

File Identification and Profiling: Initial Analysis of a Suspect File on a Windows System

Solutions in this chapter:

- Case Scenario: “Hot New Video!”
- Overview of the File Profiling Process
- Working with Executables
- File Similarity Indexing
- File Signature Identification and Classification
- Symbolic and Debug Information
- File Obfuscation: Packing and Encryption Identification
- Embedded Artifact Extraction Revisited

Introduction

This chapter addresses the methodology, techniques, and tools for conducting an initial analysis of a suspect file. The methodology for file identification and profiling remains essentially the same for both Windows based and Linux based analysis, although some of the tools and techniques differ. This chapter introduces Windows-based file profiling analysis through an incident response scenario. In the next chapter, a parallel investigation on a Linux system is conducted. Then, in Chapters 9 and 10, the investigation of suspect files will continue with hands-on, Windows based and Linux based behavioral analysis tools and techniques.

Some of the techniques covered in this and other chapters may constitute “reverse engineering” and thus fall within the proscriptions of certain international, federal, state, or local laws. Similarly, some of the referenced tools are considered “hacking tools” in some jurisdictions, and are subject to similar legal regulation or use restriction. Some of these legal limitations are set forth in Chapter 6, “Legal Considerations.” In addition to careful review of these considerations, consultation with appropriate legal counsel prior to implementing any of the techniques and tools discussed in these and subsequent chapters is strongly advised and encouraged.

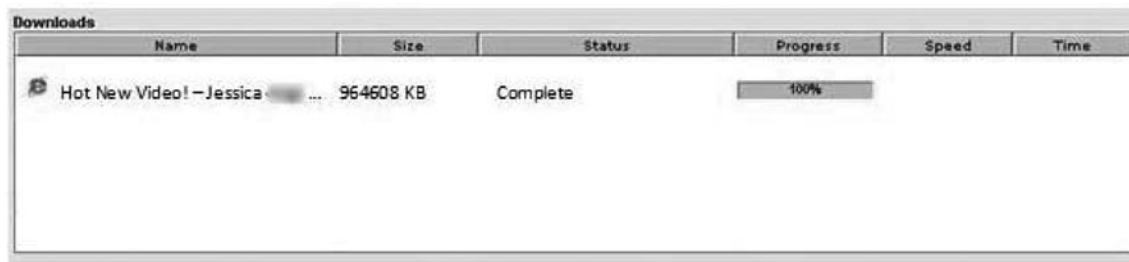
Analysis Tip

Safety First

Forensic analysis of potentially damaging code requires a safe and secure lab environment. After extracting a suspicious file from a system, place the file on an isolated or “sandboxed” system or network, to ensure that the code is contained and unable to connect to or otherwise affect any production system. Even though only a cursory static analysis of the code is contemplated at this point of the investigation, executable files nonetheless can be accidentally executed fairly easily, potentially resulting in the contamination of or damage to production systems.

Case Scenario: “Hot New Video!”

Barkley, a big fan of actress “Jessica,” was searching for new videos of her with his favorite peer-to-peer program, when he hit the jackpot. Someone was sharing a “Hot New Video!” of Jessica that had never been seen before. The listed video was described as particularly provocative and revealing, and Barkley had to have it. Barkley downloaded the file, named it “Video,” and double clicked on it, but the video would not open. Since then, Barkley has noticed that his computer sometimes runs slow. (See [Figure 7.1](#).)

Figure 7.1 The “Hot New Video”

Barkley provides you with a copy of the suspect file and requests that you analyze it to figure out what it is. Barkley advises you that he has anti-virus software on his computer, but believes that the license is expired and does not recall the last time the signatures were updated. No further details regarding the incident are provided.

You bring the suspect file back to your lab for analysis. Upon copying the file to the malware laboratory system, you learn that the icon associated with the file is for Internet Explorer, depicted in [Figure 7.2](#).

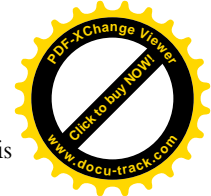
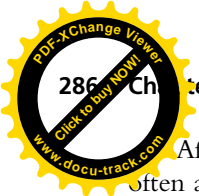
Figure 7.2 The Suspect File: Video

You are unfamiliar with the file. How do you proceed with your investigation?

Overview of the File Profiling Process

Whether during the course of responding to or investigating an incident encountered on a system within a targeted network, or clearly linked to receipt by a network user via e-mail, instant messaging, or other means of online communication or file transfer, a *suspicious file* may be fairly characterized as:

- Of unknown origin
- Unfamiliar
- Seemingly familiar, but located in an unusual place on the system
- Similarly named to a known or familiar file, but misspelled or otherwise slightly varied (a technique known as *file camouflaging*)
- Determined during the course of a system investigation to conduct network connectivity or other anomalous activity



Chapter 7 • File Identification and Profiling: Initial Analysis

After extracting the suspicious file from the system, determining its purpose and functionality is often a good starting place. This process, called *file profiling*, should answer the following questions:

- What type of file is it?
- What is the intended purpose of the file?
- What is the functionality and capability of the file?
- What does the file suggest about the sophistication level of the attacker?
- What affect does this file have on the system?
- What is the extent of the infection or compromise on the system or network?
- What remediation steps are necessary because the file exists on the system?

Although often difficult to answer all of these questions without deep forensic analysis, the right file profiling methodology often paves the way for more efficient and robust incident response overall.

Analysis Tip

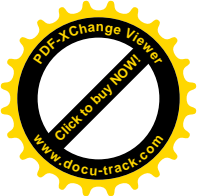
Reconnaissance

File profiling is essentially malware analysis reconnaissance, an effort necessary to gain enough information about the file specimen to render an informed and intelligent decision about what the file is, how it should be categorized or analyzed, and in turn,

The file profiling process entails an initial or cursory static analysis of the suspect code. *Static analysis* is the process of analyzing executable binary code without actually executing the file. *Dynamic* or *behavioral analysis* involves executing the code and monitoring its behavior, including its interaction and effect on the host system. Although these are two very different approaches to code analysis, most digital investigators implement both to ensure a more holistic or comprehensive analysis. Dynamic analysis of malicious code on Windows and Linux systems will be discussed in later chapters. For now, let's focus on static analysis, the core process component of file profiling.

A general approach to file profiling involves the following steps:

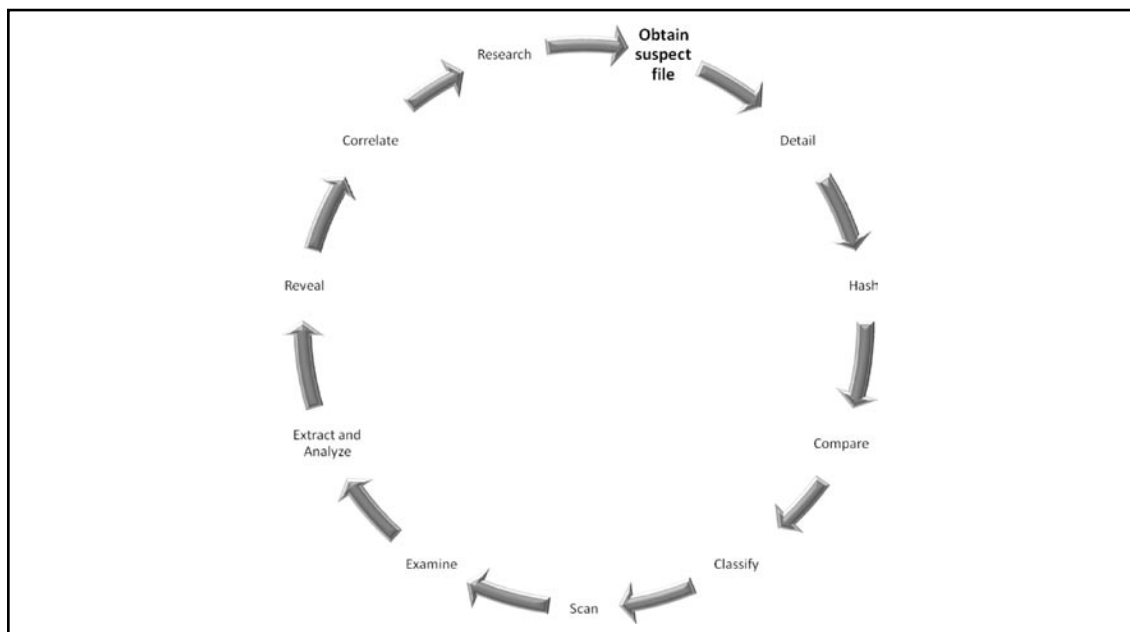
- **Detail** Identify and document system details pertaining to the system from which the suspect file was obtained.
- **Hash** Obtain a cryptographic hash value or “digital fingerprint” of the suspect file.
- **Compare** Conduct file similarity indexing of the file against known samples.

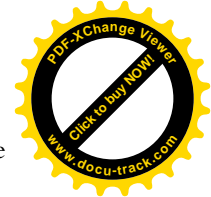


- **Classify** Identify and classify the type of file (including the file format and the target architecture/platform), the high level language used to author the code, and the compiler used to compile it.
- **Scan** Scan the suspect file with anti-virus and anti-spyware software to determine if the file has a known malicious code signature.
- **Examine** Examine the file with executable file analysis tools to ascertain whether the file has malware properties.
- **Extract and Analyze** Conduct entity extraction and analysis on the suspect file by reviewing any embedded American Standard Code for Information Interchange (ASCII) or Unicode strings contained within the file, and by identifying and reviewing any file metadata and symbolic information
- **Reveal** Identify any code obfuscation or *armoring* techniques protecting the file from examination, including packers, wrappers, or encryption.
- **Correlate** Determine whether the file is dynamically or statically linked, and identify whether the file has dependencies.
- **Research** Conduct online research relating to the information you gathered from the suspect file and determine whether the file has already been identified and analyzed by security consultants, or conversely, whether the file information is referenced on hacker or other nefarious Web sites, forums, or blogs.

Figure 7.3 graphically depicts the important components of the file profiling process.

Figure 7.3 Steps in the File Profiling Process





Although all of these steps are valuable ways to learn more about the suspect file, they may be executed in varying order or in modified form, depending upon the preexisting information or circumstances surrounding the code. Be thorough and flexible. As this phase of investigation consists primarily of a preliminary static analysis of the suspect file, the examination environment is not contingent upon any particular operating system. For purposes of this chapter, however, tools and techniques exclusive to a Windows environment are considered. Similar methodology will be followed in Chapter 8, “File Identification and Profiling: Initial Analysis of a Suspect File on a Linux System.” Note that a common middle ground is to conduct the examination on a Windows system in a Linux-like environment, using emulation software such as Cygwin,¹ WinAVR,² or MYSYS/MinGW.³

As each phase of the file profiling process is examined, numerous tools that will assist in conducting the analysis will be examined. Familiarity with a wide variety of both command-line interface (CLI) and Graphical User Interface (GUI) tools will further broaden the scope of investigative options. Inevitably, familiarity and comfort with a particular tool, or the extent to which the reliability or efficacy of a tool is perceived as superior, often dictate whether the tool is incorporated into any given common investigative arsenal.

Working with Executables

Before taking a closer look at the file profiling process, a brief discussion of the way in which source code is compiled, linked, and becomes executable seems appropriate. The steps an attacker takes in compiling malicious code will often determine the items of evidentiary significance discovered during its examination.

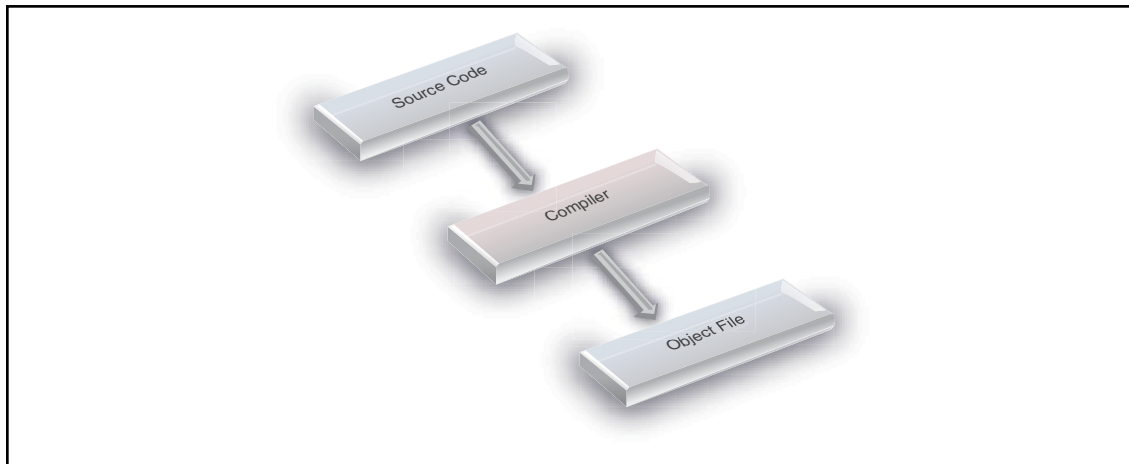
How an Executable File is Compiled

Think of the compilation of source code into an executable file like the metamorphosis of caterpillar to butterfly: the initial and final products manifest as two totally different entities, even though they are really one in the same but in different form. (See [Figure 7.4](#).)

¹ For more information about Cygwin, go to <http://www.cygwin.com/>.

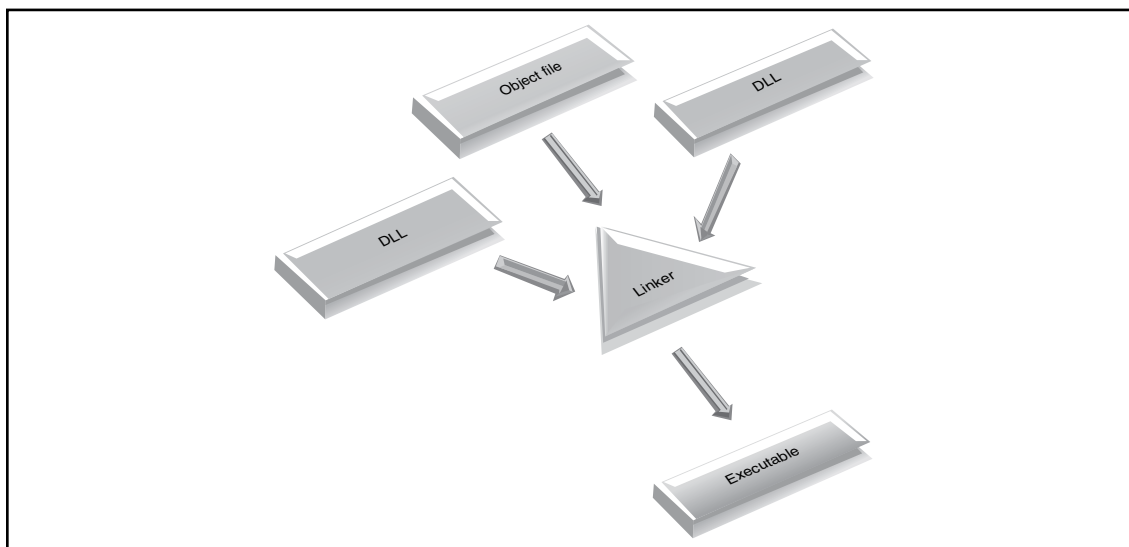
² For more information about WinAVR, go to <http://winavr.sourceforge.net/>.

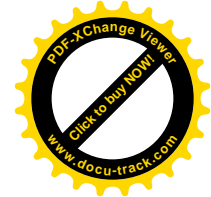
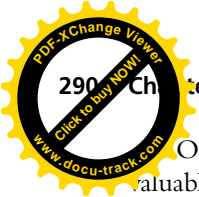
³ For more information on the Minimalist GNU for Windows and the Minimal SYStem, go to <http://www.mingw.org/>.

Figure 7.4 Compiling Source Code Into an Object File

As illustrated in [Figure 7.4](#) above, when a program is compiled, the program's source code is run through a *compiler*, a program that translates the programming statements written in a high-level language into another form. Once processed through the compiler, the source code is converted into an *object file* or machine code, as it contains a series of instructions not intended for human readability, but rather for execution by a computer processor.¹

After the source code is compiled into an object file, a *linker* assembles any required libraries with the object code to produce together an executable file that can be run on the host operating system, as seen in [Figure 7.5](#).

Figure 7.5 Linker Creation of an Executable File



Often, during compilation, bits of information are attached to the executable file that may be valuable to investigation. The amount of information present in the executable is often contingent upon how it was compiled by the attacker. Later in this chapter, the tools and techniques for unearthing these useful clues during the course of analysis will be discussed.

Static vs. Dynamic Linking

In addition to analysis of the information added to the executable during compilation, examination of the suspect program to determine whether it is a *static* or a *dynamic executable* will reveal clues about the contents and size of the file, and in turn, potentially enhance the scope of relevant discoverable evidence.

A *static executable* is compiled with all of the necessary libraries and code necessary to successfully execute, making the program “self-contained.” Conversely, *dynamic executables* are dependent upon shared libraries to successfully run. The required libraries and code needed by a dynamically linked executable are referred to as *dependencies*. In Windows programs, dependencies are most often dynamic link libraries, or DLLs (hence the .dll extension), that are imported from the host operating system during execution. By calling on the required DLLs at runtime, rather than statically linking them to the code, dynamically linked executables are smaller and consume less system memory. File dependencies in Windows executables reside in the import tables of the file structure. How to examine a suspect file to identify dependencies will be discussed later in this chapter. Import tables and file dependency analysis will be revisited and dealt with in greater detail in Chapter 9.

Symbolic and Debug Information

During the course of compiling executable binary, symbol files⁴ and debug information may be produced by the compiler and linker and are stored in debug files (.dbg) or program database files (.pdb) in the portable executable or PE file. Symbolic and debugging information often is used to troubleshoot and trace the execution of an executable image, such as to resolve program variables and function names.

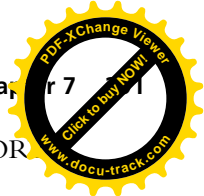
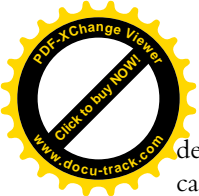
Generally, *symbolic information* can include:

- The names and addresses of all functions
- All data type, structure, and class definitions
- The names, data types, and addresses of global variables
- The names, data types, addresses, and scopes of local variables
- The line numbers in the source code that correspond to each binary instruction

Symbolic names are stored in the Portable Executable/Common Object File Format (PE/COFF) symbol table, the address of which is identified in the IMAGE_FILE_HEADER structure (COFF header format) *PointerToSymbolTable* field. Each symbol table entry contains certain information, including the symbol name, value, section number, type, and storage class.

Locating the debug information in a PE file (if it exists) is a bit more circuitous. The IMAGE_DEBUG_DIRECTORY (or simply the Debug Directory) is the structure that identifies whether

⁴ For more information about symbol files, go to <http://msdn2.microsoft.com/en-us/library/aa363368.aspx>.



debug information exists in the file, and where it is located. The `IMAGE_DEBUG_DIRECTORY` can be located anywhere within the structure of the PE file. The Debug Directory contains an abundance of often valuable information, including the time and date that the debugging information was created, the version number of debugging information format, and the type of existing debugging information. PE/COFF debugging information is identified by the `IMAGE_DEBUG_TYPE` value of 1.

Note that programmers often remove symbolic and debug information to reduce the size of the compiled executable. Moreover, attackers, now more than ever more cognizant of possible detection by researchers, system security specialists, and law enforcement, frequently take care to remove or “strip” their programs of symbolic and debug information. On a Linux platform, a simple run of the `strip` command against the binary file accomplishes this task. In Windows systems, although no `strip` utility is natively installed, parts of the programs in Cygwin, WinAVR, and MinGW nonetheless accomplish this. Moreover, to facilitate the removal of symbols from a binary file in lieu of `strip`, Microsoft developed *BinPlace*,ⁱⁱ a command-line tool available in the Debugging Tools for Windows suite.

Having discussed how an executable file is created, let's turn now to the first step of the file profiling process.

System Details

If the suspicious file was extracted or copied from a victim system, be certain to document the details obtained through the live response techniques mentioned in Chapter 1, including information about the system's operating system, version, service pack and patch level; the file system; and the full system path where the file resided prior to discovery. Further, details pertaining to any security software, including personal firewall, anti-virus, or anti-spyware programs, may prove valuable to subsequent analysis. Collectively, this information provides necessary file context, as malware often manifests differently depending on the permutations of the operating system and patch and software installation.

Hash Values

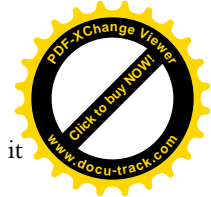
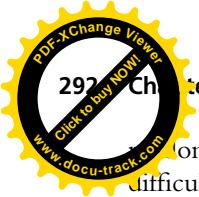
Generate a cryptographic hash value for the suspect file to both serve as a unique identifier or digital “fingerprint” for the file throughout the course of analysis, and share with other digital investigators who already may have encountered and analyzed the same specimen. The Message-Digest 5 (MD5)⁵ algorithm generates a 128-bit hash value based upon the file contents and typically is expressed in 32 hexadecimal characters. MD5 is widely considered the de facto standard for generating hash values for malicious executable identification, despite academic studies suggesting that the algorithm is susceptible to a hash collision vulnerability.⁶ Other algorithms, such as Secure Hash Algorithm Version 1.0 (SHA1)⁷ can be used for the same purpose.

Generating an MD5 hash of the malware specimen is particularly helpful for subsequent dynamic analysis of the code. Executing malicious code often causes it to remove itself from the location of execution and hide itself in a new, often non-standard location on the system. When this occurs, the malware changes file names and file properties (for instance, upon execution, the code assigns itself a

⁵ For more information on the MD5 algorithm, go to <http://www.faqs.org/rfcs/rfc1321.html>.

⁶ For more information about these studies, go to <http://www.mathstat.dal.ca/~selinger/md5collision/> and <http://th.informatik.uni-mannheim.de/People/lucks/HashCollisions/>.

⁷ For details and technical specifications pertaining to SHA1, go to <http://www.faqs.org/rfcs/rfc3174.html>.



from character name like “ahoekrlif.exe” that hides among other operating system files), making it difficult to detect and locate without a corresponding hash.

Other malware specimens upon execution engage in what is known as *process camouflaging*,ⁱⁱⁱ an anti-forensic technique wherein the code renames itself to appear as a legitimate or common process. For example, many Agobot malicious code variants rename themselves upon execution “lsass.exe,” a common operating system process in the Windows XP environment, often remaining unnoticed by an unsophisticated computer user who may only occasionally check the Windows Task Manager for anomalous processes.^{iv} Still others, upon execution of the malicious binary, may cause the malware to “phone home” and gain network connectivity, only to download additional malicious files and update itself. Such occurrences make having an MD5 hash value of the original specimen invaluable. Whether the file copies itself to a new location, extracts files from the original file, updates itself from a remote Web site, or simply camouflages itself through sound-alike renaming, comparison of MD5 values for each sample will enable determination of whether the samples are the same or new specimens that require independent analysis. There are a number of MD5 hashing tools available for accomplishing this task.

Command Line Interface (CLI) MD5 Tools

In the UNIX and Linux operating systems, the native CLI MD5 hashing utility is known as `md5sum`. Luckily for Windows users, there are a few versions of this utility ported to the Windows environment available for free (found at <http://www.weihenstephan.de/~syring/win32/win32.html> and another at <http://downloads.activestate.com/contrib/md5sum/Windows/>). Similarly, Microsoft has developed the File Checksum Integrity Verifier (FCIV),⁸ a command-line utility that computes MD5 or SHA1 cryptographic hashes for files. As an alternative to these tools, `md5deep`, a powerful MD5 hashing and analysis tool suite written by Jesse Kornblum, gives the user very granular control over the hashing options, including piecewise and recursive modes.⁹ In addition to the MD5 algorithm, the `md5deep` suite provides for alternative algorithms by providing additional utilities such as `sha1deep`, `tigerdeep`, `sha256deep`, and `whirlpooldeep`, all of which come included in the `md5deep` suite download.

GUI MD5 Tools

Despite the power and flexibility offered by these CLI MD5 tools, many digital investigators prefer to use GUI-based tools during analysis, because they provide drag-and-drop functionality and easy-to-read output. Similarly, tools that enable a Windows Explorer shell extension, or “right-click” hashing, provide a simple and efficient way to generate hash values during analysis. Here we discuss some notable GUI-based and shell extension MD5 tools.

Both the Malcode Analyst Pack (MAP)¹⁰ and HashOnClick tools offer hash calculation through Windows Explorer shell extensions. The MAP, a series of tools developed by iDefense Labs (owned by VeriSign, Inc.) to assist investigators with both static and dynamic malware analysis, provides simple, clean MD5 hash calculation upon right-clicking a target file. Hashonlick, developed by

⁸ For information on the availability and description of the FCIV, go to <http://support.microsoft.com/kb/841290>.

⁹ For more information about `md5deep`, go to <http://md5deep.sourceforge.net/>.

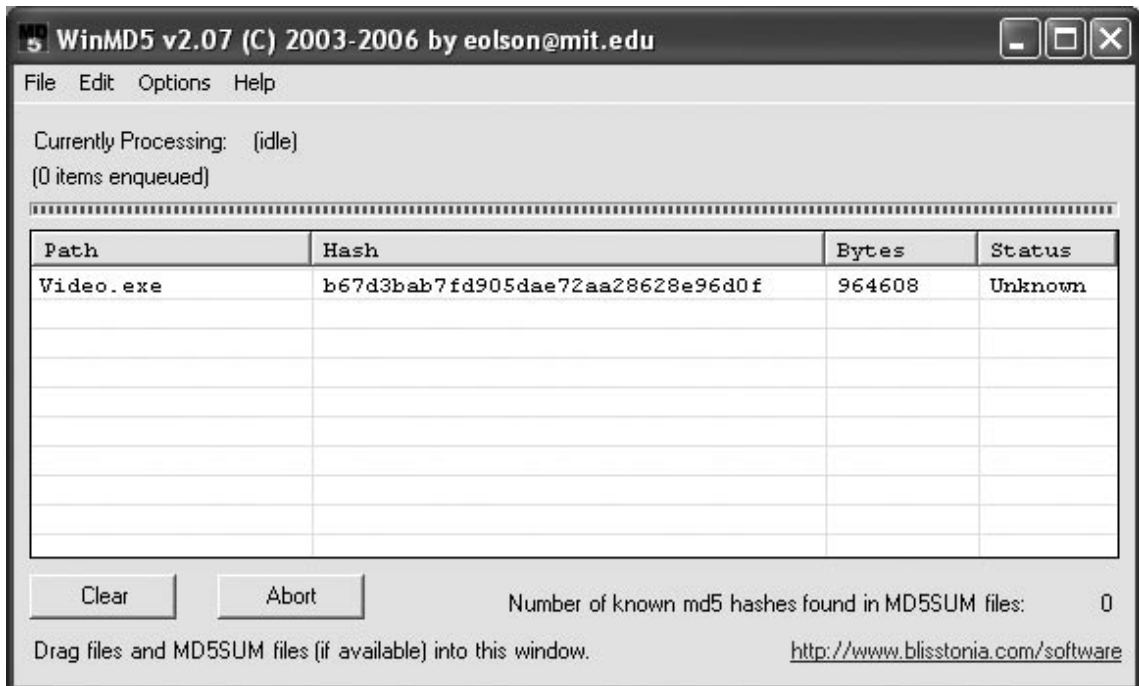
¹⁰ For more information about the Malcode Analyst Pack, go to http://labs.iddefense.com/software/malcode.php#more_malcode+analysis+pack.

2BrightSparks,¹¹ provides similar functionality, and offers the additional choices of calculating a hash value with either the SHA1 or CRC32 algorithms.

In addition, DiamondCS^v and Toast442.org^{vi} offer relatively lightweight and intuitive MD5 GUI hashing tools.

For more robust and flexible GUI-based MD5 hashing utilities, allowing for both drag-and-drop hashing of target files and folders and hash value comparison, WinMD5^{vii} (developed by Edwin Olson and pictured in Figure 7.6), Visual MD5^{viii} (developed by Protect Folder Plus Team), MD5 Fingerprint (developed by Ricardo Amaral), and Hash Quick (developed by Teddy Lindsey) are solid options. Note however, that some of these tools require installation of the .NET framework on the malware analysis machine. Querying our suspect file Video with WinMD5, learn that the hash value, as shown in Figure 7.6.

Figure 7.6 Files Being Processed in WinMD5

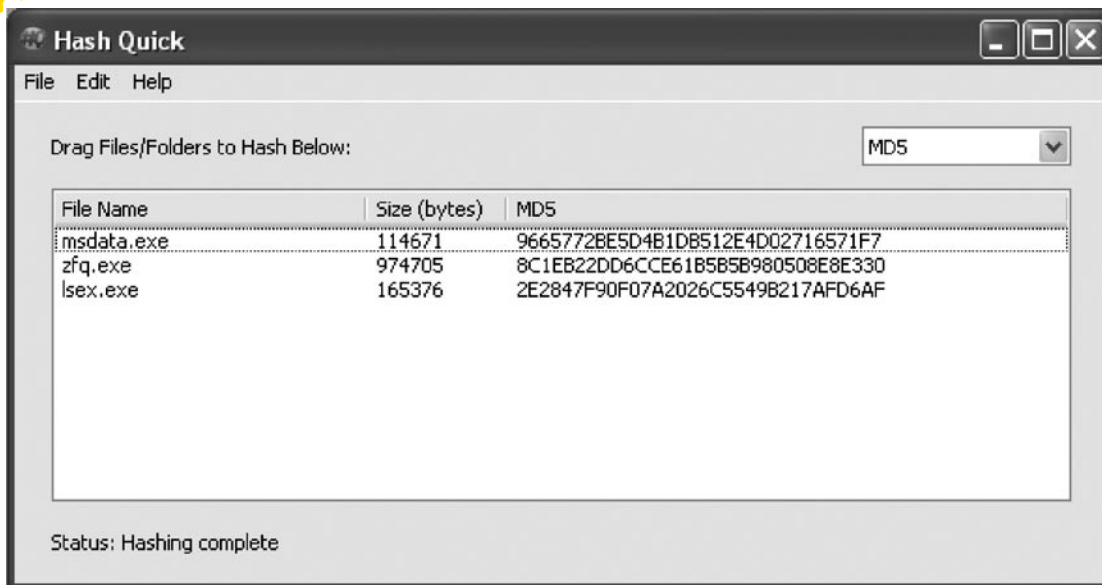


Like Jesse Kornblum's md5deep tool, some MD5 GUI tools allow batch and recursive hashing, functionality particularly helpful when examining or comparing multiples files, directories, or subdirectories. Hash Quick¹² provides this functionality with an intuitive user interface, as illustrated in Figure 7.7.

¹¹ For more information about HashonClick, go to <http://www.2brightsparks.com/onclick/hoc.html>.

¹² For more information about Hash Quick, go to <http://www.edgeintel.com/>; <http://www.lindseysystems.com/>.

Figure 7.7 Hashing Multiple Files in Hash Quick



File Similarity Indexing

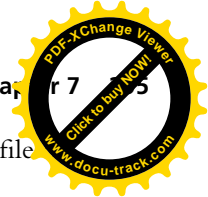
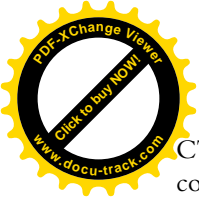
Comparing the suspect file to other malware specimens collected or maintained in a private or public repository for reference, is an important part of the file identification process. The easiest way to compare files for similarity is through a process known as *fuzzy hashing* or Context Triggered Piecewise Hashing (CTPH).^{ix}

Traditional hashing algorithms like MD5 and SHA1, generate a single checksum based upon the input or contents of the entire file. A single bit difference between files therefore, will render different hash values for two otherwise almost identical files. Whether a result of *file modification*, the intentional deletion, addition, or single-bit modification to known or otherwise identified malicious code to avoid ready detection, or because hackers often share or trade malware, thereby creating various permutations of “original” malware specimens, alternatives to MD5 and SHA1 must be implemented to identify homologous code and the functional similarities between them.^x

CTPH computes a series of randomly sized checksums for a file, allowing file association between files that are similar in file content but not identical. CTPH was first implemented in a spam e-mail detection tool, *spamsum*, developed by Dr. Andrew Triggell.^{xi,13} Through the application of CTPH, *spamsum* identifies e-mails that are similar, but not identical, to known samples of spam e-mail. Expanding on this concept, Jesse Kornblum developed *ssdeep*,¹⁴ a file hashing tool that utilizes

¹³ For more information about *spamsum*, go to <http://www.samba.org/ftp/unpacked/junkcode/spamsum/>.

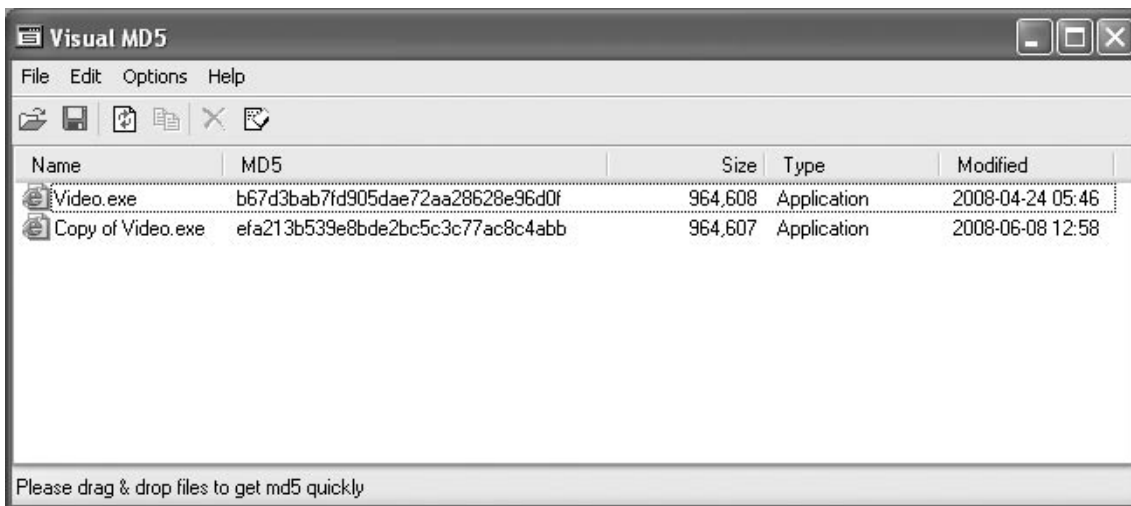
¹⁴ For more information about *ssdeep*, go to <http://ssdeep.sourceforge.net/>.



CTPH to identify homologous files. Ssdeep can be used to generate a unique hash value for a file to compare an unknown file against a known file or list of file hashes.

To demonstrate CTPH functionality, we modified our suspect file Video.exe by a single bit, saved the file, and renamed it “Copy of Video.exe”. We then hashed the two files with Visual MD5, as illustrated in [Figure 7.8](#). Despite being virtually identical files, the hash values of the files are radically different.

Figure 7.8 Single Bit File Modification Resulting in Different Hash Values



Name	MD5	Size	Type	Modified
Video.exe	b67d3bab7fd905dae72aa28628e96d0f	964,608	Application	2008-04-24 05:46
Copy of Video.exe	efa213b539e8bde2bc5c3c77ac8c4abb	964,607	Application	2008-06-08 12:58

Please drag & drop files to get md5 quickly

Examining the same two files using some of the modes available in ssdeep produce somewhat more useful results from a similarity index standpoint. As depicted in [Figure 7.9](#), the first employed mode creates a unique hash for each file and displays the full file path for the respective files:

Figure 7.9 First Employed ssdeep Mode

```
C:\Documents and Settings\Malware Lab\Desktop>ssdeep Video.exe "Copy of Video.exe"
```

```
ssdeep,1.0--blocksize:hash:hash,filename
```

```
1536:qHwOnbNQKLjWDyy1o5ReScJUEboopRrKKRqCK1:q1NQKPWDyDReScJltZrpRqCu,  
"C:\Documents and Settings\Malware Lab\Desktop\Video.exe"
```

```
1536:1HwOnbNQKLjWDyy1o5ReScJUEboopRrKKRqCK1:11NQKPWDyDReScJltZrpRqCu,  
"C:\Documents and Settings\Malware Lab\Desktop\Copy of Video.exe"
```

Chapter 7 • File Identification and Profiling: Initial Analysis

Notice that the ssdeep checksums are virtually identical, but for one value in each respective specimen's checksum (outlined in red boxes in [Figure 7.9](#) above).

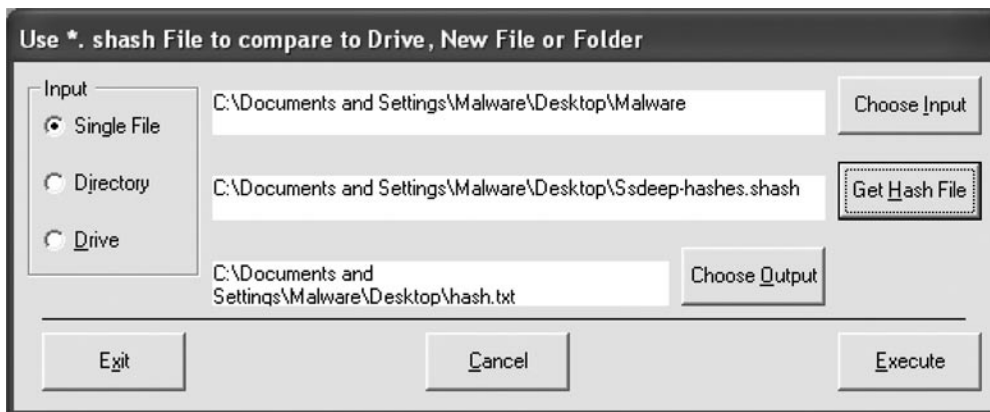
In addition, in the vast arsenal of ssdeep's file comparison modes exists a “pretty matching mode,” wherein a file is compared against another file and scored based upon similarity (a score of 100 constituting an identical match). In our test, the “pretty matching mode” assigned similarity scores of 99, as depicted in [Figure 7.10](#).

Figure 7.10 ssdeep “Pretty Matching Mode”

```
C:\Documents and Settings\Malware Lab\Desktop>ssdeep -pb Video.exe
"Copy of Video.exe"
Video.exe matches Copy of Video.exe (99)
Copy of Video.exe matches Video.exe (99)
```

Richard F McQuown of www.forensiczone.com has developed SSDeepFE,¹⁵ a slick GUI front-end for ssdeep, which allows for quick and efficient file hashing. SSDeepFE is particularly useful for comparing unknown files against a preexisting piecewise hash file list, as illustrated in [Figure 7.11](#).

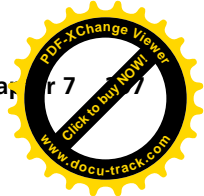
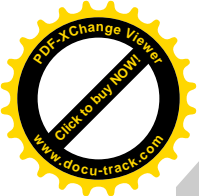
Figure 7.11 Using SSDeepFE



Through these and other similar tools employing the CTPH functionality, valuable information about a suspect file may be gathered during the file identification process to associate the suspect file with a particular specimen of malware, a “family” of code, or a particular attack or set of attacks.¹⁶

¹⁵ For more information about ssdeepFE, go to http://sourceforge.net/project/showfiles.php?group_id=215906&package_id=267714.

¹⁶ For additional resources pertaining to malware classification, see, Digital Genome Mapping: Advanced Binary Malware Analysis, http://dkbza.org/data/carrera_erdelyi_VB2004.pdf, and Automated Classification and Analysis of Internet Malware, http://www.eecs.umich.edu/~zmao/Papers/raid07_final.pdf.



NOTE

"All in the Family": Malware Classification

A number of studies have been conducted on malware classification and the categorization of malware into "families"; respective positions on the matter have been rather passionate.

Tony Lee, a member of the Microsoft Anti-malware team, proposed a behavior-based automated classification method for malware based on distance measure and machine learning. Mr. Lee's paper is available for download at <http://www.microsoft.com/downloads/details.aspx?FamilyId=7B5D8CC8-B336-4091-ABB5-2CC500A6C41A&displaylang=en>.

Conversely, Thomas Dullien of Zynamics (formerly SABRE Security) better known as "Halvar Flake," wrote a series of blog entries pertaining to his automated classification of malware using a static analysis technique incorporating IDA Pro, BinDiff2, and a phylogenetic clustering algorithm. Dullien's study can be found on <http://addxorrol.blogspot.com/2006/04/automated-classification-of-malware-is.html>, and <http://addxorrol.blogspot.com/2006/04/more-on-automated-malware.html>. Since Flake's study, SABRE now offers VxClass, an automated malware classification tool, available at <http://www.zynamics.com>.

In addition to Lee and Flake's research, Professor Arun Lakhotia, Director of the Software Research Lab, Center for Advanced Computer Studies, University of Louisiana at Lafayette, has co-authored numerous papers relating to malware phylogeny: including *Malware Phylogeny Generation Using Permutations of Code*, European Research Journal of Computer Virology, 2005, and *Malware Phylogeny Using Maximal Pi-Patterns*, EICAR Conference, 2005. Professor Lakhotia's papers are available on his Web site, http://www.cacs.louisiana.edu/labs/SRL/publications.html#REF_2005-jicv-karim-walenstein-lakhotia-parida.

File Signature Identification and Classification

After gathering system details, acquiring a digital fingerprint, and conducting a file index similarity inquiry, additional profiling to identify and classify the suspect file will prove an important part of any preliminary static analysis. This step in the file identification process often produces a clearer idea about the nature and purpose of the malware, and in turn, the type of damage the attack was intended to cause the victim system.

At this point in the file identification process, focus shifts to, among other things, identifying the *file type*; that is, determining the nature of the file from its file format or *signature* based upon available data contained within the file. File type analysis, coupled with *file classification*, or a determination of the native operating system and the architecture the code was intended for, are fundamental aspects of malware analysis that often dictate how and the direction in which your analytical and investigative methodology will unfold. If, for example, the suspect file is an executable and linking format (ELF)

any file, examination would be impractical on a Microsoft Windows XP system (unless the examiner is using virtualization software such as VMware to host a virtual Linux system) and would be better suited in a Linux environment with the techniques and tools more likely to reveal relevant behavioral and other characteristics of such a file.

File Types

The suspect file's extension cannot serve as the sole indicator of its contents; instead examination of the file's signature is paramount. A *file signature* is a unique sequence of identifying bytes written to a file's header. On a Windows system, a file signature is normally contained within the first 20 bytes of the file. Different file types have different file signatures; for example, a Windows Bitmap image file (.bmp extension) begins with the hexadecimal characters `42 4D` in the first 2 bytes of the file, characters that translate to the letters "BM." Most Windows-based malware specimens are executable files, often ending in the extensions .exe, .dll, .com, .pif, .drv, .qtx, .qts, .ocx, or .sys. The file signature for these files is "MZ," or the hexadecimal characters `4D 5A`, found in the first 2 bytes of the file. Humorously, the letters "MZ" are the initials for Mark Zbikowski, one of the principal architects of MS-DOS and the Windows/DOS executable file format.

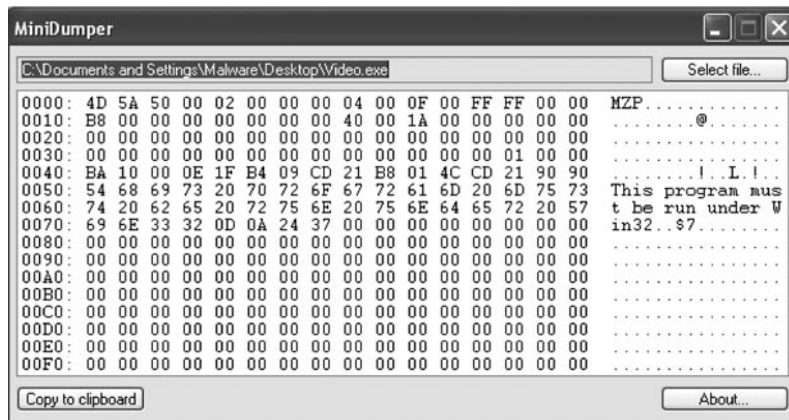
Analysis Tip

File Camouflaging

In conducting digital investigations, never presume that a file extension is an accurate representation. *File camouflaging*, or technique that obfuscates the true nature of a file by changing and hiding file extensions in locations with similar real file types, is a trick commonly used by hackers and bot herders to avoid detection of malicious code distribution.

Generally, there are two ways to identify a file's signature. First, query the file with a file identification tool. Second, open and inspect the file in a hexadecimal viewer or editor. Hexidecimal (or hex, as it is commonly referred) is a numeral system with a base of 16, written with the letters A–F and numbers 0–9 to represent the decimal values 0 to 15. In computing, hexadecimal is used to represent a byte as 2 hexadecimal characters thereby translating binary code into a human-readable format.

By viewing a file in a hex editor, every byte of the file is visible, assuming its contents are not obfuscated by packing, encryption, or compression. MiniDumper^{xii} by Marco Pontello is a convenient tool for examining a file in hexadecimal format, as it displays a dump of the file header only, as illustrated in our test of the "Hot New Video" suspect file Video, illustrated in [Figure 7.12](#).

Figure 7.12 Dumping a Suspect Executable File in MiniDumper

Other hexadecimal viewers for Windows provide additional functionality to achieve a more granular analysis of a file, including strings identification, hash value computation, and multiple file comparison. Such viewers include BreakPoint Software's Hex Workshop¹⁷ and WinHex, developed by X-Ways Software.^{xiii}

Online Resources

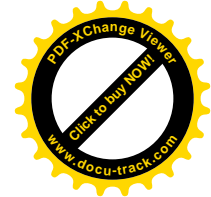
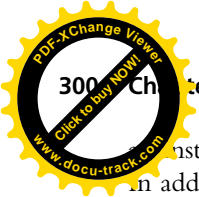
File Formats

- File Signatures Table: http://www.garykessler.net/library/file_sigs.html
- File Extensions: <http://www.fileinfo.net/>
<http://filext.com/>
<http://www.file-extensions.org/>
<http://www.dotwhat.net/>
<http://file-extension.net/seeker/>

File Signature Identification and Classification Tools

Most distributions of the Linux operating system come with the utility *file* preinstalled. The *file* command classifies a queried file specimen based on the data contained in the file as compared

¹⁷ For more information about HexWorkshop, go to www.bpsoft.com.



test the `/etc/magic` file. The *magic* file contains a comprehensive list of known file headers. In addition to identifying file type, the *file* command also provides other valuable information about the file, which is discussed later in this chapter.

Unfortunately, there is no inherent equivalent of the *file* command in Microsoft Windows operating systems. There is a Windows port of *file* available at (<http://gnuwin32.sourceforge.net/packages/file.htm>), and a similar tool, *exetype.exe*, which Microsoft developed and made available in the Microsoft Windows 98 Resource Kit¹⁸ and later Windows NT Resource Kits,¹⁹ but the tool does not recognize as many file types as *file*. Despite this apparent void in this genre of analytical tools, there are a number of CLI and GUI tools that have been developed to address file identification and analysis for Windows systems.

CLI File Identification Tools

Perhaps the closest tool to the Linux version of *file* is File Identifier (version 0.6.1 at the time of this writing), developed by Optima SC.²⁰ Similar to *file*, File Identifier compares a queried file against a *magic*-like database file.²¹ In addition to conducting file identification through signature matching, File Identifier also extracts file metadata, as illustrated in our test of the “Hot New Video” suspect file Video, depicted in Figure 7.13.

Figure 7.13 File Identifier Metadata Extraction

```
C:\Documents and Settings\Malware Lab\Desktop>file Video.exe
File identify [Freeware] Version 0.6.1 Copyright (c) Optima SC Inc. 2002-2006
Video.exe      [exe] Windows NT portable executable file, w/Symbol info
               [info] file class : code
               [info] file path  : C:\Documents and Settings\Malware Lab\Desktop\
1/1 files identified
100.00 % found.
0 seconds
```

In addition to providing a variety of different file scanning modes, including a recursive mode for applying the tool against directories and subdirectories of files, File Identifier also offers Hypertext Markup Language (HTML) and CVS report generation.

The CLI file signature and analysis tool GT2,²² developed by Philip Helger (also known as PHaX), is the latest and arguably the best of a long lineage of file format detection utilities that Helger has released.²³ In addition to identifying an unknown binary’s file format, GT2 details the file’s target operating system and architecture, file resources, dependencies, and metadata, as illustrated in Figure 7.14 (output modified for brevity):

¹⁸ <http://support.microsoft.com/kb/247024>.

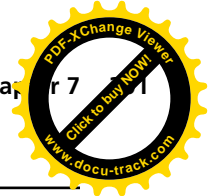
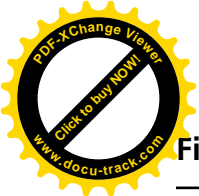
¹⁹ <http://www.microsoft.com/resources/documentation/windowsnt/4/server/reskit/en-us/reskt4u4/rku4list.mspx?mfr=true>.

²⁰ For more information about the File Identifier tool, go to <http://www.optimasc.com/products/fileid/index.html>.

²¹ For more information about the Optima SC magic file, go to <http://www.optimasc.com/products/fileid/magic-format.pdf> and www.magicdb.org.

²² For more information about GT2, go to <http://philip.helger.com/gt/program.php?tool=gt2>.

²³ For more about Philip Helger’s programs, including discontinued programs, go to <http://philip.helger.com/gt/program.php>.

**Figure 7.14** GT2 File Format Detection Utility Output

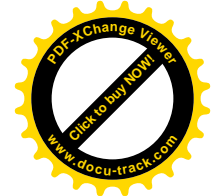
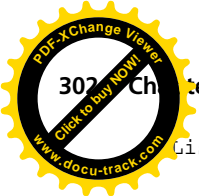
```
gt2 0.34 (c) 1999-2004 by PHaX (coding@helger.com)

- C:\Documents and Settings\Malware Lab\Desktop\Video.exe (964608 bytes)
- binary

Is a DOS executable
Size of header:      00000040h/64 bytes
File size in header: 00000250h/592 bytes
Entrypoint:         00000040h/64
Overlay size:       000EB5B0h/964016 bytes
No relocation entries

PE EXE at offset 00000100h/256
Entrypoint:         000E3E01h / 933377
Entrypoint RVA:     00C9E001h
Entrypoint section: '.aspack'
Calculated PE EXE size: 000EB800h / 964608 bytes
Image base:        00400000h
Required CPU type: 80386
Required OS:       4.00 - Win 95 or NT 4
Subsystem:         Windows GUI
Linker version:    2.25
Stack reserve:     00100000h / 1048576
Stack commit:      00004000h / 16384
Heap reserve:      00100000h / 1048576
Heap commit:       00001000h / 4096
Flags:
  File is executable
  Line numbers stripped from file
  Local symbols stripped from file
  Little endian
  Machine based on 32-bit-word architecture
  Big endian

Sections according to section table (section align: 00001000h):
Name      RVA      Virt size  Phys offs  Phys size  Phys end  Flags
CODE      00001000h  000DC000h  00000400h  0004F200h  0004F600h  C0000040h
DATA      000DD000h  00003000h  0004F600h  00001600h  00050C00h  C0000040h
BSS       000E0000h  00002000h  00050C00h  00000000h  00050C00h  C0000040h
.idata    000E2000h  00003000h  00050C00h  00001200h  00051E00h  C0000040h
.tls      000E5000h  00001000h  00051E00h  00000000h  00051E00h  C0000040h
.rdata    000E6000h  00001000h  00051E00h  00000200h  00052000h  C0000040h
.reloc    000E7000h  0000F000h  00052000h  00000000h  00052000h  C0000040h
.rsrc     000F6000h  00BA8000h  00052000h  00091E00h  000E3E00h  C0000040h
.aspack   00C9E000h  00008000h  000E3E00h  00007A00h  000EB800h  C0000040h
.adata    00CA6000h  00001000h  000EB800h  00000000h  000EB800h  C0000040h
```



Chapter 7 • File Identification and Profiling: Initial Analysis

Listing of all used data directory entries (used: 4, total: 16):

	Name	Phys offs	RVA	Phys size	Section
	Import Table	000E4DACH	00C9EFACCh	00000498h	.aspack
	Ressource Table	00052000h	000F6000h	00BA7C00h	.rsrc
	Base relocation Table	000E4D54h	00C9EF54h	00000008h	.aspack
	TLS Table	000E4D3Ch	00C9EF3Ch	00000018h	.aspack

Functions from the following DLLs are imported:

```
[0] kernel32.dll
[1] user32.dll
[2] advapi32.dll
[3] oleaut32.dll
[4] advapi32.dll
[5] version.dll
[6] gdi32.dll
[7] user32.dll
[8] ole32.dll
[9] oleaut32.dll
[10] ole32.dll
[11] oleaut32.dll
[12] comctl32.dll
[13] shell32.dll
[14] wininet.dll
[15] urlmon.dll
[16] shell32.dll
[17] comdlg32.dll
[18] shlwapi.dll
[19] user32.dll
```

Icon Group:

ID: 80001040h/2147487808

RVA: 00C9F6E4h; Offset: 000E54E4h; Size: 132 bytes

Version Info:

ID: 00000001h/1

RVA: 00C9F444h; Offset: 000E5244h; Size: 672 bytes

VersionInfo resource:

FileVersion: 1.0.0.0

ProductVersion: 1.0.0.0

Target OS: 32 bit Windows

Language '041604E4'

CompanyName: 'Primo'

FileDescription: ''

FileVersion: '1.0.0.0'

InternalName: ''

LegalCopyright: ''

LegalTrademarks: ''

OriginalFilename: ''

ProductName: ''

ProductVersion: '1.0.0.0'

Comments: 'Registrado P. Primo'

Total resource size: 12220567 bytes (data: 12216831 bytes, TOC: 3736 bytes

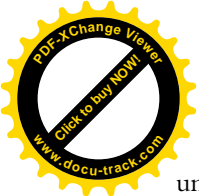
)

TLS at offset 000E4D3Ch (RVA 00C9EF3Ch) for 24 bytes

1 TLS directory entries

Processed with:

Found packer 'ASPack 2.12'



TrID,²⁴ a CLI file identifier written by Marco Pontello, does not limit the classification of an unknown file to one possible file type based on the file's signature, unlike other similar tools. Rather, it compares the unknown file against a file signature database and provides a series of possible results, ranked by order or probability, as depicted in the analysis of the *Video* suspect file in Figure 7.15.

Figure 7.15 TrID Probability Ranking

```
C:\Documents and Settings\Malware\Desktop>trid Video.exe
TrID/32 - File Identifier v2.00 - (C) 2003-06 By M.Pontello
Definitions found: 3256
Analyzing...

Collecting data from file: C:\Documents and Settings\Malware Lab\Desktop\Video.exe
90.1% (.EXE) ASPack compressed Win32 Executable (generic) (133819/79/30)
 5.7% (.EXE) Win32 Executable Generic (8527/13/3)
 1.3% (.EXE) Win16/32 Executable Delphi generic (2072/23)
 1.3% (.EXE) Generic Win/DOS Executable (2002/3)
 1.3% (.EXE) DOS Executable Generic (2000/1)
```

The TrID file database consists of approximately 3,400 different file signatures,²⁵ and is constantly expanding, due in part to Pontello's distribution of TrIDScan, a TrID counterpart tool that offers the ability to easily create new file signatures that can be incorporated into the TrID file signature database.²⁶

OTHER FILE ANALYZING TOOLS TO CONSIDER

Filetype v. 0.1.3	http://sourceforge.net/project/showfiles.php?group_id=23617&package_id=163264
Infoexe v. 1.32	http://www.exetools.com/file-analyzers.htm
Peace v. 1.00	http://www.exetools.com/file-analyzers.htm
Fileinfo v. 2.43	http://www.exetools.com/file-analyzers.htm

GUI File Identification Tools

There are a number of GUI-based file identification and classification programs for use in the Windows environment; many are intuitive to use and convenient for an initial static analysis of any suspect file.

Marco Pontello developed TrIDNet,²⁷ a GUI version of TrID, as shown in Figure 7.16. Like the CLI version, TrIDNet compares the suspect file against a file database of nearly 3,400 file signatures, scores the queried file based upon its characteristics, and reveals *Video*, a probability-based identification

²⁴ For more information about TrID, go to <http://mark0.net/soft-trid-e.html>.

²⁵ For a list of the file signatures and definitions, go to <http://mark0.net/soft-trid-deflist.html>.

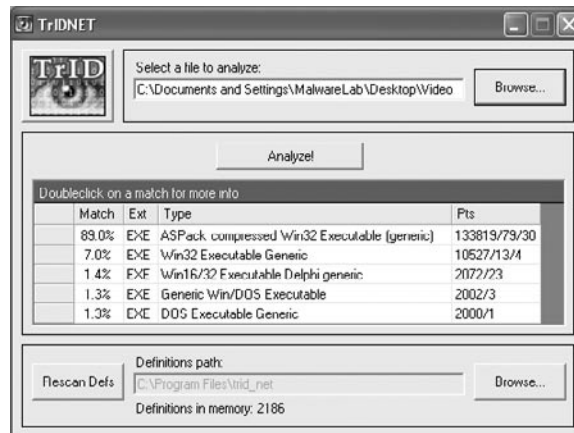
²⁶ For more information about TrIDScan, go to <http://mark0.net/soft-tridscan-e.html>.

²⁷ For more information about TrIDNet, go to <http://mark0.net/soft-tridnet-e.html>.

Chapter 7 • File Identification and Profiling: Initial Analysis

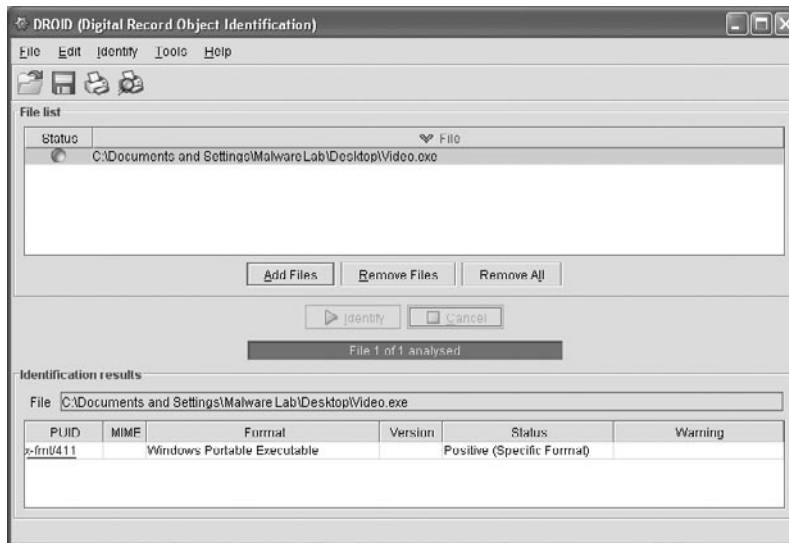
the file. The tool identified our suspect file `video` as an executable binary for Microsoft operating systems. Further the file is identified as being compressed with ASPack, the significance of which we will discuss later in this chapter.

Figure 7.16 Video.exe Classified in TrIDNet



The Digital Record Object Identifier (DROID)²⁸ is a GUI tool with similar functionality to TrIDNet. Developed by the British National Archives, Digital Preservation Department, as part of its PRONOM technical registry project,²⁹ DROID performs automated batch identification of file formats. As shown in Figure 7.17, DROID also identified our suspect file as a Windows executable binary.

Figure 7.17 DROID Identifies the Suspect File



²⁸ For more information about DROID, go to <http://www.nationalarchives.gov.uk/aboutapps/PRONOM/tools.htm> and for tool download, go to <http://droid.sourceforge.net/wiki/index.php/Introduction>.

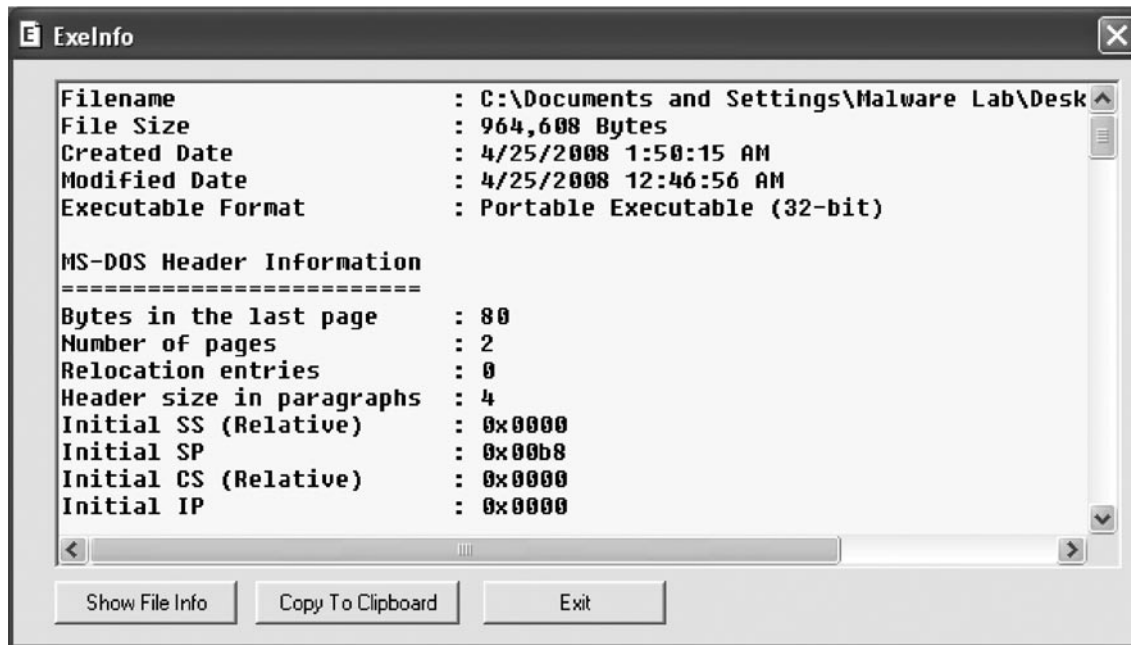
²⁹ <http://www.nationalarchives.gov.uk/pronom>.

A less robust alternative to DROID is Andrew J Glina's beta software, WhatFile,³⁰ a file identification extracting tool that can identify up to 20 files types.

Another useful GUI-based utility for file identification and analysis is FileAlyzer,³¹ a freeware tool developed by Patrick Kolla of Safer-Networking.com, which allows for basic file analysis, including type identification, hash value, properties, contents, and structure. A multipurpose tool, FileAlyzer also serves as a hex viewer, strings extractor, and PE file viewer.

At this point, inspecting our suspect file with numerous file identification tools reveals that video is likely a Windows executable binary file. Additional profiling efforts at this point might include the collection of basic executable file information, a necessary component of the any cursory extraction analysis (as opposed to the full-fledged analysis of executable file structure and contents discussed in later sections of this chapter). A great drag-and-drop GUI tool for obtaining these details, including .dlls and driver files, is Nirsoft's Exeinfo.³² Simply drag a suspect file into the interface and the tool will query the file and print the results within the interface, as illustrated in Figure 7.18. In addition to identifying the file type, Exeinfo presents basic executable structure details, Created and Modified dates and times, and file metadata, if available.

Figure 7.18 Nirsoft's Exeinfo Tool Examination of video.exe



³⁰ For more information about WhatFile, go to <http://www.sinnercomputing.com/det.php?prog=WhatFile>.

³¹ For more information about Filealyzer, go to <http://www.safer-networking.org/en/filealyzer/index.html>.

³² For more information about Exeinfo, go to <http://nirsoft.mirrorz.com>.

Other Tools to Consider: *Miss Identify*

Written by Jesse Kornblum, *Miss Identify* is a utility for finding Win32 executable programs, regardless of file extension (<http://missidentify.sourceforge.net/>). This is particularly helpful for malware analysis wherein the attacker is trying to conceal his malicious programs by using pseudo extensions in an effort to trick victims into executing the malicious program, particularly when the victims have the Windows “Hide Extensions for known file types” option for when folder options is applied. The utility is also useful in detecting misnamed executable files hidden on a hard drive. In the example below, the files appeared to have benign file extensions in Windows Explorer:

```
C:\Documents and Settings\Malware Lab\>missidentify.exe -ar "c:\Documents and
Settings\Malware Lab\Desktop\Malcode"
c:\Documents and Settings\Malware Lab\Malcode\lsex.jpg.exe
c:\Documents and Settings\Malware Lab\Malcode\msdata.doc.exe
c:\Documents and Settings\Malware Lab\Malcode\zfq.bmp.exe
```

Anti-virus Signatures

After identifying and classifying a suspect file, the next step in the file profiling process is to query the file against anti-virus engines to see if it is detected as malicious code. Approach this phase of the analysis in two separate steps. First, manually scan the file with a number of anti-virus programs locally installed on the malware analysis test system, to determine whether any alerts are generated for the file. This manual step affords control over the configuration of each program, ensures that the signature database is up-to-date, and allows access to the additional features of locally installed anti-virus tools (like links to the vendor Web site), which may provide more complete technical details about a detected specimen. Second, submit the specimen to a number of free online malware scanning services for a more comprehensive view of any signatures associated with the file.

Local Malware Scanning

To scan malware locally, implement anti-virus software that can be configured to scan on demand, as opposed to every time a file is placed on the test system. Also make sure that the AV program affords choice in resolving malicious code detected by the anti-virus program; many automatically delete, “repair,” or quarantine the malware upon detection. Some examples of freeware anti-virus software for installation on your local test system include ClamWin³³ Avira AntiVir³⁴ and Grisoft AVG.³⁵

³³ For more information about ClamWin free anti-virus, go to <http://www.clamwin.com>.

³⁴ For more information about Avira AntiVir, go to <http://www.free-av.com/>.

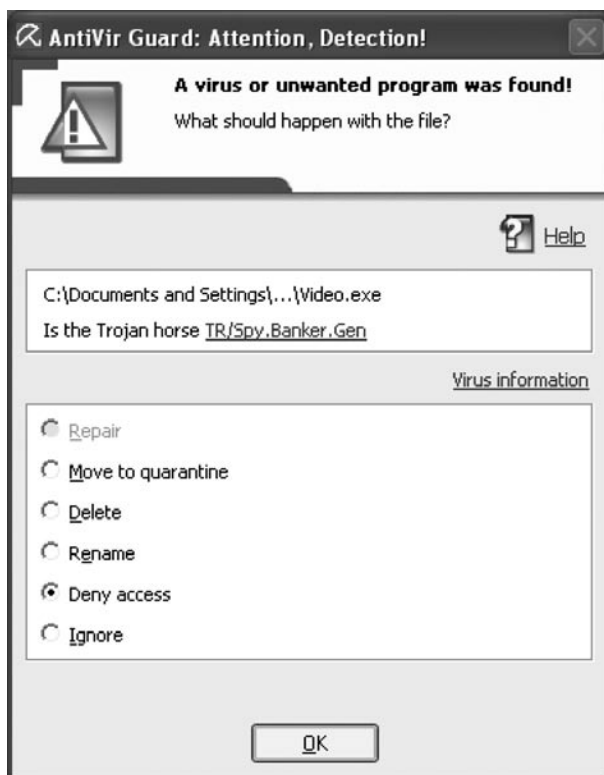
³⁵ For more information about Grisoft AVG, go to <http://free.grisoft.com/doc/5390/us/frt/0?prd=aff>.

Well understanding the machinations of how anti-virus products work and what they scan for in a file to identify it as malicious, most attackers take great care in protecting malicious files by compressing, packing, encrypting, or otherwise obfuscating their contents to ensure the files cannot be identified by anti-virus software. As such, the fact that installed anti-virus software does not identify the suspect file as malicious code, does not mean it is not. Rather, it may mean simply that a signature for the suspect file has not been generated by the vendor of the anti-virus product, or that the attacker is “armoring” or otherwise implanting a file protecting mechanism to thwart detection.

Even though the attacker in our “Hot New Video” scenario seemingly defeated the victim’s anti-virus software, the suspect file `Video` can nonetheless be scanned both locally and online to learn more about the file from any existing signature for it.

Scanning `Video` through Avira AntiVir, as depicted in [Figure 7.19](#), reveals identification by the signature `TR/Spy.Banker.Gen`, suggesting that our suspect file contains Trojan horse functionality that may relate to banks or banking. Although the signature does not necessarily dictate the nature and capability of the program, it does shed potential insight into the purpose of the program.

Figure 7.19 Results of Running AntiVir Against `Video.exe`



Given that when a malicious code specimen is obtained and when a signature is developed for it may vary between anti-virus companies, scanning a suspect file with multiple anti-virus engines is recommended. Implementing this redundant approach helps ensure that a malware specimen is identified by an existing virus signature and provides a broader, more thorough inspection of the file. In this

Since, however, querying Video through ClamWin, as depicted in Figure 7.20, does not generate a signature match. We can further investigate whether the suspect file matches known virus signatures by submitting the file to Web-based Malware Scanning Services.

Figure 7.20 Results of Running ClamWin Against Video



Web-based Malware Scanning Services

After running a suspect file through local anti-virus program engines, consider submitting the malware specimen to an online malware scanning service. Unlike vendor-specific malware specimen submission Web sites, VirusTotal,³⁶ Jotti Online Malware Scanner,³⁷ and VirScan³⁸ will scan submitted specimens against numerous anti-virus engines to identify whether the submitted specimen is detected as hostile code. During the course of inspecting the file, the scan results for the respective anti-virus engines are presented in real-time on the Web page. These Web sites are distinct from online malware analysis sandboxes that execute and process the malware in an emulated Internet, or “sandboxed” network. The use of online malware sandboxes will be addressed later in Chapter 9. In the meantime, remember that submission of any specimen containing personal, sensitive, proprietary, or otherwise confidential information may violate the victim company’s corporate policies or otherwise offend the ownership,

³⁶ For more information about VirusTotal, go to <http://www.virustotal.com/>.

³⁷ For more information about Jotti Online Malware Scanner, go to <http://virusscan.jotti.org/>.

³⁸ For more information about VirScan, go to www.virscan.org.

privacy, or other corporate or individual rights associated with that information. Be careful to seek appropriate legal guidance in this regard, before releasing any such specimen for third-party examination.

Assuming you have determined it is appropriate to do so, submit the suspect file by uploading the file through the Web site submission portal, as illustrated in Figures 7.21 and 7.22.

Figure 7.21 Submitting a File to VirusTotal for Inspection

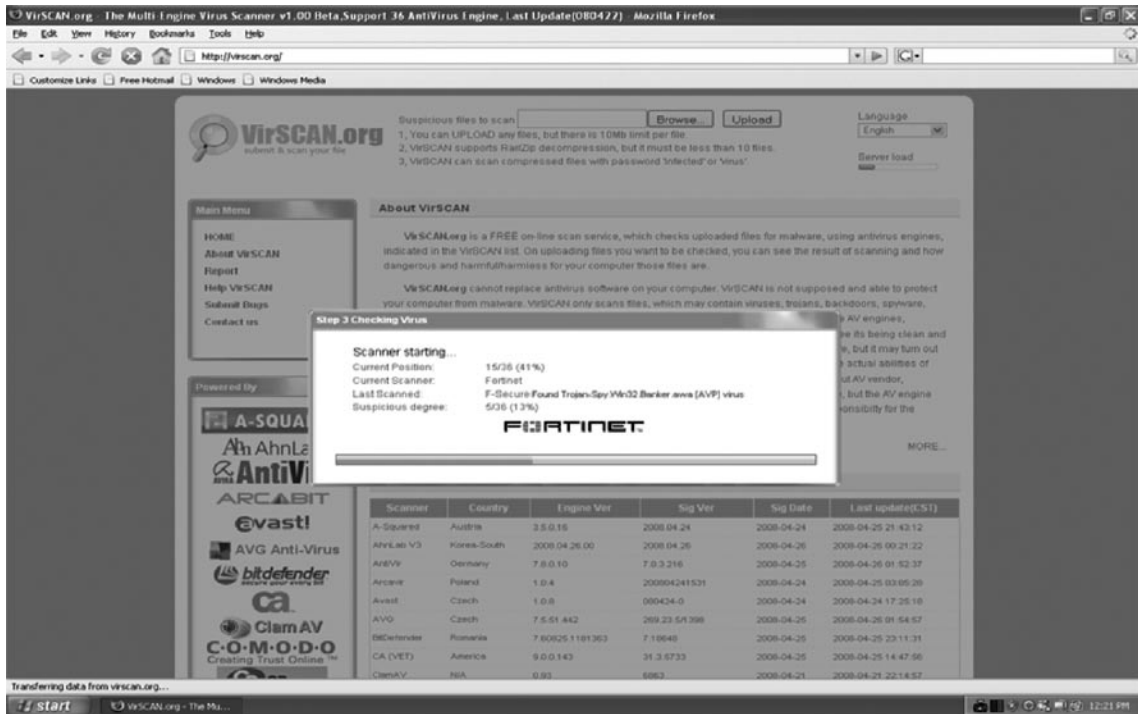


Figure 7.22 Submitting a File to VirScan for Inspection



Upon submission, the anti-virus engines will run against the suspect file. As each engine passes over the submitted specimen, the file may be identified, as manifested by a signature identification alert similar to that depicted in [Figure 7.23](#).

Figure 7.23 F-Secure AV Engine Identifies the Suspect File During the Course of a VirScan Specimen Scan



If the file is not identified by any anti-virus engine, the field next to the respective anti-virus software company will either remain blank (in the case of VirusTotal and VirScan), or state that no malicious code was detected (in the case of Jotti Online Malware Scanner), as illustrated in [Figures 7.24 through 7.26](#).

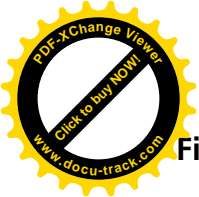


Figure 7.24 VirusTotal Results After Scanning Suspect File Video.exe



Figure 7.25 VirScan Results After Scanning Suspect File Video.exe

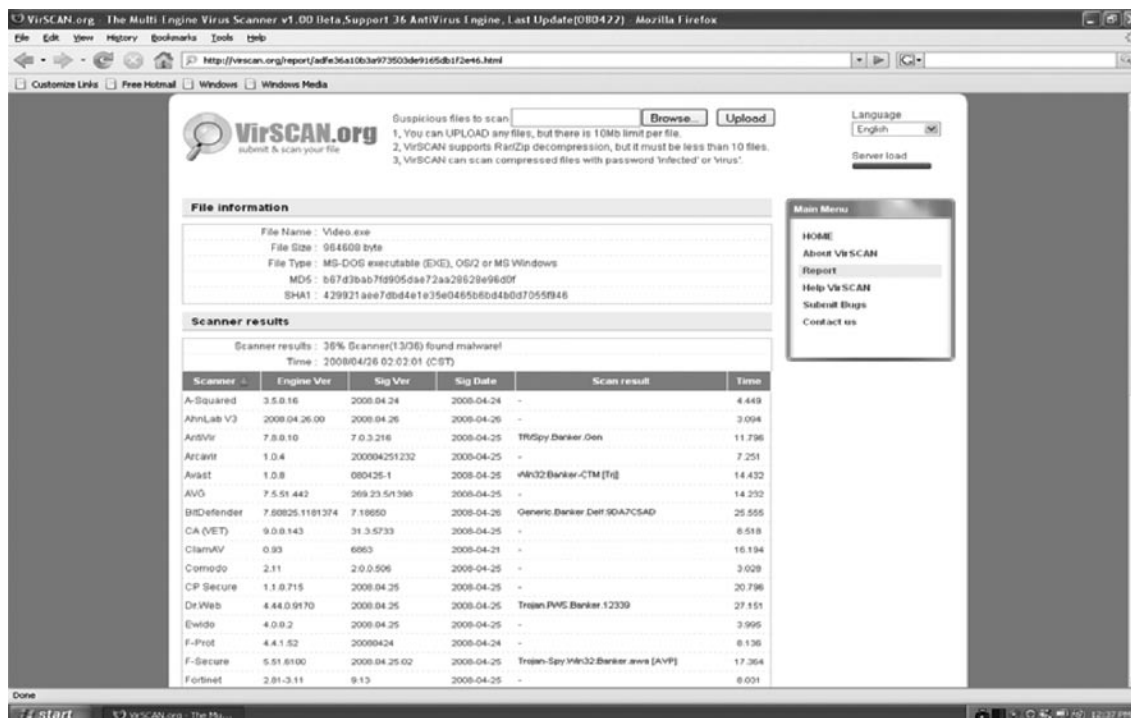
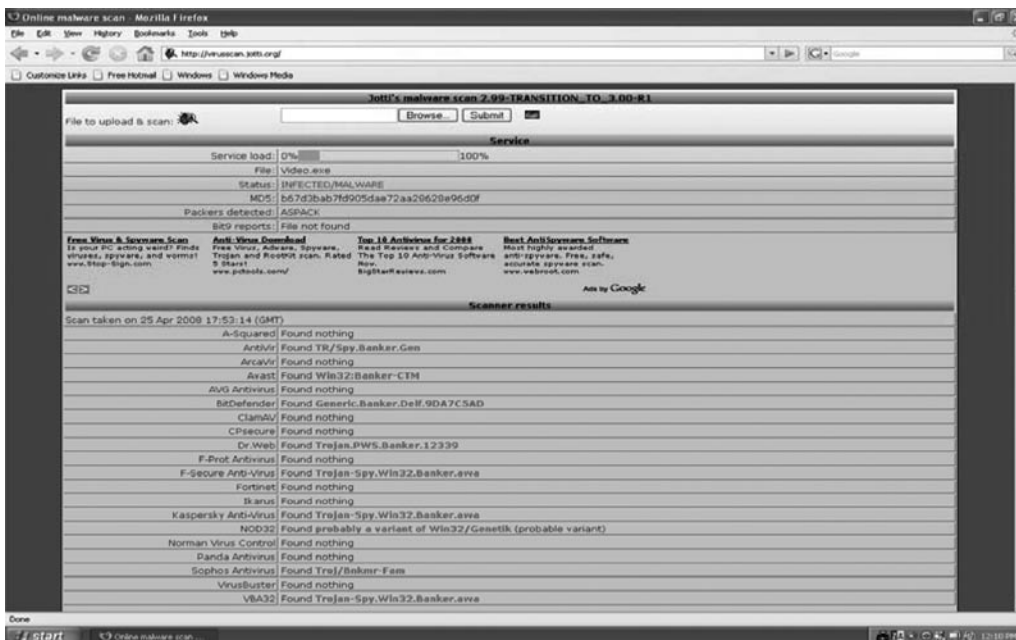


Figure 7.26 Jotti Results After Scanning Suspect File Video.exe



Scanning the suspect file through numerous anti-virus engines revealed that a number of malicious code signatures exist for the file. What next? The signature names attributed to the file provide an excellent way to gain additional information about what the file is and what it is capable of. By visiting the respective anti-virus vendor Web sites and searching for the signature or the offending file name, more often than not a technical summary of the malware specimen can be located. Alternatively, through search engine queries of the anti-virus signature, hash value, or file name, information security-related Web site descriptions or blogs describing a researcher's analysis of the hostile program also may be encountered. Such information may vastly contribute to the discovery of additional investigative leads and potentially reduce analysis time on the specimen. Conversely, there is no better way to get a sense of your malicious code specimen than thoroughly analyzing it yourself; relying entirely on third-party analysis to resolve a malicious code incident often has practical and real-world limitations.

Online Resources

Submitting Samples to Anti-Virus Vendors

All anti-virus companies accept submissions of suspicious file specimens for analysis. Most offer an online submission portal that allow direct upload of the suspect file.

Continued

Others require submission of a password-protected file within a compressed archive file that is also password-protected. Sometimes the scan is conducted and the results are reported live. Other vendors require a valid e-mail address to receive the results electronically. Below are the submission addresses for a number of AV companies:

Arcabit:	www.arcabit.com/send.html
A-Squared:	www.emsisoft.com/en/support/contact/
Avast:	http://onlinescan.avast.com/
AVG:	virus@grisoft.com
Avira/ Antivir:	http://analysis.avira.com/samples/index.php
BitDefender:	www.bitdefender.com/scan8/ie.html
ClamAV:	www.clamwin.com/content/view/89/85/
Computer Associates:	http://ca.com/us/securityadvisor/virusinfo/scan.aspx
Ewido:	www.ewido.net/en/onlinescan/
F-Prot:	www.f-prot.com/virusinfo/submission_form.html
F-Secure:	http://support.f-secure.com/enu/home/virusproblem/sample/
Fortinet:	www.fortiguardcenter.com/antivirus/virus_scanner.html
Kaspersky:	www.kaspersky.com/scanforvirus
IKARUS:	analyse@ikarus.at
McAfee:	www.webimmune.net http://vil.nai.com/vil/submit-sample.aspx
Microsoft:	www.microsoft.com/security/portal/
Norman Antivirus:	www.norman.com/microsites/nsic/Submit/en-us/
PandaSoftware:	virus@pandasoftware.com
Rising Antivirus:	http://sample.rising-global.com/webmail/upload_en.htm
Sophos:	www.sophos.com/support/samples/
Sunbelt Software:	http://research.sunbelt-software.com/Submit.aspx
Symantec:	www.symantec.com/enterprise/security_response/submit-samples.jsp
Virus Buster:	www.virusbuster.hu/en/support/contact/redirect_virus