Data Hiding

By Clayton Hoskinson, CFE, CFCE, and Jim Sleezer, CISA

The stories of end users who thought they erased a file only to learn that a computer specialist was able to make a full recovery are endless. Neither the file nor the data were really erased. All the user had done was make the filename transparent to the operating system.

This article examines techniques that a slightly more sophisticated end user might employ to hide data rather than trying to erase it.

Remember to exercise caution when investigating for hidden data. For example, if there is the possibility of a criminal offense or civil litigation, the auditor/analyst should contact a Certified Forensic Computer Examiner (CFCE). The recovery of information that may be used in court or a hearing must be accomplished according to specific standards, and those standards are not the subject of this article. The recovery process requires special skills and expertise that can be gained only through experience, education and training.

The most important rule for the auditor/analyst who is going to examine data on a computer is to make a bit-stream image of the target drive and examine that image. *Never work on the original drive.*

The following are techniques that can be used to hide data. They include renaming files, changing file extensions, turning on the hidden attribute and concealing information in a part of the disk where data would not normally exist.

Renaming Files

Because filenames are user-enabled, the end user can create any filename desired for a given file. If the end user is familiar with the operating system and its files, he or she could use similar names to conceal certain files. For example, the Microsoft Windows Operating System has a hierarchy of file extensions. If files named *start.com, start.exe* and *start.bat* exist and *start* is typed with no extension at the command prompt, the file with the extension *.com* will be initiated first. If there is not a file ending in *.com* then the file ending in *.exe*, and finally the file ending in *.bat* would be executed. In this example, the end user could name a data file *start.exe* or *start.bat* and be relatively sure to have concealed information in the file that would never impact the system. Further, such a file might not draw the attention of anyone who was looking for unusual file information.

Most people know that files that end in *.com* are command files, files that end in *.exe* are executable files and files that end in *.bat* are batch files. It would be highly unlikely to find recoverable data in *.com* or *.exe* files. This technique provides the end user who is trying to hide data with the perfect spot to place secret information.

Changing File Extensions

Hiding data by changing file extensions is similar to renaming files, except that the end user changes only the three-letter extension at the end of the file name. Command files once ended only in *.exe, .com* or *.bat*. With the advent of Windows9x, WindowsNT/2000 and WindowsXP, command files also end in *.dll* or *.sys*. Document files, on the other hand, can have any three-character alphanumeric extension to be considered a legal filename. Therefore, the end user can create any file extension on any file in an attempt to disguise it. For example, a computer user looking for a Microsoft Excel spreadsheet file would probably search for files with the extension of *.xls*, which is the default extension for Microsoft Excel files. However, if the end user named an Excel file with an extension other than *.xls*, the file would not appear in a search for Excel files. Of course, if the user searched the drive or subdirectory for all files, then, as long as the hidden attribute was not on, these files would appear in the list. Just because certain files do not appear in the search results list does not mean that those files are not on the computer. Personal experience suggests that advanced users are likely to create custom file extensions. In fact, prior to the inclusion of long filenames (LFN) in Microsoft Windows, custom extensions were almost mandatory for keeping similarly named files separate.

Renaming files and changing the filename extensions are accomplished using the same operating system command: rename or *ren*. The graphic file in **figure 1** details the use of the *ren* command and its results.

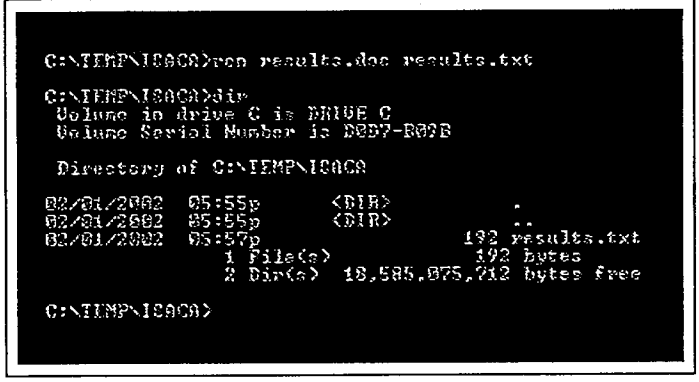
Figure 1

```
C:\TEMP\ISACA>dir
 Volume in drive C is DRIVE_C
 Volume Serial Number is D0D7-B09B

 Directory of C:\TEMP\ISACA

02/01/2002  05:55p      <DIR>          .
02/01/2002  05:55p      <DIR>          ..
02/01/2002  05:57p                   192 jim.txt
               1 File(s)              192 bytes
               2 Dir(s)  18,592,350,208 bytes free

C:\TEMP\ISACA>ren jim.txt results.doc

C:\TEMP\ISACA>dir
 Volume in drive C is DRIVE_C
 Volume Serial Number is D0D7-B09B

 Directory of C:\TEMP\ISACA

02/01/2002  05:55p      <DIR>          .
02/01/2002  05:55p      <DIR>          ..
02/01/2002  05:57p                   192 results.doc
               1 File(s)              192 bytes
               2 Dir(s)  18,592,251,904 bytes free

C:\TEMP\ISACA>
```

Figure 1 shows the directory that contains the file *jim.txt*. It is 192 bytes and was created on 1 February 2002. The *ren* command was used to change the filename to *results.doc*. A subsequent listing of the contents of the subdirectory shows that the filename has been changed. It is interesting to note that although the filename has changed, the creation date and time reported by the operating system are still the original date and time. However, while using a program like Norton Diskedit to look at the file, the modification and last access date and times would be reflected with the change date and time.

If the end user just wanted to change the filename extension, the command syntax would be *ren results.doc results.txt <enter>*. The screen shot in **figure 2** shows that change.

Figure 2



There is no good way to know that a filename or extension has been changed. However, the auditor/analyst might look for a filename that sounds like it contains information (a document) but has an extension that suggests the file could be executed at the command prompt. An example might be *taxes 2002.exe*. Such a file name extension suggests that the file could be run from the command line, but the filename does not sound like an executable file. A reasonable test would be to try to run the file from the command prompt. The auditor/analyst who gets the error message "Bad command or file name" would know that the file was not designed to run as a program. This would be reason enough to further investigate the contents of the file.

When viewing files at the hex level with an editor like Norton Diskedit, the auditor/analyst will find a few telltale signs in the file header information. For instance, all files that have an *.exe* extension begin with MZ as the first two bytes. Any *.exe* file that begins with information other than that indicates unusual data. All Microsoft Word documents start with DOC, all WordPerfect documents start with WPD, all Microsoft Excel spreadsheet files begin with XLS. By contrast, files that end in *.com* have no designation at the beginning of the file.

Turning on the Hidden Attribute

Microsoft Windows assigns attributes to each file to keep track of the file and its purpose. The attributes are as follows:

- **A**—Archive
- **S**—System
- **D**—Directory
- **V**—Volume
- **R**—Read-only
- **H**—Hidden

The archive (A) attribute lets the system know whether a file has changed since the last backup. This attribute has been around since the beginning of DOS. Knowing that end users periodically backed up their data, the designers of the operating system did not think they would want to back up the same data over and over. So the operating system designers created the archive attribute, which is reset each time the data file is backed up. When users back up their files the archive attribute is turned off. The next time the file is updated the attribute is turned on. When the operating system is asked to back up data files, it checks the file attributes to see if the archive attribute is turned on or not. Depending on the results, the operating system knows whether to back up a specific file.

The system (S) attribute marks the file as part of the system. By default, files that are required by the operating system are marked as such. Generally speaking, system files are marked as read-only (R) and hidden (H). System files that are used to start and run the operating system generally should not be accessible by the end user and should be protected from corruption or deletion. The designers of operating systems decided to protect such files by adding the system file attribute. However, this attribute readily can be added by an end user. Changing a file's attributes to make it a system file does not have any impact on the operating system.

The directory (D) attribute is added when the media directory is created. The operating system uses this attribute to keep track of all of the directories that are created on a particular media, such as a drive or a diskette. This attribute is not easily changed by the end user.

The volume (V) attribute is created by the operating system to keep track of the name of the disk or drive. This attribute is readily changeable by the end user but does not have any real effect on the computer or how it operates.

The read-only attribute tells the operating system not to let any other files write directly to a file that has this attribute turned on. Although this attribute is easily changed by the end user, rarely do end users change files with this attribute. In the early days of DOS, when the operating system was controlled at the command line, many users would set the read-only attribute to "on" for files like *command.com*. This would keep viruses from writing to or changing the code in the file. *Command.com* still appears in the Windows9x, WindowsNT, Windows2K and WindowsXP operating systems. This file, in which the operating system maintains all of its internal commands, is required for the computer to operate.

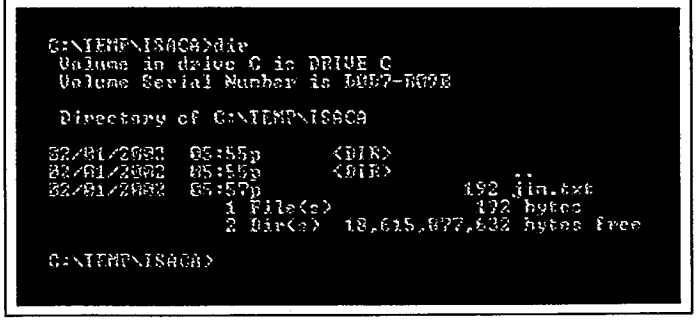
The hidden attribute tells the operating system to hide the file. Hidden files do not appear on the screen when the user runs a directory listing. To the user, the file is not present. The auditor/analyst, who knows the file is there and knows the path to locate the file, can access the file normally. However, to the end user who does not know the file is there, the file does not exist.

This particular technique is one of the easiest methods to hide a file. The command syntax is simple. At a command prompt or DOS prompt, the end user types the command

"attrib + /-" to turn on or turn off an attribute, the initial letter of the attribute, and then the file name. For example, to turn the hidden attribute on for a file named *jim.txt*, the syntax *attrib +h jim.txt* would be used.

This would turn on the hidden attribute. The file name would not be visible in the directory listing. The screen shot in **figure 3** shows this procedure.

Figure 3



Again using the file named *jim.txt*, which is 192 bytes in size and was created on 1 February 2002, the only attribute that is currently turned on for the file is the archive bit, indicated by the "A" at the beginning of the second line in **figure 4**. As discussed earlier, this attribute is turned on until the file is backed up, and then it is turned off.

Figure 4

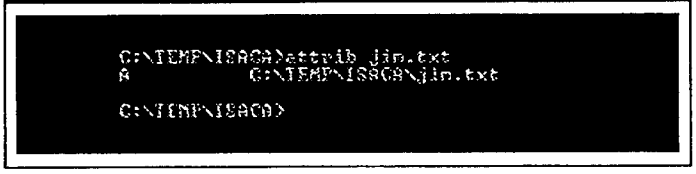


Figure 5

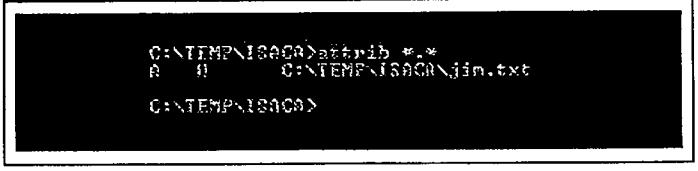
In **figure 5**, the screen shot shows a directory listing to see the filename, set the hidden attribute on and then request the directory listing again to show that the file is hidden.

Note that at the top of **figure 5**, the directory listing command *dir* was used. With this command, the computer listed the file *jim.txt* as part of the subdirectory ISACA. In the middle of the graphic, the command *attrib +h jim.txt* was used to turn on the hidden attribute that hides the file from view.

The final command was the directory listing command again, which showed the directory as empty. The file *jim.txt* is hidden and does not show up.

Figure 6 shows how to check a directory listing of all files, even the hidden ones.

Figure 6



The command syntax *attrib* *.* is used to check the attribute of all files in a subdirectory. This is a wildcard command that tells the operating system to display the attributes and file names for all files with all extensions.

Concealing Information

A number of techniques can be used to conceal data in a part of the disk where data normally do not exist. A few will be discussed here.

One technique that recently has achieved notoriety is the addition of tracks or sectors to a drive. A 3.5 inch double-density diskette with 720K of available storage space has 40 tracks, while a high-density diskette of the same physical size has 1.44MB of available space and 80 tracks. All the manufacturer did was compress the size of the tracks to put more information on the diskette. In both cases, the physical number of tracks is an artificial limit, based on the amount of space DOS can recognize on a diskette. Specialized software, such as Anadisk, by New Technologies, Inc., allows the end user to create additional tracks. This technique of adding tracks would allow end users to conceal data that are not normally viewable via the operating system.

Another technique involves the use of a hex editor, such as Norton Diskedit. The data area always begins in cluster #2. The system area precedes cluster #2. To find changes to the system area, go to the root directory and view the last directory entry with a text editor. As long as the first bit on each line is 00h, the operating system will allow additional information to be added without impacting the operating system or the root directory. The user will still see the directory without error, but a knowledgeable end user working with a hex editor could view the additional information that had been added to the diskette.

A third technique for concealing data, which also involves the use of a hex editor, is working with the file allocation table

(FAT). The FAT is like a road map that keeps track of data and filenames via a measurement called "clusters." Every file occupies space based on the number of clusters assigned to that file. When viewing a FAT with a hex editor the user sees columns of numbers that indicate an occupied cluster and a pointer to the next cluster assigned to the file. Over time the surface of the hard drive may become worn and lose some of its magnetic charge. When this happens, the operating system marks the cluster as bad and no longer addresses that space. With a hex editor, the sophisticated user could go into the FAT and choose a cluster that is good and mark it bad. Once the cluster is so marked, the operating system no longer works with that area of the drive. Then, using a hex editor, the user can access the cluster or clusters that have been marked as bad and write data to those areas. Anyone who knew a cluster had been marked bad could read the data in those areas.

A sophisticated user might use one or more of these techniques to conceal and transfer data to another user. A recent example of adding an additional track is the reported use of this method by a US Federal Bureau of Investigation (FBI) agent, who recently pleaded guilty to having traded secrets with a foreign government for years. The agent had learned the add-a-track technique at a computer crime investigation training program and had used it to communicate with his contact.

Encryption

Cryptography is more involved than can be explained in this article, but a general description and the options for an auditor/analyst follow.

The general explanation of encryption is replacement writing. If the end user encrypts a file, the contents of the file are run through an algorithm or formula that converts the original information to alternate (and possibly unreadable) characters. Users who are authorized to see the information use a key to decrypt the file, which makes the characters readable again.

There are two general kinds of encryption, one-way (synchronous key) and two-way (public key). In a one-way scheme, the information that is sent to the recipient is encoded and decoded with the same key. In a two-way scheme, the information is encoded with a public key and decoded with a private key. The person who creates the keys gives everyone the public key. If another person wants to send something to the original person, the information is encrypted with the public key. The only way to decrypt such encrypted information is to run the private key against it.

The only way to recover information that has been encrypted is for the auditor/analyst to convince the person who created the keys to provide the public and private keys. If the encryption algorithm used is a 48- or 56-bit key, those keys have been broken by frontal assault. If, however, the files have a 128-bit or larger encryption algorithm, they are in essence encrypted for all time. It is a secure and safe way to transmit information.

Steganography

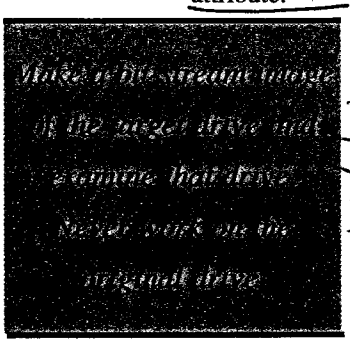
Steganography, which actually means secret writing, is one of the latest techniques for concealing information. This technique involves hiding data within data.

Researchers who studied the sizes of graphic files discovered that a large amount of the space in a graphic file was unused. Computer scientists tested this unused space and learned that it could be utilized without seriously compromising the quality of the original graphic file. For example, end users can download program files from the Internet that will allow them to transfer information within graphic or other files. Graphic files seem to be the file type of choice because they have so much unused space.

When users engage in this type of file transfer it is difficult to discover because the quality of the graphic file is rarely affected. However, if too much of this available space is utilized by other data, the quality of the graphic image can be affected. The analyst/auditor would have to have a reason to believe the graphic was degraded before noticing that something was amiss.

Finding the Data

A variety of software applications can be used to find hidden data. Some are nothing more than using the operating system's internal and external commands, such as rename and attribute.



A few sophisticated applications, among their other uses, have the ability to hide data. One such program, Anadisk, originally was created to allow for investigative and audit analysis of diskettes. In addition to a hex level editor, it has a facility that allows the end user to view the data at the individual sector level or at the track level. This same application allows the user to write changes to the diskette.

The most readily available program for discovering changes in the FAT or root directory area is Norton Diskedit. This program allows the user to view those areas of the system disk.

When any of the techniques described previously are used correctly, the auditor/analyst who comes across an altered diskette or hard disk drive may not notice anything out of the ordinary. There are no telltale signs. The auditor/analyst would normally require independent information that unusual computer use or abuse was going on to decide to examine a diskette or drive further. If there was reliable information about unauthorized computer activity and nothing had been located in the system to indicate a problem, it would be a good idea to delve further into the drives and files. Additional checks might point toward the need for an extensive computer forensic examination.

With the large size of hard drives these days, it is difficult to check every file on a system for potential abuse. That is why audit intelligence is so important.

Again, remember to never work directly on the target drive. Always make a bit-stream image of the target drive and examine the image.

Clayton Hoskinson, CFE, CFCE
is a principal in Clayton Hoskinson & Associates, providing services to a variety of companies and agencies. He has a Master of Criminal Justice Administration (MCJA) from Oklahoma City University and a BA in criminal justice from the University of Central Oklahoma. He has worked for the University of Oklahoma Police Department, the Norman (Oklahoma, USA) Police Department, the Oklahoma Department of Human Services OIG, and is presently an instructor at Redlands Community College. He can be reached at *clayton@claytonhoskinson.com.*

Jim Sleezer, CISA
is a senior information systems auditor/consultant for the Oklahoma State University/A&M System of Colleges and Universities. He has a Master of Business Administration (MBA) from the University of Minnesota, USA, and is president of the ISACA Central Oklahoma Chapter. He can be reached at *jhs8@okstate.edu.*