Chapter 9

DIY Malware Analysis

Solutions in this chapter:

- **Anti-Malware Tools of the Trade 101**
- **The Basics – Identifying a Malicious File**
- **Process and Network Service Detection Tools**
- **Web-based Inspection and Virus Analysis Tools**
- **Using Packet Analyzers to Gather Information**
- **Examining Your Malware Sample with Executable Inspection Tools**
- **Advanced Analysis and Forensics**
- **Advanced Malware Analysis**
- **Static (Code) Analysis**
- **Dynamic (Behavior) Analysis**

☑ **Summary**

☑ **Solutions Fast Track**

☑ **Frequently Asked Questions**

Introduction

It's 3:00 on a Friday afternoon and your phone rings; it's your Security Operations Center (SOC). They tell you that they see an extraordinary amount of traffic going out to the Internet on port 443 from one particular Internet Protocol (IP) address on the network, which belongs to a machine in Santa Clara, California. Your job is to check out this machine to see what is causing this high traffic volume. You don't have physical access to this PC as you're in Boston, Massachusetts. How do you analyze this machine remotely? What tools do you use to accomplish this task? After reading Michael Blanchard's introductory section on "Anti-malware Tools of the Trade 101," you will know the basics of how to scan a machine for the presence of malicious software (malware), what to do once it's identified, and how to perform basic analysis on the malicious program itself, using a variety of tools.

Anti-malware software is at a pitch of sophistication that would never have been envisaged a few years ago. Unfortunately, it's never faced the volume and complexity of threats that it does now, and it isn't practical to rely purely on "signature"-based detection. Even where more proactive, generic measures are in place, system administrators in many organizations need to have an understanding of forensics and analysis in order to do their jobs as well as possible.

The sheer volume of new variants of bots and other malicious software associated with today's explosion of spam puts a severe strain on the ability of antivirus (AV) vendors to deploy timely reactive and proactive countermeasures. This in turn puts a strain on in-house security administrators, who often have to analyze code themselves, in order to maintain services during the "window of opportunity" exploited by malware variants that antimalware programs don't yet detect. In the second section of the chapter, Bojan Zdrnja shares some of his considerable forensic experience, looking at tools and techniques for advanced analysis, both static and dynamic.

Anti-Malware Tools of the Trade 101

It is not our intention in this chapter to explore every feature of the tools listed. Primarily, we'll focus on the use of the tools mentioned to help with the identification of malware on a host machine. We will also discuss how to determine exactly what the malware does, in order to take the necessary steps to isolate infected machines. This chapter will outline the methods that we've found to be the easiest way to start looking for malware on machines. In many cases the tools we'll be discussing can be used in many different ways. The examples that we use are by no means the only possible usage, just the ways that we've found to work very well both for the beginner and for the advanced analyst.

First steps should always be to have the SOC shut off Internet access for the suspect machine using firewall rules or router Access Control Lists (ACL's) and allow it to only communicate with the IP address from which you will be performing the analysis.

This lessens any risk to the company's network and other systems, while maintaining your access to the suspect machine for analysis. If this type of a lock down were not possible in a particular environment, physical access to the suspect machine would be required. Leaving the machine connected to a live network until the analysis is complete may also be an option, but a very risky one.

The Basics: Identifying a Malicious File

There are a number of tools that can be run on the remote system to aid you in determining what process or file is causing trouble on the machine. PsExec, part of the PSTOOLS package, is available from www.sysinternals.com (now owned by Microsoft), and is an almost essential tool for enabling remote analysis. Most remote tools available today provide full and transparent Graphical User Interface (GUI) access. PsExec allows you a slightly stealthy connection to a remote machine, however. Provided that you can offer the proper credentials to access the remote machine, using PsExec to open a remote command shell, the user of the remote machine may not even realize that you're connected to it, unless he or she is actively checking connections.

PsExec is the single most useful tool that can be added to any analyst's jump kit (for Europeans and others who may not be familiar with this term, it usually denotes a comprehensive first-aid kit of the type carried by paramedics and others). It will allow you to execute programs on a remote computer to which you have access. Most importantly, it allows you to launch a remote console and execute the needed tools on the remote machine, without needing to use a full graphical controller such as XP Remote Desktop (www.microsoft.com/windowsxp/using/mobility/getstarted/remoteintro.mspx), Virtual Network Computing (VNC), and so on.

Tools & Traps

VNC

This is a platform-independent desktop sharing system, originally developed at AT&T, for controlling a remote machine over a network. The original source code is open source, as are many recent implementations. It works on a client/server model, where the server is the program on the controlled machine, and the controlling machine connects to it via the client or viewer. The fact that the viewer and server don't have to use the same operating system makes it potentially useful in mixed environments, and its display capabilities make it more versatile than ye olde worlde Telnet (and not quite as insecure).

Continued

However, in today's insecure world, it needs to be boosted security-wise. Tunneling over Secure Shell (SSH) or Virtual Private Network (VPN) is an option, and there is an open-source plug-in for UltraVNC. RealVNC offer a commercial implementation that includes "Integrated Session Security," though we haven't tested this.

- http://ultravnc.sourceforge.net/
- www.realvnc.com/
- www.realvnc.com/products/enterprise/

PsExec.exe has many useful features, but for this example we will only use it to open up a remote command shell on the system in question. This is very easy to do by using a quick command line string on your system. You will only need to know the remote machine's IP address or Fully Qualified Domain Name (FQDN). Figure 9.1 shows an example of opening up a remote command shell.

The example shown in Figure 9.1 assumes that your current logged-in session has command line-level access to the remote machine. This can be either via native local permissions on the remote machine, Active Directory, other NT domain permissions, or via a mapped resource using other credentials that can be passed through using PsExec.

Figure 9.1 Opening a Remote Command Shell Using *psexec.exe*

If your account doesn't already enable you to get access to the remote machine, but you are aware of an account that does, you can use the example shown in Figure 9.2. Figure 9.2 shows the use of one of the many command line switches available with PsExec. The username "*/u*" switch and the password "*/p*" switch, (which is optional - if it isn't used

you'll be prompted for the password), show how to pass user-level credentials along with the shell request all within the same command line.

Figure 9.2 Using the "*/u*" switch to Connect Using Different Credentials

Here is an example command line showing how to open a remote shell using PsExec:

```
PSEXEC \\192.168.1.100 cmd.exe
```

Here is a summary of the syntax and command line switches for PsExec:

```
psexec [\\computer[,computer[,..] | @file ][-u user [-p psswd]]
[-n s][-l] [-s|-e][-i][-x][-c [-f|-v]][-d][-w directory]
[-<priority>][-a n,n,...] cmd [arguments]
```

If you omit the computer name, PsExec runs the application on the local system, and if you enter a computer name of "*" PsExec runs the applications on all computers in the current domain.

- ***@file*** Run the command on each computer listed in the text file specified.
- ***-a*** Run the application on the processor's specified, separated commas where 1 is the lowest numbered Central Processing Unit (CPU). Thus, to run the application on CPU 2 and CPU 4, enter: "*-a 2,4*"
- ***-c*** Copy a specified program to the remote system to execute. If this option is omitted, the application has to be in the execution path on the remote system.
- ***-d*** Don't wait for application to terminate; for use with non-interactive applications.
- ***-e*** Loads the profile for the specified account.

- ***-f*** Copy program to the remote system even if the file already exists there.
- ***-I*** Run the program to interact with the desktop on the remote system.
- ***-l*** Run process as an unprivileged user.
- ***-n*** Specify timeout in seconds for connection to remote computers.
- ***-p*** Specify an optional password for this user name. If this is omitted, you are prompted to enter a hidden password.
- ***-s*** Run remote process as the System account.
- ***-u*** Specify an optional user name for a remote login.
- ***-v*** Copy the specified file to the remote system only if it has a higher version number or is dated more recently.
- ***-w*** Set working directory of the specified process on the remote computer.
- ***-x*** Display user interface on the Winlogon desktop on the local system only.
- ***-priority*** Specifies priority level as -low, -belownormal, -abovenormal, -high or -realtime
- ***Program*** Specify name of program to execute here.
- ***Arguments*** Arguments to pass (file paths must be absolute paths on the target system).

NOTE

For more information on PsExec see the help file and www.microsoft.com/technet/sysinternals/Security/PsExec.mspx. You may also find Mark Russinovich's article on advanced usage at http://www.windowsitpro.com/Windows/Article/ArticleID/42919/42919.html interesting and useful.

Once you've successfully opened up a remote command shell, you'll have full access to that machine and will be able to run other tools that will enable you to determine exactly what else is running on that machine. More importantly, you'll be able to identify malicious files. Not only does this enable you to take remedial actions, but also indicates which files to copy for further sample analysis, forwarding to vendors, and so on.

Tools & Traps

The PSTools Utility Suite

We strongly urge you to download the entire PSTOOLS suite of tools rather than just PsExec. Even though we've only discussed the usage of one of the tools available in the suite, there are many other tools contained in the download package. You are sure to find that many of them come in handy from time to time. The tools are compact and free to use.

The suite was put together by Mark Russinovich, and the tools it currently includes are:

- **PsExec** A tool for executing remote processes
- **PsFile** A tool for showing files opened remotely
- **PsGetSid** Displays a user's or computer's Security Identifier (SID)
- **PsInfo** A tool for displaying system information, including how long the system has been up since reboot
- **PsKill** A tool that kills processes, identified either by name or by process ID
- **PsList** A tool to display detailed information about processes
- **PsLoggedOn** A tool to show who is logged on locally and via resource shares
- **PsLogList** A tool for dumping event log records
- **PsPasswd** A tool for account password management
- **PsService** A tool for remote services management
- **PsShutdown** A tool for shutting down and rebooting a remote system
- **PsSuspend** A tool for suspending processes

While the PsTools don't, of course, contain malware, their code and functionality has been misused by malicious programs, which may result in their being flagged by AV software as malware. You therefore need to be sure that you can distinguish a false positive in this scenario from the presence of real malware.

Now you have a remote shell open on a machine where you suspect that malware is running. Where do you go from here? The first thing that we usually do is create a directory on the root of the *C*:\ drive. This directory will serve as your repository for all the data that you gather using our emergency toolset, or "jumpkit tools." You need to keep in mind that any tool that you wish to run on the remote system must exist on the remote machine before it can be executed. Even though some of the tools can be run over the network, we prefer to keep the network

Continued

connections to a minimum while analyzing a machine for malware. However, in scenarios where forensics may be required with a view to future judicial review, this may not be an option. The section on advanced analysis and forensics later in this chapter will clarify this issue.

Once the repository directory is created, you can then copy all of your tools to this directory, and run them out of this directory when the time comes. The results can be saved in this directory as well, and when you've collected all the data you can, the directory can be moved to your main machine as a whole, without having to find the results files from all the tools.

One valuable tool that is native to every Windows installation is Netstat, which is an application that will display NETwork STATistics for the current machine. In other words, it will display all network connections to and from the machine. Let's look at how to use Netstat to help us determine what network ports are open on the system.

Here's a summary of the command line syntax for Netstat:

```
NETSTAT [-a] [-b] [-e] [-n] [-o] [-p proto] [-r] [-s] [-v] [interval]
```

And here's a summary of what those switches do.

- ***-a*** Displays all active Transmission Control Protocol (TCP) connections, the TCP and User Datagram Protocol (UDP) ports on which the computer is listening.
- ***-b*** Displays the name of the program's name responsible for each connection or listening port (SP2 or higher).
- ***-e*** Displays Ethernet statistics (e.g., number of bytes and packets sent/received). This parameter can be combined with -s.
- ***-n*** Displays active TCP connections expressed numerically.
- ***-o*** Displays active TCP connections with the process ID (PID) for each connection. The Processes tab in Windows Task Manager shows the application associated with the PID. You can combine this parameter with *-a*, *-n*, and *-p*. (Windows XP or later)
- ***-p*** [Protocol] Shows connections for the [Protocol], which can TCP, UDP, TCPV6, OR UDPV6. Use with *-s* to display statistics by protocol, which can then be TCP, UDP, Internet Control Message Protocol (ICMP), IP, TCPV6, UDPV6, ICMPV6, OR IPV6.
- ***-r*** Display the [IP routing table]. (Equivalent to the route print command.)
- ***-s*** Displays statistics by protocol. (See "*-p*" parameter.) Statistics are shown for the TCP, UDP, ICMP, and IP protocols, by default. If the IPv6 protocol is installed (Windows XP), statistics for TCP over IPv6 and UDP over IPv6, ICMPv6, and IPv6 are shown.
- ***-v*** In conjunction with *–b*, displays the sequence of components that create a connection or the listening port for all executables.

- Interval Refresh the selected information display every [Interval] seconds. Ctrl+C stops the redisplay.
- */?* Displays help text.

You'll find that if you simply run most of these tools, their output is larger than the command window buffer. For this reason, when running any tool from the command line, append a > *filename.ext* to the command (example: *netstat –a >report.txt*). This will send all of the output to a text file instead of to the screen. In the example shown, the output will be placed in a file named *report.txt*. This also has the huge benefit of saving the tool's output for review at a later time. This can also be a fully qualified path so it isn't limited to the current directory. We would recommend putting the full path to your repository directory if you're not running the command from within your repository directory. In other words, use the full (absolute) pathname from the root directory downward.

Figure 9.3 Typical Output of the Netstat Command

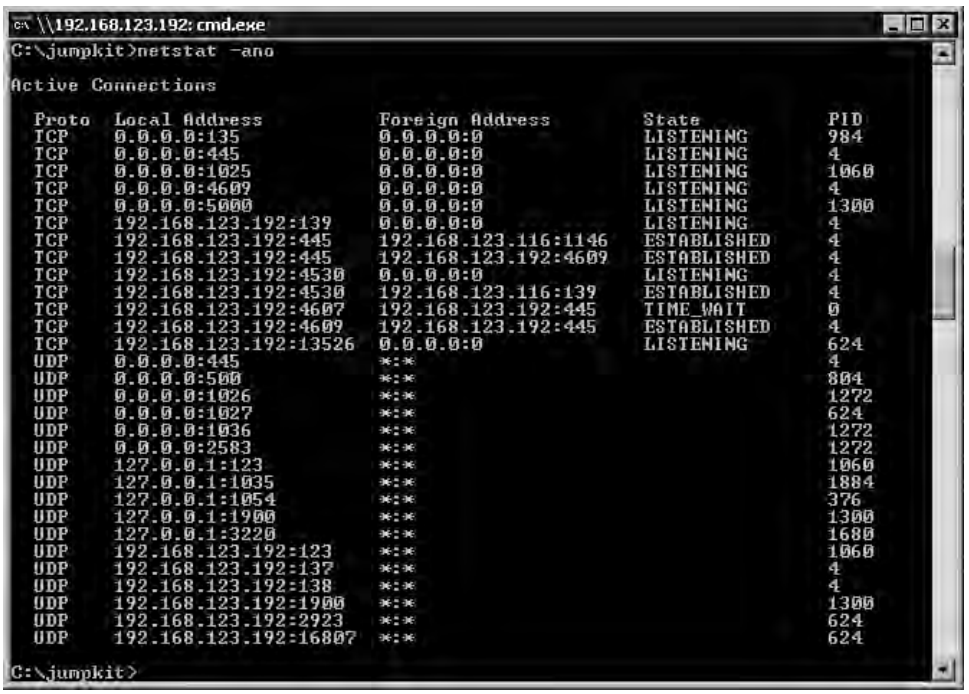


Figure 9.3 shows typical output from the Netstat command. By using the *-a* switch as shown , you can easily identify what ports are open and what machine is making use of them. By column, the screenshot shows the following data.

- **Proto** This identifies the protocol that is in use, either TCP or UDP.
- **Local Address** This identifies the local hostname and port number/common name.
- **Foreign Address** This identifies the remote machine and port that is connected.
- **State** This identifies whether or not the port is listening for connections, has an established connection, or perhaps neither.
- **PID** This identifies the process identifier of the process that is using that connection.

The Netstat command does provide some good, if cursory, information that is easily and natively available. Quickly, without adding any applications to the suspect machine, you have a listing of all the network connections that are in use on that machine. Netstat does fall short of providing all the information you'll need, but there are other tools that will be covered that will fill in the gaps.

NOTE

Only Windows XP and higher versions of Netstat have the "*-o*" option available. Only Windows XP SP2 and higher have the "*-b*" command line option available.

Analyzing the output from Netstat is pretty straightforward. You're basically looking for any line item that shows any network port open that shouldn't be. However, it takes some experience to know what ports should or shouldn't be open. In order to narrow the field, read the output from the right to the left. Go down the "state" column, and stop on any line item where it says "established." Any connection that is in this state is actively communicating on that port. Once you've identified an established connection, go over to the "foreign address" column. This column will tell you to what address or system the connection is established. If that column has "localhost" or "127.0.0.1" in it, you can rule that one out at this time, as the machine is connecting back into itself. When the foreign address column shows an external hostname or IP address, and the state column shows that it's established, that suggests a likely malware candidate. I would suggest that you go ahead and write down the information on that row, to compare it to the results of running the more advanced tools that we'll be going through later on in this chapter.

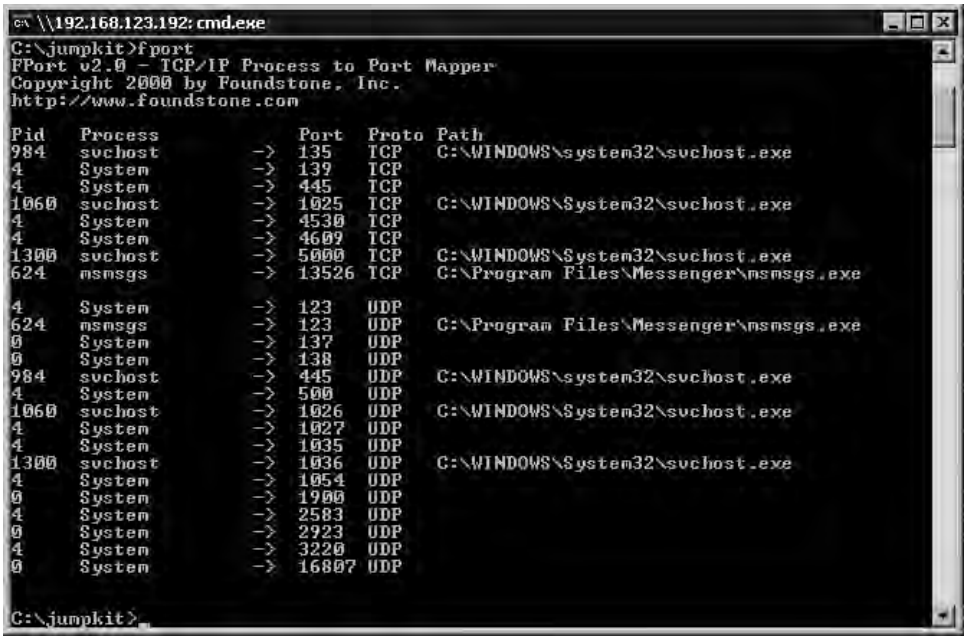
Once you review the output from Netstat, you may have an idea of what port the malicious code is using, but you're still pretty much in the dark as to exactly where that program is running from, or exactly what it is. More importantly, you still don't have a sample of the malware to analyze.

Process and Network Service Detection Tools

In this section, we'll be discussing three core tools for process and network service detection. No analyst's toolbox should be without FPort, tcpvcon, and Handle.

FPort is available free of charge from Foundstone's free resources page at www.foundstone.com/resources/freetools.html. FPort is a tool that provides some of the same information that Netstat does, but with one very significant difference: it gives you the complete path to the process that is making use of the port, along with the PID. Figure 9.4 shows what you can expect to see in a typical report from FPort, run with default options.

Figure 9.4 Typical FPort Output



You can see from the example in Figure 9.4, that the information is very similar to that of the NETSTAT command.

- **PID** This is the PID of the running process.
- **Process** This is the name of the process running.
- **Port** This is the network port that the process is using.
- **Proto** This is the protocol that the process is using to connect on the port.
- **Path** This is the fully qualified path, showing exactly where the executable is located.

Here are the command line switches available for FPORT:

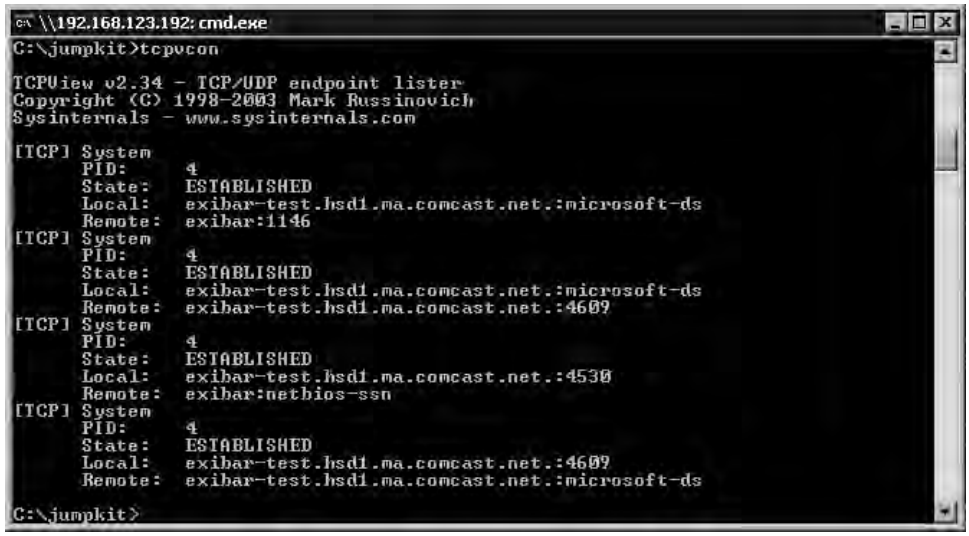
- ***/p*** Sort by port
- ***/a*** Sort by application
- ***/i*** Sort by PID
- ***/ap*** Sort by application path

By reviewing FPort's output, you can gather some very useful information. Within the output of FPort, you have the PID, port, and path items. These are extremely useful for finding the process that is causing problems, and retrieving that all-important malware sample.

For instance, if you happen to see a process called *svchost.exe* running, you might not think twice about it if you're using the Windows task manager's process view, or if you're looking at the output from Netstat. But with the output from FPort, you can see where *svchost.exe* is running from. *Svchost.exe* will normally run from the *c:\windows\system32* (*<systemroot>\system32*) directory. If you see it running from any other directory, that would normally be a piece of malware trying to trick you into thinking it's the real SVCHost process, one of a number of very common SVCHost impersonation tricks.

TCPVCON is another tool that should be in your malware jumpkit, and is available freely from www.sysinternals.com as part of the TCPView application download. tcpvcon provides another view on network connections. Figure 9.5 shows an example of tcpvcon's default output. If there aren't any command line switches applied, tcpvcon will show you only the established connections.

Figure 9.5 Example of Default Output from Running *tcpvcon*



tcpvcon has another very useful feature. You can provide the PID of a process and tell tcpvcon to show you only the network connections that the supplied PID has created. You can do this by using the "*netstat -ao*" command that was discussed earlier in this chapter.

NOTE

TCPVCON is downloaded as part of the GUI application TCPView. TCPView performs the same checks and displays the same information as tcpvcon, but does so courtesy of a nice GUI version that has some added features such as real-time auto refresh, sorting, nice graphical layout, and so forth. If you're looking for malware and have full GUI control of the target machine, or are at the keyboard of the target machine itself, TCPVIEW is certainly a tool worth using.

These are tcpvcon's syntax and command line switches:

- ***tcpvcon*** *[-a] [-c] [-n]* [process name or PID]
- ***-a*** Show all endpoints.
- ***-c*** Print output as CSV.
- ***-n*** Don't resolve addressed process. Only show endpoints owned by the process specified.

Tcpvcon gives you a nice quick look at what network processes are running, and the path that they are running from. This enables you to establish quickly if a process with a familiar name is running from a directory that it shouldn't be running from, and allows you to take further action on that process.

After running the aforementioned tools you'll usually have a pretty good idea of which process is your item of malware. But, in some cases, you might be looking at malware that isn't readily apparent, or perhaps it has some rootkit-like qualities and is attempting to hide itself. Also, some malware will not open up a network connection on its own, but will communicate over the Internet using whatever Windows utilities or installed software applications are available. Perhaps the malware you're looking for is attempting to gather personal information and send it via e-mail or Instant Messaging (IM) to a location on the Internet, or even saving it to your local hard drive for later retrieval?

This is where *Handle.exe* comes into its own (available from www.microsoft.com/technet/sysinternals/Processesandthreadsutilities.mspx). Handle is the command-line version of Sysinternals' Process Explorer. When it is run on a system, by default it displays all the file objects or "handles" that are open or in use by a particular process. Any program that has a file or directory open will show up on the default listing. This includes Data Link Libraries (DLLs), files, directories, and so on. Running Handle with the "*-a*" command-line switch will cause it to display everything that is in use, not just objects that refer to files. Reviewing the output when the "*-a*" switch is used, can be a little bit overwhelming at first. Anything else that may be in use in memory, such as ports, registry keys, synchronization primitives, threads, and individual processes, will also be shown.

NOTE

The output from running Handle can be quite large on most systems, especially if the "*-a*" switch is used. For this reason, I recommend redirecting the output from the screen to a file as with other tools by appending a redirection instruction like *>handle_output.txt* to the end of the *handle.exe* command line in order to more easily review the output.

Here's the syntax for the Handle command line:

```
handle [[-a] [-u] | [-c <handle> [-y]] | [-s]] [-p <processname>|<pid>> [name]
```

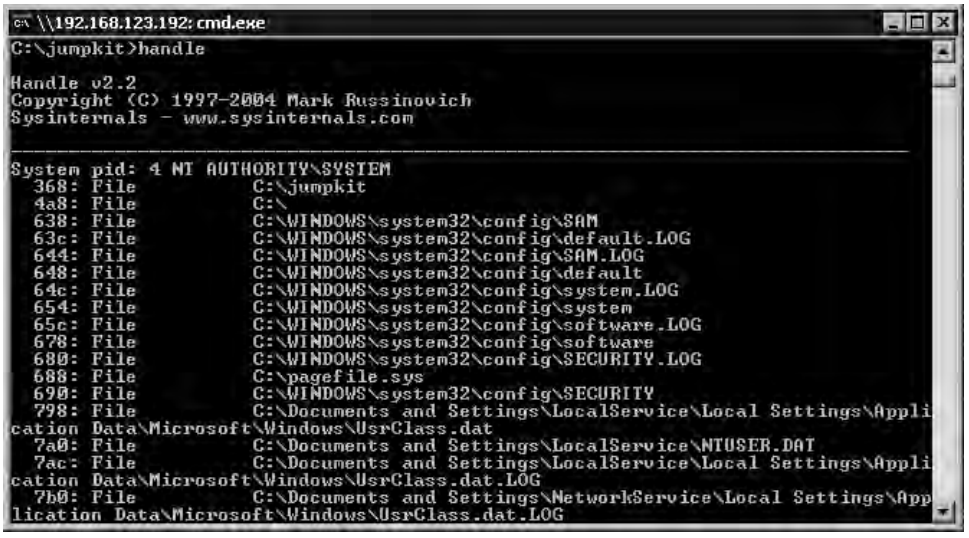
- ***-a*** Show information about all types of handles, including ports, Registry keys, synchronization primitives, threads, and processes as well as files.
- ***-c*** Closes the specified handle, identified by its PID.
- ***-y*** Close handle without prompting for confirmation.

- ***-s*** Show count of each type of open handle.
- ***-u*** Show the user name that owns an open handles.
- ***-p*** Don't examine all the handles: restrict scan to those processes that begin with "*<processname>*." For example, "*handle -p svc*" dumps open files for all processes whose names begin with *svc*.
- ***Name*** Search for references to an object with a particular name. For example, "*handle windows\system*" would find processes that have opened. Search is case-insensitive.

You can get additional information on Handle's syntax and parameters at www.microsoft.com/technet/sysinternals/ProcessesAndThreads/Handle.mspx.

Figure 9.6 shows a small example of what can be expected from Handle's output, which is broken down into sections separated by a row of dashes. The first line in each section shows the System PID, and the name of the process. In this case, the system PID is "*4*" and the name of the process is "NT AUTHORITY\SYSTEM." The data that follows after that first line in the section contains all the handles that are associated with that process. The first column is the specific handle identifier represented as a hexadecimal number, and is unique to that PID. The next column identifies the type of handle. Some of the most common identifiers that will appear in that column represent a process, an individual thread, a registry key, a port, an event, a section of memory, a token, a file, and a directory. The next column gives you the fully qualified path of what is in use by that particular PID, whether it's a register key, file, or directory.

Figure 9.6 Small Portion of the Default Output from Running Handle



Tools & Traps

Avoiding the EULA trap

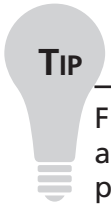
In mid 2006, Microsoft purchased Sysinternals, the makers of PSTools, Handle, and tcpvcon, which are described in detail above. There are some changes occurring to the Sysinternals tools. Most of these changes involve the addition of features to the tools. One change that is being made is the addition of an End User License Agreement (EULA) that must be agreed to before each tool is run for the first time on a new system. This usually wouldn't be a problem, but with the command-line tools that we're running on a remote system, there is a potential issue. When the tool is run the first time, it will attempt to display an EULA on the remote machine, which will never actually be displayed. The tool will appear to hang and do nothing as it's waiting for the EULA to be clicked on and agreed upon. (It's also worth remembering that when you redirect output to a file using the > or >> modifiers, you need to test for unexpected input that will result in a hang like this, usually requiring a Ctrl-C to get out of.)

There are three workarounds for this situation that will allow us to run our tools in the non-interactive manner that suits our needs. Any one of the following will prevent the displaying of the EULA, according to the Sysinternals and Microsoft Web sites:

- Microsoft has added a new command-line switch to all of the utilities in the PSTOOLS suite, that will prevent the tool from waiting for you to confirm agreement to the EULA. The command-line switch that is needed is "*-accepteula*." Microsoft has also said that this switch will be added to all the Sysinternals command-line tools. However, at the time of this writing, only the PSTOOLS recognize this command-line switch.
- If you run the following commands from within the remote command shell, as explained earlier, they will insert the registry keys necessary to allow the tools to run without displaying the EULA and short-circuiting your investigative processes by counterfeiting an application hang.
  - **PsExec** reg add *HKCU\Software\Sysinternals\PsExec /v EulaAccepted /d 1*
  - **TCPVCON** reg add *HKCU\Software\Sysinternals\Tcpvcon /v EulaAccepted /d 1*
  - **Handle** reg add *HKCU\Software\Sysinternals\Handle /v EulaAccepted /d 1*
- Use a version of the above tools that was released prior to November 1, 2006, which was when the EULA code was added to almost all Sysinternals products.

With the use of the aforementioned tools, you should be able to identify the particular malware that is causing the problems on your organization's machine or network. Once you've identified the specific file, the first thing that you need to do is to retrieve a sample of that file and secure it.

Gathering and securing the sample file is very easily done. What you *must* do is archive it and add a password to that archive. Archiving addresses a couple of critical issues. First and foremost, it secures the sample in a manner that ensures that it isn't easily executed by accident. You certainly wouldn't want to infect your own machine accidentally! Secondly, it prevents the sample from being accidentally removed by one of the AV or anti-spyware programs that should be running on your own machine. In many scenarios, you'll only have access to an infected machine for a short time, and you'll often only have access to a single live sample of the malware. It is very important to keep that item of malware secure, until you're sure you don't need it any longer and you've been able to put the proper countermeasures in place on your network. You also need to be able to distribute the proper cleaning procedures to those that will be performing the actual disinfection of the malware from your networked machines.



TIP

Floppy disks, though gradually disappearing from computer store shelves, are very convenient to use to store live samples on. You can enable write protection very easily, thus ensuring that your hard-earned live sample isn't accidentally deleted.

This author always has a supply of bright red floppies on hand that are used solely for storing live samples on. The bright red color makes them stand out, and along with the words "LIVE VIRUS," or similar, written on the label certainly flags them as something out of the ordinary, and stresses the need to handle the contents with care.

The archiving method that some like to use is the old tried and true command line version PKZip, from pkware (www.pkware.com). You may use any archival program that you prefer, but I like to stick with the standards whenever possible. All of the AV vendors and all of the online malware inspection tools accept *.ZIP* files for analysis, which makes our jobs a bit easier if we archive using the *.ZIP* format right from the start of sample collection. You might also prefer *pkzip* due to the fact that it's a stand-alone command line tool. You can copy that tool over to the machine that you are investigating without much hassle, and it's available to use right away. Figure 9.7 shows the typical use of *pkzip* in sample collection.

An example of the typical usage of *pkzip*, versions 2.5, is given here. However, you probably won't get much support for DOS and old Windows versions from PKWare nowadays. On the other hand, the format for the *.zip* archiving standard is freely available, and many other archiving programs fully support it.

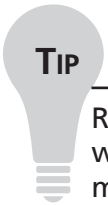
```
pkzip25 -add <name of archive>.zip -pass=<password> <filename to be archived>
```

Figure 9.7 A Typical Usage of *pkzip25*

Command line switches used in the example in Figure 9.7 are as follows:

- ***PKZIP25*** Executable command
- ***-add*** Instructs *pkzip* to add a file to an archive
- ***<name of archive>.zip*** This is the archive file (*.ZIP* file) that you wish to add your malware sample to; if the file doesn't exist, it will be created for you.
- ***-pass=<password>*** This switch allows you to add a password to the *.ZIP* file that you are creating
- ***<filename to be archived>*** This is the file that you wish to add to the *.ZIP* file, your malware sample itself.

pkzip25 has many features and switches aside from the ones described above. We will not go into all of them here, as it's out of scope for this book. If you are interested in using *pkzip* to perform your command-line archiving, we strongly suggest you check out all the features available in current versions by going to www.pkware.com, where there are a number of 30-day evaluation versions.



TIP

Regardless of what tool you wish to use to create your *.ZIP* file, you'll always want to apply a password to the *.ZIP* file when archiving or submitting malicious files. The AV industry has unofficially adopted as a *de facto* standard the password "infected" for compressed files. That password is almost universally used throughout the industry. Most online scanners, online analysis tools, or submissions to an AV vendor will assume that a .ZIP file has that password when it's submitted for analysis. However, you should always check the terms and conditions and other requirements on such Web sites before submission.

Web-based Inspection and Virus Analysis Tools

Now that you have a sample of the malicious software that is causing trouble, what do you do with it? How can you tell what it's doing, what communication channels it's set up, and what it's doing to the machine it's running on? This section will describe the next actions to take once you have secured your sample. It's worth noting here that you should always have a backup of your sample on a floppy disk or other removable or write-protected media. Often, the original file gets corrupted, deleted, or rendered unusable during analytical processing.

AV Vendors Accept Submissions

All AV vendors accept live samples either through a Web form, or through e-mail. Figure 9.8 contains a list of the major AV vendors and their malware submission e-mail or Web address for easy reference. The first place to which you should send your live malware sample is the AV vendor whose product is used in your organization. If your AV vendor isn't in the list in Figure 9.8**,** it is usually very easy to find the address by going to your vendor's Web site and searching for something like *sample submission* on their site.

Figure 9.8 Common AV Vendors and Their Sample Submission Addresses

AV Vendor	Sample Submission E-Mail or Web address
ClamAV	http://clamav.catt.com/cgi-bin/sendvirus.cgi
Command Software	virus@commandcom.com
Computer Associates (e-Trust)	virus@ca.com
Eset (NOD32)	sample@nod32.com
F-Secure Corp.	samples@f-secure.com
Frisk Software (F-PROT)	viruslab@f-prot.com
Grisoft (AVG)	virus@grisoft.cz
H+BEDV (AntiVir)	virus@antivir.de
Kaspersky Labs	newvirus@kaspersky.com
Network Associates (McAfee)	virus_research@avertlabs.com
Norman (Norman Virus Control)	analysis@norman.no
Panda Software (Panda)	virus@pandasoftware.com
Sophos (SAV)	samples@sophos.com
Symantec (Norton)	avsubmit@symantec.com
Trend Micro (PC-cillin)	http://subwiz.trendmicro.com/SubWiz/Default/asp

By getting the sample off to your AV vendor right away, you will ensure that they'll be able to start working on the appropriate definition (signature) files that are needed to detect and remove the malware that you sent them. Once they have completed their analysis of the sample and they've sent you the necessary definition update files, you'll want to test and deploy them throughout your enterprise right away.

After you've sent off your sample, you'll want more information about what the program does to a machine and/or network, and whether it's already a known threat. There are a few Web-based sites that accept malware sample submissions, which will analyze them for you, up to a point.

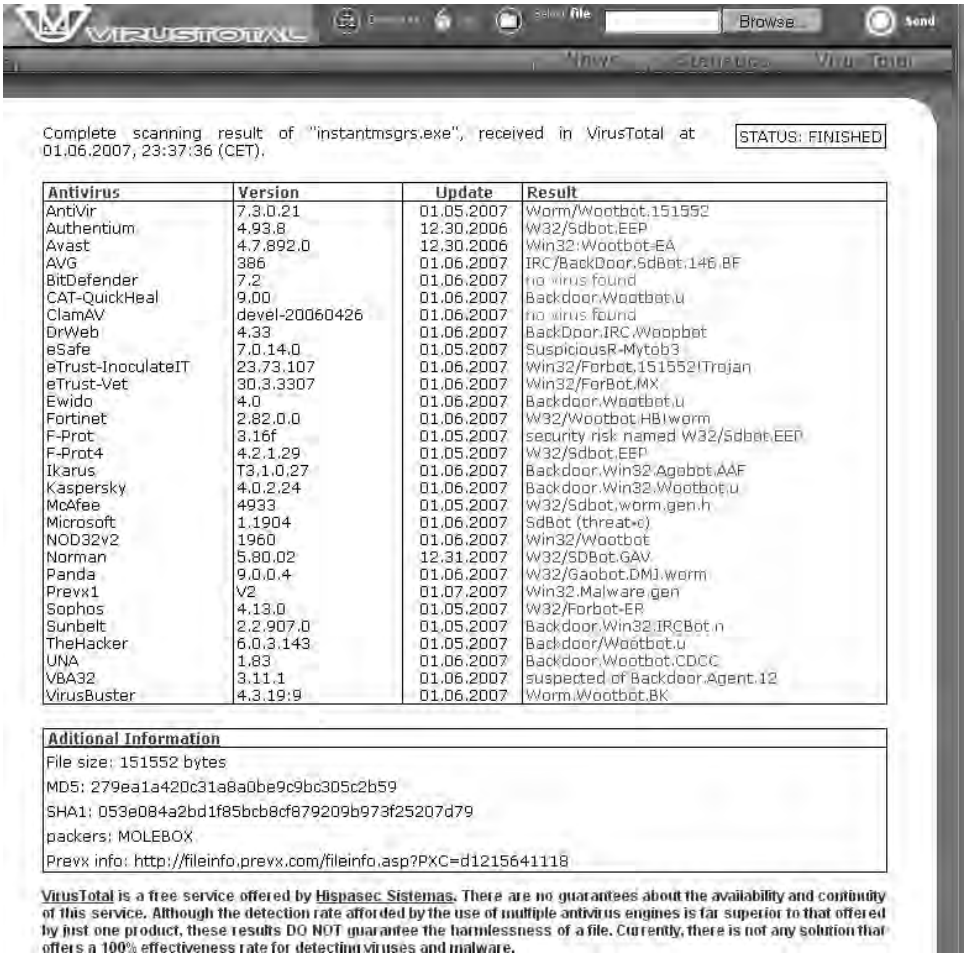
The first two Web sites that we'd like to discuss, allow anyone to submit samples for scanning, and will report their findings. They both use multiple AV scan engines and will report back to you if any of those engines detect your sample as a known threat, and will tell you what that particular vendor has named it. This is very valuable as a front line

check. The results of having VirusTotal (www.virustotal.com) and Jotti Malware Scanner (http://virusscan.jotti.org) scan a submitted item of malware are shown in Figure 9.9 and Figure 9.10, respectively. (You might also want to try Virus.Org (http://scanner.virus.org, which offers a similar service.)

The file that I submitted for this example is a variant of W32/WOOTBOT, which was discovered in March of 2005. Something that you may notice right away is that not all the products identify the sample as a known threat, and that where a threat is identified, the name used may vary widely from product to product. This is often the case. There are so many new malicious programs and variants being created, that it's likely that not all AV vendors will be able to detect 100 percent of them, especially when they first appear. This reinforces the reasoning behind sending your sample to the vendor of your enterprise AV solution as quickly as possible. The second item is that there isn't a standard naming convention within the industry for malware threats. Or to be precise, there is more than one, and it's not practically possible to enforce the universal use of a single standard. (This issue is addressed at greater length in Chapter 1.) Each scanning engine may detect different pieces of a blended or multipolar threat, and the same code can turn up in more than one threat family. This complicates the issue, when different vendors ascribe the same malware to different threat families. We would recommend using the same naming convention within your enterprise that your AV vendor uses, for internal documentation and consistency. However, when discussing malicious code with other vendors, users of other software, mailing lists like those available to AVIEN and AVIEWS members, and so on, you will probably find that it's useful to cross-reference this name with other names, such as that used by the WildList Organization (www.wildlist.org), the Computer AV Research Organization (CARO - www.caro.org), or even the Common Malware Enumeration (CME list) (http://cme.mitre.org/)

Using VirusTotal also has a benefit in that they will submit your sample to the AV vendors that they use for their Web-hosted scans. I would still submit it yourself to your AV vendor, even knowing that VirusTotal will submit as well. Many AV vendors rank the prevalence of each piece of malware according to how many unique submissions they receive from their customers, so it's important to have your sample counted toward their total.

Figure 9.9 Results of Submitting "*instantmsgrs.exe*" Malware Sample to www.virustotal.com



VIRUSTOTAL

Browse... Send

Complete scanning result of "instantmsgrs.exe", received in VirusTotal at 01.06.2007, 23:37:36 (CET). STATUS: FINISHED

Antivirus	Version	Update	Result
AntiVir	7.3.0.21	01.05.2007	Worm/Wootbot.151552
Authentium	4.93.8	12.30.2006	W32/Sdbot.EEP
Avast	4.7.892.0	12.30.2006	Win32:Wootbot-EA
AVG	386	01.06.2007	IRC/BackDoor.SdBot.146.BF
BitDefender	7.2	01.06.2007	no virus found
CAT-QuickHeal	9.00	01.06.2007	Backdoor.Wootbot.u
ClamAV	devel-20060426	01.06.2007	no virus found
DrWeb	4.33	01.06.2007	BackDoor.IRC.Woopbot
eSafe	7.0.14.0	01.05.2007	SuspiciousR-Mytob3
eTrust-InoculateIT	23.73.107	01.06.2007	Win32/Forbot.151552!Trojan
eTrust-Vet	30.3.3307	01.06.2007	Win32/ForBot.MX
Ewido	4.0	01.06.2007	Backdoor.Wootbot.u
Fortinet	2.82.0.0	01.06.2007	W32/Wootbot.HBI!worm
F-Prot	3.16f	01.05.2007	security risk named W32/Sdbot.EEP
F-Prot4	4.2.1.29	01.05.2007	W32/Sdbot.EEP
Ikarus	T3.1.0.27	01.06.2007	Backdoor.Win32.Agobot.AAF
Kaspersky	4.0.2.24	01.06.2007	Backdoor.Win32.Wootbot.u
McAfee	4933	01.05.2007	W32/Sdbot.worm.gen.h
Microsoft	1.1904	01.06.2007	SdBot (threat=c)
NOD32v2	1960	01.06.2007	Win32/Wootbot
Norman	5.80.02	12.31.2007	W32/SDBot.GAV
Panda	9.0.0.4	01.06.2007	W32/Gaobot.DMJ.worm
Prevx1	V2	01.07.2007	Win32.Malware.gen
Sophos	4.13.0	01.05.2007	W32/Forbot-ER
Sunbelt	2.2.907.0	01.05.2007	Backdoor.Win32.IRCBot.n
TheHacker	6.0.3.143	01.05.2007	Backdoor/Wootbot.u
UNA	1.83	01.06.2007	Backdoor.Wootbot.CDCC
VBA32	3.11.1	01.06.2007	suspected of Backdoor.Agent.12
VirusBuster	4.3.19:9	01.06.2007	Worm.Wootbot.BK

Aditional Information
File size: 151552 bytes
MD5: 279ea1a420c31a8a0be9c9bc305c2b59
SHA1: 053e084a2bd1f85bcb8cf879209b973f25207d79
packers: MOLEBOX
Prevx info: http://fileinfo.prevx.com/fileinfo.asp?PXC=d1215641118

VirusTotal is a free service offered by Hispasec Sistemas. There are no guarantees about the availability and continuity of this service. Although the detection rate afforded by the use of multiple antivirus engines is far superior to that offered by just one product, these results DO NOT guarantee the harmlessness of a file. Currently, there is not any solution that offers a 100% effectiveness rate for detecting viruses and malware.

Tools & Traps

Malware Submission Sites – The Downside

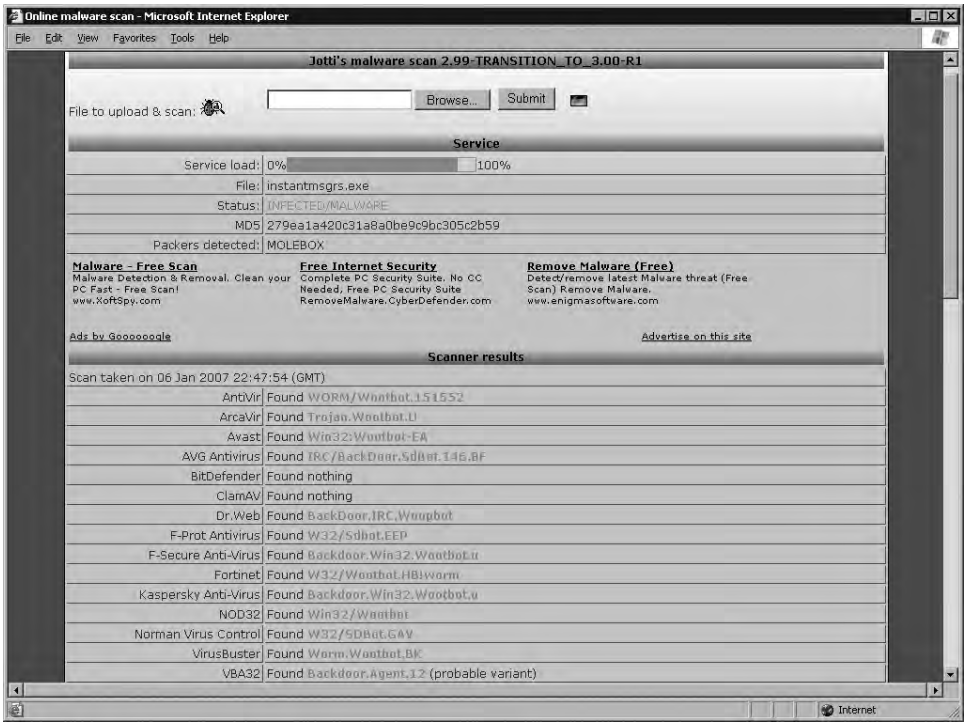
Sites like Jotti and VirusTotal can be useful in the early stages of an outbreak, as part of the process of identifying the precise nature of a possible threat. However, there are dangers in putting too much trust in these sites. Most obviously, there is no

way of guaranteeing that a file not identified by any of the engines used as being malicious might not turn out to be malicious, nonetheless. Not proven is not the same as innocent.

However, you're not just putting your trust in the vendors whose scanning engines are used. You're also placing your trust in the site to update the scanner promptly, configure it appropriately, and use an appropriate methodology. For example, such sites are usually reticent on precisely how they implement their scanning in terms of the level of heuristics employed. This is not altogether a bad thing. It makes it (a little) harder for the bad guys to misuse a site to test their malware fully against their scanners for free.

It does, however, make it harder to evaluate their accuracy and fitness for particular purposes. A false positive, false negative, or misdiagnosis can be hard to interpret correctly. In particular, it's not a good idea to use these sites as a basis for comparative evaluation of scanners.

Figure 9.10 Results of Scanning Sample "*instantmsgrs.exe*" on http://virusscan.jotti.org



At the time of this writing, we could only find three AV vendors that would accept a single sample online, perform a check on that individual file, and display the results, though several vendors (Trend, ESET et al.) have online scanners that will check a whole system. Granted, the results are very basic and limited, usually to the display of no more than "possibly not-infected, further analysis needed" or "infected with *<malware name>*." In McAfee's case, if it's a new malicious program that isn't within their daily definition files, they will send you a special definition file (an "*extra.dat*" file) that you can use to detect and clean the submitted malware, which is very useful if your organization is a McAfee customer. Figures 9.11, 9.12, and 9.13 are screenshots showing typical results of submitting a known malicious program to each of these three vendors for instant online analysis. If you're a customer of one of the three, there is a particular benefit to using their online analysis. The first benefit is that you're sending them a suspicious file that their virus engineers will be able to analyze for incorporation into their next update, if appropriate. The second, if it's a known threat to them, is that you'll find out right away what they are calling this threat, and what update is needed to detect and clean it. If you're not a customer of any of the three, then it may be enough simply to use VirusTotal or Jotti Malware Scanner and let those Web sites check your sample against many different AV products. There is, however, the possibility that a more accurate result will be obtained from the configuration on the vendor's site.

Figure 9.11 Fortinet.com Results After Scanning "*instantmsgrs.exe*"

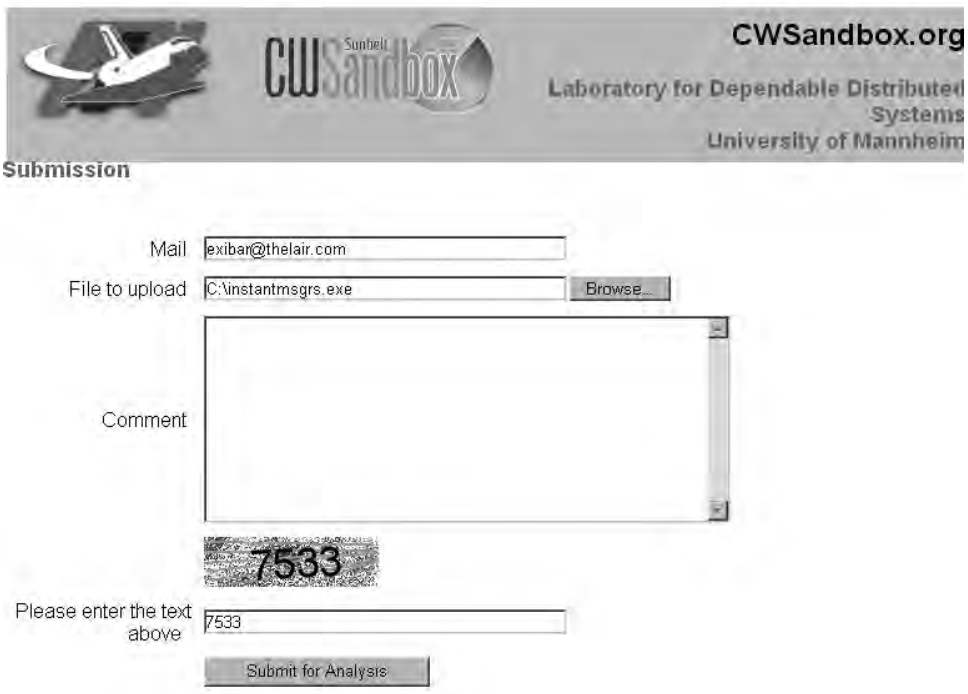
Figure 9.12 Kaspersky.com Results After Submitting "*instantmsgrs.exe*"

Figure 9.13 Results after Submitting *instantmsgrs.exe* to McAfee AVERT Labs Submission Site, www.webimmune.net

Using an Online Malware Inspection Sandbox

There are a few sites at the time of this writing that accept malware samples. These sites will do a fairly comprehensive "sandbox" analysis for you, and will e-mail you the results. These sites are extremely useful when you find that you have a brand new malicious program or variant, one that none of the AV vendors can detect and clean at the time of discovery. These sandbox analysis reports enable you to put quick countermeasures in place, even before you're able to run your sample on your own test machine (known as a "goat" or "sacrificial goat" machine). (More on using a goat machine later in this chapter.)

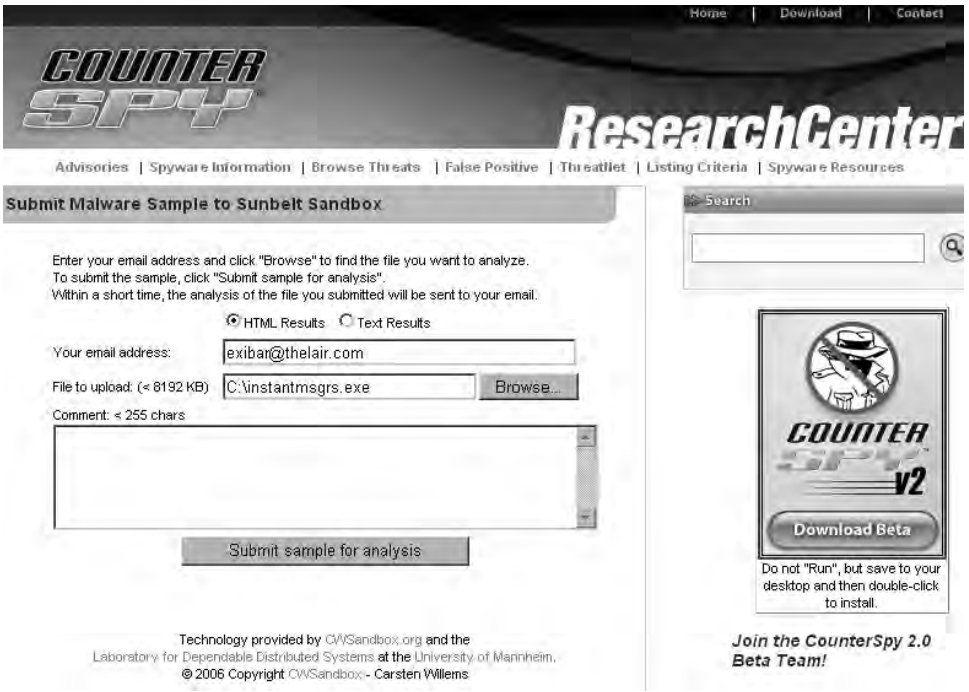
Figure 9.14 CWSandbox's Submission Page Found at www.cwsandbox.org/



There are three such Sandbox sites available to the public, free of charge, at the time of this writing. Figure 9.14 and Figure 9.15 show two of these sites. They use the same back-end engine and produce the same results. The first site, CWSandbox (www.cwsandbox.org/), will only send you the malware analysis report in eXtensible

Markup Language (XML) format, which needs to be fed into an XML processor in order to be easily readable. The second, CounterSpy's Sunbelt sandbox (http://research.sunbelt-software.com/submit.aspx), will send you the analysis report in a choice of Hypertext Markup Language (HTML) or plain text.

Figure 9.15 Counter Spy's Research Submission Page Found at http://research.sunbelt-software.com/submit.aspx



The third Sandbox, shown in Figure 9.16, is Norman AV' sandbox (http://sandbox.norman.no/live.html). Norman Antivirus' sandbox will send you the results in text format only.

Figure 9.16 Norman Sandbox Submission Page, Found Here: http://sandbox.norman.no/live.html

Out of the three sandboxes, I prefer the results from CounterSpy's Sunbelt Sandbox. I feel it is the most usable and provides the most information without having to process an XML file. CounterSpy's Sunbelt Sandbox and CWSandbox both provide a very in-depth analysis on the malware that is submitted to those sites. Although Norman's sandbox provides very good analysis on the initial file that is submitted, it does not appear to process any files that are created by that initial file. So let's say that the initial file you submit creates two files and executes them on your system. CounterSpy and CWSandbox will process those two other files as well as your original sample. This can provide a huge benefit when you're trying to control an outbreak situation.

I've included samples of a CounterSpy's Sunbelt Sandbox report and a Norman Sandbox report. Within these samples you can clearly see that there is more information provided by the Counterspy report over the Norman report. I have not included a sample report from CWSandbox, as it is essentially the same report as the one from Counterspy, but in XML format. It's also worth noting that the sample report from CounterSpy is only an extract

from the entire report, which is nearly seven full pages long! Within the CounterSpy report, you can clearly see the registry keys that are created, any mutexes that are created, any network connections that are created or used, and many other useful pieces of information. Everything provided in the reports from the Sandbox Web sites can be of considerable value when you're trying to prevent a massive outbreak within your enterprise.

Here is a partial sample report from CounterSpy Sunbelt Sandbox (**http://www.cwsandbox.org/**) received after submitting "instantmsgrs.exe," an SDBOT variant.

Analysis Summary:

Analysis Date	1/7/2007 4:57:43 AM
Sandbox Version	1.106
Filename	279ea1a420c31a8a0be9c9bc305c2b59.exe

Technical Details:

The following process was started by process: 1

Analysis Number	2
Parent ID	1
Process ID	1424
Filename	C:\WINDOWS\system32\instantmsgrs.exe -bai c:\279ea1a420c31a8a0be9c9bc305c2b59.exe
Filesize	151552 bytes
MD5	279ea1a420c31a8a0be9c9bc305c2b59
Start Reason	CreateProcess
Termination Reason	Timeout
Start Time	00:04.844
Stop Time	01:00.344
Detection	Trojan - W32/Sdbot.EEP (Authentium Command Antivirus - EngVer: 4.92.123.35 - SigVer: 20061222 35)
	Infected - Generic.Sdbot.B2963C9B (BitDefender Antivirus - EngVer: 7.0.0.2311 - SigVer: 7.10647)
	Backdoor - Backdoor.Win32.IRCBot.n (CounterSpy - EngVer: 2.1.628.0 - SigVer: 469)
	Infected -Backdoor:Win32/Sdbot!2111 (Microsoft Malware Protection -EngVer: 1.1.1904.0 - SigVer: Tue Dec 26 01:26:26 2006)
	Virus - W32.Spybot.Worm (Norton AntiVirus - EngVer: 20061.3.0.12 - SigVer: 20061226 13:19:02)

	Loaded DLLs
DLL-Handling	C:\WINDOWS\system32\instantmsgrs.exe C:\WINDOWS\system32\ntdll.dll C:\WINDOWS\system32\kernel32.dll C:\WINDOWS\system32\user32.dll C:\WINDOWS\system32\GDI32.dll C:\WINDOWS\system32\advapi32.dll C:\WINDOWS\system32\RPCRT4.dll C:\WINDOWS\system32\oleaut32.dll C:\WINDOWS\system32\msvcrt.dll C:\WINDOWS\system32\ole32.dll C:\WINDOWS\system32\comctl32.dll C:\WINDOWS\system32\wsock32.dll C:\WINDOWS\system32\WS2_32.dll C:\WINDOWS\system32\WS2HELP.dll C:\WINDOWS\system32\pstorec.dll C:\WINDOWS\system32\ATL.DLL C:\WINDOWS\system32\Wship6.dll C:\WINDOWS\system32\Secur32.dll user32.dll gdi32.dll shlwapi.dll MSVCRT.dll MSVCP60.dll iphlpapi.dll KERNEL32.dll USER32.dll ADVAPI32.dll SHELL32.dll WS2_32.dll MPR.dll PSAPI.DLL DNSAPI.dll KERNEL32.DLL kernel32.dll

C:\WINDOWS\system32\mswsock.dll
hnetcfg.dll
C:\WINDOWS\System32\wshtcpip.dll
C:\WINDOWS\System32\mswsock.dll
C:\WINDOWS\System32\winrnr.dll
rasadhlp.dll

New Files

\Device\Tcp - Stored:
\Device\Ip - Stored:
\Device\Ip - Stored:
\Device\RasAcd - Stored:

Opened Files

\\.\Ip - Stored:
\\.\pipe\net\NtControlPipe8 - Stored:

Filesystem

Deleted Files

c:\279ea1a420c31a8a0be9c9bc305c2b59.exe

Chronological order

Create/Open File: \Device\Tcp (OPEN_ALWAYS)
Create/Open File: \Device\Ip (OPEN_ALWAYS)
Create/Open File: \Device\Ip (OPEN_ALWAYS)
Open File: \\.\Ip (OPEN_EXISTING)
Delete File: c:\279ea1a420c31a8a0be9c9bc305c2b59.exe
Create/Open File: \Device\RasAcd (OPEN_ALWAYS)
Open File: \\.\pipe\net\NtControlPipe8 (OPEN_EXISTING)

Changes

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run "mousedrive.exe" = instantmsgrs.exe
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce "mousedrive.exe" = instantmsgrs.exe
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run "mousedrive.exe" = instantmsgrs.exe

Registry

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce "mousedrive.exe" = instantmsgrs.exe
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices "mousedrive.exe" = instantmsgrs.exe

	Reads
	HKEY_LOCAL_MACHINE\Software\Microsoft\Rpc\ SecurityService "DefaultAuthLevel"
	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\ Services\SharedAccess "Start"
Process	Enum Processes
Management	Enum Modules - Target PID: (892)
	Enum Modules - Target PID: (1688)
	Enum Modules - Target PID: (1796)
	Enum Modules - Target PID: (1100)
	Enum Modules - Target PID: (1108)
	Enum Modules - Target PID: (1148)
	Enum Modules - Target PID: (1424)
	Open Process - Filename () Target PID: (4)
	Open Process - Filename () Target PID: (596)
	Open Process - Filename () Target PID: (644)
	Open Process - Filename () Target PID: (668)
	Open Process - Filename () Target PID: (712)
	Open Process - Filename () Target PID: (728)
	Open Process - Filename () Target PID: (740)
	Open Process - Filename (C:\Program Files\Faronics\ Deep Freeze\Install C-0\DF5Serv.exe) Target PID: (892)
	Open Process - Filename () Target PID: (936)
	Open Process - Filename () Target PID: (1036)
	Open Process - Filename () Target PID: (1124)
	Open Process - Filename () Target PID: (1176)
	Open Process - Filename () Target PID: (1336)
	Open Process - Filename () Target PID: (1524)
	Open Process - Filename (C:\WINDOWS\Explorer.EXE) Target PID: (1688)
	Open Process - Filename (C:\WINDOWS\system32\ VTTimer.exe) Target PID: (1796)
	Open Process - Filename (C:\Program Files\Faronics\ Deep Freeze\Install C-0_$Df\FrzState2k.exe) Target PID: (1100)
	Open Process - Filename (C:\WINDOWS\system32\ rundll32.exe) Target PID: (1108)

	Open Process - Filename (C:\WINDOWS\system32\WgaTray.exe) Target PID: (1148)	
	Open Process - Filename () Target PID: (1236)	
Service Management	Open Service Manager - Name: "SCM"	
System Info	Get System Directory	
	DNS Lookup	
	Host Name	**IP Address**
Network	donna.teh1.com	
Activity	donna.teh1.com	192.168.1.105
	Opened listening TCP connection on port: 3626	
	Outgoing connection to remote server: 255.255.255.255 TCP port 6667	
	Outgoing connection to remote server: donna.teh1.com TCP port 6667	

Next, here is a report received from Norman Sandbox (http://sandbox.norman.no/live.html) after submitting "*instantmsgrs.exe*," a W32/WootBot (AKA: SdBot) variant:

```
Your message ID (for later reference): 20070114-565

Hello,

Thanks for taking the time to submit your samples to the Norman Sandbox
Information Center. Customer delight is our top priority at Norman. With that
in mind we have developed Sandbox Solutions for organizations that are committed
to speedy analysis and debugging.

Norman Sandbox Solutions give your organization the opportunity to analyze files
immediately in your own environment.

To find out how to bring the power of Norman Sandbox into your test environments
follow the links below.

Norman Sandbox Solutions

http://www.norman.com/Product/Sandbox-products/

Norman Sandbox Analyzer

http://www.norman.com/Product/Sandbox-products/Analyzer/

Norman Sandbox Analyzer Pro

http://www.norman.com/Product/Sandbox-products/Analyzer-pro/

Norman SandBox Reporter

http://www.norman.com/Product/Sandbox-products/Reporter/

instantmsgrs.exe : W32/SDBot.gen2 (Signature: W32/SDBot.GAV)
```

```
[ General information ]
    *File length: 151552 bytes.
    *MD5 hash: 279ea1a420c31a8a0be9c9bc305c2b59.

[ Security issues ]
    *Modified OS kernel function code.

(C) 2004-2006 Norman ASA. All Rights Reserved.

The material presented is distributed by Norman ASA as an information source only.

Sent by exibar@thelair.com. Received 14.Jan 2007 at 03.16 -processed 14.Jan 2007
at 03.18.
```

Tools and Traps

Signing Up for Norman Sandbox's Daily Digest

Even though the report from my submission to Norman Sandbox of the SDBOT variant, as shown above, is a little sparse, Norman has a wealth of information to share with qualified individuals. If you head out to http://sandbox.norman.no/live_2.html, you'll see that the Norman Sandbox shows you the last 30 or so files that have been submitted. Within those reports, all the Uniform Resource Locators (URLs) have been removed to prevent malicious use. The URLs have all been replaced with the word "REMOVED," which itself is a link to http://sandbox.norman.no/form.html.

If you follow that link, you will be presented with a sign-up form that will enable you to receive a daily digest of all the submissions that occurred during that day. You will have to be approved by them before you start receiving the daily digests, however.

The reports contained within the digest contain all the URLs, Internet IP's, and so forth to which the submissions attempt to open a connection. The reason that this is so valuable is that you can create some parsing scripts that can compare the daily digest URLs and IP addresses to your firewall logs. Any machine within your enterprise that is attempting to go out to any of those addresses or URLs, is surely controlled by malware!

Using Packet Analyzers to Gather Information

All of the tools previously mentioned in this section, enable you to get to a point where someone else can analyze the malware you've found in a sanitized environment. In the next section, we'll look at tools that you can use in your own enterprise environment. Ideally, you'll use them on a typical enterprise system; not an actual production machine but a "goat" machine.

A "sacrificial goat" machine or simply a "goat" machine is nothing more than a victim machine whose sole purpose is to be the victim machine on which you'll be actually running your sample. The purpose of this machine is simply to sacrifice its own integrity so you can gather as much threat data as possible in a safe environment, without fear that the sample might escape onto your production network.

This machine should not normally be connected to your production network; its only network connection should be directly to the Internet or to a 'faked' server providing DNS, for instance. This machine should very closely resemble your production machine setup, perhaps created using the same image process. It should contain all of the tools that you require to perform your testing, along with some way to copy your sample over to it, either via floppy disk or via a secure Universal Serial Bus (USB) flash drive.

Your goat machine doesn't even have to be an actual physical machine. Using software such as VMware's VMware Workstation (www.vmware.com) or Microsoft's Virtual PC (www.microsoft.com), you can have multiple, fully functional operating systems running on one host operating system simultaneously.

It's important to note that some malware writers have placed special checks in their code that will check whether it's being run in a virtual environment. If it is, it will stop running. So far, this behavior has not been much observed, but it is happening more often.

The use of a packet analyzer is one of the fundamentals techniques of malware analysis. A packet analyzer is a program that will enable you to look directly at information passing across the network to and from the machine on which it's running. Three of the most popular packet analyzers available today are tcpdump for UNIX-type machines, Windump for Windows platforms, and Wireshark (formerly known as Ethereal and available for both windows and UNIX.)

Before you can run any packet capture programs (and many of the more advanced tools coming up), you must have the correct drivers installed. In this case, you must have "packet capture libraries" installed. For Linux you must install Libcap. Windows platforms require the installation of WinPcap. Both installations are quick and simple, and the programs are available at the same location as tcpdump and windump (see below).

Tcpdump and Windump are basically the same application. Windump is a port of tcpdump from UNIX to Windows. The output from both programs and their command line switches are identical. TCPdump can be found at www.tcpdump.org, and WinPcap can be found at www.winpcap.org.

Typical tcpdump and windump command line options are discussed here. For the complete manual on tcpdump, please refer to the *man* pages found at www.rt.com/man/tcpdump.1.html. We've presented the most common command-line options and expressions below.

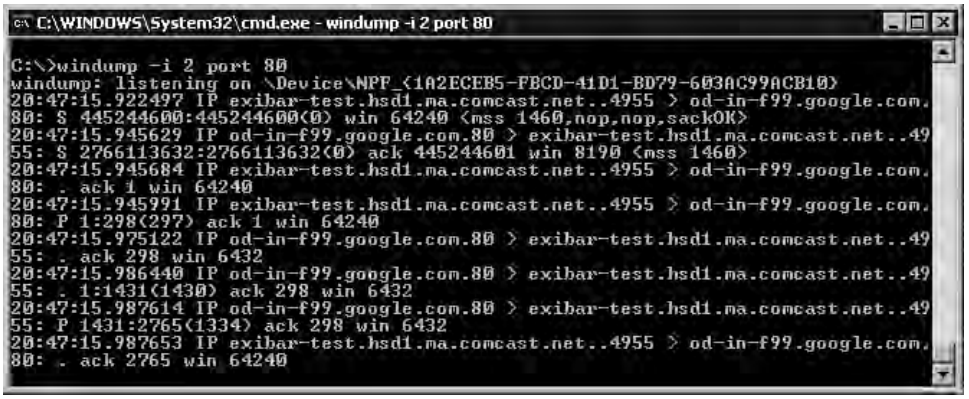
Results of Running *windump* at the Command Line to Show Proper Syntax Formatting

```
windump version 3.9.5, based on tcpdump version 3.9.5
WinPcap version 3.1 (packet.dll version 3, 1, 0, 27), based on libpcap
version 0.9[.x]
Usage: windump [-aAdDeflLnNOpqRStuUvxX] [ -B size ] [-c count]
[ -C file_size ][ -E algo:secret ] [ -F file ] [ -i interface ]
[ -M secret ] [ -r file ] [ -s snaplen ] [ -T type ] [ -w file]
[ -W filecount ] [ -y datalinktype ] [ -Z user ] [ expression ]
```

Here is a listing of tcpdump and windump common options and expressions

- ***-?*** List command line syntax options (see Figure 9.17).
- ***-c*** End capture after capturing *[count]* number of packets.
- ***-i*** Listen for packets on *[interface]*. Default is to use the lowest numbered interface. On Windows boxes, this is usually the dial-up adapter.
- ***-n*** Don't convert numbers to common names. IP addresses, ports, and so on, will remain in numeric format, and not be converted to common names. This speeds up some captures due to the lack of Domain Name Domain name system (DNS) lookups.
- ***-r*** Read from a packet capture file instead of from the network interface. Use the "*-w*" option to create a file.
- ***-v*** Verbose output.
- ***-vv*** More verbose output.
- ***-vvv*** Extremely verbose output.
- ***-w*** Write the capture to a file instead of displaying the packets on the screen.
- ***host*** Limit capture to this *[host]* only. *[Host]* can be either an IP address or a qualified domain name, and can be preceded with DST or SRC for Destination or Source host, respectively.
- ***port*** Limit capture to *[port]* only. Can also be preceded with a protocol type of UDP or TCP.

Figure 9.17 Windump Output Showing a Connection to www.google.com Using IE 6.0



Don't let the plethora of options available with tcpdump and windump scare you. Using these two tools for malware analysis is really quite simple and basic. A typical use would be for capturing every packet to a file to be analyzed later, while gathering information of a piece of malware that you're analyzing.

In order to use tcpdump or windump, you'll first want to find out which interface you want to capture packets from. On a UNIX system, the command *ifconfig* will give you the information about the interfaces needed. On a Windows platform, you can run *windump –D* to will display results similar to this:

```
c:\>windump -D
1.\Device\NPF_GenericDialupAdapter (Generic dialup adapter)
2.\Device\NPF_{2F11EF65-4EDD-4FA6-B9A1-567C4AA8CD1C}
(VMware Virtual Ethernet Adapter)
3.\Device\NPF_{B5410B0A-C3B7-4A5A-82C2-B4EEFFBD5350}
(Marvell Gigabit Ethernet Controller (Microsoft's
Packet Scheduler) )
4.\Device\NPF_{8307E6E0-6B30-4387-94B8-45F212CFA29C}
(VMware Virtual Ethernet Adapter)
```

From the above sample output you can see that the "Generic Dialup Adapter" is listed first, then one of two VMware virtual adapters, while number three is the main network interface adapter from which we want to capture packets. The command line that you would use in this case is shown below. This would capture as much data as possible from that network interface adapter, and write it to a file for later analysis.

```
c:\>windump -i 3 -w c:\test.pcap
windump: listening on \Device\NPF_{B5410B0A-C3B7-4A5A-82C2-B4EEFFBD5350}

99 packets captured
99 packets received by filter
0 packets dropped by kernel
```

The above command is broken down as follows:

- ***windump*** Core command to start the packet capture
- ***-i 3*** Capture packets on Interface 3
- ***-w c:\test.pcap*** Don't display the packets on the screen. Instead, write them to file *"c:\test.pcap"* for later analysis.

As you can see from Figure 9.17, the usage of tcpdump and its output can be quite overwhelming. If you don't use the correct command-line options and expressions, you could miss key data. On the other hand, if you don't use any options or expressions and capture *all* the packets from an interface card, there is simply too much data to be able to go through in a timely manner. This leads me to consider the usage of Wireshark (formally known as Ethereal and freely available at http://www.wireshark.org). Wireshark will help organize the output of the capture in a fashion that makes it much easier to use.

Tools & Traps

Wireshark-infested Custard

Wireshark is freely available at www.wireshark.org. There is also a wealth of documentation at www.wireshark.org/docs/

Wireshark development was started back in 1998 by a man named Gerald Combs, originally under the name of Ethereal and under the terms of the GNU General Public License. Since 1998, hundreds of people have contributed to the project to make it the industry standard in packet capture applications. In 2006, the name Ethereal changed to Wireshark due to copyright and trademark issues (www.wireshark.org/faq.html -q1.2; trends.newsforge.com/article.pl?sid=06/06/09/1349255&from=rss), but improvements continue to be made. Wireshark also remains one of the most widely used packet capture applications available and is still free for use under the GNU General Public License (GPL).

We don't have space to do more than hint at the capabilities and functionality of Wireshark here. However, Syngress have published a book by Angela Orebaugh *et al* called "Wireshark and Ethereal: Network Protocol Analyzer Toolkit," which goes into considerable detail.

More details at www.syngress.com/catalog/?pid=3770.

As well as performing packet captures on its own, Wireshark has the ability to read all PCAP-formatted output files including tcpdump, windump, the Snort Intrusion Detection System (IDS), and many other applications that produce output in PCAP format. Wireshark has a nice GUI front-end and includes many built-in capabilities that make it easy and intuitive to use for both producing and analyzing packet captures. Figure 9.18 shows a connection to www.google.com in action, using Internet Explorer 6.0. You can easily see the Synchronous (SYN), Synchronous/Acknowledge (SYN-ACK), Acknowledge (ACK) of the three-way TCP handshake on lines 4, 5, and 6. Then the connection out to www.google.com via Hypertext Transfer Protocol (HTTP) on line 7 is completed and the Google homepage is displayed.

Tools and Traps

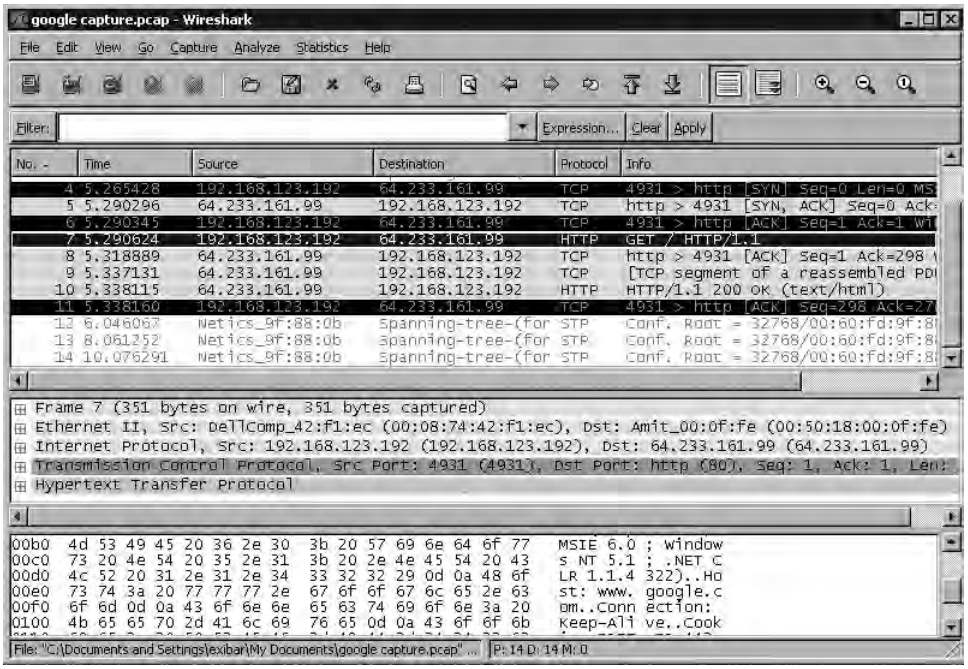
Understanding the Three-way Handshake

The three-way handshake describes the method TCP uses to establish a connection. A server opens a port up for connections (a "passive open"). When the server is "listening," a client can initiate an "active open" by sending a SYN to the server. The active open is performed by sending a SYN to the server.

Now the server acknowledges the SYN request with a SYN-ACK.

Finally, the client responds to the server's acknowledgement with an answering acknowledgement simply called an ACK. It could be said that once the client and the server have received an acknowledgement, the connection is formally open.

Figure 9.18 Wireshark Captures the Complete Connection to www.google.com



Examining Your Malware Sample with Executable Inspection Tools

One of my standard procedures when I get my malware sample for analysis is to run it against another www.sysinternals.com program, called *Strings.exe*. This program will go through the file and (by default) look for three or more consecutive Unicode and/or American Standard Code for Information Interchange (ASCII) characters in a row. This isn't an analysis procedure that I focus too much time on (and Strings is considered later in more detail in the section on advanced forensics), but may be well worth the time spent in a lot of cases. Sometimes Internet Relay Chat (IRC) server names, IP addresses, and other useful information can be gathered by looking through the malware executable itself.

Available Command-line Options for STRINGS.EXE from www.sysinternals.com

```
Strings v2.1
Copyright (C) 1999-2003 Mark Russinovich
Systems Internals -www.sysinternals.com
usage: strings [-s] [-n length] [-a] [-u] [-q] <file or directory>
-s Recurse subdirectories
-n Minimum string length (default is 3)
-a Ascii-only search (Unicode and Ascii is default)
-u Unicode-only search (Unicode and Ascii is default)
-q Quiet (no banner)
```

Up to this point, all the tools that we've discussed have been either open source or freeware tools. There are a few commercial tools that we use on a regular basis, and one such tool is from FreeSpace Internet Security (http://www.freespaceinternetsecurity.com), and is called VIP Utility. VIP Utility was written back in 2003, by Lixin Lu, who then went ahead and founded Freespace Internet Security the same year. VIP Utility produces results that are very similar to those given by the previously mentioned Web inspection sites. The benefit of using this utility is that it is run in your own environment, and you get immediate results, whereas there is some delay waiting for results e-mailed to you from Web inspection sites.

VIP Utility is basically an open sandbox that monitors the executable that is passed through it, using a combination of heuristics and behavior monitoring.

The reason that I say it's an "open" sandbox is that in some cases, VIP Utility will allow the executable to create actual files or folders on the host system. VIP Utility does a remarkable job in terms of "rolling back" anything that the executable does to the host system in this regard. In the two years or so that this author has been using VIP Utility to analyze malware code, I have never once had any problems on the host. Nonetheless, we do not recommend running VIP Utility, or any malware analysis tools, on a production machine. Always use a goat machine or a goat virtual machine for analysis purposes.

Using VIP Utility is very simple. After installation and registration of the product, you'll notice if you right-click on an executable, as shown in Figure 9.19, that there is a new "send-to" menu option**.** This is the command that you'll use to send an executable through to the VIP Inspector, which is the core inspection module of VIP Utility. Once you do this, you'll see a DOS box open up while VIP Inspector is analyzing the executable. When this DOS box closes, your analysis is complete.

Once the VIP Inspector has completed its analysis, it will produce a report and store that report in the "reports" directory, which resides inside the installation directory for VIP Utility. The reports are very straightforward and easy to read. A sample report is shown in Figure 9.20.

Figure 9.19 Sending a File Over to VIP Utility Using the "Send To" Right-click Menu

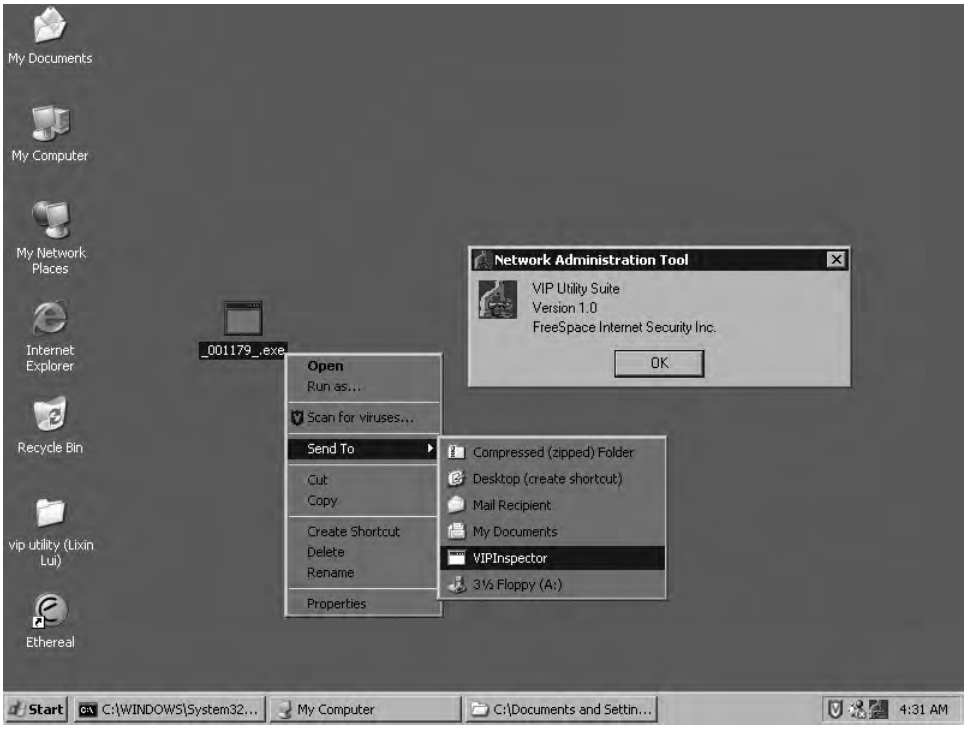


Figure 9.20 Sample Output From VIP Utility

```
File name: scpr32b.exe
Warning Level: 10
Viral Type: Infected by a new variant of Unknown virus.
Activities:

1. Tries to set unhandled exception filter.
2. Tries to start up a Winsock to communicate outside.
3. Tries to delete file autorun.inf.
4. Tries to create a mutex named as 3676C64A-W454-122E-BFC6-
083C2BF4S551.
5. Tries to query Windows folder.
6. Tries to copy itself onto the system as
C:\WINDOWS\System\CSRSS.EXE.
7. Tries to open registry
Software\Microsoft\Windows\CurrentVersion\Run.
8. Tries to set auto run for the newly dropped executable file:
.svchost.
9. Tries to set unhandled exception filter.
```

Another extremely useful commercial tool is InCtrl5, from PC Magazine, and this can be found at http://www.pcmag.com/article2/0,4149,9882,00.asp. The cost for this product is negligible, and is well worth paying, given everything it does. InCtrl5 basically takes a snapshot of all files, registry keys, text files, and INI files. It takes these snapshots before and after the file is executed, compares the differences, and generates a report based upon those differences.

WARNING

It's very important to note that when using InCtrl5 to track changes when a file executes, the file is actually executed on that machine, live. It is not run in a sandbox environment, as is the case with VIP Utility. If you use InCtrl5 to analyze an actual virus or worm, that machine will be infected after the execution is complete. This is where using VMware's "VMware Workstation" to host your virtual goat machine comes in handy. VMware has a "snapshot" capability that allows you to revert back to a previous machine state before the infection took place, at the click of a button.

Figure 9.21 shows the GUI for InCtrl5. It is very intuitive to use, and the default settings work in 99 percent of all situations that you are likely to encounter. The only feature that we would add would be under the "Text Files" section. By default, the program will only track inside *autoexec.nt* and *config.nt*. We would recommend that you add the *c:\windows\system32\drivers\etc\hosts* file to this listing as well, as many malicious programs will attempt to modify this file.

Figure 9.21 The Main GUI Front End for InCtrl5 from PCMagazine

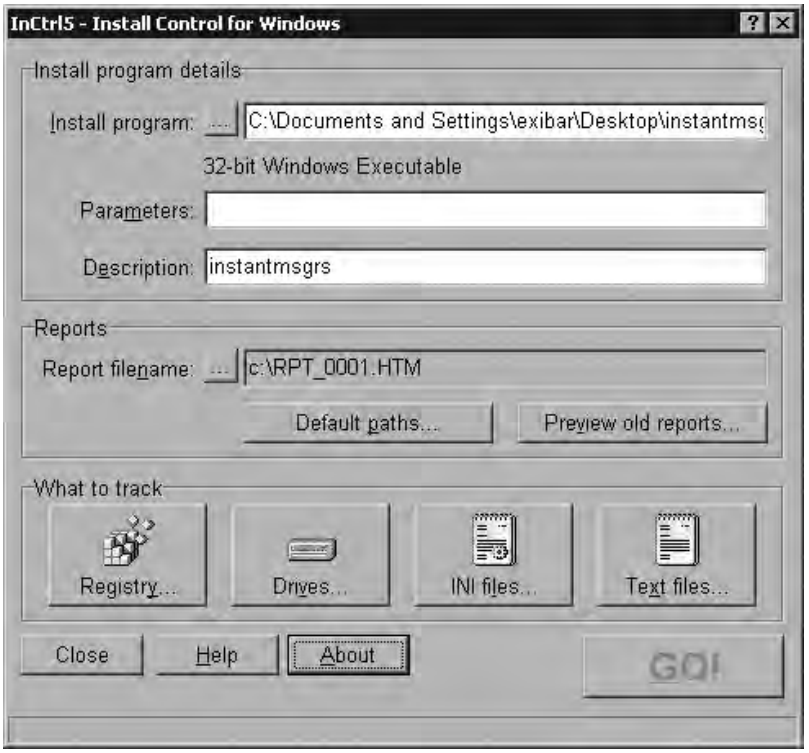


Figure 9.22 shows InCtrl5 in action as it goes through the machine, analyzing the state of the machine before the executable is run.

Figure 9.22 InCtrl5 Collects the Data Before Launching the Executable

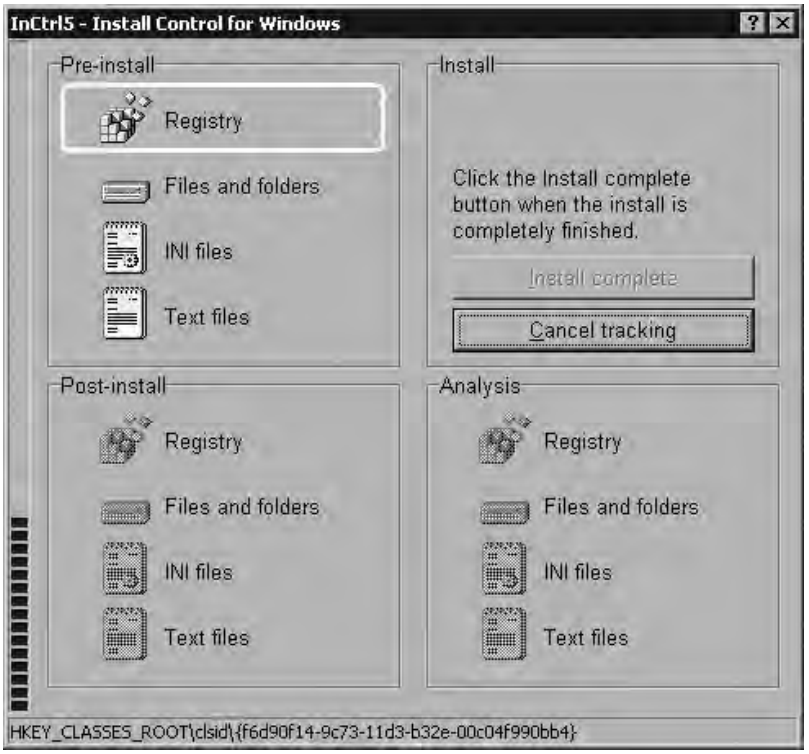
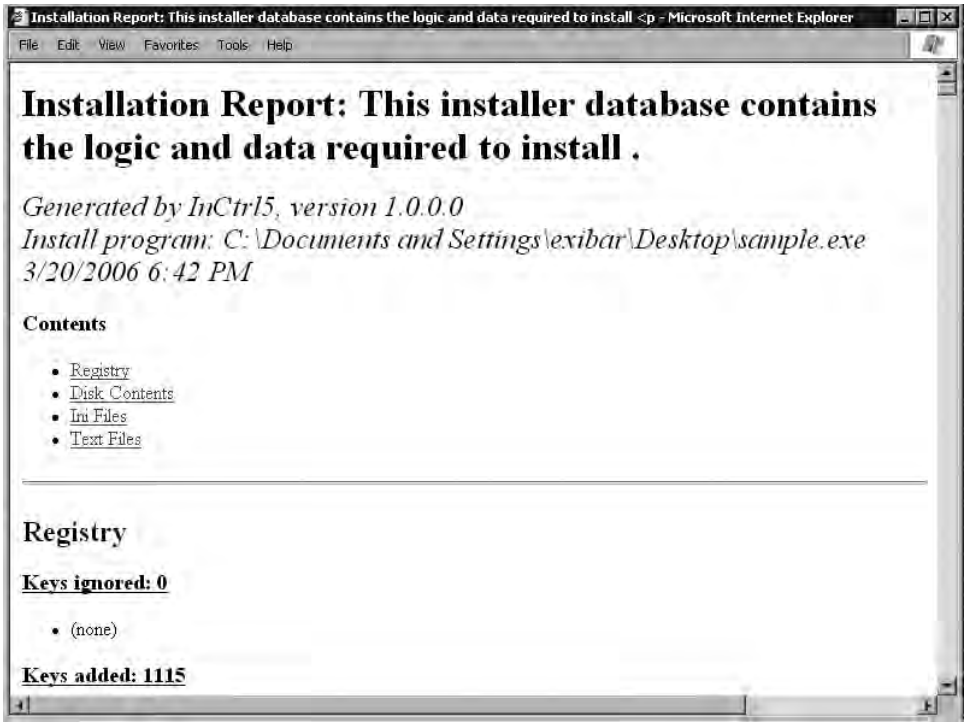


Figure 9.23 shows an extract from a sample report from InCtrl5. Notice that the executable *sample.exe* added 1,115 new registry keys!

Figure 9.23 Sample Report After Running InCtrl5



TIP

In order to minimize the number of times that a piece of malware code is executed on a live system, we'll always have WireShark up and running before we launch the executable with InCtrl5. This way, we'll be able to gather data on any network activity that the executable is generating, along with the machine changes that it's causing. By doing this, we only have to physically run our sample once, and we gather all the data that we need to analyze it properly.

Using Vulnerability Assessment and Port Scanning Tools

Many pieces of malware code today make use of certain vulnerabilities in order to propagate. Alternatively, some of the more notorious malicious programs will open up a port on your machine in order to allow others to connect to it. Part of the process

of analysis requires you to be able to search out those vulnerable machines on your network. After all, it's not good enough just knowing that your sample malware opens up port TCP/1337 on your computer system if you can't actually hunt down vulnerable and infected systems and clean them up, using the data that you gained as a result of the analysis. This next section will give you a feeling for the most popular (and free!) vulnerability assessment and port scanning tools available today.

The first such tool is a port-scanning tool, probably the most well known of its type. Its presence in movies such as *The Listening,* and recently and much more notably in *The Matrix Reloaded* have elevated this tool way beyond cult status. This tool is *Nmap* from www.insecure.org. Fyodor, the creator of Nmap, set the standard in port scanners when its source code made its first appearance in Phrack Magazine, volume 7, Issue 51 back in September 1997. Since that fateful day in September 1997, there have been literally hundreds of improvements including an optional GUI front-end version called *Nmapfe*, also available from www.insecure.org.

There are many command-line options that can be used with Nmap; I've listed the most useful for our purposes below**.** For a complete listing of command-line options for use with Nmap, please refer to www.insecure.org and the help pages found there.

Most Common Command Line Switches for Scanning with Nmap

```
Usage: nmap [Scan Type(s)] [Options] {target machine or range}

-P0: Do not ping the target machine first, simply check for open ports
-n: Don't resolve hostnames during scan. Can speed up performance
significantly
-sT: Perform a full TCP connect scan
-sU: Scan UDP ports only
-p <port ranges>: Scan these ports only. ("-p 1-65535" will scan
all TCP ports)
-sV: Attempt to determine what service and version is running on
the open ports found
-A: Attempt to determine remote host's Operating System and version of OS
-T[0-5]: Timing of scan, 0 is slower and stealthier, 5 is faster
and noisier
-v: verbose output
-vv: very verbose output
```

Figure 9.24 shows Nmap in action after scanning a machine for open ports. You can see how intuitive the output from Nmap is and how well laid out it is. No wonder Nmap has become the standard in port scanners!

Figure 9.24 Output After Running a Port Scan Using Nmap

Nmap's usual command-line options, shown in Figure 9.24, will suffice for most of the port scanning activities you're likely to need in your enterprise. But there may be times when you require a more aggressive or a more passive scan. My advice is to download a copy from www.insecure.org, print out the *man* page for it, and go through the options that Nmap provides while testing on a spare machine. This way you'll be able to see what each option does, and how they all affect the scan output.

If you'd prefer a port scanner that offers a GUI, and don't want to use Nmapfe, then Superscan is the port scanner for you. Superscan was created by the fellows over at Foundstone, and is free to download from their Web site at http://www.foundstone.com. This is a port scanner that performs basically the same duties as Nmap, but in a nice, easy to read, graphical format. Superscan has an advantage over Nmap in that it has a GUI for all scan options. You simply go through each available tab, and check off the options that you want Superscan to look for while it scans. Although the default options will pick up most items of interest, there are a couple of items that I change when using Superscan to perform a port scan. These closely resemble the options we'll use when running Nmap from the command line.

By default, Superscan doesn't scan all TCP and UDP ports (see www.skullbox.net/tcpudp.php for a succinct and mildly humorous explanation of the differences between TCP and UDP. When looking for unknown malware that has opened up a new port on the target machine, it's important to check all of these on the suspect machine, unless you were able to determine exactly what port was opened by analyzing a sample with the tools previously mentioned.

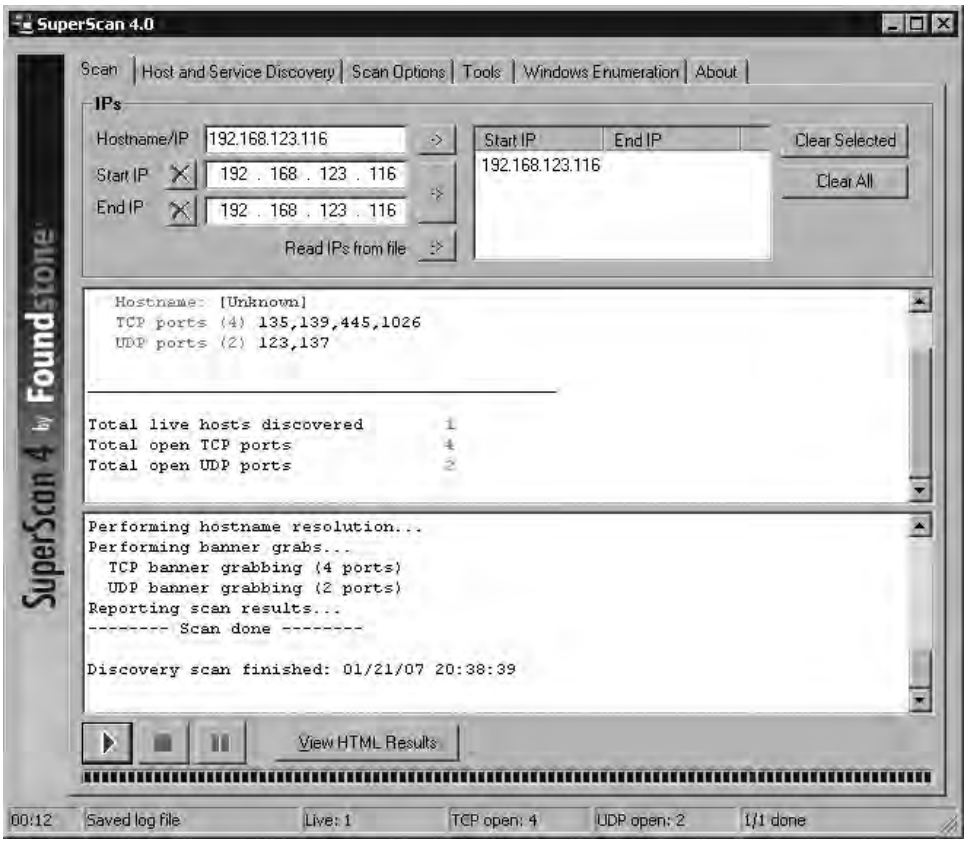
If you're scanning an entire subnet or subnets, you might also set all Host Discovery options to active, by placing a checkmark next to each one. By doing this, you'll stand a better chance of detecting that a remote machine is live and on the network, even if it's not returning ICMP pings.

The final option that you might change from the default can cause the scan to take a bit longer to complete if you're scanning many class C or larger network subnets. Nevertheless, consider turning on a full TCP connect scan. With this option turned on, Superscan will

perform a full three-way handshake with each port before it attempts to gather information about that port. You can certainly see why this would increase the scan time compared to a simple SYN TCP scan. To compensate for the time increase waiting for the full three-way handshake, it is possible to decrease the TCP port scan timeout from the default of 4000 ms, down to 1000 ms. The danger with decreasing the timeout is that on slower networks, or networks where there are many hops between the scanning machine and the subnets being scanned, you could exceed that timeout and Superscan would assume that the port is not active. In reality, the port could be active in this scenario and simply taking longer than 1000 ms for the replies to reach your machine.

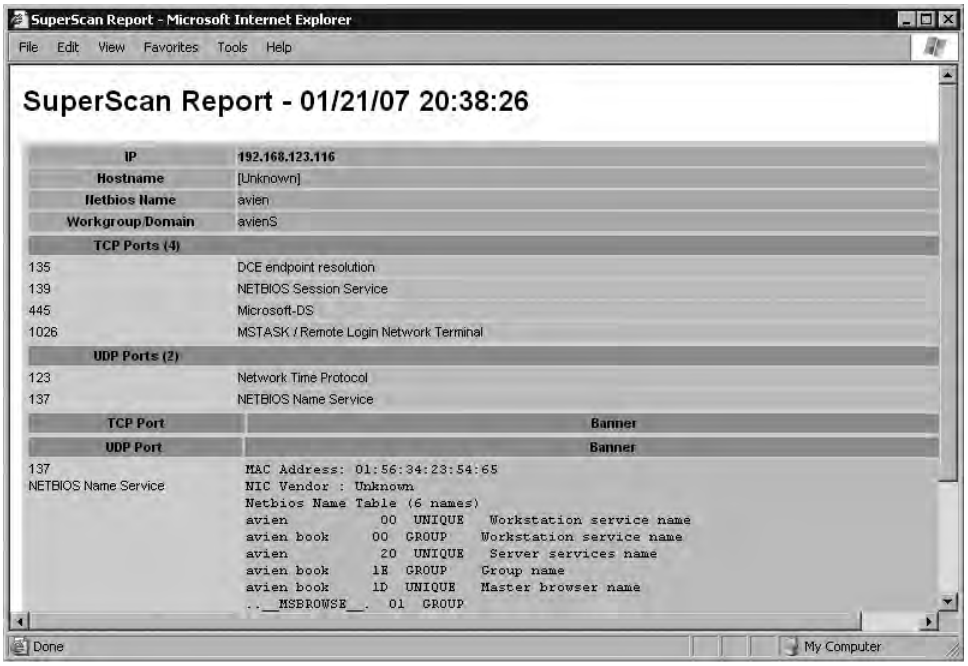
Figure 9.25 shows you a view of the main GUI page. Here there is only one item that has to be entered, the hostname or the IP address of the machine that you wish to scan. If you wish to scan a range or multiple ranges, this is where you'll enter that information as well. To start the scan, click on the little blue arrow button. During the scan you can watch scanning progress in the two large windows. Once the scan is complete, click on the large **View HTML Results** button and a report similar to the one shown in Figure 9.26 will open up in your default Internet browser.

Figure 9.25 Main GUI Interface Page of Superscan 4.0



In Figure 9.26, you can see how the report is laid out. The first item on the report is general information about the machine that was scanned, then follows a listing of TCP and UDP ports that Superscan found open. The section on the report is a listing of the individual ports that were found open, their common name, and the banner that is retrieved from the connection. The banner will usually contain identifying information about the application that is listening on that port, usually providing, as a minimum, the name and version information it has on that application.

Figure 9.26 HTML Report Generated After Running Superscan 4.0



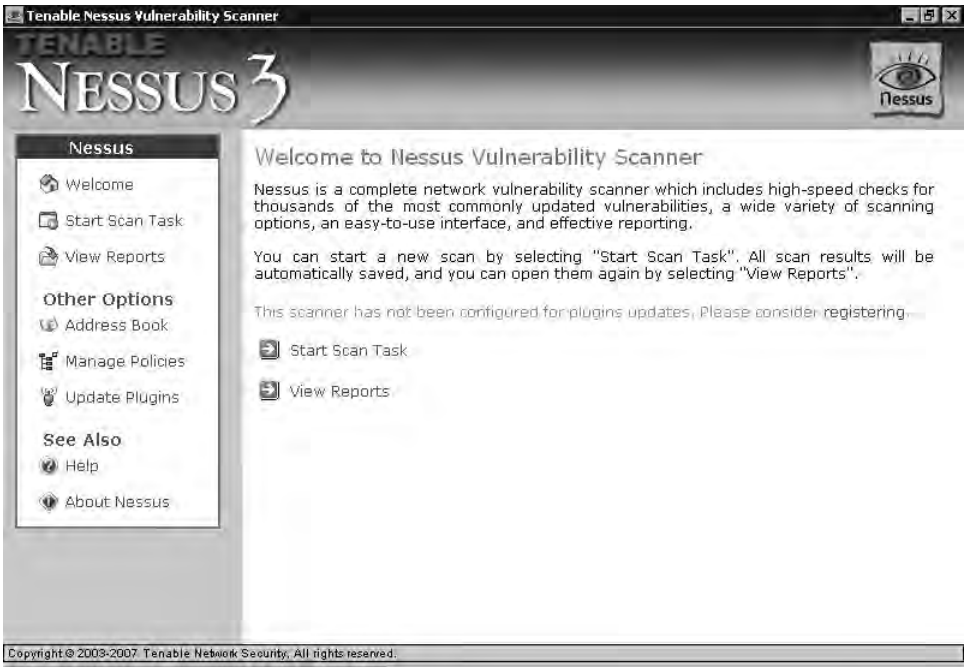
There will come a time that you'll want to examine a remote machine for more than just open ports. You'll want to know if the machine has any known vulnerabilities, perhaps due to a Windows or Linux patch that either wasn't applied, or was not applied correctly. This is where you'll want to use a full vulnerability assessment tool. The most popular vulnerability assessment tool is, without a doubt, Nessus, from Tenable Network Security. Renaud Deraison started Nessus as the "Nessus Project" back in 1998. Four years later, he teamed up with Ron Gula and Jack Huffard to form Tenable Network Security, which offers Nessus free for download on their Website www.nessus.org.

Nessus is updated on a daily basis with the latest vulnerability checks. However, in order to get these same-day updates, Tenable Network Security charges a nominal subscription. However, Nessus can still be used without having to pay a fee. Users that who choose not to

pay for daily updates can still use Nessus and update it as the paying customers do. However, your vulnerability database will be seven days old. In an enterprise setting, the fee for the subscription is well worth the cost, but Nessus is still quite usable without having the up-to-date subscription, if you don't mind the fact that your vulnerability database will be a week out of date.

Nessus is available for numerous different platforms, including Linux, FreeBSD, Solaris, Mac OS X, and Windows. It is extremely easy to use, and yet it has many advanced features that can be used to tailor scans according to the environment. Figure 9.27 shows the initial screen that you're presented with once you've installed Nessus. After clicking **Start Scan Task**, you enter the IP or machine name of the machine that you wish to scan. Then you're asked to choose the plug-in set to use. The default policies are very basic, yet very effective. Non-intrusive scans and intrusive scans make up the two basic policies. If you choose the non-intrusive scans, the machine that you are scanning should not be negatively affected in any way. With the non-intrusive scans, only passive checks are performed. If you choose the intrusive checks, you take the chance of performing a Denial of Service (DoS) on the machine that you're scanning, as you'll be attempting to brute-force passwords, possibly even attempting to overflow buffers. This could cause the machine being scanned to crash or re-boot, or trigger alerts from other security software, so care must be taken when performing intrusive scans.

Figure 9.27 The Initial GUI Screen Upon Starting Nessus



When the scan is complete, you'll be presented with an HTML report that contains the results of the scan. You may be surprised at what is found on your network! Figure 9.28 shows you a sample of one of the reports. The reports provided by Nessus are plain, basic, and chock-full of information about the machines that were scanned. The reports identify what ports are open, what vulnerabilities were found, where to go for more information about the vulnerability, and how to remedy the vulnerability found. The vulnerability remediation function is very useful. After all, what good is knowing there's a vulnerability if you don't know how to fix it?

Nessus also offers its own scripting language. This allows you to create your own vulnerability checks. Nessus Attack Scripting Language (NASL) is easy to learn to use for performing all the vulnerability checks you set up, and is the language used to create your own Nessus plug-ins. All of the existing NASL plug-ins are in plaintext. This means that you can open them up with Notepad or another text editor and see exactly how they work. This makes it easy to borrow code from existing NASL plug-ins for use in your own plug-ins.

Figure 9.28 Sample Report After Performing a Scan Using Nessus

Advanced Tools: An Overview of Windows Code Debuggers

Sooner or later you will want to know absolutely everything about an executable file. You may want to know, for instance:

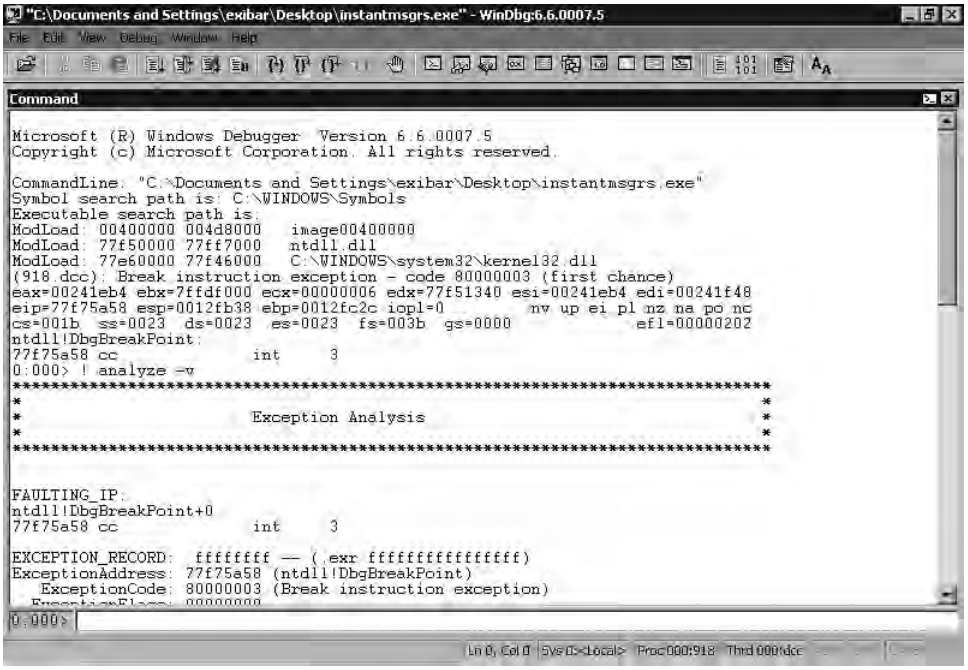
- The exact memory address that it is calling
- The exact region of memory that it is writing to
- What region it's reading from
- Which registers it's making use of

Debuggers will aid you in reverse-engineering a file for which you don't have the source code, by disassembling the file in question. (For more on the relationship between programming and debugging tools, see the later section on advanced forensic analysis.) This comes in handy when you're analyzing malware, as you almost never have access to the executable's original source code. The goal of this section is not to coach you in depth on how to use these debuggers, but simply to show you that they are out there and available for you to use. Debuggers are very powerful tools that take a long time to learn to use to their fullest extent.

There are three popular debuggers for the Windows platforms that we'll be discussing: two of them are free and one is a commercial product. The first debugger in this section is Windows Debugger (WinDbg), available free of charge from Microsoft. It's part of the Debugging Tools for Windows (www.microsoft.com/whdc/devtools/debugging/default.mspx). WinDbg, although basic in its functionality, will allow you to fully debug and analyze Windows applications and other user-mode items. Along with services, drivers, and other kernel mode items, it can be used to analyze crash dumps created by a Blue Screen of Death (BSoD). This is probably its most popular use. The first thing to keep in mind when using WinDbg is that you must download the proper symbol set for the operating system that you're working on. Symbol sets are available free from Microsoft at the link above. They are quite large; the average size is about 150Mb.

Figure 9.29 shows WinDbg starting to analyze a WootBot Variant with the file name "*instantmsgrs.exe*."

Figure 9.29 WinDbg Analysis of a WootBot Variant



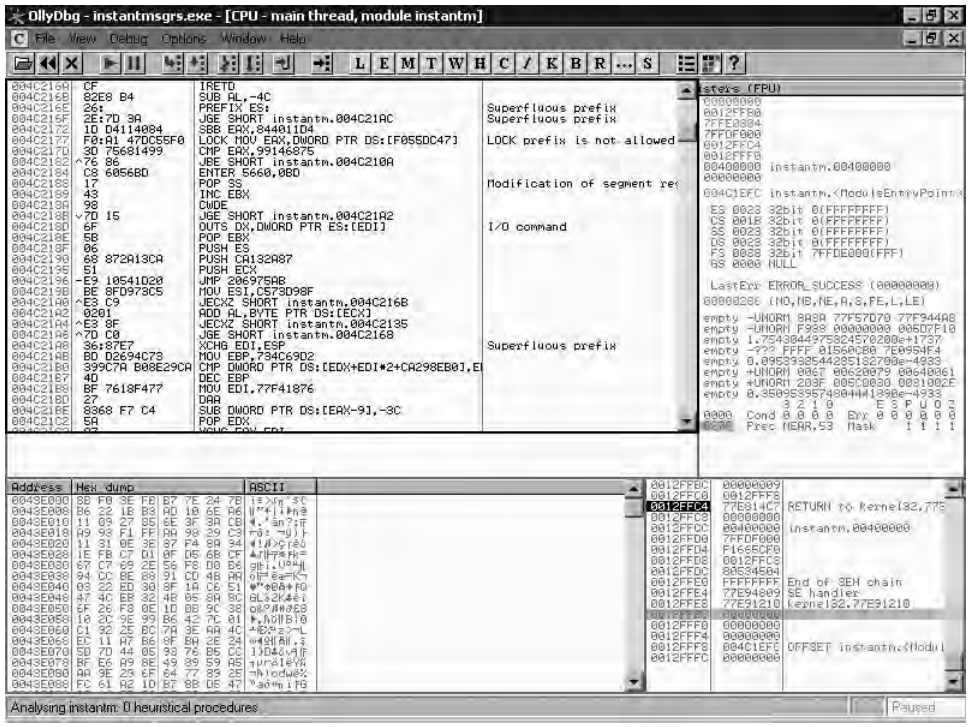
A very useful command for use with WinDbg is "*! Analyze –v*." If you load the proper symbol set and a mini-dump into WinDbg, you can retrieve information such as the exact instruction that was being executed at the time of the crash, the particular threads in use, loaded modules, and so on. If you were to load a full crash dump file, you have access to even more, including the contents of the process heap itself at the time of the crash.

NOTE

The same PE format is used by both file types. The only difference between the two is the use of a single distinguishing byte .exe and .dll, so that the operating system knows how to execute them. Note also that a number of other file types are also PE format files, for instance Control Panels (*.cpl*), and can be executed by the OS as if they were *.exes* or *.dlls*. Hence the frequent generic blocking of *.scrs*, *.ocxs* and so on by e-mail filters. For a fuller explanation of the PE format, try Matt Pietrek's Inside Windows article "An In-Depth Look into the Win32 Portable Executable File Format," at http://msdn.microsoft.com/msdnmag/issues/02/02/PE/

The next Debugger has gained in popularity tremendously over the past year since SoftICE was discontinued in April 2006. Oleh Yuschuk (Olly) released OllyDbg in November 2000, for free download. OllyDbg is a very powerful, full featured 32-bit assembler level-debugger. Figure 9.30 shows a WootBot variant being disassembled within OllyDbg. This utility basically picks up where WinDbg leaves off. Where WinDbg really shines at analyzing crash dumps and makes diagnosing a BSoD much easier, OllyDbg excels at binary code analysis, tracing memory registers and recognizing procedures, tables, strings, and more. OllyDbg is perfect for debugging or reverse engineering Windows malware code, as long as it's in the Windows Portable Executable (PE) format used in current generations of Windows executables (*.exe*) and Dynamic Link Libraries. (*.dll*).

Figure 9.30 OllyDbg Analyzing *instantmsgrs.exe*, a WootBot Variant



Let's say that you want to analyze a piece of malware that isn't an .exe or a .dll, or a Crash Dump. You can't use WinDbg, nor can you use OllyDbg, as neither of those debuggers can perform analysis on other types of files. This is where you need the "cream of the crop" in debuggers. To be precise, you need Interactive Disassembler Pro (IDA Pro), available from DataRescue. IDA Pro should be your first choice of debuggers for an enterprise environment. It isn't really expensive, and is well worth the nominal outlay for the features it offers.

TIP

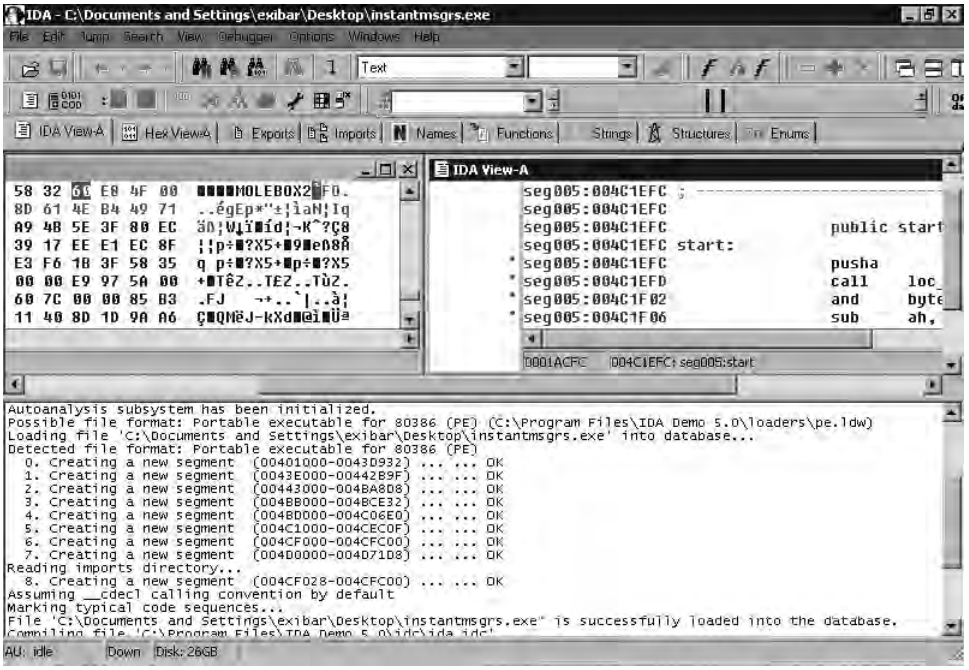
DataRescue offers a demo version from their Web site at www.datarescue.com/idabase/index.htm. This version can only work with a limited range of file and processor types, is time limited, runs only as a Windows GUI application, and so on.

IDA Pro is much more than a simple debugger. It is a programmable, interactive disassembler and debugger. With IDA Pro you can reverse-engineer just about any type of executable or application file in existence. IDA Pro can handle files from console machines such as Xbox, Playstation, Nintendo, to Macintosh computer systems, to PDA platforms, Windows, UNIX, and a whole lot more. Figure 9.31 shows the initial load screen wizard when you first start IDA Pro. Notice all the file types and tabs that will help you select the proper analysis for the file type that you wish to disassemble.

Figure 9.31 IDA Pro's Disassembly Database Chooser Loads Upon Start

In Figure 9.32, IDA Pro has loaded and is disassembling a WootBot variant with file name *instantmsgrs.exe*. Part of what we can see from Figure 9.33 is that *instanmsgrs.exe* was packed using an executable packer called Molebox. You can also plainly see the memory calls that it's making, and the Windows DLLs that are being called. This type of information can be invaluable when it comes to fighting off a virus or malware outbreak, especially if you need to make a custom cleaner in order to repair your systems.

Figure 9.32 IDA Pro Disassembles *instantmsgrs.exe*, a WootBot Variant



Advanced Analysis and Forensics

The end of 2006 saw an explosion in spam e-mails. Security analysts agree that the vast majority of spam sent today is broadcast by huge zombie/bot armies. These are PCs (usually) that are infected with various forms of malware, allowing them to be accessed covertly and used remotely for spam distribution and other unwanted, unsuspected activities. The AV industry (and related antimalware companies) is barely coping with the amount of new malware variants being released daily, as both reactive and proactive defensive measures fail to stop the intrusions.

When an incident happens in an organization, the security administrator often has to assess the impact of the incident quickly and decide what should be done with the infected machine(s). While the correct answer, unfortunately, is often to reinstall the machine from scratch, this does not prevent future infections, unless the infection vector has been determined.

It usually takes hours or longer for AV companies to release definitions for new malware and days, if not weeks, to post technical documents about most common malware. And it is often the case that particular malware that has hit the organization is not widespread, in which case the technical analysis from the AV vendor may not be available at all.

In cases like this, the security administrator often has to analyze the malware himself and gather as much information as possible in order to ensure that his organization's core business functions remain stable and uninterrupted.

Tools & Traps

Lies, Damned Lies, and (Spam in 2006) Statistics

Spam statistics are notoriously variable according to source. In this instance, traffic on Anti Virus Information Exchange Network (AVIEN) and other lists clearly suggested a spike in spam volumes over the last part of 2006, and spam vendor reports such as those at www.marshal.com/trace/spam_statistics.asp and www.messagelabs.com/Threat_Watch/Threat_Statistics supported that supposition.

However, the media seemed to be more interested in a report by SoftScan that there was actually a 30 percent drop in spam volumes at the beginning of 2007 (www.softscan.co.uk/composite-491.html). Several theories were put forward to account for this, including the disruption of network services by an Asian earthquake and the possible unavailability of a major robot network (botnet). The media chose to focus on the alternative suggestion of the replacement of large quantities of compromised PCs by new machines received as Christmas presents.

There's nothing wrong with some healthy speculation; however, it may be misleading to build too many theories on data received from a single source.

Advanced Malware Analysis

Analysis of any code can be a very demanding and complex process. It becomes even more difficult when dealing with unknown, potentially malicious, and often obfuscated code. Malware can be analyzed statically when we review the code, in order to determine what it will do. Alternatively, it can be reviewed dynamically, when we observe the behavior of the malware when it is being executed. Depending on exactly how you plan to deal with what you find and how much time you are prepared to invest in analysis, you might want to use either method, a combination of both approaches, or elements of both approaches.

Static (Code) Analysis

The majority of systems and application programs today are developed in high-level programming languages such as C or C++. These programs are then compiled into machine code translating into a stand-alone executable program. An aspiring virus writer can choose to use any high-level programming language to create a virus. However, it has not been unusual to see code, even malicious code, written completely in assembly language. This has become less common in the last couple of years, since malicious writers are also looking at

"return of investment (ROI)" when writing malware. It is much quicker and easier to code in high-level programming languages.

Static or code analysis is (usually) based on manual review of unknown code. When analyzing malicious programs, you will almost always encounter binary executables. This means that code analysis must be performed on disassembled instructions, a process that is not trivial and that requires a lot of knowledge about machine code, the lowest level of binary code. There are various utilities that can make this process easier.

Tools & Traps

The 10-Second Guide to Programming and Disassembly

Dedicated, high-level programming languages like C, C++, Delphi, and so on are often roughly divided into compiled languages and interpreted languages. In fact, this is an artificial distinction. There's rarely any absolute reason why a given language needs to be one or the other, though the specific characteristics of a given language may make it difficult to include all of its features in both forms. Many languages are, in fact, implemented as interpreters and as compilers, sometimes packaged together.

The term compiler is popularly applied in PC programming to a program that translates human read-writable source code into object code, usually with the intention of creating a standalone executable program. (Real computer scientists are probably already sniffing contemptuously at this gross oversimplification of an esoteric concept.)

An interpreter runs through a program translating one statement at a time into machine code, rather than translating the entire program into a single executable. The execution of the program could be said to take place within the application environment rather than in system space. Tools like Visual Basic for Applications (VBA) provide advanced programming functionality within an application like Microsoft Word that is not primarily seen as a programming tool in itself.

Machine code is the lowest level of code, the level at which the computer is able to read it directly. Assembly language (often referred to informally as "assembler") is the next level up from machine code. It allows the programmer to use mnemonics like "MOV BL, 34H" or "JMP MYLABEL" instead of pure human-hostile machine code ("001010011"), while retaining the advantages of working "close to the metal."

A disassembler translates machine code to assembly language, which is a little easier to read for humans, and is therefore useful for reverse engineering. A decompiler is somewhat similar in intent, but translates to something closer to a high-level language rather than assembly language.

Reverse engineering is the process of ascertaining the workings and technological principles of a system (in this case, a possibly malicious program) by examination and analysis.

Packers and Memory Dumping

The first obstacles that malicious authors put in our path are runtime packers. The idea of packing (compressing) an executable is not new. Back in the old days of Commodore 64, programs were packed with simple packers in order to decrease their size, so the transfer of files would take less time. This is still one of the reasons malware is packed today: malicious authors want their programs to be as small as possible to reduce the loading time.

The other and most important goal of a packer is to make analysis or reverse engineering of the code more difficult. All packed malware consists of the unpacker code and packed data, which typically look like random data. The benefit of packed malware, from a malicious author's point of view, is that it cannot be analyzed until it is properly unpacked. This presents a problem for AV software as well, and is the main protection that malicious authors employ today. You might ask why AV vendors do not just detect the unpacker code. The answer is simple: there are legitimate purposes for runtime compression, and it is often the case that malicious authors use legitimate packers on purpose, so as to hide their code. (Actually, there has been ongoing discussion for some years as to whether it's acceptable to filter all packed code on the grounds that it's probably malicious, but that is still sometimes viewed as controversial.)

Unpacking can therefore be the most difficult part of a static analysis. Malicious authors usually go through a very lengthy and complex process to obfuscate the unpacker and make it as difficult to analyze as possible. After all, if it is difficult for a human analyst to unpack the malware, it will be even more difficult for an automated process running inside an antimalware program.

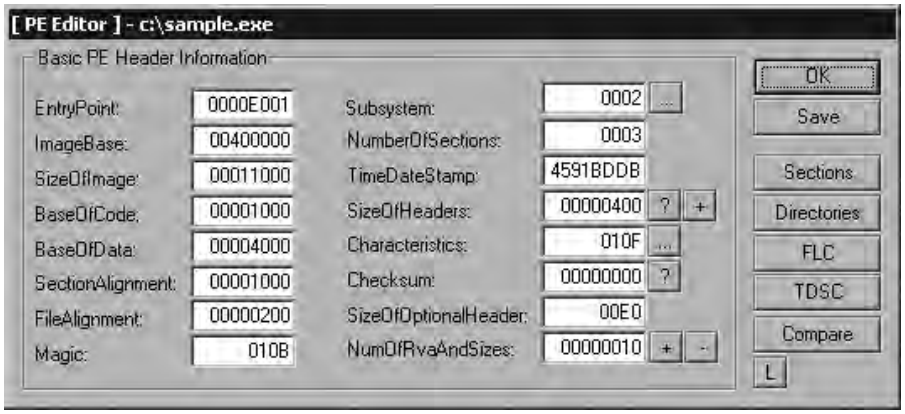
Before we start unpacking malware, you need to be aware of some basics about executable file format on Windows operating systems. All executables and DLLs, in order to be valid and interpreted correctly by the operating environment as executable code, have to be in a special format called Packed Executable (PE). This format defines information that the operating system loader needs in order to start the program.

NOTE

You can find detailed information about the PE format at www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx. The document "Visual Studio, Microsoft Portable Executable and Common Object File Format Specification," describes the structure of executable (image, PE) files and object files (Common Object File Format) under Windows®.

It is easy to identify the PE header. Every PE file has to start with the two bytes "MZ" (0x4D 0x5A), in order to be identified by the operating system correctly. PE headers contain crucial information about the file that includes the Original Entry Point (OEP), ImageBase, BaseOfCode, BaseOfData, NumberOfSections, and other data. When the loader loads data from the file into the memory, it uses the ImageBase address as the base, and all other addresses from the PE header are taken as an offset from ImageBase. In other words, the real entry address (the address that the program starts at) will be calculated as the sum of OEP and ImageBase. As previously mentioned, when an executable is packed, the OEP actually points to the unpacker code that will prepare (unpack) the file and then jump to the unpacked code. In the case of real malware, this will be the beginning of the actual malicious code. The unpacker will, in fact, create a new OEP address. This is an important part of the unpacking process. The other options in the PE header include BaseOfCode, which specifies the address in memory on which program code is stored, BaseOfData (which is the memory section holding program's data), and NumberOfSections, which defines the total number of sections in the PE file.

Figure 9.33 Section Table of a Malicious Program



The section information here was listed with LordPE (available at http://scifi.pages.at/yoda9k/LordPE/LPE-DLX.ZIP), a program that allows you to review and edit PE header information.

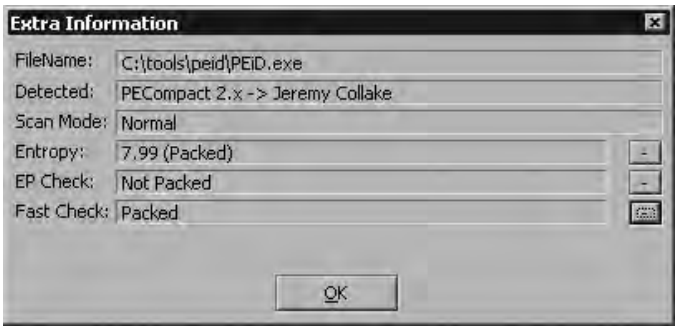
The first step, when analyzing an unknown malicious program, is to determine if it is indeed packed and, if possible, to determine what packer has been used. At the time of writing this book there are around 600 packers and the most popular ones, such as UPX or ASPack are frequently used. Knowing what packer has been used will allow you to decide how to unpack the program, and whether it is worth going through the trouble

of manually unpacking it. A utility that you should consider as an essential component of your reverse-engineering arsenal is PEiD (available at http://peid.has.it/. PEiD is a very powerful utility that has a database of fingerprints for identifying most popular packers. The database is also expanded regularly, so when a new packer is encountered, the PEiD community quickly generates a unique fingerprint that identifies it. In cases when PEiD does not identify the packer, you can still use it to see if the malware was packed or not. As previously mentioned, a packed program has some characteristics that can be easily identified, and the most important one is that no program code or data can be seen when it is packed or encrypted. PEiD can calculate program entropy and, depending on the result of the calculation, conclude whether or not a program is packed.

NOTE

Program entropy in the context of data compression is a measure of the amount of non-redundant, non-compressible data a file contains. Files that contain large quantities of redundant material (e.g. repetitive byte sequences) can be compressed very small, even with a fairly unsophisticated compression algorithm.

Figure 9.34 PEiD Calculating Entropy of a Packed Program



The process of unpacking can be very complex, depending on the packer,. Besides the code obfuscation that is typical of most packers, the authors of malicious software add various anti-debugging techniques. They will, for instance, attempt to detect when the program is being run inside a virtual machine, which usually means that someone is attempting to analyze/reverse-engineer the code. Again, depending on the time and effort you want to invest into the analysis of a malicious program, and whether an unpacker utility is readily available, you might want to perform dynamic analysis (see later section) or utilize dumps of process memory, which will in most cases circumvent packers.

Quick Assessment

In some cases, it is not necessary to perform a full blown analysis and (if appropriate) manual unpacking when you want to assess quickly what an unknown executable is doing. However, you do need to unpack program code and data, so as to see what it is doing. One easy way to retrieve unpacked program code is to dump its memory when it is running. The assumption and hope here is that the program will not just do something and exit immediately, as in this case it is difficult to "capture" it. The benefit of this approach is that, as you execute the malware, it will automatically unpack itself and you will be able to dump the unpacked code and data to a file for further analysis.

The utility that is most commonly used for this purpose is LordPE, but any competent process dump utility will do the same job. Of course, it must be stressed again that you should do this only on an isolated machine, not one which is connected to a network. Alternatively, you may choose to do it in an appropriate virtual environment (see the section on dynamic analysis for virtual machine requirements). The main reason for this is that you do not know what the malware you are analyzing does, and you do not, therefore, want to risk getting other machines in your network infected by an unanalyzed threat. The procedure is as follows: you should start LordPE first, then execute the unknown file, find its process in memory and dump it with LordPE to disk. This will create a memory image of the process on your disk. As the file has to unpack itself in order to run the real code, the image will contain raw program code and data. Unless, that is, further obfuscation has been used, as happens with a well-known packer called Armadillo. This program packs data in multiple sections and unpacks only what is needed for the current execution. All other parts of the program are still packed and/or encrypted. In cases like this, a memory dump will not contain the whole program, but can still be useful.

A dumped process image cannot be directly started, because it will not contain a proper OEP address. (Remember, the first OEP address points to the unpacker code. Once the program is unpacked, OEP needs to be changed to point to the real start address.)

Now that you have an image of the unpacked program code and data (or alternatively, once you unpacked it manually or with an unpacker utility, such as the one available for UPX), you can do a quick assessment of what the program actually does. Most program files under Windows operating systems make great use of DLLs. A lot of DLLs come preinstalled with Windows, and offer basic operating system and network functionality. In order to use functions that are available in DLLs (functions are *exported* by DLLs), every program has to *import* them. All DLLs that need to be imported by a program are listed in the Import Table, which is a part of the PE header. One problem springs to mind here: if you just dumped the process' memory with a utility such as LordPE, you will not have a proper Import Table, so you cannot just refer to this part of the header.

A very brief and quick assessment of an unknown file can be carried out with a utility called "*strings*" (www.microsoft.com/technet/sysinternals/Miscellaneous/Strings.mspx).

The *strings* utility lists all textual (ASCII) strings that were found in a file and are, by default, longer than three characters.

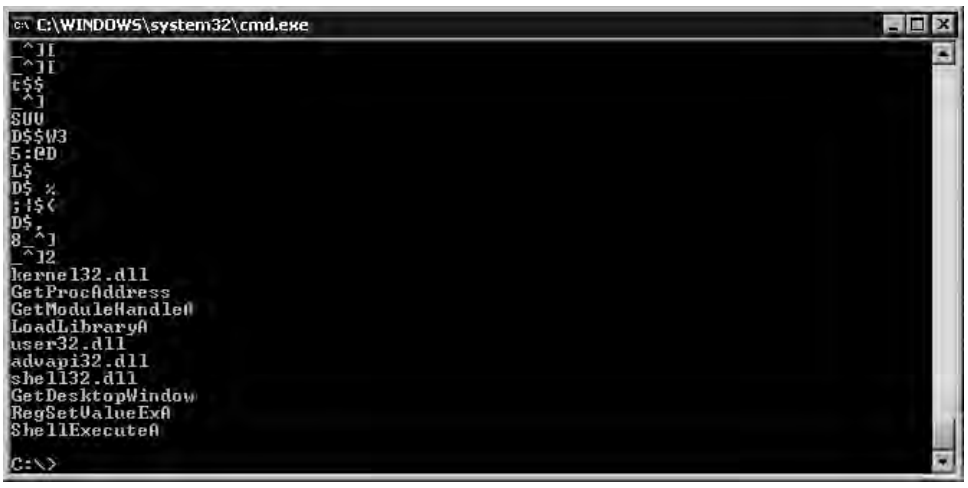
Obviously, if the program has been unpacked, running *strings* on it will result in a wealth of information. It is not unusual to find the DNS address of a command and control (C&C) IRC server, or the URL that the malware visits, shown as clear text in the program data. The *strings* utility can also list DLLs that the program imports, as well as the functions. In some cases, malware authors try to obfuscate text strings such as URLs so that they can't be viewed with simple utilities (such as *strings*): however, all of these text strings can still be spotted, with a little bit of practice and experience. Figure 9.35 shows the *strings* utility run on a malicious, packed file; you can see imported DLLs in clear text. String obfuscation, if present, is typically done through simple Exclusive Or (XOR) operations, or just by swapping bytes.

NOTE

The Windows *strings* utility has been ported from UNIX. However, its functionality and options are a little different. The UNIX utility, by default, looks for strings of four or more printable characters ending with a newline or null, and uses the options *–a* (search entire file), *-o* (display offset position before string), and *–n l* (find strings of minimum length *l*)

Mark Russinovich's port to Windows includes the options:
-s (recurse subdirectories)
-a (search for ASCII only)
-u (search for Unicode only)

Figure 9.35 Running the *strings* Utility on a Packed File



At this stage, you will have enough high-level information about what the unknown file does. If you need more information than this, you will have to start disassembling it, or running it through a debugger. This might be necessary when certain parts of the malware are obfuscated. If the obfuscation is obvious, (e.g., XOR encoding), you can try cracking it manually, or using one of the brute force tools.

Disassembling Malware

Disassembly tools have come a long way since their beginning, and are a great help when reverse engineering any unknown code. In most cases, just browsing through the disassembled code without any further action will be enough to see what it does. Malicious authors try to prevent disassembling by creating obfuscated or self-modifying code so, in some cases, disassembly will not be enough. In such a case you will have to go one step further and debug the application.

Probably the best disassembler tool available today for Windows operating systems is DataRescue's IDA Pro. While the latest version of this tool has an integrated debugger, at the moment we will concentrate on its use as a disassembler. IDA Pro has some very powerful features for analyzing the code, so you can get a lot of information just by reading IDA Pro's comments. To make the best use of a disassembler, you will need to be familiar with x86 machine instruction codes.

NOTE

X86 is a generic term for the microprocessor family derived from Intel's 8086 (and 8088), including separate math co-processors for early models, compatible processors by other makers such as AMD, and more recent processors such as the Pentium sub-family, which no longer have the X86 model numbering. The details of the instruction set are beyond the scope of this chapter, but these two URLs will give you a flavor.

http://en.wikipedia.org/wiki/X86_instruction_listings
http://en.wikipedia.org/wiki/X86_assembly_language

One amazingly useful function of IDA Pro is its ability to generate program flow charts. IDA Pro can analyze the code and create a nice flow chart from it (essentially a representation of the functions and code blocks called during execution). This is particularly useful when analyzing packers, as in some cases it can clearly show the process flow of an unpacker directing you towards the new OEP. Besides this, IDA Pro, like almost all disassemblers and debuggers, will show and parse imported DLLs and functions, so you can easily jump to

code that interests you. For example, when analyzing a downloader, you will most likely be interested in any Internet-related functions. Imported functions are easy to locate, so all you have to do is check input parameters for the function call. This will enable you to identify second-stage malware downloaded by the downloader module you are analyzing.

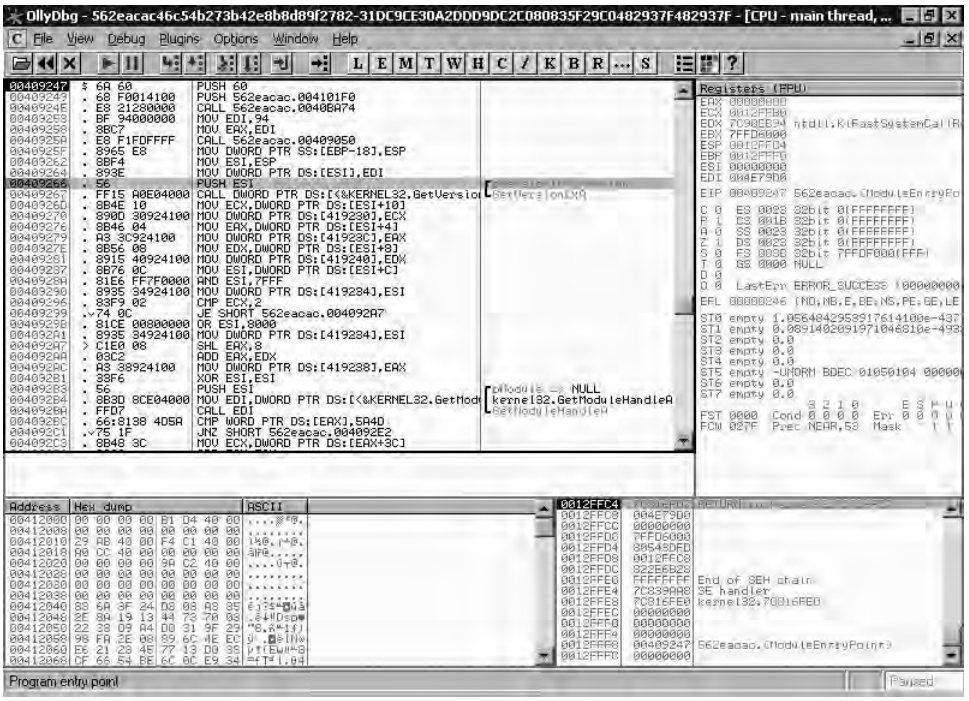
Figure 9.36 IDA Pro Disassembling the YaY Trojan

Debugging Malware

The final step in any analysis of unknown code is debugging. This step requires the most detailed knowledge of machine code instructions, as well as of internal operating system functions. This is often the most time-consuming part of the analysis. Generally, debugging is needed only when you want to analyze a piece of malware thoroughly, or when you encounter obfuscated code that has to be debugged (observed during execution) in order to see what it is really doing. Even in cases like this, you might first want to perform a dynamic (behavioral) analysis of malware, which might provide you with enough information to avoid the need to spend hours on manual analysis using a debugger.

If you still wish to proceed along this route, you should get familiar with OllyDbg (www.ollydbg.de), the most popular freeware debugger today. OllyDbg, like IDA Pro, also has pretty powerful analysis features that will be of great help when analyzing unknown malware. OllyDbg can enumerate all imported DLLs, and can also recognize API calls.

Figure 9.37 OllyDbg Analyzing a Malicious Program



To get back to the previous example, once you have located Internet-related functions, analysis with OllyDbg can be extremely easy. All you need to do is set a breakpoint on all function calls. This will cause the program execution to stop when an instruction with a breakpoint is reached. If you set breakpoints on API calls, you can examine their parameters just before they are called. All parameters will be located on the stack, because that is the way parameters are passed to functions.

Tools & Traps

API Emulation

Some malware goes so far as to avoid importing any functions from available DLLs. Instead, it emulates all of the APIs it needs. This means that you cannot list the functions, so you cannot easily set breakpoints on them, as all API calls will just be a part of the malware code.

Continued

In some cases, malware authors even use this to trap reverse engineers: they may import functions that are never used (having used the emulated ones instead). If you set a breakpoint on such fake function calls and allow the program to execute, you might find that your machine has been infected and the breakpoint was never reached.

The most helpful feature for analyzing unknown code that any debugger offers, apart from setting breakpoints, is step-by-step execution of code. With this asset on hand, the only limitation you're likely to meet on exploring the unknown world of malware will be the amount of time that you're prepared to invest.

It is quite common for malware authors to go one step further when they produce obfuscated code. They very often try to detect the presence of a debugger process in order to prevent you from reverse engineering their programs. In cases like this, it might be much faster and easier just to observe the behavior of the unknown program. That might give you enough information on what it does, and you can leave detailed analysis to AV vendors.

In some cases, you will have to resort to using a kernel debugger. The previous tool of choice for most researchers has been SoftIce, but as it is not being developed or supported any more, your best bet in this arena will be Microsoft's debugger (www.microsoft.com/whdc/devtools/debugging/installx86.mspx). This is a full-blown kernel debugger, but it has fewer features and its interface is more difficult to use, compared to the programs previously described.

Dynamic (Behavior) Analysis

While static (code) analysis can demand a great deal of time and knowledge, dynamic or behavior analysis can, in some cases, be much faster and yield practically the same information. Depending on the malware that you are analyzing, behavior analysis can sometimes be as simple as determining by eye what is going on (for example, most adware programs simply open popup windows with advertisements). With the help of some appropriate tools, behavior analysis will take you a long way down the road to accurate analysis. When resorting to it, you should first prepare the environment for running any unknown and potentially malicious code.

Isolated Environments

By now it should be clear that behavioral analysis of any potentially malicious program must be carried out in a strictly isolated environment. The main reasons for this is to prevent any uncontrolled replication of malware, and the avoidance of potential attacks towards your

(or someone else's) infrastructure that might be the result of executing malware on a live system. For example, various bots are often channels for DoS attacks. Historically, malware researchers used physical machines that were either disconnected from the network or connected to an isolated segment. Using a physical machine makes reversion to the previous, known state of the operating system much more difficult. You need to either use an imaging utility (such as Symantec Ghost – see www.symantec.com/enterprise/products/overview.jsp?pcid=1025&pvid=865_1), or completely reinstall the operating system.

Virtual machines have recently gained popularity in this area. The ability to run multiple operating systems on virtual machines, hosted on one physical machine, is extremely attractive in a tool for behavior analysis. In fact, most AV labs and researchers today use virtual machines for quick behavior analysis of unknown and suspicious code. In order to set up a virtual malware research lab, you can use practically any virtual machine software, such as Virtual PC (now published by Microsoft: see www.microsoft.com/windows/virtualpc/default.mspx) or VMWare's Server (www.vmware.com/). There are even a few open source options for braver souls. This offers the opportunity to emulate a real world environment with multiple virtual machines.

In the simplest case, your virtual environment will consist of one virtual machine that will be used as a sacrificial goat. An obvious advantage of virtual machines is easy reinstallation. As the whole hard disk is kept as a file on the host operating system, all you need to do is make a copy of this file and use it to return the system back to a known, safe state. If you want to go even further, most of today's virtual machine programs allow you to set up non-persistent virtual hard disks. In other words, as soon as you turn off the virtual machine, all changes to the hard disk are gone. This type of setup makes virtual machines perfect for behavior analysis.

When emulating a real environment, a typical configuration consists of a sacrificial virtual machine connected to an internal (virtual) network and a router/server virtual machine (typically running some sort of Linux operating system). The virtual router/server machine emulates all real-world services, such as DNS, HTTP, Simple Mail Transfer Protocol (SMTP), and so on. When the sacrificial virtual machine is infected with the malware you are analyzing, this setup allows you to monitor all (internal) network interaction from the virtual router/server, so that you can see exactly what is happening on the virtual victim machine.

Finally, you need to be aware of certain dangers and limitations that come with virtual machines. A bug in the implementation of your virtual machine software might allow malware to escape and infect the host machine. While this is an extreme case, it is by no means impossible. Indeed, a known vulnerability found in VMWare Workstation and GSX products in December 2005, enabled a specially crafted program to execute arbitrary code on the host operating system.

Tools & Traps

Malware That Detects Virtual Environments

A major drawback of this approach is that some malware may detect, or attempt to detect, whether it is running in a virtual machine or not. This has become a pretty common case today, as malware authors are aware of benefits that virtual machines offer for malware analysis. Malware that detects virtual machines will typically just exit upon detection, or perform some benign or innocuous action, making useful behavior analysis in a virtual environment almost impossible. In such a case, you might need to revert to physical machines or use static (code) analysis as a replacement or supplement to the virtual environment.

Themida is an example of a well-known packer that detects virtual machines and stops the executable from running.

Peter Ferrie's paper "Attacks on Virtual Machine Emulators" considers known attacks on VMware and Virtual PC, as well as other virtual machines such as Hydra and Xen (http://pferrie.tripod.com/papers/attacks.pdf).

Behavior Monitoring

Once a suspicious has been executed, you will want to gather as much information about its activities as possible. Whether you are performing this on a virtual machine, or on a physical scapegoat system, the procedures and tools you use for gathering information will be the same.

The goals of behavior monitoring are to gather enough information about changes to the environment that have been introduced by the unknown program you are analyzing. Typical changes that you will want to monitor on Windows operating systems include file access and modification, registry modification, and any network connections established by a malicious program.

Most of the tools for behavior monitoring must be installed in the "scapegoat" or sacrificial virtual machine. You will need the following tools:

- Filemon from Microsoft Sysinternals (www.microsoft.com/technet/sysinternals/utilities/filemon.mspx). This utility will let you record all file activities in real time. It can generate a lot of data, but it will give you a good overview of how the unknown program you are analyzing is manipulating the file system.

- Regmon from Microsoft Sysinternals (www.microsoft.com/technet/sysinternals/utilities/Regmon.mspx). This is a utility similar to Filemon, but it monitors all registry-related activities.
- AutoRuns from Microsoft Sysinternals (www.microsoft.com/technet/sysinternals/utilities/Autoruns.mspx) or HiJack This by Merijn (www.spywareinfo.com/~merijn/programs.php). You should use one of these utilities after you have executed the malware, so as to see if it has modified any of the startup parameters. (Most malware adds itself to the Run registry key in order to be executed after system reboots.)
- Wireshark (www.wireshark.org). Wireshark is a network sniffer. You can run this utility on:
  - The sacrificial virtual machine
  - The machine acting as the router/server
  - The host machine, if you allow the virtual machine to access external sites

These four utilities will allow you to gather a wealth of information about the malware you are analyzing. Just by going through logs created by Filemon and Regmon, you can easily spot typical infection cycles. The AutoRuns utility will show you all programs that are set to start on system boot. Going through this list will require a bit of experience, but just by comparing the known state of the system before with the state of the system after you executed the file, you can easily see what has changed. A word of caution: depending on what the malware actually does, it can subvert all these programs. Some malware authors even go so far as to enumerate running processes and kill any known debugging/analyzing programs.

Finally, Wireshark will offer you details about network traffic. If you are running it on the router/server virtual machine, it cannot be subverted by any rootkits on the scapegoat machine. You will always be able to see all network traffic generated by the scapegoat machine. However, if the malware is encrypting its network traffic, the only value you will get from Wireshark logs will be ascertaining the end points to which malware tries to talk.

TIP

Wireshark and its uses in security are considered at length in "Wireshark & Ethereal Network Protocol Analyzer Toolkit" by Angela Orebaugh *et al* (Syngress).

Forensic Analysis

A commonly asked question related to new malware (apart from what it does once it is executed on a machine), is the nature of its infection vector. This question is even more important for administrators of big organizational networks. In these environments, the infected machines are normally just reinstalled (re-imaged), but without knowing what the initial vector was, you cannot be sure that the infection will not happen again.

Analyzing the malware can sometimes show the infection vector (e.g., a network worm that exploits a vulnerability in the target operating system), but this is not necessarily the case. In cases when malware spreads in multiple stages, it might be difficult to determine the infection vector, as can happen when a downloader deletes itself after downloading malware that initiates subsequent stages in the process of infecting and compromising the target system.

It is often also important to determine whether the infection vector included the exploitation of a previously unknown vulnerability (e.g., a vulnerability in the end-user's browser or e-mail client), or if the infection just happened because the user knowingly downloaded and executed a file.

Forensic analysis of malware is intended to determine the infection vector. In this case, we are not necessarily interested in collecting information that could lead to potential legal actions (although in some cases, especially in big organizations, that will be the case). Rather, our aim here is to collect as much information about the spreading mechanism of the malware.

Tools & Traps

The Changing Face of Forensics

The term "forensics" is traditionally applied to the application of scientific methodology for judicial review. However, recent definitions of digital forensics talk about "digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations" (Digital Forensics Research Workshop. "A Road Map for Digital Forensics Research" 2001. www.dfrws.org/2001/dfrws-rm-final.pdf)

Forensic aims may include identification, preservation, analysis, and presentation of evidence. Digital investigations that are or may be presented in a court of law, must meet the applicable standards of admissible evidence. Admissibility is a concept that varies according to jurisdiction, but is founded on relevancy and reliability.

Collecting Volatile Data

Any program that is executed on a machine, leaves a wealth of information about itself. When investigating new malware, this information can be crucial in determining the initial infection vector and it can also reduce the amount of analysis necessary. This is why the process of volatile data collection is important and different from cases when a computer is seized with the ultimate aim of pursuing some form of legal action. In the latter case, the emphasis is on the preservation of the integrity of evidence, rather than on the pragmatic accumulation of information with the intention of reducing the impact of an incident on your business processes.

When collecting volatile data, our primary goal is to enumerate all current activities on the system. In this phase, you should also investigate machine memory, because all process information is lost after it is powered down to collect the non-volatile data.

Tools & Traps

Forensics and the Law

The following are some of the major issues when it comes to forensic processes for presenting evidence in court:

- Evidential integrity is paramount. Risks from extraneous malicious code, mechanical damage, and accidental modification need to be (as far as is practicable) eliminated.
- Standard practice is to work with a disk image rather than original data, in order to avoid cross-contamination and challenges to its legal validity.
- The chain of custody should be thoroughly documented, to show how, when, where, and by whom it was obtained, what it consists of, and who holds the responsibility for securing it.

Even if you don't anticipate submitting your findings to judicial review, it can sometimes be appropriate to conduct your analysis as if you might be required to. Sometimes an incident can turn out to have a legal dimension that wasn't obvious in the beginning. In any case, if your procedures meet judicial rules for the admissibility of legal evidence, it's not likely that it will be rejected internally to the organization you work for.

Rootkits

The biggest problem when collecting volatile data is the amount of trust you have to put in the infected machine. Malware authors have learned to go to extreme lengths to hide their creations from the system owner, and recent developments and take up of rootkit and stealthkit technology have truly deserved the meaning of the word *stealth*.

One of the first steps when performing a forensic analysis on an infected machine is to determine if there are any rootkits installed on the machine. This does not necessarily have to be done as a first step, but if there is a rootkit installed on the machine, you will have to stop it before collecting volatile data, otherwise it will almost certainly not be complete and you will miss crucial information about the malware.

Are You 0wned?

Rootkit Detection

It is very difficult to detect beyond all doubt whether a rootkit is present on the machine. As a general rule of thumb, always run multiple rootkit detection tools and compare their results.

In some cases, it will be almost impossible to remove a rootkit without rebooting the machine, and this will cause all volatile data to be lost. If that is the case, you will have to rely on non-volatile data and boot the machine off safe, trusted media

Some anti-rootkit tools:

- Sophos anti-rootkit software: www.sophos.com/products/free-tools/sophos-anti-rootkit.html
- GMER anti-rootkit: www.gmer.net/index.php
- McAfee Rootkit Detective (beta at time of writing): http://vil.nai.com/vil/stinger/rkstinger.aspx
- Panda anti-rootkit software (beta at time of writing): www.pandasoftware.com/download/documents/help/rkc/en/rkc_en.html
- Trend Micro's Rootkit Buster (beta at time of writing): www.trendmicro.com/download/rbuster.asp
- F-Secure's BlackLight anti-rootkit software: www.f-secure.com/blacklight/
- Microsoft Sysinternals' RootkitRevealer: www.microsoft.com/technet/sysinternals/utilities/RootkitRevealer.mspx

Rootkit information resources:

- Rootkit: www.rootkit.com/
- Antirootkit.com: www.antirootkit.com/
- "The Root of All Evil? – Rootkits Revealed." David Harley, Andrew Lee: www.eset.com/download/whitepapers.php
- "Hide 'n Seek Revisited – Full Stealth is Back." Kimmo Kasslin, Mika Stahlberg,
- Samuli Larvala and Antti Tikkanen. In Proceedings of the 15th Virus Bulletin International Conference, 2005.
- "ROOTKITS, Subverting the Windows Kernel," Greg Hoglund and Jamie Butler (Addison-Wesley)

There are many rootkit detectors available today that are more or less based on a common technique: they perform a high-level scan of the system using available APIs, and a low-level scan of the system (typically a direct read of the hard disk). Once the system has been scanned, results are compared and if there are any discrepancies, there is a high possibility that a rootkit is present on the system. For this purpose, one of the best tools is again Microsoft Sysinternals' RootkitRevealer.

Another recommended tool is Joanna Rutkowska's System Virginity Verifier (http://invisiblethings.org/tools.html). While still in development, it has a different approach as it compares memory locations of APIs in the active operating system memory with those written in files on the disk.

Using these tools, perhaps along with some of the other tools listed above, will make you more confident that a rootkit is (or is not) present on the machine you are analyzing.

Collecting Process and Network Data

Information about processes running on the infected machine and established network connections, is the most interesting volatile information that we want and need to acquire. If you have successfully established whether you can trust the operating system (e.g., that a rootkit is not present), then it is relatively straightforward to collect these data.

When enumerating both processes and network connections, with time you will get enough experience to spot "strange" instances and anomalies easily. It is, however, generally easier to enumerate network connections. Most users and administrators will be well acquainted with their own, local computer network and the services they are accessing. Besides that, most current operating systems include utilities that allow you to enumerate network connections easily. These operating systems enable you to use the netstat utility or an equivalent, in order to list all network connections and listening services. Below, we list

network connections on an infected machine that is connected to an Internet IRC C&C server:

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\User>netstat -ano
Active Connections

Proto     Local Address        Foreign Address          State          PID
TCP       0.0.0.0:135          0.0.0.0:0                LISTENING      1904
TCP       0.0.0.0:445          0.0.0.0:0                LISTENING      4
TCP       0.0.0.0:3389         0.0.0.0:0                LISTENING      1848
TCP       0.0.0.0:9688         0.0.0.0:0                LISTENING      948
TCP       192.168.0.1:139      0.0.0.0:0                LISTENING      4
TCP       192.168.0.1:1043     192.168.100.100:6667     ESTABLISHED    3040
```

From the listing above you can see that there is a connection established from the local address of 192.168.0.1 (the infected machine), port 1043 to the remote machine with the IP address of 192.168.100.100 on port 6667 (which is the default IRC port.) Notice the use of the very useful *–o* option, which allows you to list the PID associated with this connection. This option is, unfortunately, present only on Microsoft Windows XP and Server 2003 operating systems and later. In order to get the same functionality on other, older, Windows operating systems, you will need to install a third-party application, such as Foundstone's Fport (www.foundstone.com/resources/proddesc/fport.html). In this particular case, it was very easy to identify the malicious process as well, since it had a connection established. We were therefore able to see which process caused this behavior, just by checking the PID listed.

In other cases, you will want to inspect processes currently running. While you can theoretically use even the humble Windows Task Manager to do this, we recommend another, more powerful utility called Process Explorer (available from www.microsoft.com/technet/sysinternals/utilities/ProcessExplorer.mspx). Process Explorer lists all active processes on the system, and gives you a wealth of information about them. Just as network connection enumeration becomes easier with practice, with a bit of experience you will be able to easily spot rogue processes in the list produced by Process Explorer. The ability to show the full path to the executable from which the currently running process originates is also a big help. This way, when you see a suspicious process, you can see where it has been started from, and this is likely to give you enough information to determine whether the process is rogue or not. Process Explorer can also show you network connections established by any process (enabling you to confirm whether the data you saw with Netstat corresponds to that shown by Process Explorer). There is also a very useful feature that allows you to print all strings from process memory. This feature is similar to the strings utility we used previously, but this time it works on the memory image of the process, not on a file on disk.

After collecting network and currently running process data, you should have enough information about the current state of the machine to enable you to go forward and collect non-volatile data, in order to determine the initial infection vector.

Collecting Non-volatile Data

Non-volatile data (meaning data stored in files on the infected machine) is easier to collect, since trusting the operating system is less of an issue. The typical approach to collecting non-volatile data includes booting the machine off a trusted (known safe) CD-ROM or other non-writeable medium. By doing this, you can be sure that the underlying operating system has not been compromised by the malware you are hunting.

The following steps depend on actions you want to pursue in the future. If there is a chance that this forensic analysis might involve legal action you should follow proper, specialized incident procedures that you should have in place and written before the incident actually happened. (Best get writing now!) This step typically involves creating at least two images of the hard disk from the compromised system, together with their checksums (the use of SHA-1 checksums is recommended). One of these images is then safely stored so that you can perform forensic investigation on the other image. This way, in the event that the incident ends up as the subject of a court case, there is a chain of custody and it can be proved that the original image has not been tampered with.

If, on the other hand, you are just performing an internal malware analysis and there is no likelihood of legal involvement, you might want to mount the hard disk of the infected machine directly under a live Linux distribution (such as Knoppix-www.knoppix.org), and begin collecting files and data. If you do not feel comfortable working under Linux, you can still create the hard disk image of the infected machine and mount it on a different Windows machine, where you can use Windows tools during the analysis. While this has certain advantages, like being able to scan the acquired hard disk image with your AV directly, be aware that you may be exposing the host machine to malicious software, especially if you decide to execute any files from the acquired hard disk. It is best to have a separate, isolated machine for this purpose.

Determining the Initial Vector

At this point, if you have collected all volatile data, you should have enough information about the infection status of the machine, and you will now be interested in determining the infection vector. Before going into deep technical analysis of the malware, it is always recommended that you approach the user or owner of the machine and ask him or her for as much detail as possible about possible infection vectors. In many cases, the user will at least be able to tell you if he executed a file or casually browsed the Internet. However, do not be surprised if the user breached internal policy (for example, by using a peer-to-peer program) and is hesitant and evasive about what really happened.

The vast majority of infections today happen according to one of the following scenarios:

- The user received an infected e-mail and executed the attachment.
- The user downloaded a file with his Web browser or peer-to-peer software and executed the file.
- The user was infected through a vulnerability in his Web browser.

As the first step, you should try to determine the time the machine was infected. Determining a precise time is not critical here (although it helps), because you will be looking at a time window in which the machine could have been infected. If there is an obvious suspicious program that you detected in the previous step, you should check the time at which that file was modified and then search for any activities on the machine that happened in a narrow time frame before and after the infection. Be aware that malware can change the recorded time associated with its host file, so you cannot always rely on timestamps. Nonetheless, in most cases, the time will be correct.

After you locate all files that have been modified or created near the infection time, you will often have enough information to confirm which of the scenarios described above happened. If the user just executed a file, it is quite possible that will be the only activity recorded, but you might also find additional files that have been dropped by this malware.

A Lesson from History

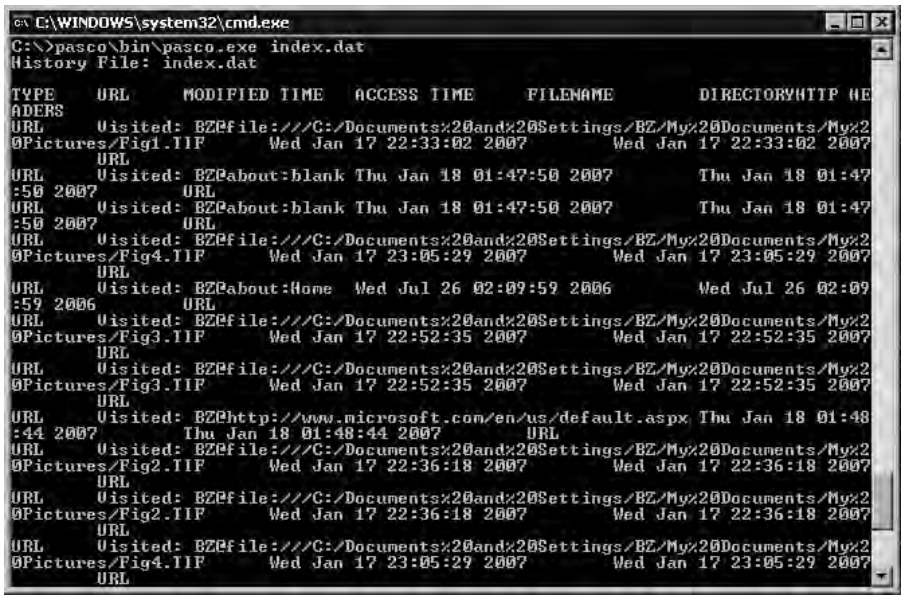
If you suspect that the user got infected through a browser (either through a vulnerability in the browser or by downloading and executing a file), you need to analyze browser history and cache files. These files allow you to reconstruct the user's Internet-related activities, unless they have been deliberately deleted. It is very rare for malware to modify the browser history or cache, so if these have been modified, it is most likely a result of a user's action.

While historically Internet Explorer has been riddled with vulnerabilities, other vulnerabilities in Mozilla FireFox, Opera, and so on have also been made public, and malware authors are happily trying to exploit all of them. As you probably already know, all browsers, by default, cache a certain amount of all content downloaded from the Internet, in order to speed up future accesses by the user to the same sites. This leaves, potentially, a wealth of information that can be analyzed, and is extremely helpful when you are trying to determine the initial infection vector where a browser vulnerability has been exploited.

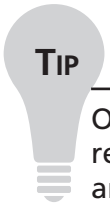
We will start with Internet Explorer as it is the most commonly exploited client on infected machines. By default, Internet Explorer will store all cached files in the directory *C:\Documents and Settings\<username>\Local Settings\Temporary Internet Files\Content.IE5*. There are additional directories inside the *Content.IE5* directory that contain all downloaded files. You can already see that just by inspecting these files, you can get all the information you need about a user's Internet behavior. Typically, just comparing file dates can point you in

the right direction. In the event of needing detailed information about a user's Web activities, you can analyze Internet Explorer's history file. This file is located in the *C:\Documents* and *Settings\<username>\Local Settings\History\History.IE5* directory and is called *index.dat*. Unfortunately, this is a binary file, so you cannot inspect it visually, but there are numerous utilities that can parse this file. The most commonly used (and free) one is Pasco by Foundstone (www.foundstone.com/resources/proddesc/pasco.html). Pasco reads *index.dat* files and outputs a tab-delimited history of the user's Internet activities. This file can then easily be loaded into a program such as Excel and analyzed further. In most cases, this will be enough to show you the site where the infection was picked up.

Figure 9.38 Pasco Parsing Internet Explorer History File



Parsing Mozilla Firefox's history file is a bit easier, as it stores all information in a text (XML) file, so theoretically you do not need specialized tools to do this. Firefox will store the history file in *C:\Documents* and *Settings\<username>\Application Data\Mozilla\Firefox\Profiles\<random>\history.dat*. The random part will be different on every machine (installation), as Firefox uses it to prevent attacks that rely on starting locally cached data by guessing its full pathname.



TIP

Once you have identified the site that caused the infection, you can try to replicate the infective behavior. If you decide to do this, make sure that you are either using an isolated goat machine that does not have access to any other site or network, or use a safe utility such as Wget for pulling files without risking executing them.

Another place that you will definitely want to check is the Recycle Bin. Normally, deleted files are just stored in the Recycle Bin, which has another nice feature: it stores the exact time when the file was deleted, as well as the original pathname. Foundstone built another useful forensics utility called Rifiuti (www.foundstone.com/resources/proddesc/rifiuti.html) that can be used to analyze Recycle Bin contents.

Case Study: An IRCbot-infected Machine

We will finish this section with a case study of a machine that has been compromised by an IRCbot, a worm which connects to a command and control IRC server, and sits in an IRC channel waiting for instructions from the owner. In most real life scenarios, these bots (zombies) are used either as component systems for Distributed Denial of Service (DDoS) attacks or for sending spam, and it is not uncommon for attackers to use them as their base for future attacks (especially if they can compromise your local network or more remote networks by doing so).

In this case, the administrator has been informed of a machine that is generating high volumes of network traffic and causing problems in its own subnet. The administrator immediately suspects that the machine was compromised, and decides to follow proper forensics procedures until it can be determined what kind of compromise he is dealing with.

The first step in the forensic analysis is to collect volatile data. As we already said above, in order to do that you have to know that you can trust the operating system. Nevertheless, if there is an active attack in progress, as it is the case here, you might want to see if there is anything obviously suspect running on the machine. The first thing the administrator would do in this case is to check all active network connections on the machine using Netstat. The (truncated) output from Netstat is shown below:

```
C:\Documents and Settings\User>netstat -ano
Active Connections

Proto    Local Address       Foreign Address        State          PID
[...]
TCP      192.168.0.1:1044    192.168.200.200:80     SYN_SENT       4111
TCP      192.168.0.1:1045    192.168.200.200:80     SYN_SENT       4111
TCP      192.168.0.1:1046    192.168.200.200:80     SYN_SENT       4111
TCP      192.168.0.1:1047    192.168.200.200:80     SYN_SENT       4111
[...]
TCP      192.168.0.1:5440    192.168.100.100:8555   ESTABLISHED    4111
```

As you can probably see, this looks much like a DDoS attack on a remote site (via port 80, so the Web server is being attacked) and SYN packets have been sent by a process with PID 4111. The administrator logged netstat output in a file, as the volume of output can be pretty big when a DDoS attack has been launched.

At this point in a similar attack, you should disconnect the machine from the network. Although you risk losing some information (for example, whether the attacker was connected to the machine over the network), you still have all the information about network connections logged by netstat. By disconnecting the machine from the network, you will block the DDoS attack as well (at least the part coming from your network).

Now that initial information has been gathered, the operating system should be checked to determine whether it can be trusted. This can be done by running a couple of rootkit detector utilities. In this case, the scan did not detect anything on this machine, so the administrator proceeds with the investigation.

It is obvious that the process with a PID of 4111 caused network attacks and, as can be seen on the last line of netstat's output, it has also established a weird connection to port 8555 on a remote site. By using Process Explorer, the administrator successfully locates this process, which turns out to be a binary named *svchosts.exe*. and located in *C:\WINDOWS\System32* directory. If you are familiar with Windows operating systems (or even read the example earlier in this chapter), you will recognize this as an attempt to hide malware among legitimate system files. Current versions of Windows come with a system process called *svchost.exe*. (Notice the missing "s" character at the end of the main filename; i.e., the part that comes before the filename extension.). Process Explorer's memory string search function is also very useful here, having shown that this is indeed an IRCBot. All of the typical IRC commands are listed (such as */JOIN*, */NICK* and so on) as well as various commands generally associated with DDoS attacks.

After killing the process, all DDoS-related activities immediately stop, so it is clear that the IRCbot is responsible for the DDoS attack and for the network traffic originating from this machine. The executable file can now be archived for later analysis (if needed, just the collection of volatile data provided us with a lot of information about the nature

of the compromise). The startup sequence should be checked to ensure that nothing else has been added. In this instance, HijackThis showed only one new process added in the Run registry key, and the executable it started was the IRCbot itself.

At this point, you can decide what to do next. In many cases, when we are talking about a bot installation, we recommend that you simply archive files unequivocally identified as innocent data, and reinstall the operating system and all applications, as you cannot be completely sure what additional malware may have been dropped by the bot after the machine was infected. Besides reinstallation of the system, it is also recommended that you determine the initial infection vector.

As our case study administrator knows the approximate time when the DDoS attack started, he can search the file system on the compromised machine thoroughly, looking for any files that have been created or modified that same day. After checking the results, the administrator locates a small, 5kb file in the Temporary Internet Files directory, with the timestamp of 5 minutes prior to the DDoS attack. Running the *strings* utility on the binary produced no results, and checking with PEiD confirms that the file was packed.

The administrator can now use Pasco to list all Web sites that the user visited. He locates those with an access time close to the DDoS attack, and quickly identifies a suspicious Web site with only an IP address and no name. Finding that the user visited a legitimate site only a second before the suspicious Web site was visited, the administrator cautiously downloads a copy of the legitimate site's HTML Web page, only to find out that it has been compromised and that there was an IFRAME pointing to the site with an exploit. After downloading the exploit Web page, the administrator is able to identify the exploit that has been used; Internet Explorer on this user's machine has not been fully patched.

Notes from the Underground

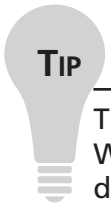
JavaScript Obfuscation

Almost all browser exploits today use either JavaScript or VBScript. In order to make detection more difficult for inline Web AV programs, malware authors obfuscate their JavaScript programs. It is typically easy to spot JavaScript obfuscation. It will always consist of a function with some mathematical operation and the function call with a big, obfuscated input argument. This JavaScript is automatically evaluated and processed by a Web browser.

In order to safely analyze obfuscated JavaScript without risking infecting the machine, you can use a command-line JavaScript interpreter called SpiderMonkey (www.mozilla.org/js/spidermonkey). SpiderMonkey is part of the Mozilla project,

and is the same interpreter that is used in the Firefox browser. When you use the command-line interpreter, the evaluated JavaScript code that the browser will execute is not executed, as SpiderMonkey does not in itself provide a host environment for JavaScript: it will just be printed on your screen so you can further analyze it. One caveat: JavaScript interpreters are different in Internet Explorer and Mozilla FireFox, and there have been cases when malware authors abused this to prevent analysis in Mozilla FireFox.

By analyzing all activities related to this incident, the administrator was able to determine the initial vector (and replicate its behavior in a safe, isolated environment). While determining whether a single patch was missing is relatively easy, by identifying the exact infection vector, our administrator can be sure that his network is now protected from this attack.



TIP

There's another interesting case study in Syngress's "Botnets: The Killer Web App" by Craig Schiller and Jim Binkley et al, in the chapter on botnet detection. (See www.syngress.com/catalog/?pid=4270)

Summary

With the knowledge that you gained in the 101 section of this chapter, you should be much better armed the next time that you get that phone call from your SOC. We would strongly recommend that you practice using these tools until you know how to use them perfectly. The quicker and more proficient you become in the proper use of these tools, the sooner the outbreak will be stopped, and the fewer infected machines you'll have to deal with during the aftermath.

The tools mentioned in this chapter are also by no means the only tools that will accomplish the needed tasks. They are tools that are used by the authors and that are recommended for your use, but there are newer and better tools being created each and every day. A good place to watch for new tools, or perhaps tried and true tools that you didn't know existed, is the listing found on http://sectools.org/. The site maintains a listing of the top 100 security tools. It is well worth looking over the listing, downloading a few, and trying them out. You may just find a new favorite tool that you'll add to your emergency jumpkit.

In the last couple of years, malware has become increasingly difficult to analyze and remove. Most malware authors today are not students who want to show how good they are in programming, but organized crime gangs that seek profit. They go an extra step in making it difficult to remove their malware, to hide it, and to make reverse engineering more complex. This is why the only sure way to deal with infected machines is to reinstall them. However, before doing that, you should make sure that you know what the infection vector was, because otherwise you probably face the same re-infection in the future.

A prepared and tested incident response plan is a must for every organization today. Malware incidents will happen, no matter how much you have invested in protection. When previously unknown malware strikes your organization, you will have to assess the impact and decide on the countermeasures. This can be very difficult in the first few hours of malware spread, as most AV vendors will not yet have definitions and malware descriptions.

By analyzing malware when you need to, you will be able to assess the impact (and the threat) correctly, and ultimately decide on money spent by your organization on damage recovery which can range from reinstalling the infected machine to dealing with stolen intellectual property or customer data.

Solutions Fast Track

Anti-Malware Tools of the Trade 101

- ☑ The first step to take when an incident is suspected, is to isolate a problem machine so that the only remote system it can access is one from which you plan to run your analysis.
- ☑ This should be possible using firewall rules or Access Control Lists (ACLs)

The Basics – Identifying a Malicious File

- ☑ The PSTools package contains a number of extremely useful tools for the administrator's jumpkit. psexec.exe is particularly useful for quickly launching a remote console and running programs remotely.
- ☑ The VNC system allows desktops to be shared and is platform-independent, and therefore, it can be particularly useful in a mixed environment.
- ☑ Once you have a remote control shell open, you can run other tools locally to the machine under analysis: for instance, the other PSTools packages and netstat.

Process and Network Service Detection Tools

- ☑ Three particularly useful tools for analyzing processes and network services are FPORT, tcpvcon, and Handle. FPORT is similar in function to netstat, but includes some significant enhancements.
- ☑ tcpvcon is one of the very useful tools available from www.sysinternals.com, and traces network processes back to the originating pathname.
- ☑ Handle offers rich output on all types of extant handles, including ports, registry keys, synchronization primitives, threads, files, and processes.
- ☑ Archiving sample files with a password ensures that malware samples aren't executed accidentally, or inadvertently removed or modified by security programs.

Web Based Inspection and Virus Analysis Tools

- ☑ It's better to submit suspicious samples directly to your own AV vendor as soon as possible, rather than rely on Web sites that are running multiple online scanners to forward samples in a timely fashion. Sites like Jotti, VirusTotal, and Virus.Org can be very useful as a supplementary check in the early stages of identification, but there are risks in relying on totally trusting their results.You may also find it useful to try samples against the online scanners some anti-virus vendors offer.
- ☑ Some sites also offer sandbox analysis of submitted samples. Using a service like this gives you the opportunity to test a sample and take appropriate countermeasures even before carrying out comprehensive analysis in-house.

Using Packet Analyzers to Gather Information

- ☑ Sacrificial goat machines should not be connected to a production network, for safety reasons, but should be configured so that they closely resembles your production machines. Using a package such as VMware allows you to run a goat machine within a virtual environment.
- ☑ A packet analyzer such as tcpdump, windump or Wireshark is a must-have for your toolkit. It enables you to inspect information traversing the network to and from the machine being inspected.
- ☑ As well as capturing and analyzing packets, Wireshark reads PCAP-formatted output files from tcpdump, windump, snort et al.

Examining your Malware Sample with Executable Inspection Tools

- ☑ Strings is a program that searches a file for strings of Unicode or ASCII characters, which may provide clues to IRC server names, IP addresses, and so on.
- ☑ Not all useful tools are open source or freeware. VIP Utility is a commercial open sandbox program that monitors an executable, using a combination of heuristics and behavior analysis outputting somewhat similar results to some of the Web-based tools described earlier.
- ☑ InCtrl5 takes a snapshot of files, registry keys, text files, and INI files before and after the file being analyzed is executed.

Using Vulnerability Assessment and Port Scanning Tools

- ☑ The most popular and best known port scanning tool is probably Nmap. Its default command-line options can be supplanted or modified for a more aggressive or more passive scan.
- ☑ Superscan offers similar functionality to Nmap's, but with a GUI front end (Nmap also has an optional GUI).
- ☑ The most widely used vulnerability scanner is Nessus, which is flexible and full-featured, yet easy to use. It includes its own scripting language.

Advanced Tools: An Overview of Windows Code Debuggers

- ☑ A debugger helps you to reverse-engineer a suspicious file. WinDbg is a basic but competent component of the "Debugging Tools for Windows."
- ☑ OllyDbg is a highly-rated, very popular and full-featured 32-bit assembler level debugger, highly suitable for taking apart suspected malware in the Windows PE format.
- ☑ DataRescue's IDA Pro is a reasonably priced best-of-breed debugger/disassembler that can handle a huge range of executable formats.

Advanced Analysis and Forensics

- ☑ The sheer volume of new variants of bots and other malware associated with today's explosion of spam and other breaches, compromises the ability of AV vendors to deploy timely reactive and proactive countermeasures. This in turn puts a strain on in-house security administrators who often have to analyze malicious code themselves, in order to maintain services and deploy countermeasures specific to their organization.
- ☑ Depending on how detailed an analysis is required, you might want to use static analysis, dynamic analysis, or a hybrid approach. Malware can be analyzed statically or dynamically. Static malware analysis offers in-depth information about the target program, but can be very time consuming, whereas dynamic analysis can be quicker, but may be thwarted by malware using a range of deceptive techniques.

Static (Code) Analysis

- ☑ Static analysis involves reviewing, more or less by hand, the actual code of a program in order to determine what it does. To review code, we must be able to disassemble it, and to be well-acquainted with the inner workings of assembly language and machine code.
- ☑ The biggest obstacle in static analysis is the use by malware authors of runtime packers. Packed malware must be unpacked in order to analyze the raw code.
- ☑ Runtime packers are used by malware authors to compress the size of the executable, shorten the loading time, and, most significantly, to obfuscate the code and make it harder to analyze.
- ☑ Once the file has been unpacked, it can be examined with a range of utilities, from simple tools like *strings* to top-flight utilities like IDA Pro.

Dynamic (Behavior) Analysis

- ☑ Dynamic or behavior analysis can, in some instances, be much faster than static or code analysis, yet yield almost as much information. It involves monitoring the behavior of the suspect program while it's executed in order to ascertain what it really does.
- ☑ Dynamic analysis needs to be undertaken in isolation. Nowadays, the use of virtual environments makes it easier to simulate a real environment without significant risk to a production environment, but may not be feasible for analyzing malware that checks whether it's running on a virtual host.
- ☑ A number of tools need to be installed on a sacrificial virtual machine, and are likely to include Filemon, Regmon and AutoRuns from Microsoft Sysinternals and the WireShark network sniffer.

Forensic Analysis

- ☑ Documenting and testing an incident response plan is vital. This has to be done before the incident happens, and documentation and procedures need to be reviewed and tested both routinely, and in the light of experience of specific incidents.
- ☑ Volatile data gives you information about malware's behavior when it's active on the machine.
- ☑ Collecting volatile data does not justify leaving the machine actively attacking other machines/network.
- ☑ Always make several copies of the infected hard disk so you preserve original information while working on a copy.

Frequently Asked Questions

Q: Where can I get more information about issues like batch file syntax and command line redirection using >> and >, for instance?

A: You can still get some of this information from Web sites like http://www.computerhope.com/msdos.htm and http://en.wikipedia.org/wiki/Batch_file. Note, however, that information that may be correct when it refers to specific DOS versions may not be accurate for programs run at the DOS prompt in modern Windows operating systems, where the DOS shell is more emulation than core system. For such systems, it's worth examining the help files for Windows as well as the program help and summaries available using the help options at the DOS prompt. In brief, not everything works as it did in real MS-DOS.

Q: What does 127.0.0.1 mean?

A: It's the IP address that denotes the machine you're using. In other words, it's used for a loopback connection between a machine and itself. This has many diagnostic uses, though it can also be used maliciously.

Q: Why do you use an obsolete version of PKZIP for archiving examples, rather than modern tools like WinZip, or one of the many freeware tools?

A: There's nothing to stop you using WinZip, Stuffit and so on, or even Windows' own compression/decompression. PKZIP is convenient because, like many of the security tools used here, it's a command line tool that can be easily incorporated into scripts and batch files, and it's reasonable to expect it to conform absolutely to the ZIP archive standard. If you use other compression formats for submission of samples, for example, you may not be able to assume that they'll be read or decrypted properly at the other end, depending on what software they have access to. In particular, you may need to use standard ZIP 2.0 passwords, not one of the more secure but less standard alternatives such as AES encryption.

Q: If I can use an online virus scanner, why do I need virus scanning on my machine?

A: Online scanners, although they may have more up-to-date virus definitions than desktop scanners, aren't in all respects as reliable in terms of detection mechanisms. Nor do they offer the sort of real-time protection that an on-access scanner on your system can, or the configurability.

Q: What's all this goat stuff?

A: The use of the sacrificial goat metaphor in AV and anti-malware literature probably stems from the practice of hunting tigers by tethering a goat in tiger country and waiting for the tiger to come along in search of an easy meal. (There are many instances of goats used in sacrificial rites, of course.) Another use of the metaphor in this field is the so-called goat file, a file intended to attract the attention of specific types of viral program. The term scapegoat also has a sacrificial origin (Leviticus 16:22, the Talmud and so on), though the use of the term in the context of malware analysis is less common.

Q: What network ports should be open in general?

A: That's too complex a question and too specific to how a particular operation works. A sensible approach is to block everything except services you know you need. That can be very time consuming to implement properly, though. Of course, when you're specifically looking for malicious traffic, you need an environment with a more laissez faire approach.

Q: What platforms are appropriate for malware analysis?

A: Obviously, you need actual or virtual machines representing the vulnerable platforms in your production environment, or your customers'. You might also want access to platforms that aren't believed vulnerable to the same types of threat, not only for safety, but to lessen the risk of inaccuracies introduced by spoofing.

Q: Doesn't forensics always deal with criminal action and legal cases?

A: Traditionally, yes, but the term is now very commonly used in digital forensics and computer forensics to refer to detection and analysis for non-judicial purposes. You should note that some of the approaches discussed here are not compatible with "traditional" forensics for legal purposes, unless they're carried out in the context of strict adherence to standards relating to the preservation of evidential integrity. In other words, if you're likely to have to present evidence that can be challenged in court or elsewhere, you have to be scrupulous about preserving the chain of custody, documentation, and so on, in order that your evidence remains admissible. That's a fascinating subject, but it's out of scope for this chapter. There's a reasonable overview at http://en.wikipedia.org/wiki/Computer_forensics, if you want to start exploring it.

Q: How can you be sure you aren't being fooled by rootkits and such?

A: There are many ways of detecting known and unknown rootkits, though you can fool some of the detector programs some of the time. It's unlikely that there will ever be

malware so effective that it can never be detected, but you have to live with the fact that there is always a risk that some of what you're seeing is what some malware wants you to see.

Q: What is packing?

A: Packing is a way of reducing the size of compiled program code and obfuscating it. It is most commonly used by malware authors to increase the complexity of reverse-engineering their program.

Q: Can a process dumper be used on all malware so we do not have to unpack it?

A: No, some advanced packers, such as Armadillo, will unpack only parts of the code and will keep the rest packed in memory. When that part of code is needed it will be unpacked. This means that if such a process is dumped, only a small part will be visible.

Q: Why can some malware not be debugged with OllyDbg?

A: Some malware authors will implement various anti-debugging mechanisms to make analysis more difficult and time consuming. In cases like this, you will either need to use Microsoft's kernel debugger or disable the anti-debugging part of the malware.

Q: Can virtual machines always be used for behavior analysis?

A: No, there are packers and malware that detect the presence of a virtual machine and will either exit or run a benign function. In cases like this, you will have to either statically analyze the malware or run it on a physical machine.

Q: How do rootkits affect volatile data collection?

A: Rootkits can be configured to hide certain network activity, files on the disk, and registry keys. If a rootkit is present on the machine you are analyzing, and you have not disabled it, you will most likely not be able to see any malware-related files and activities.

Q: Do browsers always leave cache/history traces?

A: No, a user can manually delete cache contents and browser history.

Q: How can I de-obfuscate JavaScript files and see what they do?

A: You can run them through SpiderMonkey, a command-line JavaScript interpreter. Additionally, you can find the line which evaluates the output and disable or change it to display the data instead of evaluating it.

Chapter 10

Antimalware Evaluation and Testing

Solutions in this chapter:

- **Antimalware Product Evaluation**
- **Evaluation Checklist**
- **Antimalware Product Testing**
- **Sources of Independent Tests and Certification Bodies**

- ☑ **Summary**
- ☑ **Solutions Fast Track**
- ☑ **Frequently Asked Questions**