Digital Investigation (2005) 2, 19–22

Digital Investigation

www.elsevier.com/locate/diin

Malware analysis for windows administrators

Harlan Carvey

14371 Fowlers Mill Dr, Gainesville, VA 20155, United States

KEYWORDS
Malware;
Analysis;
Windows;
VMWare;
Anti-virus

Abstract Malware is continually evolving, and anti-virus vendors have a hard time keeping up. In some cases, the vendors may opt not to include a signature for a particular piece of malware. However, this does not prevent Windows administrators from using freeware tools and techniques to analyze the files and develop their own prevention and detection mechanisms.
© 2005 Elsevier Ltd. All rights reserved.

Purpose

The purpose of this article is to provide Windows administrators with tools and techniques that can be used to analyze suspicious files found on their systems. This analysis can lead to security mechanisms that can be implemented to protect the network infrastructure.

Introduction

You're a Windows system administrator, and you've found a file on a system that you don't recognize. You think it may be something malicious, a bit of malware that made its way onto the system and created the havoc that got your attention in the first place. You have a limited budget (if you have a budget, that is) and no experience with assembly language programming or code debugging at all. What do you do? What can you do? Well, fear not. There are a number of extremely useful freeware tools that can provide you with insight as to what the file does, as well as determining exactly what effect it has on a system.

To start off with, you should have already collected background information about the suspicious file as part of your usual incident response activities. You should have collected volatile information (i.e., process information, network connections, etc.) from the ''victim'' system, as well as some non-volatile (i.e., startup entries in the Registry, etc.) information, and you noticed references to this particular file. You were able to correlate the Registry startup entry with a running process, and the executable image for that process was located in the system32 directory. This file had an odd name that you didn't recognize, so you collected the file's MAC (i.e., last modified, last access, and creation) times, attributes, full path

E-mail address: keydet89@yahoo.com

1742-2876/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.diin.2005.01.006

information, and determined whether there were any NTFS alternate data streams or links associated with the file. As a precaution, you also ran handle.exe and listdlls.exe against the running process, before dumping the process's memory contents to a file using pmdump.exe (from NTSecurity.nu). With that done, you killed the process via the Task Manager, and made a copy of the file in question.

You became concerned when you scanned the file with an updated version of the anti-virus (A/V) software you use, and the A/V software didn't recognize it as malicious software. So you've got some information to go on, and now you need to identify what the file does on a system. Doing so will not only allow you to determine how to detect other systems what may have been affected by the malware, but also how to protect your systems, as well.

At this point you'd like to analyze the file, and see what it does and what affect it has on a system. There are several reasons for analyzing malware, but the most important is to see what effect it has on a system so you can develop prevention and detection mechanisms for both your hosts and the network itself.

Setting up your test environment

Before taking a look at the suspicious file, we need to make sure that we have a suitable test environment. This system will be a ''throw away'' system, or a non-critical system that can be easily recreated or reinstalled. All of the necessary tools, as discussed later in this article, should be installed on the system. One way to accomplish this is to install the operating system from scratch, update it with all necessary patches and updates, and have the tools you're going to use available on a CD, or download from a network share. Another method would be to create a system image, and update that image as necessary. A third option is to use VMWare, installing versions of Windows (2000, XP, 2003) as guest operating systems. Once all updates and tools have been installed, create a snapshot of the configuration. After the snapshot has been created, begin your analysis. Once you've completed all tasks, you can simply revert to the snapshot, and all effects of everything done after the snapshot was created will be removed.

Once you've selected a system, and a method for managing the image, you'll need to decide how you want to set the system up on a network, or even if you want to do so. In many cases, you may want to have a standalone system, but this limits your dynamic analysis of the malware. What happens if the malware attempts to make a connection to the network, or out onto the Internet? Should your analysis end there? Perhaps not. If the malware tries to contact a website, wouldn't it be useful to know what it's trying to do; i.e., is it trying to upload or download information?

Systems used to analyze malware need to be isolated from production networks. Malware that attempts to spread via Web server exploits or networked shares can quickly wreak havoc on an infrastructure if not closely monitored. Testbed systems should be set up on an isolated VLAN or on a DMZ subnet (in both cases, all traffic should be directed away from production facilities), or on a completely separate network, altogether.

The method you choose to set up your test environment may depend on a variety of facts, such as availability of funds or systems, size of the infrastructure, or simply available network ports.

Static analysis

The first step in analyzing the file is to conduct static analysis, which involves examining and analyzing the contents of the file without launching it, either as a standalone executable or through an application. In some cases, such as scripts or configuration files, static analysis is all that is necessary.

When conducting static analysis of malware, the purpose is to look at the file and get an idea of what it does without actually launching it. What can you do, short of using a debugger or disassembling the code itself? Well, without a suitable budget and/or an understanding of assembly language programming, there are a number of freeware tools you can use to peek at the inner workings of that malware file.

The first thing you need to do with the suspicious file is ensure that it is an executable by performing file signature analysis. Executable files (ending with .exe, .sys, .ocx, or .dll extensions) on Windows systems include the letters ''MZ'' in the first couple of bytes of the file. You can perform file signature analysis by opening the file in Notepad (or a hex editor) or by using a file signature analysis utility (such as sigs.exe from Windows-ir.com).

Next, you will want to look for strings embedded within the file, using tools such as strings.exe (from SysInternals.com) or BinText (from Foundstone.com). These tools allow you to view a listing

of all ASCII and Unicode strings, of a user-specified length, from within a file. These strings can give clues as to the purpose of the file. As most executable files are binary data, any strings will stand out, such as messages displayed for the user, Windows API functions accessed, or even the author's manifesto (as found in the IE-0199/ATAKA executable), if one was added to the file.

Specific strings you should look for include file version information embedded within the file. Many commercial organizations include such information with their files, and some malware consists of legitimate applications. For example, the russiantopz IRCbot consists of mIRC32.exe, a legitimate IRC client. Perl scripts can be written that use the Win32::File::Ver or Win32::Exe modules to retrieve file version information.

There are a variety of other tools that you can use to investigate a suspicious file. If you suspect that the file has been compressed in some manner (many malware files are compressed using tools such as UPX or ASPack), tools such as PEid will assist you in identifying the particular compression tool employed. If the file has been compressed, and you determine the compression used, you can then download the appropriate program to expand the file.

Once you have an expanded file to examine, Resource Hacker will let you view the resources (i.e., icons, dialogs, message and string tables, etc.) compiled into the executable, which may provide valuable clues as to the function of the file. LordPE will let you examine elements of the PE (portable executable) format, as will PEExplorer. Note: There is a fee for using PEExplorer beyond its trial period.

Another useful tool is Dependency Walker. This tool allows you to open a file and see which dynamically loaded modules (DLLs) the executable relies on, giving clues as to the function of the file.

Dynamic analysis

Static analysis of executable files has a number of limitations. Short of decompiling or disassembling the executable and being fluent in assembly language, you will not know what the executable does to a system without launching it. Dynamic analysis of malware involves loading the file onto a testbed system and launching it, while monitoring it to determine what effects it has on the system. The testbed system can be set up in a number of ways, as discussed in the ''Setting up your test environment'' section. The steps inherent to dynamic analysis of malware are to document the configuration of the test system, run monitoring tools while executing the malware (if this is possible), and then at some point (either the malware processes have completed, or activity seen in the monitoring tools has died off) to note any modifications made to the system by the malware.

There are several tools available for making snapshots of system configurations, and then comparing those snapshots to subsequent snapshots created after the execution of malware. WinAlysis is available as trialware, but InControl5 is a freeware utility that is extremely easy to use. Once installed on a system, InControl5 is run in two-phase mode. The first phase creates the snapshot of the system, and the second phase compares the configuration of the system at a later date or time to the original snapshot. InControl5 is capable of showing any added, removed, or modified files, directories, or Registry entries. InControl5 provides its reports in HTML or CSV format.

Once you've put the malware file on the testbed system (transferred via CD, USB-connected thumbdrive, diskette, etc.), and run the initial phase of your snapshot utility, you'll need to start your monitoring tools before you execute the malware. You'll want to monitor as much activity on the system as possible, including file and Registry access, network communications, process creation, etc. There are several excellent tools available for these purposes. You can get FileMon (monitor file and directory access activity) and RegMon (monitor Registry access activity) from SysInternals.com, as well as Process Explorer (monitor process activity) and TCPView (monitor which processes use which network connections). Ethereal (from Ethereal.com) is a freeware, open source network sniffer that can be used to capture network traffic and reconstruct TCP streams, among other functionality. The information collected by Ethereal (as well as other sniffer tools, such as WinDump and tethereal) can be used to create snort (snort is a freeware, open source network-based intrusion detection application) or firewall rules to detect or prevent the malicious activity. All of these monitoring tools are GUI-based, and need to be started prior to launching the malware file. Once monitoring has been completed, the information collected by these tools can be saved to files and archived for later analysis.

Once you've launched the malware, you may want to get further information about the processes that are created as a result of the

malware. Some malware may create several processes once launched, to include retrieving additional information from sites on the Internet. More detailed information can be retrieved about these processes using tools such as tlist.exe (from the Microsoft Debugger Tools), handle.exe, listdlls.exe, and tcpvcon.exe (all from SysInternals.com), as well as pmdump.exe (as mentioned previously in this article). Tools such as fport.exe (from Foundstone.com), openports.exe (from DiamondCS.com.au), and Microsoft's own portqry.exe (version 2) can be used to perform port-to-process mapping, showing which processes are bound to which network ports. There is another tool available for monitoring which applications make network connections from Microsoft called Port Reporter. Port Reporter installs as a service on Windows 2000, XP, and 2003 systems, and is able to log TCP and UDP network port activity. On Windows XP and 2003 systems, Port Reporter will also log the process that uses each particular port. There is also a tool available called Port Reporter Parser, which will parse the Port Reporter logs and provide some level of analysis. Port Reporter is capable of collecting a great deal of information, and the parser tool can assist in identifying unusual or suspicious activity.

The best approach during dynamic analysis is to collect as much information as possible, and to even use multiple tools that collect similar information. In some cases, one tool may use different application programming interface (API) calls to collect information, and may be capable of providing some information that another tool misses, as in the case of a rootkit. Multiple tools can be used to collect information on processes, such as tlist.exe, pslist.exe (from SysInternals.com), scripts that implement WMI to access the Win32_Process and Win32_Thread classes, as well as port-to-process mapping tools. Once all of this information has been collected, Perl scripts can be written to quickly parse through the information to identify disparities, such as processes that appear in the output of some tools, but not others. Perl can also be used to provide a level of data reduction and reporting, by implementing Marcus Ranum's ''artificial ignorance'' (a method for identifying anomalies by filtering out everything that is ''normal'' or a ''known good'').

Summary

This article is not intended to be a step-by-step guide for analyzing all potential malware, as such a thing is hardly possible. There are many forms of malware, including adware, spyware, worms, and rootkits, to name a few. Each of these is continually evolving as techniques are produced to prevent and detect infections by the malware. For example, one method used by rootkits to infect a system is called ''DLL injection'', which occurs when the DLL for a rootkit is injected into the memory space of a legitimate application. Rootkits have since evolved to perform direct kernel object manipulation, where the linked list that maintains the list of processes on a system is manipulated to hide the rootkit's process.

The purpose of this article is to provide an introduction to what can be done to analyze suspicious files. Many times, an administrator will find a file on a system that isn't identified as malicious by an anti-virus software package. The constant evolution of malware leads to a lag time from detection to the implementation of a signature by anti-virus vendors. Windows administrators can use the tools and techniques described in this article to analyze suspicious files and subsequently protect their systems.

Resources

VMWare, http://www.vmware.com
pmdump.exe, http://www.ntsecurity.nu
Strings.exe, FileMon, RegMon, Process Explorer, TCPView, handle.exe, listdlls.exe, tcpvcon.exe, http://www.sysinternals.com
BinText, fport.exe, http://www.foundstone.com
Perl, http://www.activestate.com
PEiD, http://peid.tk
Resource Hacker, http://www.users.on.net/johnson/resourcehacker/
LordPE, http://mitglied.lycos.de/yoda2k/LordPE/info.htm
PEExplorer, http://www.heaventools.com/
Dependency Walker, http://www.dependencywalker.com
WinAlysis, http://www.winalysis.com
Ethereal (archive includes tethereal), http://www.ethereal.com
WinPCap, WinDump. http://winpcap.polito.it/
Openports.exe, http://www.diamondcs.com.au/index.php?page=products
Port Reporter, Port Reporter Parser, tlist.exe, http://www.microsoft.com

Harlan Carvey is a computer security consultant located in the Metro DC/Northern Virginia area. He specializes in vulnerability assessments and incident response with regard to Windows systems, and is the author of ''Windows Forensics and Incident Recovery'' (available on Amazon, etc.). He also provides incident response training for Windows administrators and consultants.