Different Interpretations of ISO9660 File Systems

By

Brian Carrier

From the proceedings of

The Digital Forensic Research Conference

DFRWS 2010 USA

Portland, OR (Aug 2nd - 4th)

available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/diin

Digital Investigation

Different interpretations of ISO9660 file systems

Brian D. Carrier

Basis Technology, One Alewife Center, Cambridge, MA 02140, USA

Keywords:
File System Forensic Analysis
Data Hiding
Tool Testing

ABSTRACT

In this paper, we examine the potential to hide data in an ISO9660 file system, which is used to store data on CD-ROMs. By design, this file system allows for multiple directory trees and different byte orderings of essential data. We describe how data could be hidden in an ISO9660 file system and create test images using the described techniques. We test commonly used forensics tools to determine if the hidden data can be seen. The test results show that different tools show and hide different data. Some tools show all of the data, some tools show some of the data, and other tools show none of the data.

1. Introduction

Typically, there is only one way to interpret a file system. The first sectors of the volume have data that point to the various essential file system structures, such as the root directory, free block list, and file metadata. With that information, you can go to the root directory and start searching for specific files and examine the file content.

The ISO9660 file system, which is used to store data on CD-ROMs, is different in that it, by design, allows for different interpretations. From the first sectors of the volume, you can find multiple directory tree hierarchies and data structures store essential file system data in both big- and little-endian byte orderings.

This paper examines how different digital forensic tools and operating systems interpret the data in an ISO9660 file system. The motivation for this work is that if the tools have different analysis assumptions and produce different interpretations, then it may be possible for someone to exploit these differences and hide data from an investigation.

The goals of this paper are to outline ways that the ISO9660 design can be used to hide data and to determine if existing tools will allow data to be hidden and not identified. Section 2 of this paper provides an overview of the ISO9660 file system. Section 3 outlines some techniques for data hiding and Section 4 outlines test results from commonly used digital forensic tools.

2. ISO9660 file system

This section describes the high-level concepts of an ISO9660 file system. In general, this paper does not require detailed knowledge of the low-level data structures, but they can be found in the specification (ECMA, 1987).

Note that an ISO9660 file system is not the lowest level of abstraction on a CD-ROM. At the lowest data levels, there are one or more sessions. Each session contains one or more tracks. These session and tracks can be considered similar to partition tables on a hard disk. The ISO9660 file system exists inside of a track. Sessions and tracks are out of scope for this file system-level work.

An ISO9660 (ECMA, 1987) file system contains one primary and zero or more secondary volume descriptors. The volume descriptors start in sector 16 of the file system and contain the data that identify the file system's size and layout. They are similar to boot sectors in FAT and NTFS (Carrier, 2005). The volume descriptor will point to, among other things, the block where the root directory starts and the location of the path table. The reason for having multiple volume descriptors will be addressed at the end of this section.

ISO9660 file systems store data in blocks, which are groupings of consecutive sectors. The block size is stored in a volume descriptor. File content is, in general, stored in consecutive blocks. The exception is if the file system uses an

E-mail address: brianc@basistech.com
1742-2876/$ – see front matter © 2010 Digital Forensic Research Workshop. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.diin.2010.05.016

interleave gap and places a constant gap between groupings of blocks.

File metadata is stored in the blocks allocated to the file's parent directory. Each file has a directory entry data structure in the directory and it contains the file's name and other metadata, such as size, dates, and starting block location. Because the file content is stored in consecutive blocks, the file content can be located using only the starting block address, the file size, and the interleave gap.

Some CDs store additional metadata about each file using the "Rock Ridge" extensions (IEEE CDROM, 1994). These CDs store additional metadata for each file immediately after the file's directory entry.

ISO9660 file systems support names with a maximum of 8 Latin characters in the file name and 3 Latin characters in the extension. Some ISO9660 file systems have the Joliet extension (Microsoft, 1995) and can store longer names in Unicode.

As previously mentioned, the starting block of the root directory is listed in the volume descriptor. The directory tree allows you to locate a file by opening the root directory, finding the first directory in the path, opening it, and repeating the process for each directory. An alternative method for locating a file is using the path table.

The path table contains an entry for every directory in the file system and the entry points to the directory's starting block. The path table allows one to find a specific subdirectory faster than processing all of the directory contents because it can be read into memory when the file system is loaded. The location of the path table is given in the volume descriptor. Fig. 1 shows the relationship between the volume descriptor, path table, root directory, and file content.

As described, a volume descriptor points to all of the needed file system structural data. An ISO9660 file system will have multiple volume descriptors if it uses one of the extensions that exist to compensate for the basic ISO9660 limitations. For example, an ISO9660 file system that uses the Joliet extension to store Unicode file names will have a secondary volume descriptor that points to a different path table and root directory that have Unicode file names.

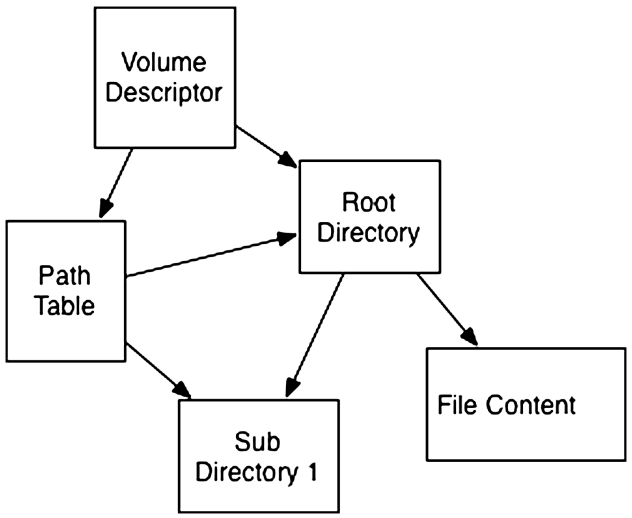

Fig. 1 – The ISO9660 volume descriptor points to the essential file system structures.

When multiple volume descriptors exist, a file will typically have a directory entry in each of the corresponding directory trees, but it will have only one copy of the file content and both directory entries will refer to it. For example, the root directory for the primary volume descriptor may have a file with a name of "ABCDEF~1.TXT" and the same file will have a name of "Abcdefghijklmnop.txt" in the root directory of the secondary volume descriptor. Both entries will refer to the same starting block for the file content. Fig. 2 shows two volume descriptors, two root directories, and a single set of blocks that contain the file content.

A final note about the ISO9660 file system is that it stores essential multi-byte file system data in both big- and little-endian orderings. Data stored in big-endian ordering store the most significant byte first and data stored in little-endian ordering store the least significant byte first. Most file systems store data in only one of the orderings.

3. Data hiding methods

In this paper, we focus on the ability to hide files by a) having multiple volume descriptors and b) having different values in the different byte ordered storage locations. We do not address issues that can occur at the lower-levels of abstraction, such as open sessions. Other papers (Marberry and Craiger, 2006) have examined tool support for variations at these levels.

Note that in all of the data hiding techniques outlined in this paper, the "hidden" data exists in blocks in the file system. We found that even if the tool did not display one or more of the hidden files, the file content could be found from keyword searching or carving of unallocated space.

3.1. *Inconsistent directory trees*

The ISO9660 specification does not require that the directory trees associated with the different volume descriptors have

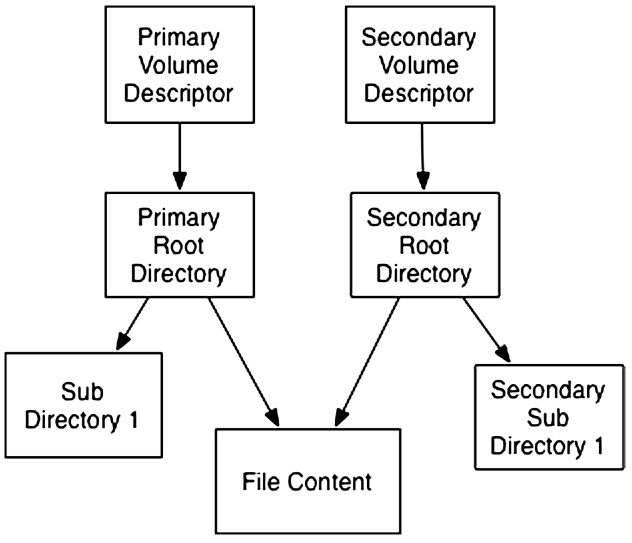

Fig. 2 – ISO9660 file systems can have multiple volume descriptors and directory trees. The directory entries typically point to the same file content blocks though.

the same files and subdirectories. Therefore, one approach to hiding data is to have different data in each directory tree. If most operating systems and forensics tools show only one of the trees, but the data hider has the ability to see the contents of the other tree, then he can access the hidden data and it may not be detected during an investigation. Fig. 3 shows an example where the primary volume descriptor points to a root directory with three files in it and the secondary volume descriptor points to a root directory with two files.

3.2. *Empty directory tree*

Another data hiding approach is a variation of the previous one, except that it uses two or more secondary volume descriptors. The first secondary volume descriptor has an empty root directory and the second secondary volume descriptor contains the files to be hidden.

This approach was motivated by the observation that some tools and operating systems will process only the first secondary volume descriptor, because it typically contains the Unicode file names from the Joliet extension. The tools may never look at the second descriptor, because they rarely exist. Therefore, files in the second descriptor's root directory may not be shown to the investigator. Fig. 4 shows an example where the primary volume descriptor has a root directory with three files, the first secondary volume descriptor has no files, and the second secondary volume descriptor has two files.

3.3. *Endian ordering*

The third approach exploits the fact that ISO9660 data structures store values in both big- and little-endian orderings. Data could be hidden if the starting block value in a directory entry has different values in the big- and little-endian storage locations. If analysis tools examine only one of the locations and the hider's tools use the other, then the hidden data may not be found. Fig. 5 shows an example where the starting block in the big-endian field is 0x00000020 and the starting block in the little-endian field is 0x00000030.

Many ISO9660 data structures store both little- and big-endian ordered data that could be used to hide data. Other examples include block addresses in the path table, file length, root directory starting location, and root directory size.

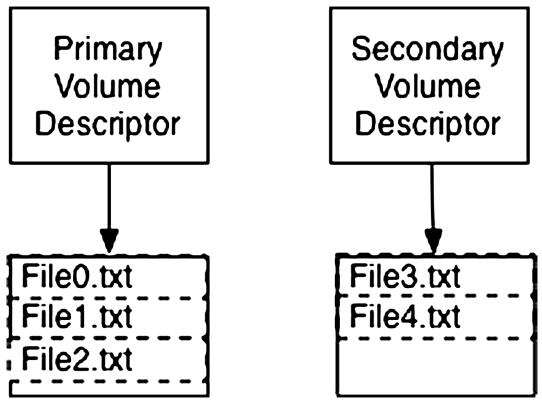

Fig. 3 – Different directory trees can have different files in them.

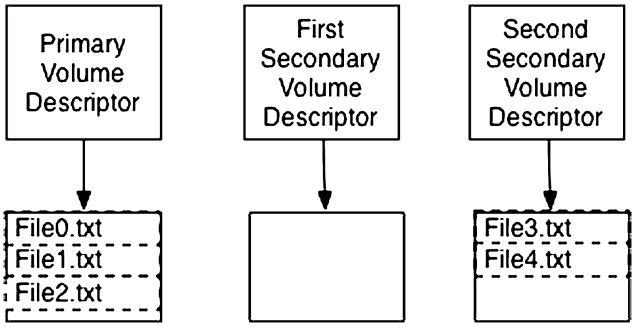

Fig. 4 – One hiding technique uses an empty root directory in the first secondary volume descriptor.

4. Tool testing results

Tests were created for the three data hiding techniques outlined in the previous section. ISO9660 images were created that used each of the data hiding techniques and the image was loaded into various tools and operating systems. The following sections outline the test details and results. The images associated with these tests are available for further testing (Carrier).

The following tools were included in the tests:

- EnCase 6.15 (Guidance Software)
- Forensic Tool Kit (FTK) 1.60 (Access Data)
- ISOBuster 2.7 (X-Ways Inc)
- Linux 2.6 (Lefebvre)
- OS X 10.4.11 (Power PC) (Apple Inc)
- OS X 10.6.2 (Intel) (Apple Inc)
- Sleuth Kit (TSK) (Carrier)
- Windows XP (Microsoft)
- Windows Vista (Smart Projects)
- Win Hex Forensics (Microsoft)

EnCase, FTK, ISOBuster and TSK did not have configuration settings that were ISO9660-specific. The image file was loaded into the software and default settings were used. Win Hex Forensics has a "Read ISO9660 even if Joliet Present" option and the image file was processed with this option enabled and disabled.

To test Windows XP and Windows Vista, a CD was burnt from the image file and the CD was inserted into the system and automatically mounted by the OS. OS X was tested using both the CD and the image file. The image file was mounted using the Disk Utility application.

Linux had several options. First, the CD was inserted and automatically mounted with no user interaction. The image file was also manually mounted using the loopback mode. First it was mounted using the default options. Next, it was mounted using the "nojoliet" option, which will cause the Joliet metadata extensions to be ignored. The image was also mounted using the "norock" option, which will cause the Rock Ridge Unicode extensions to be ignored.

Name	Starting Block (big-endian)	Starting Block (little-endian)
File1.txt	00 00 00 20	30 00 00 00

Fig. 5 – A simplified version of a directory entry with different starting block addresses in the big- and little-endian locations.

4.1. Inconsistent directory trees

4.1.1. Test outline

The first test was to determine what would be visible when an image had two inconsistent directory trees. To test this data hiding approach, an image was created with two volume descriptors. Each volume descriptor had a corresponding root directory with a single file. The file name and corresponding file content were different in each root directory. The test would show which of the two files was shown.

4.1.2. Data creation

A base ISO9660 image was made with Joliet extensions. It had two files, "File1.txt" and "File2.txt", in the root directory and because it was created with Joliet extensions, it had a secondary volume descriptor. The image ("iso-dirtree1.iso") was made with 'mkisofs' in Linux.

Using a hex editor, we manually edited both of the root directories in the image file to remove one of the two files. The location of the root directories was determined using the 'fsstat' command in The Sleuth Kit. The root directory for the primary volume descriptor was located in block 28. It was edited so that the directory entry for "File2.txt" was overwritten with 0s. The root directory for the secondary volume descriptor was located in block 29. It was edited by copy and pasting the "File2.txt" directory entry over the "File1.txt" directory entry and the original "File2.txt" entry was overwritten with 0s. In the end, the root directory for the primary volume descriptor had "File1.txt" in it and the root directory for the secondary volume descriptor had "File2.txt" in it.

4.1.3. Test results

The modified ISO9660 image was burnt to a CD, loaded into forensics tools, and mounted by different operating systems. As can be seen in Table 1, we got different results from different software.

Forensic Tool Kit, ISOBuster, Sleuth Kit and Win Hex Forensics (using non-default options) showed both files. The method for showing the files differed though. FTK, ISOBuster, and Win Hex had a hierarchy and showed the individual files under the heading of each of the two volume descriptors. Sleuth Kit showed File2.txt as a normal file and File1.txt as an "orphan file" (meaning that it could not be reached from the root directory).

The other ten tools showed only one of the files. Five of them showed only the primary volume descriptor contents and five of them showed only the secondary volume descriptor contents.

Two conclusions can be drawn from these results. One is that common analysis techniques for CDs do not produce results that are consistent among them and do not show all possible data. However, a second conclusion is that, because of the inconsistent approaches taken by analysis tools, it would be risky for someone to use this data hiding approach because they don't know what tool that their adversary is using.

4.2. Empty secondary volume descriptor

4.2.1. Test outline

The second test focused on the approach of having an empty root directory in the first secondary volume descriptor. This approach may be able to hide data because some tools seem to rely only on the first secondary volume descriptor.

The data for this test was based on the data for the previous test, but it would have three root directories. The first and third would have unique files. By identifying which of the files was shown, we could determine which of the volume descriptors was being processed.

4.2.2. Data creation

The image for this test ("iso-dirtest2.iso") was based on the image from the first test. A copy of the secondary volume descriptor was made in a hex editor from sector 17 and copied into sector 18.

A copy of the secondary volume descriptor's root directory and path table was made and placed in unused blocks, 33 and 34 respectively. The new secondary volume descriptor in sector 18 was updated with the addresses of the new root directory and path table. The root directory for the first secondary volume descriptor was manually edited to overwrite the "File2.txt" directory entry with 0s. In the end, the root directory for the primary volume descriptor had "File1.txt" in it, the root directory for the first secondary volume descriptor had no files in it, and the root directory for the second secondary volume descriptor had "File2.txt" in it.

4.2.3. Test results

The modified ISO9660 image was burnt to a CD, loaded into forensics tools, and mounted by different operating systems. The results can be seen in Table 2.

Table 1 – Inconsistent directory test results. A 'Y' means that the file was displayed.

Tool/OS	File1.txt	File2.txt
EnCase		Y
Forensic Tool Kit	Y	Y
ISOBuster	Y	Y
Linux (mount image, default options)	Y	
Linux (mount image, nojoliet option)	Y	
Linux (mount image, norock option)		Y
Linux (via CD)	Y	
OS X (mount image)	Y	
OS X (via CD)	Y	
Sleuth kit	Y	Y
Windows XP (via CD)		Y
Windows Vista (via CD)		Y
Win Hex Forensics (default)		Y
Win Hex Forensics ("Read ISO9660 even if Joliet Present" option enabled)	Y	Y

Table 2 – Empty secondary volume descriptor test results. A 'Y' means that the file was displayed.

Tool/OS	File1.txt	File2.txt
EnCase		
Forensic Tool Kit	Y	Y
ISOBuster	Y	Y
Linux (mount image, default options)	Y	
Linux (mount image, nojoliet option)	Y	
Linux (mount image, norock option)		
Linux (via CD)	Y	
OS X (mount image)	Y	
OS X (via CD)	Y	
Sleuth Kit	Y	Y
Windows XP (via CD)		
Windows Vista (via CD)		
Win Hex Forensics (default)		Y
Win Hex Forensics ("Read ISO9660 even if Joliet Present" option enabled)	Y	Y

The results are similar to the results from the previous test with four exceptions. EnCase, Linux mounting with the noRock Ridge option enabled, Windows XP, and Windows Vista displayed none of the files. The other tools showed either both or only one of the files.

Similar conclusions can be drawn from these results. One is that the tools do not use the same procedure and this can result in missing data. In this test, some tools failed to show either of the two files. The second conclusion is that it would be risky to use this data hiding technique because the hider may not know what his adversary is using.

4.3. *Endian ordering*

4.3.1. *Test outline*

The third test focused on if tools read one or both of the various values that are stored in both big- and little-endian orderings. This test used an image with a single file in the root directory (the primary and secondary directory trees were consistent for this test). The file's directory entry had different starting block values in the big- and little-endian storage locations. The test would be to determine which blocks the tool processed and opened.

4.3.2. *Data creation*

The image for this test ("iso-endian.iso") started as a basic image with a single text file, "File1.txt". The file contained the text "This tool uses the little-endian value" in block 31. Using a hex editor, the contents of block 31 were copied into block 32, which was not being used, and edited to have the text "This tool uses the big-endian value." Spaces were added to ensure that the text was the same length in both blocks.

The 'fsstat' command was used to locate the root directories for the primary and secondary volume descriptors. The directory entry for "File1.txt" in both directories was edited using a hex editor so that the big-endian value of the starting block was 32 and the little-endian value was 31.

4.3.3. *Test results*

The modified ISO9660 image was burnt to a CD, loaded into forensics tools, and mounted by different operating systems. The results can be seen in Table 3.

Table 3 – Endian ordering test results. A 'Y' means that the tool used that ordering.

Tool/OS	Big endian	Little endian
EnCase		Y
Forensic Tool Kit	Y	
ISOBuster	Y	
Linux (mount image, default options)		Y
Linux (mount image, nojoliet option)		Y
Linux (mount image, norock option)		Y
Linux (via CD)		Y
OS X, 10.4.11 (via CD)		Y
OS X 10.6.2 (mount image)		Y
OS X 10.6.2 (via CD)		Y
Sleuth Kit	Y	
Windows XP (via CD)		Y
Windows Vista (via CD)		Y
Win Hex Forensics (default)		Y
Win Hex Forensics ("Read ISO9660 even if Joliet Present" option enabled)		Y

The results show that most of the tools, including ones that run on big-endian platforms, process the little-endian value. Only Forensic Tool Kit, ISOBuster, and Sleuth Kit process the big-endian value. None of the tools reported the inconsistency to the user. Like the previous data hiding approaches, the inconsistent approaches from the various tools would make this a risky data hiding approach because the hider would not know which tool his adversary is using.

5. Conclusion

The design of an ISO9660 file system allows for the data to have multiple interpretations. This paper has shown that existing operating systems and tools use different methods for choosing how to interpret ISO9660 file systems. The impact of this is that an analysis tool may not show the same data that the media's users were seeing. This can result in an investigator missing important data.

To reduce the risk of missing data, investigators should use multiple tools that use different analysis techniques. For example, one might want to use one of the tools that showed both files in the first two tests and one of the tools that showed only one of the files. In the future, tools will hopefully clearly show when different interpretations of an ISO9660 file system produce conflicting results.

The threat of the data hiding techniques outlined in this paper are mitigated by the fact that the files that are not displayed to the investigator in the directory tree can still be found by carving the unused blocks or keyword searching the entire image. Also, because the existing tools are so inconsistent with what they show, it could be risky for someone to use these data hiding techniques because they don't know what tool their adversary will be using to view the data.

The images used for testing in this paper were manually created. Further work needs to be done to study CDs created by frequently used CD mastering software to determine if they use techniques similar to the ones presented in this paper. For example, one could imagine that a CD may use different starting

block addresses in the different endian ordering fields if the file content was endian-order specific and the CD creator wanted the OS to access the relevant file content. Further study would allow an investigator to make better-informed hypotheses about intent when techniques such as those used in this paper are found.

Because there is not a standard set of ISO9660 file system forensic tool requirements, we cannot say that any of the tools used in this work are flawed. Instead, we can say that some tools choose only one interpretation, that different tools choose different interpretations, and that other tools consider multiple interpretations and show them all to the user. Investigators should know what to expect from their tools and should choose tools carefully so that they are seeing the various interpretations.

REFERENCES

Access Data. Forensic toolkit. Available online at: http://www.accessdata.com/forensictoolkit.html.

Apple Inc. OS X. Available online at: http://www.apple.com/macosx/.

Carrier B. Digital forensics tool testing image #14: ISO9660 data hiding. Available online at: http://dftt.sourceforge.net.

Carrier B. File system forensic analysis. Addison Wesley; 2005.

Carrier B. The Sleuth kit. Available online at: http://www.sleuthkit.org/.

ECMA. Volume and file structure of CDROM for information interchange. In: Standard ECMA-119. 2nd ed.; December 1987.

Guidance Software. EnCase Forensic. Available online at: http://www.guidancesoftware.com/.

IEEE CDROM File Systems Format Working Group. Rock ridge interchange protocol. Version 1.12. July 8, 1994.

Lefebvre C. Linux Mint. Available online at: http://www.linuxmint.com/.

Marberry C, Craiger P. CD-R acquisition hashes affected by write options. Journal of Digital Forensic Practice 2006;1(4).

Microsoft, Inc. Joliet specification. Version 1. May 22, 1995.

Microsoft, Inc. Windows XP. Available online at: http://www.microsoft.com/windows/windows-xp/.

Microsoft, Inc. Windows vista. Available online at: http://www.microsoft.com/windows/windows-vista/.

Smart Projects. ISOBuster 2.7. Available online at: http://www.isobuster.com/.

X-Ways Inc. WinHex Forensics. Available at: http://www.x-ways.net/winhex.