clouseau.

This chapter will discuss how to analyze honeypot data, covering a range of tools and techniques. It also includes examples of the analysis of two real-life honeypot systems, to demonstrate the various methods.

Why Analyze?

Why should you bother analyzing the honeypot data? Isn’t it enough to simply scan the log files, note the events that seem interesting, make some quick conclusions, and react rapidly? Well, that’s one way of looking at it, and if you are operating a honeypot for the sole purpose of acting as an early warning system (EWS) for your network, that may be the best first approach.

An EWS honeypot is placed inside your network to warn you of hackers and malware that go through your other defenses. It should alert the administrator to any traffic it receives. Since an EWS’s primary purpose is early warning, rapid response is a natural conclusion. Learn about the hacker or malware, stop it, and then close the hole that allowed it into your network in the first place. But even a rapid response requires a more sophisticated approach.

Here are a few questions to ask yourself before the analysis:

- What is the primary purpose of the honeypot?
- What are you trying to protect?
- Are you interested in any attacks, or just ones that could be successful?
- Are you interested in learning how the hacker or malware was initially successful?
- Are you interested in identifying the hacker or origination point of the malware?
- Are you interested in what the hacker or malware did (or wanted to do) after the initial exploit gained entrance to the honeypot?
- Are you interested in what tools, techniques, or mechanisms were used?

EWS honeypot administrators are more concerned about how the initial exploit happened than what the hacker or malware did after the attack. Administrators of high-interaction honeypots are more interested in what the hacker did after the initial exploit. By providing a rich content environment, you give hackers a place to upload and download files, and practice their craft.

Is the honeypot trying to protect specific computer assets, or is it designed to guard the whole network? If you're interested in protecting just specific assets, then attempts or attacks that would be successful only against other assets aren't as relevant. For example, if you are trying to protect web servers only, your honeypot should mimic a production web server.

If you use an EWS honeypot, then you want to close the hole that allowed the hacker or malware to be successful in the first place. A quick forensic solution is to locate the destination IP address of the attack and investigate that asset. For example, if an Internet-scanning worm begins to probe the honeypot from another local machine, investigate the originating local machine. It's the one with the weakness. How did the worm enter the system: through e-mail, an open port, a file attachment, or a malicious HTML link? What defenses did the hacker or malware bypass in order to be successful? How can it (or they) be stopped? Are more computers on the network infected or exploited? In order to answer the last question, you'll need to do an IP address distribution analysis (covered in the "Analyzing Network Traffic" section later in this chapter). With an EWS honeypot, you'll discover the hole that allowed the malware or hacker to gain entrance in the first place, close the hole, and then wait again.

The rest of this chapter will assume that you are interested in more than just closing the hole that allowed the malware or hacker to thrive. It will assume that you want to learn exactly how the exploit happened, and even more important, what happened after that.

Honeypot Analysis Investigations

Honeypot analysis actually involves three related but separate forensic investigations:

- Was the attack automated or manual?
- How did the initial compromise happen?
- What did the hacker or malware do after initial compromise?

Automated vs. Manual

Attacks appear because of roving automated malware (such as viruses, worms, and trojans), because of a specific manual attack directed by a hacker, or through some combination of these techniques. In most cases, manual attacks are more of a concern than random, automated attacks. Either type of attack can cause damage, but the manual attack is unpredictable. Most automated malware is known (zero-day attacks are not frequent). You can use an Internet search engine or an antivirus database to search on the malware and learn everything about it. In contrast, no one knows how to predict what hackers will do when they are in control of your honeypot system.

The telltale signs of an automated attack are as follows:

- Several different types of attack, in quick succession
- Exploits not designed specifically for the platform attacked

- The same attack tried over and over again in quick succession, without changing any parameters
- Typing too fast to be done by person, without any typos

The following are the telltale signs of a manual attack:

- Exploit code used is specific for the platform attacked
- Random typos in commands, with a lot of retyping
- Random periods of time between different mechanisms of attack
- Signs of prior intelligence gathering (such as pinging or port scans)

Initial Compromise

Almost all hackers and malware employ two different mechanisms of action: one used to gain initial access and the other to accomplish their true intent. Breaking in is often just a means to an end, although many hackers and malware are content to simply break in.

For example, the Slammer worm (`http://securityresponse.symantec.com/avcenter/venc/data/w32.sqlexp.worm.html`) used a buffer overflow exploit to compromise Microsoft SQL Server machines (and clients running the Microsoft Desktop Engine, or MSDE). After gaining initial access, it used the resources of the exploited machine to attack other computers with the same exploit. It contained no damaging routine and infected no files. It did no other damage than that resulting from overflowing the server and launching as many exploratory attacks against new hosts as possible. Its replicating routine so overwhelmed the exploited host and network that its spread was actually hampered as it unintentionally caused its own chokepoints. The effects of the Slammer worm could have been devastating if it had spread a little less quickly and if it had erased data.

After the Initial Compromise

The hacker's or malware's intent after the initial exploit is often more important to the honeypot administrator. Did the intruder want to compromise that particular machine, or was it just an exploitable host?

Hackers could, if they wanted to, analyze the computer's data and eventually gain access to valuable information. Imagine the damage hackers could do to a corporate network or data center by capturing passwords or silently corrupting data. Maybe they could sell the data to an interested third party, or hold the data hostage. Even most home computers contain valuable information. Computer users often access their online bank accounts and conduct online commercial transactions. If hackers wanted to, they could steal credit card information and go on a buying spree. Certainly, a small percentage of hackers do just that.

However, most hackers and malware simply want the resources of the computer. They don't know (and don't care) what computer they are breaking into. They want to use the CPU cycles and disk space. Maybe the computer will be used to store pirated DVDs, games, or other hackers' warez. Other times, the computer is commandeered to attack other computer systems, like a zombie trojan botnet.

Note *Zombie trojans* are malware programs deposited on exploited machines, which then patiently wait for commands from the originating hacker. Hackers often exploit dozens to thousands of computers with these trojans in preparation for a larger attack, making a network of bots (or a *botnet*). The entire resources of the malicious botnet can then be directed against a single computer or web site. Along the same lines, today's spammers use worms or viruses to direct otherwise innocent computers to send out millions of unsolicited messages. Some antispam resources, like MessageLabs (`http://www.messagelabs.com`), say that more than 60% of the spam delivered today is sent out by spam bots.

Computers may also be used to commit corporate crime. I was involved in a case where a competitor infiltrated a company's computer to gain competitive advantage. The hacker company was able to learn what price its competitor was bidding on different fish contracts and beat the other company every time by pennies per pound. In six months, the aggrieved company was out of business, and the competitor had stolen millions of dollars in contracts. I was able to prove the grievance in court. The harmful competitor was placed in jail, but my client's company was gone, and he was bankrupt.

As another example, The Honeynet Project (`http://www.honeynet.org/papers/profiles/cc-fraud.pdf`) recorded a credit card fraud network, including the participants, tools, and involved businesses. Although The Honeynet Project has a policy of not getting involved with law enforcement agencies, the millions of dollars of potential damage made this analysis an exception. Even more interesting was the fact that this particular crime ring, although involved in high-stakes computer crime, made no significant efforts to hide its activities. Communications were not encrypted. Network transactions happened on an IRC network using clear-text transmissions. All it took to capture the fraud was an exploitable honeypot.

Regardless of the intent of the hackers or malware, honeypot analysis should be done using a structured approach, which is the main topic of this chapter.

A Structured Forensic Analysis Approach

Low-emulation honeypots usually have only network traffic logs, IDS log files, and honeypot log files to analyze. High-emulation honeypots, such as real Windows systems or virtual machine sessions, have many other areas to monitor (as covered in Chapter 10). I'll cover all the possibilities here, which you can use for your own honeypot analysis as applicable.

Analyzing your honeypot should follow a step-by-step structured approach. In general, the steps are as follows:

1. Take the honeypot offline.
2. Save RAM contents, if possible.
3. Make copies of the hard drive.
4. Analyze captured network traffic.
5. Analyze the file system.
6. Analyze malicious code, if any.

7. Analyze the OS.
8. Analyze the logs.
9. Draw conclusions.
10. Make modifications/corrections to honeypot system, if needed.
11. Redeploy honeypot.

Taking the Honeypot Offline

Each forensic analysis session begins by temporarily taking the honeypot offline. You need to create a snapshot of the logs, network traffic traces, and the system itself. Many utilities will allow you to make complete system backups while the system is online, but in general, you want to stop the honeypot from being modified to get a snapshot in time. At the very least, this means disabling the network connection or physically disconnecting the network cable. I prefer the latter method to ensure there is no network activity at all.

Taking the honeypot offline will also prevent remote hackers from discovering your forensic analysis and instituting an offensive erasure or formatting tactic. Often, hackers will install a batch file or single command that, when executed, removes all traces of their activities and/or formats the hard drive.

At this point, you need to decide whether you want to shut down the computer to make the copy. And, if so, do you want to formally shut down the system or just power it down? Keep in mind that formally shutting down the system may flush memory buffers and erase temporary files, removing telltale traces of hacking activity.

Recovering RAM Data

Unlike in the Linux/Unix world, there are no utilities (that I know of) specifically designed to copy Windows RAM contents for forensic examination, but there are some alternative techniques. (The Linux/Unix world has Memfetch.)

One way is to be satisfied with the RAM memory portion that is written to the paging file during memory-swap operations. To do this, don't use the formal Start ➤ Shutdown method to power down the computer. Instead, shut down the system using its power button (you may need to hold in the power button five seconds or longer). This will prevent the page files from flushing data from the virtual RAM back to other disk files and keep temporary files from being cleaned up. Once the page file and temporary files are left behind, you cannot boot up on the same disk, because that risks overwriting and losing data. To recover the resulting page and temporary file data left behind, you will need to find a way to back up or examine the disk while the OS on it is not booting. In most cases, this means mounting the disk to another system as an additional slave drive. Then, using forensic tools installed on the booting drive, you can analyze the original drive.

You can also use a debugger program (covered in Chapter 12) to dump RAM memory, but these types of programs aren't made to do wholesale memory dumps. For example, you can use Windows's built-in Debug.exe to do limited investigations and to write memory to disk.

Another possible way to capture all Windows memory is to intentionally create a Windows STOP error (bringing up the Blue Screen of Death) after first previously instructing Windows to save a complete memory dump to an unused, but mounted drive. In order to do this, you must enable a Registry key beforehand:

```
HKEY_LM \System\ CurrentControlSet\Services\i8042prt\ParametersValue Name:
CrashOnCtrlScrollData Type: REG_DWORDData:(1 = enabled)
```

Then, by simply holding down the Ctrl key on the right side of the keyboard and tapping the Scroll Lock key twice, you can initiate a STOP error, and Windows will dump all memory (up to 2GB) to a previously defined location. See `http://support.microsoft.com/kb/q244139` and `http://support.microsoft.com/kb/254649` for details. The included links also contain instructions on how to examine the resulting memory dumps.

■ **Tip** Researchers are working on hardware-based solutions for capturing and storing RAM data. One such solution is called Tribble (`http://www.grandideastudios.com/portfolio/index.php?id=1&prod=14`).

Making Copies of the Hard Drive

Next, you need to save copies of the disk or image and any resulting log files. For an emulated honeypot, simply shutting down the software and copying all its related files is enough. For virtual machine honeypots, you can shut down the virtual machine session and copy the virtual hard drive and configuration settings files (see the "Virtual Machine Options" subsection). With real honeypots, a common method is to use disk-cloning software, like Symantec's Norton Ghost (`http://www.symantec.com/ghost`) or server-based deployment tools, like Microsoft's Automated Deployment Services (`http://www.microsoft.com/windowsserver2003/technologies/management/ads/default.mspx`). Using Norton's Ghost is a popular choice for making identical images in the Windows world, but forensic experts often use more specialized tools.

No matter how the drive or image is copied, you probably want two images. One can be for analysis, and the other can be stored as an unaltered copy. If there is a possibility that the drive or image copies may be used in court, save the copies to unalterable media (for example, a write-once CD-ROM disc) and make a before-and-after comparison checksum. Some utilities will do a hash on the whole disk; others file by file. Commercial forensic tools will automate this process. (Utilities that do hashing are discussed in the "Analyzing the File System" section later in this chapter.)

The Dd.exe Command-Line Tool

One of the most popular disk-copying utilities for forensic investigations is the command-line tool, Dd.exe. You can find it at `http://uranus.it.swin.edu.au/~jn/linux/rawwrite/dd.htm`, or as part of the UnxUtils package (`http://unxutils.sourceforge.net`) or Cygwin (`http://www.cygwin.com`). Dd is a Windows port of a popular Unix utility of the same name. Dd's claim to fame is its block-for-block copying between two media. You can copy a disk partition to another disk, multiple floppy disks, or other media. Dd is perfect for copying data from a disk or partition to another disk or partition when you cannot physically clone the disk. You can even copy drive images over mapped network shares.

The basic syntax of Dd is as follows:

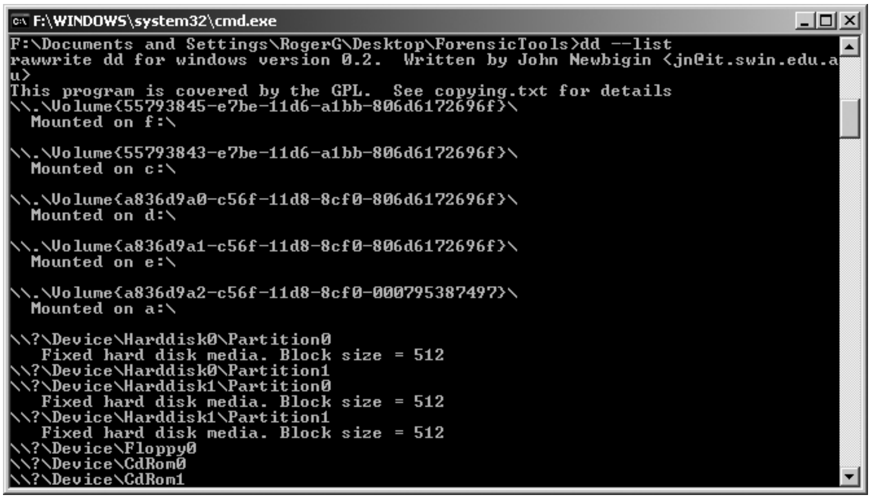
```
dd if=\<sourcemediadevice> of=\<destinationdirectory>
```

The `if` argument specifies the data input, which can be a single file, part of a file, a partition, or a logical or physical disk. Dd can make a byte-for-byte copy of nearly any data structure, except RAM (the Unix/Linux version can copy RAM contents). The `\\.` parameter can be used to indicate a local computer. The `of` parameter specifies the output file. The `bs` option specifies the block size (needed for copying to tape media).

To use Dd, you must refer to all disks and storage devices (disk partitions, CD-ROMs, and USB devices) using their drive letters, physical drive number (start counting at 0), or unique Windows-assigned GUID numbers. The resulting Dd syntax and GUIDs can be tricky for a first-time Dd user. Use the following command to list all available storage devices and their GUIDs:

```
dd  --list
```

The outputted list will display found storage devices, their drive letters (if assigned), GUIDs, and block size, as shown in the example in Figure 11-1.



```
F:\WINDOWS\system32\cmd.exe
F:\Documents and Settings\RogerG\Desktop\ForensicTools>dd --list
rawwrite dd for windows version 0.2.  Written by John Newbigin <jn@it.swin.edu.a
u>
This program is covered by the GPL.  See copying.txt for details
\\.\Volume{55793845-e7be-11d6-a1bb-806d6172696f}\
  Mounted on f:\

\\.\Volume{55793843-e7be-11d6-a1bb-806d6172696f}\
  Mounted on c:\

\\.\Volume{a836d9a0-c56f-11d8-8cf0-806d6172696f}\
  Mounted on d:\

\\.\Volume{a836d9a1-c56f-11d8-8cf0-806d6172696f}\
  Mounted on e:\

\\.\Volume{a836d9a2-c56f-11d8-8cf0-000795387497}\
  Mounted on a:\

\\?\Device\Harddisk0\Partition0
   Fixed hard disk media. Block size = 512
\\?\Device\Harddisk0\Partition1
\\?\Device\Harddisk1\Partition0
   Fixed hard disk media. Block size = 512
\\?\Device\Harddisk1\Partition1
   Fixed hard disk media. Block size = 512
\\?\Device\Floppy0
\\?\Device\CdRom0
\\?\Device\CdRom1
```

Figure 11-1. *Example of* `dd --list` *command output*

Here are some other examples of Dd commands:

- To make an image of a D: drive and save to a hard drive file called C:\fdisk.img:

  ```
  dd if=\\.\d: of=c:\fdisk.img
  ```

- To copy a slave IDE drive (the second hard drive) to another drive and file called DDriveimage.dat:

  ```
  dd if=\\.\PhysicalDrive1 of=f:\DDriveimage.dat
  ```

- To restore a saved disk image to the D: drive:

  ```
  dd if=c:\fdisk.img of=\\.\d:
  ```

- To make a copy of a USB memory device to C:\images\usb1.img:

  ```
  dd if=\\.\Volume{e29588c0-03a1-11d8-85ff-00312158492b} of=c:\images\usb1.img
  ```

Other Disk-Copying Tools

Other shareware disk-copying programs are available, including DCF Software's Hard Disk Copy (`http://www.pcworld.com/downloads/file_description/0,fid,1175,00.asp`) and HD95Copy (`http://www.pcworld.com/downloads/file_description/0,fid,4171,00.asp`). However, neither of these utilities has the feature set and ease of use offered by the commercial software programs designed for this purpose.

Alternatively, several commercial products can help you forensically examine an exploited honeypot and its disks. They usually include automatic data-integrity measurement (hashing), GUIs, and advanced features to recover deleted files and to find interesting data areas (slack space). These include the following programs:

- EnCase (`http://www.guidancesoftware.com/products/EnCaseForensic/index.shtm`)
- Winhex (`http://www.x-ways.net/winhex/forensics.html`)
- ProDiscover (`http://www.techpathways.com/ProDiscoverWindows.htm`)
- SafeBack (`http://www.forensics-intl.com/safeback.html`)

Guidance Software's EnCase ($2,495) is considered a best-in-class forensic tool, and it does far more than data copying. It has an intuitive GUI, which starts with the media acquisition module. It can copy data from any Windows-mountable storage media (but not RAM), and it supports every file system that Microsoft offers. EnCase even has a separately available module that will decrypt EFS-encrypted files (it does so by recovering the local administrator's password, which is then used to recover the EFS recovery agent key). It has a highly optimized search engine, supports Outlook PST file support, and supports multiple languages (converting them, if it can, using Unicode translation). All data recovered is verified by an MD5 hash and is verified again each time the data is analyzed. Found images can be displayed in a gallery view, and files can be displayed by timeline. EnCase has a scripting language and a plethora of reports. If you do forensic analysis for a living or if you want the very best, you want EnCase.

Virtual Machine Options

If you are using virtual machine software for your honeypot, you have additional disk-copying options:

- Because virtual machines use host disk space for their storage and memory space, it is possible to analyze honeypot data while the system is running, without affecting the data itself. Furthermore, because virtual machines end up swapping programs and data into and out of disk swap virtual memory areas more than a normal system, there is a greater chance that RAM data will be found on the disk image.
- Virtual machine software supports "undoable" or "differencing" disks, which contain only the differences between the original image and its current state. If you use these, you can search and analyze just the portions of the disk that have changed, rather than examining the whole disk.

■ **Tip** In VMware, you can choose between two types of disk: virtual and raw. Although virtual disks are the popular choice, using raw disks will make honeypot forensics easier.

An excellent whitepaper (`http://honeypots.sourceforge.net/monitoring_vmware_honeypots.html#raw_disk_virtual_disk`) discusses the unique peculiarities of virtual machine honeypots in the forensic process.

Analyzing Network Traffic

Network traffic analysis is the quickest way to get a good overall snapshot of what happened to a honeypot system. It will be used two ways: to gather summary statistics and to look at packet payload data for details. Unless the traffic was encrypted (which it often is), a packet sniffer will reveal every command and file that passed between the remote hacker and your honeypot.

Start every network analysis by gathering all the network packets into one big, sequential file. In many cases, you will have multiple trace files, saved over one or more days, from multiple systems. Gather them all into a single file so the data can be aggregated and then analyzed. Most network packet analysis tools support the tcpdump file format, so convert captured packets to that file format. Next, use the collected data to make summaries:

- How many packets in total were captured?
- Which IP addresses were involved?
- Who was talking to whom?
- Which ports were involved?
- What was the time distribution between captured packets?
- What were the packet sizes?

Determining the Number of Collected Packets

First, establish the overall number of packets collected so you know whether you are dealing with hundreds of packets or millions. It is not unusual for a single honeypot to generate hundreds of megabytes of packets in a single day. Zip compression typically gives me a 3:1 compression ratio, which still means that log files can be difficult to e-mail to colleagues. WinDump (`http://windump.polito.it/default.htm`), in its default mode, will tell you how many packets it has captured when you exit the program. Most graphical network sniffers will tell you how many packets they have captured on their main screen or in a report. In Ethereal, with the capture file loaded, choose the menu option Statistics ➤ Summary.

■**Tip** Linux, Unix, and Solaris users should consider using open-source Argus (`http://www.qosient.com/argus`) for network traffic analysis.

Identifying the IP Addresses and Top Talkers

In most honeypots, the amount of network traffic data collected is unwieldy to examine packet by packet. Instead, you need to analyze and prioritize the data to pull out the most significant traffic. Remote computers that talked to your honeypot system only once or sent packets without any payload data are the least appealing. To help find the most interesting candidates, sometimes known as *top talkers,* you need to summarize.

Learn which IP addresses were involved and how many packets were sent to each host. You can do this using several different tools. In Ethereal, you can summarize involved IP addresses by choosing the menu option Analyze ➤ Statistics ➤ Endpoints. Other products may call it an IP address distribution report. Once the endpoints are identified, you want to identify the top talkers—who talked to whom the most? In Ethereal, choose Statistics ➤ Conversations.

Learning Which Ports Were Involved

A port analysis will reveal which ports were probed and which protocols were tried. In Ethereal, you can sort data on the main capture screen by clicking the Protocol column heading, and then paging down through the various protocols. You can get a quick distribution screen report by choosing Statistics ➤ Protocol Hierarchy from the menu. The screen report displays summary statistics by all layers of the OSI model. Even content, such as JPG files, will be decoded (if possible) and summarized. In particular, I look for commonly attacked ports, such as RPC, SMTP, HTTP, SQL Server, and whatever is the port number of the day's most common Internet-scanning worm.

Analyzing Packet Time Distribution

Next, do a time distribution analysis. What packets were sent and when? What you are looking for is how often a particular remote computer sent data to the honeypot. Ultimately, you want to classify attacks as manual versus automated, and to look for persistent attackers. If the probes are coming in several per second, the tool being used is automated or it is a malware attack.

I've even seen port scans split over several different origination IP addresses in order to fool firewalls and IDSs. Most firewalls and IDSs will alert or document port-scan attempts. Although what defines a series of sequential packets as a port scan differs between products, usually it is defined as some number of probe packets arriving from the same source IP address in a few seconds. Some hackers, in order to defeat port-scan detection mechanisms, will send probe packets several minutes apart or will send each probe packet from a different source address. When I reviewed my honeypot logs during the time I was writing this chapter, I noticed sequential port scans coming from the same source network block of addresses, though each source IP address was different. It was not random that my honeypot received port probes to varying (never repeating) port numbers from a group of computers with nearly the same IP address. I realized it and documented the attacker's intent.

Filtering by Packet Size

When you are trying to focus on the most important packets, at least during the initial analysis, it's helpful to filter out packet data with very small payloads. Packets with payloads smaller than 64 bytes are not as likely to have important information as packets with bigger sizes. Initially, filter out packet sizes equal to zero and see how much data this trims from the network packet capture file. Then filter out packets with payload data less than 64 bytes. It is important to remove these filters when analyzing the whole stream, but during the initial analysis, it's helpful to get rid of handshake sequences, meaningless broadcasts, and protocol overhead (noise).

Discerning Patterns

Once you have done the summary analysis, you will see the patterns that deserve more attention. You will see groupings of probes and attacks. For example, you will be able to pick out IIS attacks from MS SQL buffer overflows. And while you might have been able to do this without all the summary analysis, getting a feel for the larger picture will help you focus on the juiciest attack patterns. Why waste your time analyzing 100 SQL Server attack packets that would never have been successful, while missing the 2,000 successful packets involved in an HTTP open-proxy attack used by spammers? Using a structured approach ensures you will always start out on the right foot and be able to discern the bigger trees in the forest.

Performing String Analysis on Packets

I like to do a string analysis on the network packets to see what I can find. Unless the network communications were encrypted, they usually contain a lot of juicy information. I look for login names, password information, English words, program names, and host names.

Use a program like Sysinternal's Strings.exe (`http://www.sysinternals.com/ntw2k/source/misc.shtml#strings`) to search for text within the network packets. Most sniffers allow the captured network data to be saved to one big, searchable ASCII file (for example, in tcpdump format) that can then be searched in the same way as any regular text file.

Tracking the Hacker

Once you've identified the source IP addresses of who is attacking your honeypot, you may want to investigate the intruder a little. The first thing most administrators do is to resolve the IP address to a domain name (if that is not already done). You can try using Windows's interactive command-line Nslookup.exe program to query DNS about a hacker's domain name or IP address information. Sysinternal's Hostname utility (`http://www.sysinternals.com/ntw2k/source/misc.shtml#hostname`) will convert IP addresses and host names to the other's form. TamoSoft's SmartWhois (`http://www.tamos.com/products/smartwhois`) is the best whois query tool I've used. At $29 per copy, it's a bargain. TaFWeb Whois (`http://www.tafweb.com/whois.html`) is free, although no longer updated. It specializes in tracking down foreign domain registrations.

You may also want to fingerprint the remote hacker's or malware source computer. You can use the same tools discussed in earlier chapters (such as nmapNT or Xprobe), but these might alert the hacker to your presence. It might seem suspicious to the hacker that the Windows 98 computer that he just probed is now initiating a port scan or active fingerprinting tool. If you need to identify the remote computer, consider using the P0F utility (`http://lcamtuf.coredump.cx/p0f-win32.zip`). P0f examines hacker's traffic to discern what OS he is running. It is smart enough to detect firewalls and to summarize their physical connection type and ISP. See `http://lcamtuf.coredump.cx/p0f.shtml` for details. The only problem with these types of resolutions is that they may not lead to the actual hacker. Hackers can use fake source addresses or, more often than not, are using another person's computer to launch new attacks.

I've even seen honeypot administrators post backdoor trojan programs as trojan executables on the honeypot for the hacker to download and execute. For example, the file might be called Porncrack.exe and pretend to be a pornography password-cracking database program, while it really is the Back Orifice trojan connecting back out over port 80. Although most administrators and security experts will advise against such techniques, they might prove useful if approved by a court order.

Analyzing the File System

Analyzing the file system means looking for additions, changes, and deletions to the honeypot's files and folders. It also means looking for data not stored in files, such as data in alternate data streams and in unused areas of the hard drive.

One of the best ways to discover malicious activity is to find new files or folders placed by the hacker. Chapter 10 mentioned several utilities (such as Tripwire, WinInterrogate, and Winalysis) for inventorying files and folders, and then reinventorying them again at a later date and making a comparison. A quick and easy (but not always accurate) method is to use the Search or Find Files and Folders feature built into Windows. Tell it to search on all files or folders modified since the honeypot was first deployed.

■**Note** Some files and Registry keys are updated by Windows even when no outside activity has occurred, so not all changes are malicious. The list of what Windows modifies just because Windows is running varies by computer, so learn what is normal by documenting file-change activity in your baseline before deploying the honeypot in a production environment.

The Afind program (`http://www.securityfocus.com/tools/525`) lists files by their last access time, without tampering with the data, the way that right-clicking to view file properties in Windows Explorer will. AFind allows you to search for access times between certain time frames and collect the results. Some other utilities include FileStat, part of Foundstone's (`http://www.foundstone.com`) Forensic Toolkit. Although FileStat operates on only one file at a time, you can implement it using a batch file to take whole-drive snapshots for before-and-after comparisons.

Unfortunately, it is all too easy for a hacker or malware program to manipulate the system time or file modification data, making the analysis results invalid. You can use a file/date timestamp for your cursory check, but for real forensic work, you should rely only on file hashing. Each file on your honeypot should be hashed before it goes online and compared later after the compromise. I introduced several monitoring and logging utilities that do hashing, including Tripwire and WinInterrogate, in Chapter 10. Here are some other programs that do hashing:

- FileCheckMD5 (`http://www.brandonstaggs.com/filecheckmd5.html`) recursively scans a selected folder and creates hashing data, which can then be saved and loaded later to determine if the files have changed.
- Digital Detective's hashing tool (`http://www.digital-detective.co.uk/freetools/md5.asp`) works on one file at a time.
- Secure Hash Signature Generator (`http://www.ics-iq.com/show_item_222.cfm`) is another tool that can be used to hash disk and file images.
- SuperDIR (`http://thunder.prohosting.com/~sdir/intro.html`) lists files in the same way as the normal `DOS DIR` command, but it will calculate a CRC32 checksum for every file and dump the results.

The best hashing tools are ones where hashing is done on all files and the results saved to a database for later comparison. Most hashing programs use MD5 or SHA-1 hash algorithms. See `http://www.honeypots.net/ids/integrity-management or http://www.handyarchive.com/free/md5` for more hashing program alternatives.

Tip Watch for sound-alike files or legitimate-looking file names located in the wrong location. For instance, finding Svchost.exe in the C:\Windows\Fonts folder is suspicious.

Looking for Hidden Files and Alternate Data Streams

Hackers frequently hide their files and folders. You can use the `DOS DIR` command to look for hidden files and folders. This command will list all hidden files and folders in or under the current directory path:

```
DIR /AH /S
```

Pipe the output to a text file for review or for export. For quick checks, I look in the root directory, C:\Windows, and C:\Windows\System32 folders.

You can also use the `DOS ATTRIB` command to locate hidden, system, and read-only files. The following command will remove those attributes, which a hacker can use to complicate file discovery or removal:

```
ATTRIB <filename> -S -H -R
```

Windows Explorer will look for hidden files if configured to do so, and there are many utilities that will assist in finding hidden files and alternate data streams. Here are some of the programs that can help you find these files:

- Foundstone's (`www.foundstone.com`) free HFind tool (part of the Forensic Toolkit) will scan a disk for hidden or system files and display the last accessed times.
- Foundstone's free SFind utility (also part of the Forensic Toolkit) will list hidden NTFS alternate data streams and their access times.
- Sysinternal's free Streams program (`http://www.sysinternals.com/ntw2k/source/misc.shtml#streams`) program will list any hidden NTFS streams by file or directory.
- Crucial ADS (`http://www.crucialsecurity.com/downloads.html`) is another free utility that reveals alternate data streams. For more information about ADS, read "The Dark Side of NTFS (Microsoft's Scarlet Letter)," by Harlan Carvey (`http://patriot.net/~carvdawg/docs/dark_side.html`).

Note There are many legitimate hidden files and folders on a typical Windows computer. We, of course, are just concerned with new additions placed by malicious intruders.

Using Disk Viewers

Only a full search of the entire hard drive, files, *and* nonfile areas can reveal the ultimate changes in the honeypot. How else could you locate a boot sector virus? Hackers like to hide data and programs in the unused areas of the hard drive, called the *slack space*. Slack space can be found inside files (in unused portions not affecting the overall structure) and between files. Most forensic investigators rely on a disk editor to view the disk's raw data on a byte or sector-by-sector basis. Here are few disk viewer programs:

- The Disk Investigator (`http://www.theabsolute.net/sware/dskinv.html`) is an excellent free tool for the job. It allows you to view raw disk information and search for particular byte patterns and text strings. You can even view raw data on a file or folder basis.
- Directory Snoop (`http://www.briggsoft.com/dsnoop.htm`) is another disk viewer tool.
- Symantec's Norton System Utilities (`http://www.symantec.com/sabu/sysworks/basic`), which is now part of Norton SystemWorks, is considered one of the best disk editors.

Visit `http://www.handyarchive.com/free/disk-editor` to find more disk editor programs.

Confirming File Types

Not every file is what it seems. In Windows, most file formats are associated with a particular file extension. In the legitimate world, the file extension is a somewhat reliable indicator of the true file format. Hackers like to hide their real intent by naming files with extensions that are different from their conventional filename extensions. I've seen text files listed with .SYS extensions, executables listed as .ZIP files, and DLL files renamed to Readme.txt.

In one honeypot forensic analysis contest (`http://www.honeynet.org/scans/scan32`), the malicious executable was compressed with an archiving program called UPX (discussed in Chapter 12). The fictional hackers then removed all the normal telltale signs of its UPX-origins from the resulting compressed executable to stymie forensic analysis. The winner of the honeypot contest was able to use a utility to verify the file's true format, which he then disassembled.

If you suspect that a malicious file has the wrong extension, you can use a program specifically designed to ferret out its real contents. WhatFormat (`http://www.jozy.nl/whatfmt.html`) and File Investigator (`http://www.robware.com/index.html`) can verify the contents of more than 1,200 different file types.

■ **Tip** You can learn which file extensions are associated with which programs at `http://www.filext.com` or `http://www.file-ext.com`.

Checking Permissions

It is important to find out if any file or folder permissions changed. You can use one of the utilities introduced in Chapter 10 (such as Tripwire, Cacls.exe, or Winalysis) or other programs. Microsoft has many Resource Kit utilities that can assist in documenting permission settings, including Perms.exe (`http://www.microsoft.com/windows2000/techinfo/reskit/tools/existing/perms-o.asp`) and Showacls.exe (`http://www.microsoft.com/resources/documentation/WindowsServ/2003/all/techref/en-us/Default.asp?url=/Resources/Documentation/windowsserv/2003/all/techref/en-us/showacls.asp`).

Sysinternal's AccessEnum (`http://www.sysinternals.com/ntw2k/source/accessenum.shtml`) is a GUI listing of who has what permissions to files, Registry keys, and folders. It should be run before the honeypot is exploited and after. The results of the second running can be compared with the results of the first to point out file and folder permission changes.

Recovering Deleted Files and Formatted Disks

Hackers often delete files during the course of exploiting a system. After any exploit, look for erased files. You can manually do this using a disk editor or a special utility built for undeleting files. It's no exaggeration to say that there are more than 100 programs claiming they can retrieve deleted files and folders. Here are some examples:

- Active@ UNDELETE (`http://www.active-undelete.com`)
- Active@ UNERASER (`http://www.uneraser.com/undelete.htm`)
- Back2Life (`http://www.grandutils.com/Back2Life`)
- FinalData (`http://www.finaldata.us/products/products_overview.php`)
- Norton System Utilities (`http://www.symantec.com/sabu/sysworks/basic`)

There are even specialized undeletion tools for particular file types. Photo Retriever (`http://www.finaldata.us/products/products_photoret.php`) recovers multimedia files (audio, video, and pictures) from hard disks and removable media. JpegDump (`http://web.archive.org/web/20030207111029/http://www.tx2600.com/downloads.php`) is a free utility that recovers deleted JPEG files. In the next section, I'll cover a few programs that recover deleted e-mail. Go to `http://www.handyarchive.com/free/undelete` to find more undelete utilities.

■ **Tip** Foundstone's Rifiuti utility (`http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/freetools.htm`) will examine Recycle Bin activity.

Examining E-Mail

If the hacker uses an e-mail client on the honeypot (most don't), you can read and recover those messages as part of your forensic analysis. Here are a few tools for recovering e-mail messages:

- DBXpress (`http://www.oehelp.com/DBXpress/Default.aspx`) extracts mail and news messages from individual Outlook Express databases, including deleted or corrupted files.
- The E-Mail Detective (`http://www.hotpepperinc.com/EMD.html`) views and recovers AOL e-mail, including deleted or cached mail.
- FINALeMail (`http://www.finaldata.us/products/products_finalemail.php`) recovers Outlook Express and Eudora e-mail.
- OutlookRecovery (`http://www.officerecovery.com/outlook/index.htm`) will recover e-mail from Outlook PST files.
- Microsoft's ExMerge utility (`http://www.microsoft.com/downloads/details.aspx?FamilyID=429163ec-dcdf-47dc-96da-1c12d67327d5&displaylang=en`) will recover deleted Outlook e-mail messages when Outlook uses Exchange Server to send e-mail.
- Advanced Attachments Processor (`http://www.mailutilities.com/aap`) extracts file attachments from e-mail databases for quick forensic investigation.

Tracking Internet Explorer Activity

If the hacker uses Internet Explorer, you can track that session information, too. Here are some tools for getting that information:

- Foundstone's Pasco (`http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/freetools.htm`) reads the index.dat file in the Temporary Internet Files (TIF) folder of Internet Explorer 5 or 6, or in any other folder selected, and presents data as a synoptic table, in chronological or alphabetical order. It shows the URLs of the pages stored in cache and the dates of latest visit.
- Cache Reader (`http://www.wbaudisch.de/CacheReader.htm`) works in the same way as Pasco. Cache Reader is now also part of the Index (History) Reader for Internet Explorer 4, 5, and 6. See `http://www.wbaudisch.de/HistoryReader.htm`.
- Foundstone's Galleta (`http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/freetools.htm`) will examine the contents of Internet Explorer cookies.
- CookieView (`http://www.digital-detective.co.uk/freetools/cookieview.asp`) decodes the internal cookie data such as the date and times, and it will split the data into separate cookie records.
- Cache View (`http://www.progsoc.uts.edu.au/~timj/cv`) is a shareware program that extracts the following information about files saved in Internet Explorer, Netscape, or Mozilla browser caches: URL, size (in bytes), MIME type, last modified date, date the file was downloaded, and the expiry date. You can then open the cached files for viewing, and copy or move them out of the cache. It will even reconstruct the names and directory paths of the files for you.
- CacheInf (`http://www.winsite.com/bin/Info?500000012549`) is a freeware ActiveX control that does nearly the same thing as Cache View.
- CacheX (`http://www.pcworld.com/downloads/file_description/0,fid,7904,00.asp`) is a free *PC World* magazine utility with a Windows Explorer-like interface.

■ **Tip** NT Objective's ntoinsight's (`http://www.ntobjectives.com/freeware/index.php`) is a free webcrawler capable of quickly scanning a web site to discover site content, resources, and attributes to quickly gain an understanding of site architecture and content. You can use it to discover if a web site that is involved with your honeypot contains malicious code or not. Since worms and embedded web links (in e-mail messages) often connect to an external web server to download new malicious code, this tool can come in handy.

Checking IM Activity and File Trading

Frequently, hackers compromise a computer so it will host an IM server or file-trading service. Athough a network sniffer can capture unencrypted P2P traffic, like file transfers and instant messaging conversations, here are a few utilities for checking for this type of activity:

- Spytech's commercial SpyAgent (`http://www.spytech-web.com/spyagent.shtml`) software stealthily works with all the major IM services (AOL, Yahoo, ICQ, MSN, and others) and includes a suite of other features.
- DataGrab (`http://hometown.aol.com/datagrab`) will record IRC conversations, but is only available to law enforcement.
- IM Grabber (`http://www.bitsplash.com/products.html`) will record and index AOL conversations.
- The most powerful and complete tool I know for identifying and recording IM conversations is the commercial Akonix L7 Enterprise (`http://www.akonix.com/products/l7enterprise.asp`).
- The free KaZaA .dat Viewer (`http://www.angelfire.com/ego2/idleloop/dat_view.htm`) allows you to view and manage KaZaA data.

You may also want to consider one of the many commercial "spying" programs available for parents and suspicious spouses, which must be installed in advance to monitor activity. Once installed, these tools try to remain invisible while recording keystrokes, logging IM chat sessions, watching viewed documents, and spying on e-mail sessions. You can monitor the sessions remotely, and even have the program e-mail you with captured information. Applications of this type include Realtime-Spy (`http://www.realtime-spy.com`) and iOpus Software's STARR (`http://www.spy-software-directory.com/starr.asp`).

Finding Pornography

Many computers are exploited to hide or serve up pornography images. If you suspect a computer is storing illicit images, you can speed up the search and confirmation process using a program designed specifically to locate such content. Pictuate (`http://www.pictuality.com`) and Perkeo (`http://www.perkeo.com/e_index.htm`) can find pornography files. Both are commercial programs.

Analyzing Malicious Code

Once you find new or modified files, it's time to dig deeper. In text files, you can search for strings that contain relevant information. Executable files require disassembly for analysis.

Performing String Analysis

When you have found suspicious files, it's a good idea to look for malicious-sounding text strings embedded in them. Although hackers can easily obscure any embedded text, they often don't. I've found files with text such as "victim is online" or "your computer is toast" too many times to count. You can be sure that a program file containing the words "warez" and "greetz" wasn't created by Microsoft.

You can use Microsoft's own Search feature to look for text strings in a file, or use specialized tools such as the following:

- Sysinternal's Strings utility (`http://www.sysinternals.com/ntw2k/source/misc.shtml#strings`) is a commonly used tool. It searches for ASCII and Unicode strings.
- Foundstone's BinText (`http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/proddesc/bintext.htm`) finds text and Unicode strings in a file.
- The Ssed program (`http://www.cornerstonemag.com/sed`) extracts text, as discussed in Chapter 7.

Doing Detailed Code Analysis

Deciphering plain text files is relatively easy, but analysis becomes increasingly harder if the program is an executable file. If you are so motivated, consider disassembling the program file in its native language. Although many malware programs are written in a scripting language (such as VBScript, JavaScript, or Perl), most are fully compiled and must be broken down into assembly language to examine. Windows comes with a built-in disassembler called Debug.exe, but it is very old and unfriendly to use.

The IDA Pro Disassembler and Debugger (`http://www.datarescue.com/idabase`) has quickly become the whitehat hacker's disassembler of choice. It has everything you could need for disassembly and malicious code analysis. Of course, you'll need a thorough understanding of assembly language to use it. There are several assembly language tutorials on the Internet and available in printed book form. Classes in using IDA Pro to disassemble malware are taught by several organizations, including SANS (`http://www.sans.org`) and many local universities and community colleges. Chapter 12 will discuss malware disassembling in more detail.

■ **Tip** When examining file details, it's helpful to have a hexadecimal-to-decimal converter handy. Sysinternal's Hex2dec (`http://www.sysinternals.com/ntw2k/source/misc.shtml#hex2dec`) is a popular choice.

Although I used to do a fair amount of disassembly, I leave the hard stuff up to the experts now. It's quicker (for me) to submit the malware programs I cannot read or understand to someone who codes and disassembles for a living. This is especially true for today's mutating and encrypting malware. You can submit suspected malware examples to web sites for online analysis. For example, you can zip and e-mail files to McAfee (`http://us.mcafee.com/root/faqs.asp?faq=453`) or submit them to Symantec (`http://securityresponse.symantec.com/avcenter/submit.html`). Symantec requires that you already have Norton AntiVirus installed. Both services respond within two to seven days. Unfortunately, you usually do not get a fact-filled analysis—just a yes or no answer to whether the file was malicious.

Of course, more can happen to a Windows system than just file changes.

Analyzing the Operating System

Without modifying a single file, hackers can make Windows vulnerable to exploitation. They can turn on vulnerable services, remove passwords, create open shares, and make all sorts of adjustments. Analyzing the OS means rerunning the same tools that you used to take a baseline in Chapter 10. I'm particularly fond of Winfingerprint and WinInterrogate.

Check for the following:

- Note any Registry changes, particularly the autorun keys. Run the Autoruns (`http://www.sysinternals.com/ntw2k/freeware/autoruns.shtml`) program to list startup program changes.
- Investigate any new or unusual processes or services using Process Explorer (`http://www.sysinternals.com/ntw2k/freeware/procexp.shtml`) or PsTools (`http://www.sysinternals.com/ntw2k/freeware/pstools.shtml`).
- Look for any new network ports and the services they are connected to. As noted in Chapter 10, you can use Microsoft's Netstat utility, or one of the friendlier competitors. Foundstone's (`http://www.foundstone.com`) Fport and Vision port enumerators are popular. My personal favorite is Port Explorer (`http://www.diamondcs.com.au/portexplorer`) or the command-line OpenPorts program (`http://www.diamondcs.com.au/openports`).
- Check for pending file changes. Malware will almost always install itself in such a way that it is executed when the computer reboots. Most of the time, this means it modifies the normal autorun areas, but it can also be accomplished by using the Windows Pending Move mechanism. Used by legitimate programs, like service packs and hotfixes, programs can modify a known Registry location (HKLM\System\CurrentControlSet\Control\SessionManager\PendingFileRenameOperations) and have Windows copy, move, or delete defined files upon the next reboot. Sysinternal's PendMoves (`http://www.sysinternals.com/ntw2k/source/misc.shtml#pendmoves`) utility will reveal any pending file changes.

Analyzing Logs

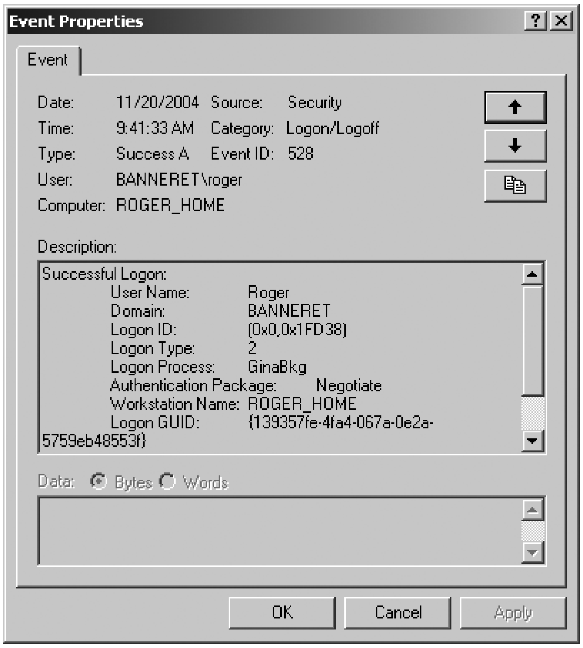
After collecting all the various log file types, now is the time to review them. Windows auditing events are great for tracking hacker activity. As discussed in Chapter 10, you should enable success and failure of Audit Logon Events and Audit Account Logon Events in the auditing policy (although for stand-alone computers, Account Logon Events are generally not applicable). Once auditing is turned on, Windows will record logon and authentication events to the Security log.

Reviewing Logon/Logoff Activity

In a real Windows honeypot, knowing when the hacker logged on and off the computer can provide useful information in the forensic analysis. Microsoft's auditing policy does a fairly good job of reporting when someone logs on and off a Windows system, as long as you can read the cryptic event ID messages.

Authentication can be NTLM or Kerberos (Windows 2000 and above). In most cases, Windows honeypots will use only NTLM authentication. Kerberos authentication requires a domain, a domain controller, and a domain account logon. Local logons into stand-alone computers use only NTLM authentication and are written as Logon events, rather than Account Logon events.

You want to look for specific event IDs, to find the success and failure logon attempts. In Windows NT, look for event ID 528, which describes a successful logon, as shown in Figure 11-2. A logoff event is event ID 538.



```
Date:      11/20/2004  Source:    Security
Time:      9:41:33 AM  Category:  Logon/Logoff
Type:      Success A   Event ID:  528
User:      BANNERET\roger
Computer:  ROGER_HOME

Description:
Successful Logon:
        User Name:          Roger
        Domain:             BANNERET
        Logon ID:           (0x0,0x1FD38)
        Logon Type:         2
        Logon Process:      GinaBkg
        Authentication Package:     Negotiate
        Workstation Name:   ROGER_HOME
        Logon GUID:         {139357fe-4fa4-067a-0e2a-
5759eb48553f}

Data: (•) Bytes ( ) Words
```

Figure 11-2. *Example of event ID 528*

Logon event messages contain a lot of detail. Table 11-1 describes the fields in the Event Properties dialog box, which appears when you double-click an event in the Event Viewer.

Table 11-1. *Logon Event Properties*

Field	Description
Date	The date on which the event occurred
Time	The time at which the event occurred
Type	The type of event: success (Success Audit) or failure (Failure Audit)
User	The security principal account involved in the logon or authentication
Computer	Account name of the computer on which the event occurred
Source	The source of the event
Category	The category of the event
Event ID	The identifier for the event
Description	A short description of the event (see Table 11-2)

The description of the event in the Event Properties dialog box provides more details, as shown in Table 11-2. Note that the last four fields in Table 11-2 may not be present for all event IDs.

Table 11-2. *Event Description Information*

Field	Description
Reason	An explanation of why the authentication failed (if it failed)
User Name	The name of the security principal account that tried to log on
Domain	The NT domain of the account that tried to log on (if there is one)
Logon ID	The domain unique identifier for a logon session
Logon Type	A numeric value indicating the logon type: 2: Interactive logon 3: Network logon 4: Batch logon 5: Service logon 6: Proxy logon 7: Unlock workstation 8: Network cleartext logon 9: Newcredentials logon 10: RemoteInteractive; Remote desktop (RDP) logon process 11: Logon on process used cached credentials
Logon Process	The name of the process that performed the logon or authentication
Authentication Package	The name of the authentication package used for the logon
Workstation Name	The account name of the workstation that the user account used for logon
Logon GUID	A globally unique logon identifier

Most honeypot logon events will be Logon Types 3, 4, or 5. When you see a Logon Type 3, you know that someone tried to access a resource on your computer from the network. When you see a Logon Type 4, you know that an account, most likely the Task Scheduler service, ran a script or program in batch mode. Logon Type 5 refers to service accounts used to log on a service. Local logons, from the keyboard (or Terminal Services in Windows 2000 or before), are recorded with Logon Type 2.

The Authentication Package field typically has the value MICROSOFT_AUTHENTICATION_PACKAGE_V1_0, Kerberos, or negotiate. The MICROSOFT_AUTHENTICATION_PACKAGE_V1_0 authentication package, also known as MSV1_0, authenticates users against the SAM database. The Negotiate value means the logon verification process was choosing between NTLM and Kerberos.

Whereas Windows NT 4.0 used event ID 528 for every type of logon, Windows 2000 and later versions use a different event ID for network logons. Network logons (mapping a drive to a server, connecting to a network printer, or otherwise connecting to a networked resource) results in an event ID 540. Unfortunately, Windows logs a lot of irrelevant (to honeypot administrators) event ID 540s, most with SYSTEM or computer (contains $ in account name). Of course, if a hacker gains LocalSystem access and begins using the SYSTEM account (which might happen in a buffer overflow situation), pay attention to the SYSTEM account events. Otherwise, just note the logon events with real user account names involved.

You can use utilities such as the following to help keep track of logon information:

- Foundstone's NTLast utility (`http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/proddesc/ntlast.htm`) will identify and track who has attempted or successfully logged on to a Windows system (it queries the event log files). It also will report on IIS authentication events.
- Sysinternal's LogonSessions (`http://www.sysinternals.com/ntw2k/source/misc.shtml#loggonsessions`) utility lists all the active logon sessions and the processes associated with each. This is helpful when logging on interactively to your honeypot to make sure the hacker is not also logged on.

Noting Other Interesting Event IDs

You are interested in more than logon events. Table 11-3 shows some event IDs to take note of when you are analyzing logs.

Table 11-3. *Interesting Event IDs*

Event ID	Description
512	Windows startup
516	Resources exhausted; security events lost
517	Security log cleared
519	Process used invalid local procedure call in an attempt to impersonate a client
520	System time changed
528	User logged on
529	Logon failure; bad name or password
530	Logon failure; outside allowed logon time
531	Logon failure; logon attempted to disabled account
532	Logon failure; logon attempted using expired account
533–539	Logon failures for various reasons
538	User logged off
540	User logged on to network
548 and 549	Logon failure; filtered SID
550	Possible DoS attack
563	Attempt was made to delete file
564	Object deleted
567	Permission was executed on object
570	Object access attempted

Event ID	Description
592	New process created
593	Process exited
595	Indirect access to an object was obtained
601	User attempted to install service or did not have permission to perform this operation
608	User right assigned
609	User right removed
612	Audit policy changed
621	System access granted to an account
624	User account created
626 and 629	Account disabled or enabled
627	User password reset attempt
628	User password set
630	User account deleted
632	Member added to global group
635	New local group created
636	Member added to local group
643	Domain policy changed
644	User account locked
645	Computer account created
658	Universal group created
660	Member added to universal group
685	Name of an account changed

These event IDs assume that you had enabled auditing policy, as recommended in Chapter 10. When collecting event logs, you should focus on these events, and add others as you become more familiar with Windows event logging. You can use Microsoft's built-in event log filtering or customize your event log reporting tool.

Tip There are a few sites on the Internet, in addition to Microsoft, attempting to fill the gap between what a Windows event log message says and what it means. One such site is `http://www.eventid.net`.

BOOTABLE FORENSIC DISTRIBUTIONS

There are dozens of bootable forensic CD-ROMs (mostly Linux-based) that you can use to examine a compromised system. You download the image from the provider, create a bootable CD-ROM disc, boot the system you want to examine from that CD, and then use the provided tools (or any you add) to analyze the system. Three popular distributions are Knoppix (`http://www.knoppix.net`), Helix (`http://www.e-fense.com/helix`), and Forensic and Incident Response Environment (`http://biatchux.dmzs.com`).

Helix is a distribution of the Knoppix Live Linux CD that includes customized Linux kernels (2.4.26 and 2.6.5), Fluxbox window manager, excellent hardware detection, and many applications. Helix has been modified to specifically not touch the host computer and to be forensically sound. Helix also has a special Windows autorun side for incident response and analysis. It is meant to be used by individuals who have a sound understanding of incident response and forensic techniques.

For an overview of all major forensic toolkits (both Unix and Windows), see `http://www.forensics.nl/toolkits`.

Drawing Conclusions

With all the evidence collected and assimilated, now is the time to draw conclusions. After you've done honeypot analysis a few times, you'll find yourself becoming interested in only what was different about the attack:

- Was a different attack method involved?
- Was encryption used?
- Is another language involved?
- Is a strange protocol being used?
- Was the attack against your honeypot specifically or just a random event (more likely)?
- What did you learn from the attack?
- How should you change your defenses to account for the new method?

Unless your computer defenses are impeccable, there are usually lessons learned and new tasks to implement.

If you are going to present this analysis to someone else (or for your own benefit), consider writing a formal report. The report should include a summary of what attacks and exploits occurred, and provide details on the interesting aspects. It is often said that a picture is worth a thousand words. Use a lot of graphs, show attack distribution analysis charts, and diagram the network flow following the steps hacker took. Refer to the Honeynet Project's Scans of the Month (`http://www.honeynet.org/scans`) if you're writing your first report.

Modifying and Redeploying the Honeypot System

After all the analysis is complete and conclusions drawn, you need to decide if the honeypot system should be modified prior to redeployment. You can make changes for several reasons, including to close exploit holes you didn't know about, to prevent the same exploit from being used successfully, or to focus the honeypot on a different objective.

I've even modified one of my honeypots to be susceptible to an attack type to which it is currently immune. For example, on one of my honeypots, I found the remote hacker persistently trying to exploit the old SQL Server SA user account exploit, where Microsoft SQL Server used to be installed with an administrator account named SA, with either a blank password or a password of `password`. I was interested in seeing what the remote hacker wanted in contacting my honeypot, so I changed the SA account password to `password` and let him exploit my honeypot. The hacker eventually copied gigabytes of hacker warez to the fake server and set up an IRC server. (Unfortunately, most of the communications and warez were in Chinese, and I didn't have a readily available translator, so I ended up clearing out the hackers and closing the hole.)

After making any desired modifications, document the changes. Update any honeypot images you have, upgrade configuration lists, and communicate the modifications to any affected staff members or coworkers. Although you might be tempted to skip this step, what might seem fresh in memory today is fuzzy or forgotten within a few weeks. So, make sure to document any modifications.

When you are ready, redeploy the honeypot and begin the cycle all over again. Make sure your honeypot system is back to its unexploited state, the logs are cleared, and monitoring tools are turned back on.

Now that you've read about all of the steps in structured forensics analysis, let's see how they can be applied. The next section of this chapter describes the analysis of two real-life honeypots.

Forensic Analysis in Action

This section uses two real-life honeypot exploits to demonstrate the structured forensic analysis of honeypots. The first is an example of a low-interaction honeypot, and the second is from a Windows 2000 real honeypot.

A KFSensor Honeypot

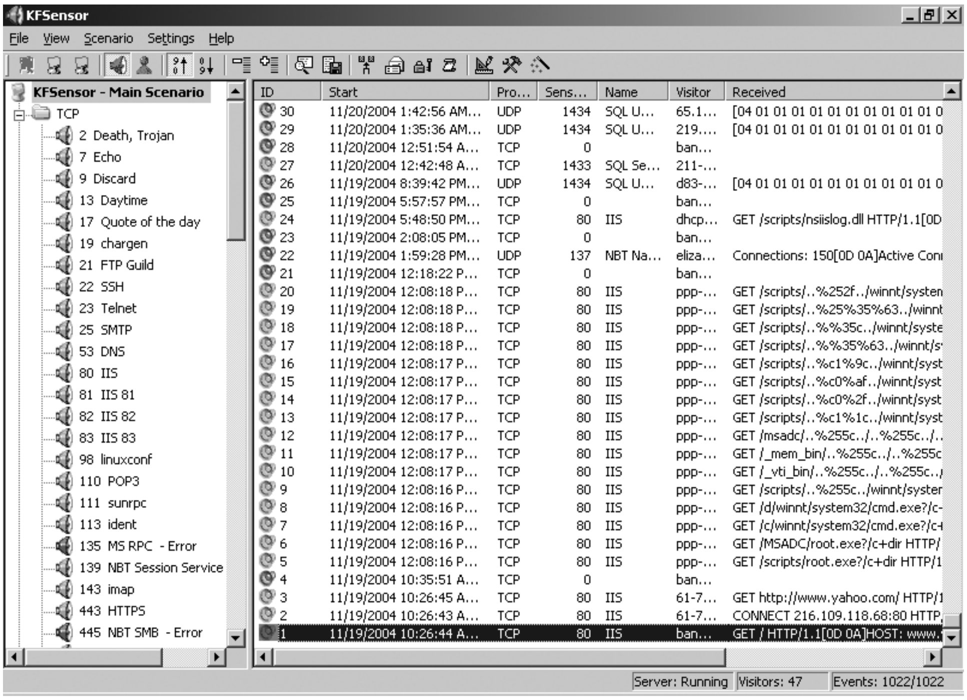
I frequently recommend honeypots as an EWS within a network—a canary in the coal mine sort of thing. Since collecting complete hacker or malware information is necessary, I can use a low- to medium-emulation honeypot. My usual choice is Honeyd (covered in Chapters 5 to 7) or KFSensor (covered in Chapter 8). If the client has the money, I'll always suggest using KFSensor. It's the best Windows honeypot offering, full of features, and easy to set up. This forensic example follows three days of honeypot activity on a KFSensor honeypot on a DMZ.

Because there were legitimate public services on the DMZ, the following ports were allowed through the external firewall: 21, 22, 25, 53, 80, 81, 443, 1433, 1434, 8080, and a few others. Ethereal was used as the network sniffer. I was using RDP (which is encrypted and authenticated by default) over nonstandard ports to administer the remote honeypot. For that reason, some traffic from my banneretcs.com domain at random intervals was detected by the honeypot.

Initial Review

The honeypot went live at 9:16 A.M. The first probe came 70 minutes after the honeynet was placed into production. Over the next three days, the honeypot recorded 1,022 different events over almost all the allowed ports, as shown in Figure 11-3. Ethereal recorded 120MB of data in almost 900,000 packets. The honeynet averaged about four packets per second.

■Note In this described session, Ethereal had to be stopped shortly after it was started to put in an Ethereal filter to filter out irrelevant bridge-related broadcasts. Filtering out legitimate traffic and fine-tuning the sniffer is a normal part of any honeypot setup.



ID	Start	Pro...	Sens...	Name	Visitor	Received
30	11/20/2004 1:42:56 AM...	UDP	1434	SQL U...	65.1...	[04 01 01 01 01 01 01 01 01 01 01 0
29	11/20/2004 1:35:36 AM...	UDP	1434	SQL U...	219....	[04 01 01 01 01 01 01 01 01 01 01 0
28	11/20/2004 12:51:54 A...	TCP	0		ban...	
27	11/20/2004 12:42:48 A...	TCP	1433	SQL Se...	211-...	
26	11/19/2004 8:39:42 PM...	UDP	1434	SQL U...	d83-...	[04 01 01 01 01 01 01 01 01 01 01 0
25	11/19/2004 5:57:57 PM...	TCP	0		ban...	
24	11/19/2004 5:48:50 PM...	TCP	80	IIS	dhcp...	GET /scripts/nsiislog.dll HTTP/1.1[0D
23	11/19/2004 2:08:05 PM...	TCP	0		ban...	
22	11/19/2004 1:59:28 PM...	UDP	137	NBT Na...	eliza...	Connections: 150[0D 0A]Active Con
21	11/19/2004 12:18:22 P...	TCP	0		ban...	
20	11/19/2004 12:08:18 P...	TCP	80	IIS	ppp-...	GET /scripts/..%252f../winnt/systen
19	11/19/2004 12:08:18 P...	TCP	80	IIS	ppp-...	GET /scripts/..%25%35%63../winnt
18	11/19/2004 12:08:18 P...	TCP	80	IIS	ppp-...	GET /scripts/..%%35c../winnt/syste
17	11/19/2004 12:08:18 P...	TCP	80	IIS	ppp-...	GET /scripts/..%%35%63../winnt/s
16	11/19/2004 12:08:17 P...	TCP	80	IIS	ppp-...	GET /scripts/..%c1%9c../winnt/syst
15	11/19/2004 12:08:17 P...	TCP	80	IIS	ppp-...	GET /scripts/..%c0%af../winnt/syst
14	11/19/2004 12:08:17 P...	TCP	80	IIS	ppp-...	GET /scripts/..%c0%2f../winnt/syst
13	11/19/2004 12:08:17 P...	TCP	80	IIS	ppp-...	GET /scripts/..%c1%1c../winnt/syst
12	11/19/2004 12:08:17 P...	TCP	80	IIS	ppp-...	GET /msadc/..%255c../..%255c../..
11	11/19/2004 12:08:17 P...	TCP	80	IIS	ppp-...	GET /_mem_bin/..%255c../..%255c
10	11/19/2004 12:08:17 P...	TCP	80	IIS	ppp-...	GET /_vti_bin/..%255c../..%255c..
9	11/19/2004 12:08:16 P...	TCP	80	IIS	ppp-...	GET /scripts/..%255c../winnt/syster
8	11/19/2004 12:08:16 P...	TCP	80	IIS	ppp-...	GET /d/winnt/system32/cmd.exe?/c-
7	11/19/2004 12:08:16 P...	TCP	80	IIS	ppp-...	GET /c/winnt/system32/cmd.exe?/c
6	11/19/2004 12:08:16 P...	TCP	80	IIS	ppp-...	GET /MSADC/root.exe?/c+dir HTTP/
5	11/19/2004 12:08:16 P...	TCP	80	IIS	ppp-...	GET /scripts/root.exe?/c+dir HTTP/1
4	11/19/2004 10:35:51 A...	TCP	0		ban...	
3	11/19/2004 10:26:45 A...	TCP	80	IIS	61-7...	GET http://www.yahoo.com/ HTTP/1
2	11/19/2004 10:26:43 A...	TCP	80	IIS	61-7...	CONNECT 216.109.118.68:80 HTTP,
1	11/19/2004 10:26:44 A...	TCP	80	IIS	ban...	GET / HTTP/1.1[0D 0A]HOST: www.

Server: Running | Visitors: 47 | Events: 1022/1022

Figure 11-3. *Main KFSensor screen showing some of the 1,022 events*

My cursory review of the KFSensor alert messages showed most traffic was to the IIS service, followed by traffic to Microsoft SQL Server ports. I could readily see buffer overflow and directory transversal attacks against IIS in rapid succession. I was surprised that I didn't have more port 25 traffic, as spammers and spam worms are rampant these days.

Analysis

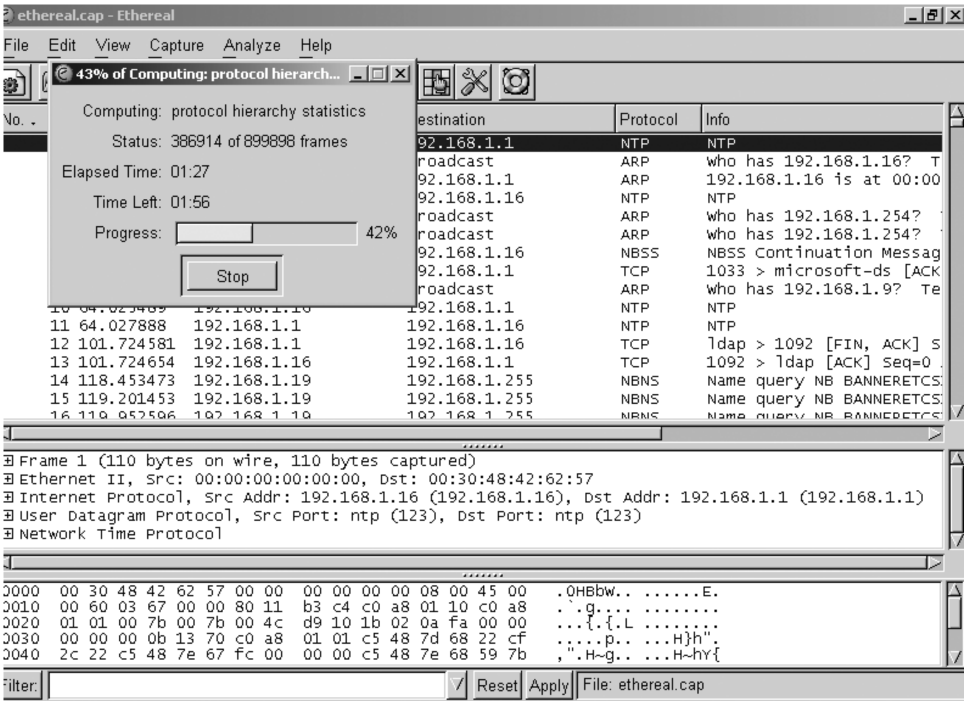
I fed the Ethereal capture files into Snort. Snort alerted on 32 different types of exploits, again most related to HTTP.

The Ethereal capture files were three separate capture files, one for each day. I used Ethereal's Mergecap.exe command-line program to merge all three files into one larger file for easier analysis. I used the following commands:

```
Mergecap.exe -v -w c:\logs\ethereal.cap - c:\ethereal_day1.cap
- c:\ethereal_day2.cap - c:\ethereal_day3.cap
```

This process took about five minutes on a mid-range Pentium computer.

I opened the larger Ethereal.cap capture file in Ethereal (the GUI product), and it took a little over one minute to load. My Ethereal summary distribution reports showed traffic came from 47 separate source IP addresses (including two that were related to my remote monitoring). A protocol distribution analysis report took several minutes to run, as shown in Figure 11-4.


ethereal.cap - Ethereal
File Edit View Capture Analyze Help
43% of Computing: protocol hierarch...
Stop
Filter:
Reset
Apply
File: ethereal.cap

```
Computing:  protocol hierarchy statistics
   Status:  386914 of 899898 frames
Elapsed Time:  01:27
Time Left:  01:56
Progress:  42%
```

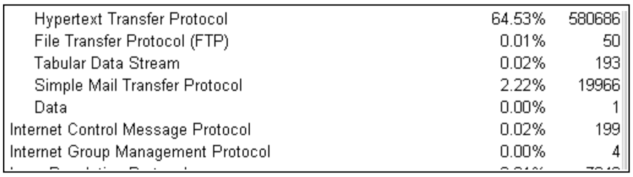
```
No. .                                      estination         Protocol  Info
                                           92.168.1.1         NTP       NTP
                                           roadcast           ARP       Who has 192.168.1.16?  T
                                           92.168.1.1         ARP       192.168.1.16 is at 00:00
                                           92.168.1.16        NTP       NTP
                                           roadcast           ARP       Who has 192.168.1.254?
                                           roadcast           ARP       Who has 192.168.1.254?
                                           92.168.1.16        NBSS      NBSS Continuation Messag
                                           92.168.1.1         TCP       1033 > microsoft-ds [ACK
                                           roadcast           ARP       Who has 192.168.1.9?  Te
                                           92.168.1.1         NTP       NTP
 11 64.027888    192.168.1.1               192.168.1.16       NTP       NTP
 12 101.724581   192.168.1.1               192.168.1.16       TCP       ldap > 1092 [FIN, ACK] S
 13 101.724654   192.168.1.16              192.168.1.1        TCP       1092 > ldap [ACK] Seq=0
 14 118.453473   192.168.1.19              192.168.1.255      NBNS      Name query NB BANNERETCS
 15 119.201453   192.168.1.19              192.168.1.255      NBNS      Name query NB BANNERETCS
```

```
⊞ Frame 1 (110 bytes on wire, 110 bytes captured)
⊞ Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:30:48:42:62:57
⊞ Internet Protocol, Src Addr: 192.168.1.16 (192.168.1.16), Dst Addr: 192.168.1.1 (192.168.1.1)
⊞ User Datagram Protocol, Src Port: ntp (123), Dst Port: ntp (123)
⊞ Network Time Protocol
```

```
0000  00 30 48 42 62 57 00 00  00 00 00 00 08 00 45 00   .0HBbW.. ......E.
0010  00 60 03 67 00 00 80 11  b3 c4 c0 a8 01 10 c0 a8   .`.g.... ........
0020  01 01 00 7b 00 7b 00 4c  d9 10 1b 02 0a fa 00 00   ...{.{.L ........
0030  00 00 00 0b 13 70 c0 a8  01 01 c5 48 7d 68 22 cf   .....p.. ...H}h".
0040  2c 22 c5 48 7e 67 fc 00  00 00 c5 48 7e 68 59 7b   ,".H~g.. ...H~hY{
```

Figure 11-4. *Ethereal generating a protocol distribution report*

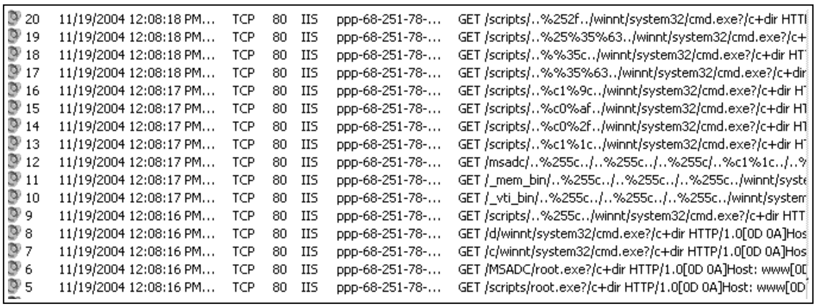
The protocol distribution report revealed that HTTP requests accounted for nearly 65% of all traffic, as shown in Figure 11-5, followed by small amounts of SMTP and FTP traffic.



Hypertext Transfer Protocol	64.53%	580686
File Transfer Protocol (FTP)	0.01%	50
Tabular Data Stream	0.02%	193
Simple Mail Transfer Protocol	2.22%	19966
Data	0.00%	1
Internet Control Message Protocol	0.02%	199
Internet Group Management Protocol	0.00%	4

Figure 11-5. *Portion of Ethereal protocol distribution report*

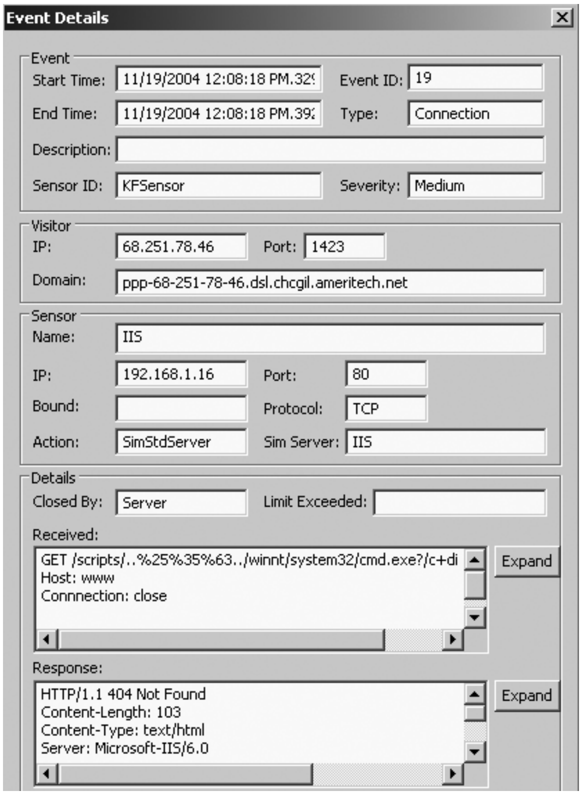
Looking at the KFSensor logs, I saw that the first sustained attack was an IIS buffer overflow and directory transversal attack. As Figure 11-6 shows, the honeypot got 15 different exploit attempts in two seconds. This told me that the attack was either automated by malware or by a scanning script. The attacks might have been directed because all exploit attempts were Windows and IIS-related, but the fact that they were automated decreased the chances of a directed attack. There were no Apache exploits in the bunch. Because of the C:\Winnt directory reference in most commands, I saw that the hackers were attempting to attack either Windows NT 4.0 and IIS 4 or Windows 2000 and IIS 5. Windows XP Professional and Server 2003 use C:\Windows as the default OS directory name, unless the computer was upgraded.



```
20  11/19/2004 12:08:18 PM...  TCP  80  IIS  ppp-68-251-78-...  GET /scripts/..%252f../winnt/system32/cmd.exe?/c+dir HTTI
19  11/19/2004 12:08:18 PM...  TCP  80  IIS  ppp-68-251-78-...  GET /scripts/..%25%35%63../winnt/system32/cmd.exe?/c+
18  11/19/2004 12:08:18 PM...  TCP  80  IIS  ppp-68-251-78-...  GET /scripts/..%%35c../winnt/system32/cmd.exe?/c+dir HT
17  11/19/2004 12:08:18 PM...  TCP  80  IIS  ppp-68-251-78-...  GET /scripts/..%%35%63../winnt/system32/cmd.exe?/c+dir
16  11/19/2004 12:08:17 PM...  TCP  80  IIS  ppp-68-251-78-...  GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir H
15  11/19/2004 12:08:17 PM...  TCP  80  IIS  ppp-68-251-78-...  GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir H
14  11/19/2004 12:08:17 PM...  TCP  80  IIS  ppp-68-251-78-...  GET /scripts/..%c0%2f../winnt/system32/cmd.exe?/c+dir H
13  11/19/2004 12:08:17 PM...  TCP  80  IIS  ppp-68-251-78-...  GET /scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir H
12  11/19/2004 12:08:17 PM...  TCP  80  IIS  ppp-68-251-78-...  GET /msadc/..%255c../..%255c../..%255c/..%c1%1c../..%
11  11/19/2004 12:08:17 PM...  TCP  80  IIS  ppp-68-251-78-...  GET /_mem_bin/..%255c../..%255c../..%255c../winnt/syste
10  11/19/2004 12:08:17 PM...  TCP  80  IIS  ppp-68-251-78-...  GET /_vti_bin/..%255c../..%255c../..%255c../winnt/system
9   11/19/2004 12:08:16 PM...  TCP  80  IIS  ppp-68-251-78-...  GET /scripts/..%255c../winnt/system32/cmd.exe?/c+dir HTT
8   11/19/2004 12:08:16 PM...  TCP  80  IIS  ppp-68-251-78-...  GET /d/winnt/system32/cmd.exe?/c+dir HTTP/1.0[0D 0A]Hos
7   11/19/2004 12:08:16 PM...  TCP  80  IIS  ppp-68-251-78-...  GET /c/winnt/system32/cmd.exe?/c+dir HTTP/1.0[0D 0A]Hos
6   11/19/2004 12:08:16 PM...  TCP  80  IIS  ppp-68-251-78-...  GET /MSADC/root.exe?/c+dir HTTP/1.0[0D 0A]Host: www[0D
5   11/19/2004 12:08:16 PM...  TCP  80  IIS  ppp-68-251-78-...  GET /scripts/root.exe?/c+dir HTTP/1.0[0D 0A]Host: www[0D
```

Figure 11-6. *KFSensor logs showing the first IIS attack*

I opened one of the logged events generated by an attack and looked at the content detail, as shown in Figure 11-7. I then copied the content from the top line of the Received box (showing what was sent by the attacker) and used Google to search on the string. Google revealed that the attack traffic was from a Nimba-style worm.



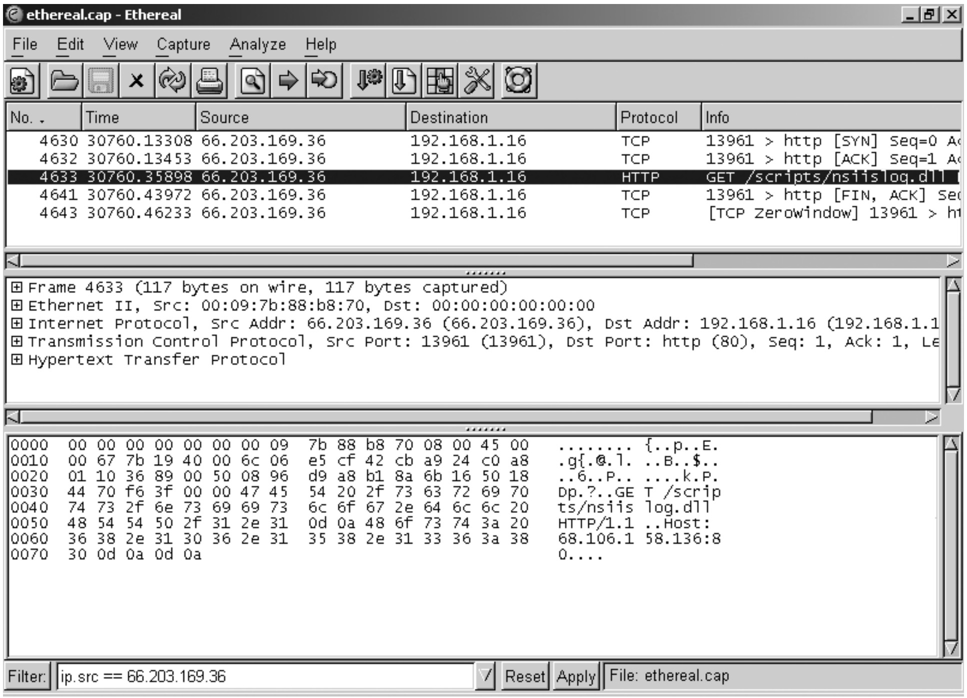
```
Event Details

Event
Start Time:  11/19/2004 12:08:18 PM.32!     Event ID:  19
End Time:    11/19/2004 12:08:18 PM.39:     Type:      Connection
Description:
Sensor ID:   KFSensor                       Severity:  Medium

Visitor
IP:          68.251.78.46       Port:  1423
Domain:      ppp-68-251-78-46.dsl.chcgil.ameritech.net

Sensor
Name:        IIS
IP:          192.168.1.16       Port:        80
Bound:                          Protocol:    TCP
Action:      SimStdServer       Sim Server:  IIS

Details
Closed By:   Server             Limit Exceeded:
Received:
GET /scripts/..%25%35%63../winnt/system32/cmd.exe?/c+di
Host: www
Connnection: close
Response:
HTTP/1.1 404 Not Found
Content-Length: 103
Content-Type: text/html
Server: Microsoft-IIS/6.0
```

Figure 11-7. *KFSensor log detail for one of the attacks*

Windows Media Services Buffer Overflow Attack

Another single IIS probe was looking for the Nsiislog.dll file. This is a Windows Media Services buffer overflow vulnerability in IIS 5 and Windows 2000. It was reported and patched in 2003. The attacker tried only once and did not come back.

Using Ethereal's View Filter feature, I looked at just the packets associated with the single source IP address, as shown in Figure 11-8. There are a total of five packets, but only one with any real payload data. The other packets are TCP handshake and Window size negotiations. Interestingly, there are other related packets, such as the ACK,SYN packet sent as part of the three-way TCP handshake. It isn't listed because the filter is one way, so it does not show traffic headed back to the attacker. In order to capture all traffic, I would need to modify the filter to look for all traffic headed to or from the source IP address.



```
No. .  Time        Source          Destination    Protocol  Info
  4630 30760.13308 66.203.169.36   192.168.1.16   TCP       13961 > http [SYN] Seq=0 A
  4632 30760.13453 66.203.169.36   192.168.1.16   TCP       13961 > http [ACK] Seq=1 A
  4633 30760.35898 66.203.169.36   192.168.1.16   HTTP      GET /scripts/nsiislog.dll
  4641 30760.43972 66.203.169.36   192.168.1.16   TCP       13961 > http [FIN, ACK] Se
  4643 30760.46233 66.203.169.36   192.168.1.16   TCP       [TCP ZeroWindow] 13961 > h
```

```
⊞ Frame 4633 (117 bytes on wire, 117 bytes captured)
⊞ Ethernet II, Src: 00:09:7b:88:b8:70, Dst: 00:00:00:00:00:00
⊞ Internet Protocol, Src Addr: 66.203.169.36 (66.203.169.36), Dst Addr: 192.168.1.16 (192.168.1.1
⊞ Transmission Control Protocol, Src Port: 13961 (13961), Dst Port: http (80), Seq: 1, Ack: 1, Le
⊞ Hypertext Transfer Protocol
```

```
0000  00 00 00 00 00 00 00 09  7b 88 b8 70 08 00 45 00   ........ {..p..E.
0010  00 67 7b 19 40 00 6c 06  e5 cf 42 cb a9 24 c0 a8   .g{.@.l. ..B..$..
0020  01 10 36 89 00 50 08 96  d9 a8 b1 8a 6b 16 50 18   ..6..P.. ....k.P.
0030  44 70 f6 3f 00 00 47 45  54 20 2f 73 63 72 69 70   Dp.?..GE T /scrip
0040  74 73 2f 6e 73 69 69 73  6c 6f 67 2e 64 6c 6c 20   ts/nsiis log.dll
0050  48 54 54 50 2f 31 2e 31  0d 0a 48 6f 73 74 3a 20   HTTP/1.1 ..Host:
0060  36 38 2e 31 30 36 2e 31  35 38 2e 31 33 36 3a 38   68.106.1 58.136:8
0070  30 0d 0a 0d 0a                                     0....
```

Filter: ip.src == 66.203.169.36 Reset Apply File: ethereal.cap

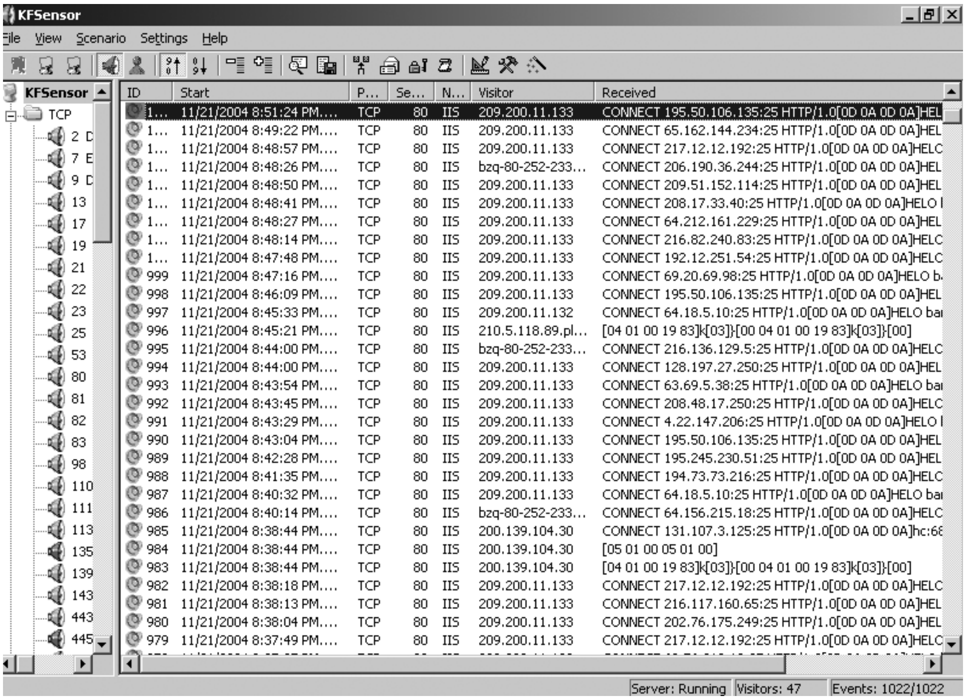
Figure 11-8. *Ethereal capture showing Windows Media Services buffer overflow attack*

SQL Server SA Password-Guessing Attack

There were dozens of SQL Slammer worm attempts and FTP password guessing attempts. KFSensor captured all of the FTP password attempts, the vast majority of which used administrator as the logon name and `password` as the password. There was also a SQL Server password-guessing attack. The attacker sent 48 different password guesses in 25 seconds. Each used the logon name SA and tried various passwords, including `sa`, blank, `password`, `sa12`, `sa123`, `123`, `12345`, `1`, `1234567`, and `super`. Each password was attempted three times. This is another automated attack, of course. A manual hacker would have tried once, and then moved on.

Open-Relay Attack

On day three at 9:26 A.M., the first probe from a spammer (actually a spam bot) arrived. Up until this point, the honeypot had generated about a 100 separate alerts. Traffic was bursty and random, with long periods of nothing happening. That was about to change, as shown in Figure 11-9. Two packets were sent from source IP address 220.242.53.140 to port 80 using the `CONNECT` command. The HTTP `CONNECT` command was originally introduced to allow access to HTTPS servers, but it can also be used to relay any type of traffic to any destination and port. If a server is found that supports HTTP `CONNECT`s to external IP addresses, the spammers have found an open relay from which to send spam.



ID	Start	P...	Se...	N...	Visitor	Received
1...	11/21/2004 8:51:24 PM....	TCP	80	IIS	209.200.11.133	CONNECT 195.50.106.135:25 HTTP/1.0[0D 0A 0D 0A]HEL
1...	11/21/2004 8:49:22 PM....	TCP	80	IIS	209.200.11.133	CONNECT 65.162.144.234:25 HTTP/1.0[0D 0A 0D 0A]HEL
1...	11/21/2004 8:48:57 PM....	TCP	80	IIS	209.200.11.133	CONNECT 217.12.12.192:25 HTTP/1.0[0D 0A 0D 0A]HELO
1...	11/21/2004 8:48:26 PM....	TCP	80	IIS	bzq-80-252-233...	CONNECT 206.190.36.244:25 HTTP/1.0[0D 0A 0D 0A]HEL
1...	11/21/2004 8:48:50 PM....	TCP	80	IIS	209.200.11.133	CONNECT 209.51.152.114:25 HTTP/1.0[0D 0A 0D 0A]HEL
1...	11/21/2004 8:48:41 PM....	TCP	80	IIS	209.200.11.133	CONNECT 208.17.33.40:25 HTTP/1.0[0D 0A 0D 0A]HELO
1...	11/21/2004 8:48:27 PM....	TCP	80	IIS	209.200.11.133	CONNECT 64.212.161.229:25 HTTP/1.0[0D 0A 0D 0A]HEL
1...	11/21/2004 8:48:14 PM....	TCP	80	IIS	209.200.11.133	CONNECT 216.82.240.83:25 HTTP/1.0[0D 0A 0D 0A]HELO
1...	11/21/2004 8:47:48 PM....	TCP	80	IIS	209.200.11.133	CONNECT 192.12.251.54:25 HTTP/1.0[0D 0A 0D 0A]HELO
999	11/21/2004 8:47:16 PM....	TCP	80	IIS	209.200.11.133	CONNECT 69.20.69.98:25 HTTP/1.0[0D 0A 0D 0A]HELO b
998	11/21/2004 8:46:09 PM....	TCP	80	IIS	209.200.11.133	CONNECT 195.50.106.135:25 HTTP/1.0[0D 0A 0D 0A]HEL
997	11/21/2004 8:45:33 PM....	TCP	80	IIS	209.200.11.132	CONNECT 64.18.5.10:25 HTTP/1.0[0D 0A 0D 0A]HELO ba
996	11/21/2004 8:45:21 PM....	TCP	80	IIS	210.5.118.89.pl...	[04 01 00 19 83]k[03]}[00 04 01 00 19 83]k[03]}[00]
995	11/21/2004 8:44:00 PM....	TCP	80	IIS	bzq-80-252-233...	CONNECT 216.136.129.5:25 HTTP/1.0[0D 0A 0D 0A]HELO
994	11/21/2004 8:44:00 PM....	TCP	80	IIS	209.200.11.133	CONNECT 128.197.27.250:25 HTTP/1.0[0D 0A 0D 0A]HEL
993	11/21/2004 8:43:54 PM....	TCP	80	IIS	209.200.11.133	CONNECT 63.69.5.38:25 HTTP/1.0[0D 0A 0D 0A]HELO ba
992	11/21/2004 8:43:45 PM....	TCP	80	IIS	209.200.11.133	CONNECT 208.48.17.250:25 HTTP/1.0[0D 0A 0D 0A]HELO
991	11/21/2004 8:43:29 PM....	TCP	80	IIS	209.200.11.133	CONNECT 4.22.147.206:25 HTTP/1.0[0D 0A 0D 0A]HELO
990	11/21/2004 8:43:04 PM....	TCP	80	IIS	209.200.11.133	CONNECT 195.50.106.135:25 HTTP/1.0[0D 0A 0D 0A]HEL
989	11/21/2004 8:42:28 PM....	TCP	80	IIS	209.200.11.133	CONNECT 195.245.230.51:25 HTTP/1.0[0D 0A 0D 0A]HEL
988	11/21/2004 8:41:35 PM....	TCP	80	IIS	209.200.11.133	CONNECT 194.73.73.216:25 HTTP/1.0[0D 0A 0D 0A]HELO
987	11/21/2004 8:40:32 PM....	TCP	80	IIS	209.200.11.133	CONNECT 64.18.5.10:25 HTTP/1.0[0D 0A 0D 0A]HELO ba
986	11/21/2004 8:40:14 PM....	TCP	80	IIS	bzq-80-252-233...	CONNECT 64.156.215.18:25 HTTP/1.0[0D 0A 0D 0A]HELO
985	11/21/2004 8:38:44 PM....	TCP	80	IIS	200.139.104.30	CONNECT 131.107.3.125:25 HTTP/1.0[0D 0A 0D 0A]hc:6
984	11/21/2004 8:38:44 PM....	TCP	80	IIS	200.139.104.30	[05 01 00 05 01 00]
983	11/21/2004 8:38:44 PM....	TCP	80	IIS	200.139.104.30	[04 01 00 19 83]k[03]}[00 04 01 00 19 83]k[03]}[00]
982	11/21/2004 8:38:18 PM....	TCP	80	IIS	209.200.11.133	CONNECT 217.12.12.192:25 HTTP/1.0[0D 0A 0D 0A]HELO
981	11/21/2004 8:38:13 PM....	TCP	80	IIS	209.200.11.133	CONNECT 216.117.160.65:25 HTTP/1.0[0D 0A 0D 0A]HEL
980	11/21/2004 8:38:04 PM....	TCP	80	IIS	209.200.11.133	CONNECT 202.76.175.249:25 HTTP/1.0[0D 0A 0D 0A]HEL
979	11/21/2004 8:37:49 PM....	TCP	80	IIS	209.200.11.133	CONNECT 217.12.12.192:25 HTTP/1.0[0D 0A 0D 0A]HELO

Server: Running | Visitors: 47 | Events: 1022/1022

Figure 11-9. *KFSensor's logs of the spam open relay*

The first two packets, I later learned, were test packets to see if an open relay existed. KFSensor allowed enough response to happen to fool the spammer. This was a spam bot designed specifically to find open proxies. It sent two test packets, and then reported the found open relay back to its web site and database. The spam bot web site acts as a coordinator for all open proxies on the Internet that spammers and hackers can use. Sixteen minutes later, traffic from the first spammer arrived; one minute later, the second arrived; and so on. Once my open relay was found, all the spammers paying the spam bot service were notified of the new victim. One by one, spammers began to appear. Within minutes, the network bandwidth of the DMZ had been fully utilized (this was when no e-mail was really leaving the DMZ, which would have doubled the traffic).

I was able to see the spam being sent (more of the mortgage variety than the porn or Viagra type) and who it was being sent to (random yahoo.com addresses). There was so much spam occurring that the different spammers were literally fighting it out to make more connections to the honeypot. If wanted to, I could have researched the spam-sending machines, but they would have no doubt led to other innocent compromised machines. Still, if I were tasked with prosecuting spammers, this would be an intriguing method to use.

Lessons Learned

This medium-emulation honeypot suffered a fair amount of attacks in the three days it was up. Most attacks were automated, and all could be defeated with simple security measures.

The customer learned that attacks were happening on the DMZ, and password guesses and old exploits were frequent. Using longer and complex passwords would have defeated many of these attacks, as well as, up-to-date patching. The biggest current threat appeared to be spammers and their spam worm creations. Once they detected the availability of an open-proxy server, almost every other attack was pushed out as the spammers took control.

The WhiteDoe Real Honeypot

I was hired to find out how hackers were successfully breaking in to a computer belonging to a large public school system (I'll call it WhiteDoe Public School System). The school system administrator was very good at her job, but not trained in computer security or antihacking techniques. She had found a hacked server a few months before my arrival. She hired an outside firm to find out how the hackers were breaking in and to remove the malicious code. The outside firm couldn't find the hackers, so they called in an "expert." My client calmly recalled the 15-year-old teenager dressed in dark, gothic clothes arriving the next day. He was "one of the best hackers in the world," she was told. She was made to leave the computer room as he practiced his craft. Of course, months later, the system was still hacked. She knew it was still hacked because of the lack of free disk space.

After the hacker wunderkind had visited, the server quickly ran out of available free disk space. The client added two new 9GB hard drives. Two days later, both drives were out of room. This was despite the fact that the server hosted only a small web site and ten Exchange Server users. Clearly, the hackers loved the new disk space, and they were greedy. A cursory review of the server revealed nothing suspicious, other than the low drive space. The client and I could not find the files that were taking up all the space. An up-to-date antivirus scan was executed, and it found nothing.

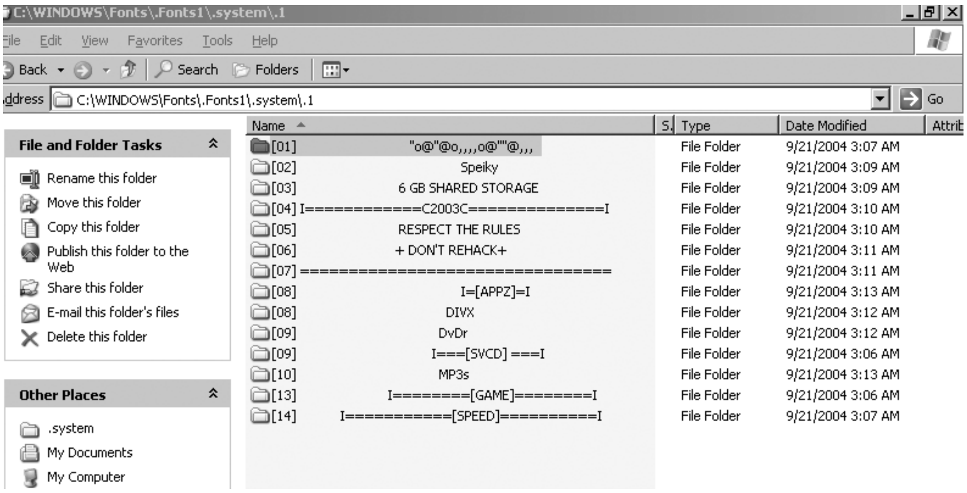
I created a production honeypot that mimicked the hacked Windows server. The real honeypot was running Windows Server 2003 Standard Edition, without hotfixes or service packs, and connected to the same domain as the other server. Identical local user accounts were created, along with identical passwords. I installed Exchange Server and IIS on the server. I used a transparent bridge and a managed switch (as discussed in Chapter 2) to isolate the honeypot and the hacked server into the same virtual LAN. The monitoring PC ran Snort and Ethereal. I took a baseline on the honeypot and on the honeypot network. Then the client and I went to lunch and waited. I wasn't even through with my hamburger when my pager went off.

Exploit Code Happens

Ethereal sent my pager a SMS message indicating activity on the honeypot. We raced back to the honeypot. We could not see anything moving on the console screen, but disk activity was high and Ethereal was busy capturing packets. Snort did not alert us at any time. In a few hours, we had hundreds of thousands of captured packets. All traffic headed to the honeypot was coming directly from the previously exploited server. After a few hours, we took the honeypot and the exploited server offline.

New snapshots were taken and compared against the original. In summary, dozens of new folders had been created, 8GB of data transferred, and two new services installed. The hackers had come in through using the administrator account and password. This proved that somehow the hackers had cracked the administrator password. It would need to be changed. After the hackers got their initial access, they immediately created two new local accounts on the server, for later access (in case we changed the administrator password). They then installed a remote-access trojan called Ghost RAdmin (`http://www3.ca.com/securityadvisor/pest/pest.aspx?id=453084944`). Ghost RAdmin is based on a legitimate remote-control program called Remote Administrator (`http://securityresponse.symantec.com/avcenter/venc/data/remacc.radmin.html`), but it has been co-opted by malicious hackers.

The hackers then set up an IRC server running on port 6667 and an FTP server. Ghost RAdmin was used to copy gigabytes of pirated DVDs, DVD-Rs, games, appz (rogue applications), and MP3s. The directory structure was hidden under C:\Windows\Fonts. The hackers created subfolders under that folder called \.Fonts1\.system\.1, as shown in Figure 11-10. Beginning the folders with a period made them appear invisible in Windows Explorer, although when I typed in their absolute names, the folders appeared.

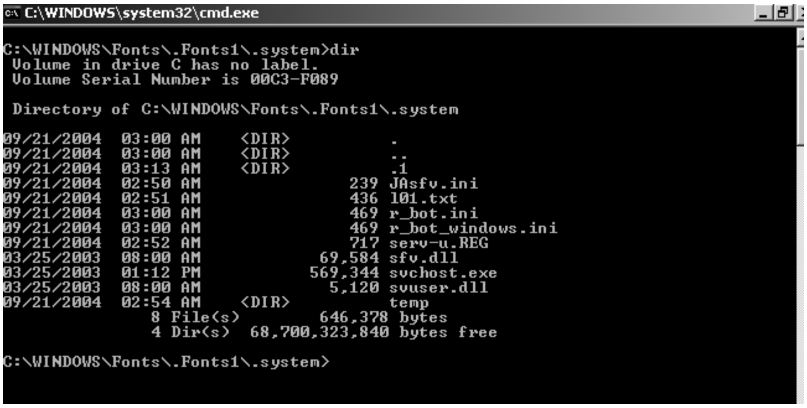


Name	S	Type	Date Modified	Attrib
[01] "o@"@o,,,,o@""@,,,		File Folder	9/21/2004 3:07 AM	
[02] Speiky		File Folder	9/21/2004 3:09 AM	
[03] 6 GB SHARED STORAGE		File Folder	9/21/2004 3:09 AM	
[04] I============C2003C==============I		File Folder	9/21/2004 3:10 AM	
[05] RESPECT THE RULES		File Folder	9/21/2004 3:10 AM	
[06] + DON'T REHACK+		File Folder	9/21/2004 3:11 AM	
[07] ================================		File Folder	9/21/2004 3:11 AM	
[08] I=[APPZ]=I		File Folder	9/21/2004 3:13 AM	
[08] DIVX		File Folder	9/21/2004 3:12 AM	
[09] DvDr		File Folder	9/21/2004 3:12 AM	
[09] I===[SVCD] ===I		File Folder	9/21/2004 3:06 AM	
[10] MP3s		File Folder	9/21/2004 3:13 AM	
[13] I========[GAME]========I		File Folder	9/21/2004 3:06 AM	
[14] I===========[SPEED]==========I		File Folder	9/21/2004 3:07 AM	

Figure 11-10. *Hacker's malicious folder structure*

Actually, only the .system folder was invisible, and consequently, anything below it. For whatever reason, .Fonts1 and the other subfolders were visible. When I first saw the .Fonts1 folder under the regular Fonts folder, I wasn't sure if the folder was malicious or if it was just some sort of Windows temporary folder (or something like that). Fortunately, my baseline comparisons told me it was the former.

When I explored the new directories, I found German-language versions of Microsoft Encarta Professional 2004 and Blue Crush DVD, among other files. The hackers were clearly intending to set up the hacked server as an IRC-accessible FTP site. Ethereal confirmed my suspicion when I saw text headed to an IRC server network advertising the new FTP server and DVDs. I found trojan and configuration files in the bogus .system directory, as shown in Figure 11-11.

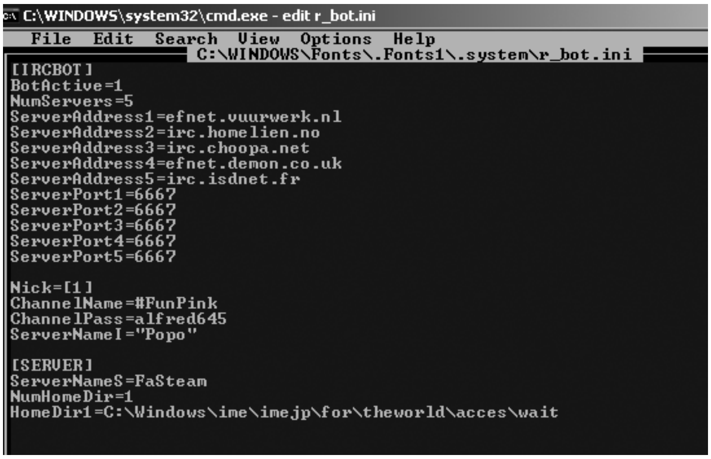


```
C:\WINDOWS\system32\cmd.exe

C:\WINDOWS\Fonts\.Fonts1\.system>dir
 Volume in drive C has no label.
 Volume Serial Number is 00C3-F089

 Directory of C:\WINDOWS\Fonts\.Fonts1\.system

09/21/2004  03:00 AM    <DIR>          .
09/21/2004  03:00 AM    <DIR>          ..
09/21/2004  03:13 AM    <DIR>          .1
09/21/2004  02:50 AM               239 JAsfv.ini
09/21/2004  02:51 AM               436 101.txt
09/21/2004  03:00 AM               469 r_bot.ini
09/21/2004  03:00 AM               469 r_bot_windows.ini
09/21/2004  02:52 AM               717 serv-u.REG
03/25/2003  08:00 AM            69,584 sfv.dll
03/25/2003  01:12 PM           569,344 svchost.exe
03/25/2003  08:00 AM             5,120 svuser.dll
09/21/2004  02:54 AM    <DIR>          temp
               8 File(s)        646,378 bytes
               4 Dir(s)  68,700,323,840 bytes free

C:\WINDOWS\Fonts\.Fonts1\.system>
```

Figure 11-11. *Bogus .system directory*

Here, I found a trojan version of Svchost.exe, named after a legitimate Windows file normally found in C:\Windows\System32. The legitimate version is used to launch the different Windows RPC services. The trojan version, at over 560KB, is significantly bigger than the normal Svchost.exe, which is 7KB or 8KB. The trojan could not have overwritten the legitimate version of Svchost.exe in the System32 directory because of Windows File Protection (WFP), so it wrote it to a bogus new directory instead. It then appeared in Task Manager as a process called Svchost.exe, along with all the other legitimate processes of the same name.

Ethereal picked up a significant amount of IRC and FTP traffic. The IRC traffic was encrypted or encoded. In the bogus directory, I found an IRC-related file affiliated with the trojan called R_bot.ini, as shown in Figure 11-12. It contained information about the involved IRC network. It listed the involved servers, port addresses, and more important, the IRC channel name being used.



```
C:\WINDOWS\system32\cmd.exe - edit r_bot.ini
  File  Edit  Search  View  Options  Help
            C:\WINDOWS\Fonts\.Fonts1\.system\r_bot.ini
[IRCBOT]
BotActive=1
NumServers=5
ServerAddress1=efnet.vuurwerk.nl
ServerAddress2=irc.homelien.no
ServerAddress3=irc.choopa.net
ServerAddress4=efnet.demon.co.uk
ServerAddress5=irc.isdnet.fr
ServerPort1=6667
ServerPort2=6667
ServerPort3=6667
ServerPort4=6667
ServerPort5=6667

Nick=[1]
ChannelName=#FunPink
ChannelPass=alfred645
ServerNameI="Popo"

[SERVER]
ServerNameS=FaSteam
NumHomeDir=1
HomeDir1=C:\Windows\ime\imejp\for\theworld\acces\wait
```

Figure 11-12. *R_bot.ini IRC configuration file*

As you can see in Figure 11-12, the IRC channel was called `FunPink` and the password was `alfred645`. I immediately used that information to join the chat and to sniff more traffic. I was glad to find (and to document) that all traffic was headed to and from the bogus directories. The hackers were using the school's computers as an FTP server, but they didn't appear to be taking confidential school files or e-mail messages and passing them to remote locations. Sniffing more, I found out the location of the other files on the original server. I had missed them before in my initial analysis, but now I knew why. The original server had directories created using unprintable German Unicode characters. Once we knew what we were looking for, removing the hackers' files and the trojans was a piece of cake.

Lessons Learned

There may have been other ways to solve this particular problem, but the honeypot proved invaluable to our investigation. The encrypted IRC channel would not have been readily apparent if I did not learn about the channel name and password.

We learned that hackers are readily using other people's computers to set up FTP and IRC servers, and they don't care if they run out of free space. At first, I thought the latter fact to be a hacker mistake. If the hackers watched the free disk space, there is a good chance the client would not have noticed and would not have called me to investigate. But when you are robbing and using other people's resources, if you get kicked off one computer, you find another.

We also learned that Windows Explorer doesn't handle German Unicode characters well or even directories beginning with periods (such as .system). The client also learned that the uber-hacker teenager probably wasn't the best choice for the job. A structured analytical approach will usually prevail in the end.

Forensic Tool Web Sites

Many web sites specialize in listing free and commercial forensic tools. Here is a sampling:

- TUCOFS–The Ultimate Collection of Forensic Software (`http://www.tucofs.com/tucofs/tucofs.asp?mode=mainmenu`)
- The Electronic Evidence Information Center (`http://www.e-evidence.info/other.html`)
- NISER Computer Forensics Laboratory (`http://www.niser.org.my/forensics/tools.html`)
- Open-source Windows forensics tools (`http://www.opensourceforensics.org/tools/windows.html`)
- Foundstone (`http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/freetools.htm`)
- Computer Forensics, Cybercrime and Steganography Resources (`http://www.forensics.nl/toolkits`)

Summary

This chapter covered the structured approach to honeypot analysis. It reviewed all the different ways to examine honeypot data, including analyzing network traffic, changes to the file system, and changes to the OS. There are hundreds of useful forensic utilities to help make the job easier.

Chapter 12 will finish the book by discussing malware code disassembly.