NIST CFTT: Testing Disk Imaging Tools

James R. Lyle, Ph.D.
Computer Scientist
National Institute of Standards and Technology

1. Introduction

There is a critical need in the law enforcement community to ensure the reliability of computer forensic tools. A capability is required to ensure that forensic software tools consistently produce accurate and objective test results. The goal of the Computer Forensic Tool Testing (CFTT) project at the National Institute of Standards and Technology (NIST) is to establish a methodology for testing computer forensic software tools by development of general tool specifications, test procedures, test criteria, test sets, and test hardware. The results provide the information necessary for toolmakers to improve tools, for users to make informed choices about acquiring and using computer forensics tools, and for interested parties to understand the tools capabilities. Our approach for testing computer forensic tools is based on well-recognized international methodologies for conformance testing and quality testing, such as ISO/IEC 17025:1999, *General requirements for the competence of testing and calibration laboratories*. This project is further described at http://www.cftt.nist.gov/.

The CFTT is a joint project of the National Institute of Justice, the research and development organization of the U.S. Department of Justice; the National Institute of Standards and Technology's Office of Law Enforcement Standards and Information Technology Laboratory; and other agencies, such as the Technical Support Working Group. The entire computer forensics community can help develop the specifications and test methods by commenting on drafts as they are published on the website.

This paper reports on the initial CFTT work conducted at NIST in the Software Diagnostics and Conformance Testing (SDCT) division of the Information Technology Lab (ITL) as applied to disk imaging tools. An overview of the testing methodology developed at NIST is presented followed by a discussion of the development of the specification for testing two disk imaging tools, **dd** and **SafeBack**[1].

2. CFTT methodology overview

The testing methodology developed by NIST is functionality driven. The activities of forensic investigations are separated into discrete functions or categories, such as hard

[1] † **Certain trade names and company products are mentioned in the text or identified. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the products are necessarily the best available for the purpose.**

disk write protection, disk imaging, string searching, etc. A test methodology is then developed for each category. Currently we have developed a methodology for disk imaging tools and are developing a methodology for software hard disk write blocking tools. Deleted file recovery tools will be the next category for development of a test methodology.

The CFTT testing process is directed by a steering committee composed of representatives from the law enforcement community, including the FBI, DoD, NIJ (representing state and local agencies), NIST/OLES and other agencies. Currently the steering committee selects tool categories for investigation and tools within a category for actual testing by CFTT staff.

Under the disk imaging category the tools selected initially for testing were: Linux **dd**, and **SafeBack**. The Royal Canadian Mounted Police **hdl** tool was selected for the hard disk write block category. Final test reports are posted to a web site maintained by NIJ.

2.1 Specification development process

After a tool category and at least one tool is selected by the steering committee the development process is as follows:

1. ITL and law enforcement staff develop requirements, assertions and test cases document (called the tool category specification).
2. The tool category specification is posted to the web for peer review by members of the computer forensics community and for public comment by other interested parties.
3. Relevant comments and feedback are incorporated into the specification.
4. A test environment is designed for the tool category.

2.2 Tool test process

After a category specification has been developed and a tool selected, the test process is as follows:

1. ITL acquires the tool to be tested.
2. ITL reviews the tool documentation.
3. ITL selects relevant test cases depending on features supported by the tool.
4. ITL develops test strategy.
5. ITL executes tests
6. ITL produces test report.
7. Steering Committee reviews test report.
8. Vendor reviews test report.
9. ITL posts support software to web.
10. NIJ posts test report to web.

3. The disk imaging specification

The base requirements developed for disk imaging tools are the following:

- The tool shall make a bit-stream duplicate or an image of an original disk or partition.
- The tool shall not alter the original disk.
- The tool shall be able to verify the integrity of a disk image file.
- The tool shall log I/O errors.

Several other optional requirements were also developed to allow for differences in tool designs and available options. The detailed specification can be found at the CFTT website.

3.1 Disk imaging tool test environment

The usual testing paradigm is a three step process of setting up a test environment, executing the tool being tested and measuring the results of tool execution. This section establishes the capabilities required to set up and measure each test. This is accomplished by first determining what must be measured to confirm compliance with the tool requirements and second what conditions must be created to allow an accurate measurement of compliance.

3.1.1 Accuracy of duplicate creation

The basic function of a disk imaging tool is to create a duplicate of either a hard disk drive or a partition from a hard disk drive. A disk imaging tool copies disk sectors from a source to a destination such that the destination is identical or nearly identical to the source. In the ideal situation, the destination is identical to the original. However, forensically valid differences do occur when the source and destination are not the same size or if partitions from the source disk must be relocated on the destination drive to keep the destination partitions accessible.

A program to compare the corresponding sectors of two disks can determine the accuracy or point of failure of the duplication process. If the destination is larger than the source, then there are excess sectors on the destination. An analysis of the excess destination sectors is useful to determine tool behavior in this case. However, to do such an analysis, the source and destination sectors need to be uniquely recognizable from the sector contents. If every sector of each disk used in a test is initialized to a known unique value then the task of determining the origin of each sector of the destination during the measurement of the results of a test run is simplified. When a destination sector is examined there are three possibilities for the contents of the sector: the original destination sector contents are unchanged by the test, the contents of a sector from the source disk is written to the destination, or the imaging tool has generated new content for the sector.

Another factor to consider in duplicate creation is the accessibility of the duplicate. It may be desirable to ensure that the information copied to a disk can be accessed or the duplicate drive be booted. However, if the duplicate drive has a different number of cylinders and heads from the original, a sector for sector copy might not align partitions to cylinder boundaries. To create a bootable copy of the source, the partitions of the destination need to be aligned on cylinder boundaries. If there is more than one partition

on the source then the destination partitions are padded with extra sectors to create a bootable destination.

3.1.2 Source Disk is Unchanged

The source disk must not be altered by running the disk imaging tool. One approach for detecting any changes to a disk is to compute a hash function of the entire contents of a disk both before and after the disk imaging tool is run. The likelihood that two disk drives with different content would produce the same hash value is related to the length of the hash function value. The secure hash algorithm (SHA-1, Federal Information Processing Standard FIPS 180-1), used by CFTT, produces a 160 bit hash, giving a 1 in 2^{80} chance that any change to the source by the tool would be undetected. The hashing technique can also be used to verify that a block of sectors has not been changed by some operation.

3.1.3 Verify image file integrity

Some disk imaging tools attempt to detect any changes to an image file after the creation of the image file. A tool is needed to introduce a change into an image file.

3.1.4 Error logging

Since forensic evidence may need to be obtained from a faulty disk, the behavior and capabilities of disk imaging tools in the presence of disk I/O errors must be evaluated. The problem is how to have a *reliably bad disk* (i.e., a disk that can produce a variety of failure modes and I/O errors useful to a planned test). One solution is to use a correctly functioning disk, but intercept disk I/O requests such that a faulty disk can be simulated to the disk imaging tool according to each test case specification. A better solution would be to develop a hardware component that can be attached between the hard drive and the computer to simulate (reliably) a faulty drive under external program control. The software solution has been implemented but NIST is also researching a hardware solution. The software approach has the drawbacks that there needs to be a separate software component for each of several software environments and it would be difficult to reliably simulate a faulty disk for a program accessing a disk directly through the ATA interface. One software version is required for access by BIOS in DOS, one each for the different Windows environment, and one for Linux. A hardware implementation would be independent of any software environment.

3.1.5 Summary of Required Capabilities

The minimum set of basic capabilities required to support testing a disk imaging tool is the following:

- Initialize each sector of a disk to a known unique value.
- Compute a SHA-1 hash of an entire disk or a block of disk sectors.
- Simulate a faulty disk drive.
- Compare two disks of either equal or unequal size.
- Compare two disk partitions of either equal or unequal size.

- Compare two disks where the destination disk has had partitions aligned to cylinder boundaries.

In addition to these basic capabilities there are several additional capabilities that are helpful. These include the following:

- Compare two sectors.
- Modify or examine the contents of a sector.
- Relocate a block of sectors.
- Examine a partition table.

3.1.6 Software Overview

To support the capabilities necessary for testing disk imaging tools, the programs listed in Figure 1 were developed. The support software can be grouped into four categories: programs to set up something required for the test, programs to measure the results of a test, programs to document some aspect of a test, and utility programs to provide capabilities outside the scope of the usual test case.

Figure 1 List of Support Programs

Program	Category	Function
DISKWIPE	Setup	Write a unique pattern to each sector of a disk.
BADDISK	Setup	Simulate a bad disk sector for legacy BIOS access.
BADX13	Setup	Simulate a bad disk sector for extended BIOS access.
CORRUPT	Setup	Change one byte in a file (corrupt an image file).
ADJCMP	Measure	Compare two disks by partitions (allowing for cylinder alignment).
DISKCMP	Measure	Compare two disks.
PARTCMP	Measure	Compare two partitions.
DISKHASH	Measure	Compute a SHA-1 for an entire disk.
SECHASH	Measure	Compute a SHA-1 for part of a disk.
LOGCASE	Document	Log information about the test run, e.g., disks used.
LOGSETUP	Document	Log information about source hard drive setup.
PARTAB	Document	Log partition tables.
DISKCHG	Utility	Change or examine the content of a single disk sector.
SECCMP	Utility	Compare two disk sectors.
SECCOPY	Utility	Copy a block of sectors from one location to another.

3.2 The disk imaging test cases

Each disk imaging test case is designed to follow a sequence of steps that set up the test, execute the disk imaging tool under test and measure the results. The general structure of a test case is outlined in Figure 2. Usually a source drive is set up (steps 1-3) once and then used in more than one test case.

Figure 2 General Structure of a Test Case

Setup for the test case:
1. Record details of source disk setup.
2. Initialize the source disk to a known value.
3. Hash the source disk and save the hash value.
4. Record details of test case setup.
5. Initialize a destination disk.
6. If the test requires a partition on the destination, create and format a partition on the destination disk.
7. If the test uses an image file, partition and format a media disk.

Execute the tool being tested:
8. If the test requires a disk I/O error, set up disk error simulation.
9. If the test requires an image file, use the tool to create an image file of the source on the media disk.
10. If the test requires a corrupted image file, corrupt the image file.
11. Use the disk imaging tool to create the destination disk either from a copy of the source disk or by restoring an image file of the source to the destination.

Measure the results:
12. Compare the source to the destination.
13. Compute a hash of the source disk, compare with the saved hash value.
14. Examine the tool log file for relevant messages.

The major challenge to developing the test cases for a tool category is the identification of the relevant parameters that can be varied across test cases. After the test parameters are identified, they must be incorporated into the specification in a meaningful way. To identify relevant parameters, major imaging tools such as **SafeBack** and **EnCase** were examined for features and options provided by the tools. In addition, general features of the Intel-based PC architecture were considered. The result is the table of parameters and values presented in Table 3.2-1.

Table 3.2-1 Disk Imaging Test Parameters and Values

Parameter	Values
Tool Functions	Copy, Image, Verify
Source interface	BIOS to SCSI, BIOS to IDE, Direct IDE, ASPI SCSI, Legacy
Destination interface	BIOS to SCSI, BIOS to IDE, Direct IDE, ASPI SCSI, Legacy
Relative size	Equal, Source < destination, Source > destination
Errors	none, src read, dst write, image read, image write, corrupt image
Object Type	Entire disk, partition: FAT12, FAT16, FAT32, NTFS, Ext2
Remote access	yes, no

If a test case were generated for each possible combination of parameter values, more than 32,000 test cases would be produced. To keep the number of test cases to a practical

level, a number of techniques were used. For example, instead of all possible combinations of parameter values, a more relaxed criterion such as *pair-wise selection* (cases are selected such that for each pair of parameter values there is at least one case). The number of test cases was also reduced by careful elimination of unlikely combinations of parameter values.

3.2.1 Tool Functions

The most basic function common to all imaging tools is for the tool to read all accessible sectors of a hard drive and store the contents in some form such that any designated sector (word, byte or bit) can be accurately recovered and examined. The challenge that arises in testing is that no imaging tool stores the sectors in exactly the same way. Some tools allow the copying of the source drive to another drive. Some tools produce an image file in a proprietary format. Since the image file format is usually unique for each tool, it is difficult to examine the image file to determine if all the sectors from the original hard drive are included in the image file. However, since most tools have a function to restore the contents of the image file to a destination drive, this function can be used to determine if all the sectors from the source are accurately represented in the image file. If a sector is present on the destination, it can be inferred that the sector is present in the image file. If a sector is not present on the destination there are two possibilities, either the sector is not in the image file or a sector is in the image file, but the restore operation failed to produce an accurate destination. This can usually be easily resolved.

3.2.2 Source and destination interfaces

There are several possible interfaces for attaching a hard drive to a computer. Each interface requires different procedures for access to the sectors on the disk. We omitted testing the earliest hard drive interfaces and restricted test case selection to IDE and SCSI drives accessed either directly or through the BIOS interface. We also consider if the BIOS includes the extensions for hard drives greater that 8GB or if the BIOS can only access smaller drives (*legacy BIOS*).

3.2.3 Relative size

The relative size of the source and destination needs to be varied among the three possible relationships. A smaller destination should generate a warning from the tool. Cases with a larger destination should also consider how the tool deals with the excess sectors. In addition, if the destination is larger and the drive geometry differs from the source, the tool may have a feature to align partitions on cylinder boundaries.

3.2.4 Errors

Disk I/O errors may occur due to bad sectors or a failing disk. There are different failure modes depending on the selected tool operation.

1. Copy of source to destination has two failure modes: read source, write destination.
2. Source to image to destination operation has four failure modes: read source, write image, read image and write destination.
3. Verify image file has a single failure mode: read image.

4. For tools that check the integrity of an image file, the tool should detect if the image file is modified.

3.2.5 Object Type

There are over a hundred different types of disk partition formats. Microsoft operating systems recognize at least 15 partition types. However, many partition types are special purpose (0x3C, Partition Magic recovery), obscure operating system (0x93, Amoeba), laptop hibernation (0xA0, IBM Thinkpad) or UNIX dialect (0xA5, FreeBSD). Because of the nature of a disk copy operation (bit for bit) the format of any partitions does not matter during an operation on an entire disk, however a partition operation could depend on the partition format. The most relevant (widely used) partition types are FAT12, FAT16 (three types), FAT32 (two types), NTFS and Linux native partition (0x83 or ext2).

3.2.6 Remote access

Since this is a very slow operation, infrequently used, no test cases were created using remote access.

4. Results Evaluation Procedure

After a test case has been run, the results must be examined to determine if the results should be accepted or if some further actions are required to complete the test case. The evaluation of results must consider if the apparent results are an accurate reflection of the tool under test. Either a successful or unsuccessful test outcome must be reviewed to ensure that an error has not occurred.

The first issue is: if a test appears to be successful should we accept the result that the tool has produced the expected result for the particular test case? There are several ways that the test could appear to produce expected results without actually doing so. This would usually involve entire steps not running and the measurement of disks that are left in the final state from an earlier successful test. This can be mitigated by always ensuring that the destination disk has been wiped at the beginning of each test. The **diskwipe** log file should show that the correct number of sectors were wiped for the given destination disk.

The second issue is: if a test produces an anomaly and appears to fail, has the tool failed or is something else wrong? Each anomalous test run is reviewed to characterize the anomaly and then a course of action is selected.

1. If a hardware or procedural problem can be found, e.g., disk drive has failed, or improper configuration for the test, then rerun the test with appropriate adjustments.
2. If no hardware or procedural problem can be found and the anomaly matches a known anomaly then accept the anomaly as genuine.
3. If the anomaly is unique then defer a decision until more test cases have been run. These test cases are referred to as *defer until more*.
4. If the anomaly matches an anomaly in the *defer until more* category then both results are examined for common factors. Based on the reviewer's judgment a new *known anomaly* may be established, otherwise the test cases remain *defer until more*.

After all test cases have been run any test cases remaining in the *defer until more* category must be resolved by either accepting the anomaly as genuine or reclassified based on additional investigation as needed.

5. Technical Difficulties

The test procedures sometimes need refinement. As an example, during **dd** testing some results seemed to indicate that the Linux environment was making a change to the source disk. After investigation we found that the problem was actually the test procedure. Under some conditions the creation of a partition on one hard drive was accompanied by a change to another hard drive.

The original procedure for creating a partition on a destination drive was to install both the source drive and the destination drive in a host computer, boot into DOS, **diskwipe** the destination, and create the partition on the destination. This procedure worked without any problems for creating FAT16 and Linux EXT2 partitions. However, when creating FAT32 or NTFS partitions on the destination, a change would be made to the source drive. This gave the appearance that booting into the Linux environment for the test was making a change to the source drive. The procedure was modified to not install the source drive until after the partition was created on the destination drive.

A second difficulty occurred with the bad sector simulation software. The initial use of the bad sector simulation software was with **SafeBack**. The tool worked as expected simulating an error for the *write* command (0x03), but produced a very interesting result the first time it was used to simulate a *read* command (0x02) error.

When the tool is executed a command, a physical drive and a disk address are specified to indicate the condition that should trigger the tool. The tool intercepts all **interrupt 13** commands and any command not matching the trigger condition is passed on to the BIOS for execution. However, any command that matches the trigger condition is not executed and an error code is returned to the calling program. A feedback message is also written to the display indicating that the tool had been triggered.

A test was run with the tool setup to trigger on a *read* command for a selected sector. **SafeBack** was executed and the trigger message appeared on the display screen. In fact, the feedback indicated that **SafeBack** attempted two retries of the *read* command. However, when the source drive and the destination drive were compared there were no differences. This was surprising since there should have been at least one sector different. On further analysis, it was observed that there was a second command, *read long* (0x10), that could be used to read from a hard drive. The test was run again with both the *read* and *read long* commands selected. This time the tool feedback indicated that both commands triggered the tool and when the source was compared to the destination the selected sector was different.

6. Current Status and Future Directions

The initial phase of the project is coming to an end. We have been developing a sound methodology based on recognized international standards. Now that we have a methodology in place we are beginning to produce results.

Test reports for **dd** in the Linux environment and **SafeBack** version 2.18 are in final review for publication on the NIJ web site. The **EnCase** imaging function is currently being tested. The specification for software hard disk write protect has been posted to the web and available for public comment. Currently test cases for software hard disk write protect tools are being developed.

In the near future NIST, with input from the law enforcement community, will develop a specification for *deleted file recovery* tools. This specification will be used in testing the most widely used deleted file recovery tools. In addition, using the software hard disk write protect specification as a basis, a specification for hardware hard disk write protect devices will be developed. Other disk imaging tools such as **dd** in the FreeBSD environment and the **iLook** imaging tool will be tested in 2003.

About the Author

James Lyle (JLYLE@NIST.GOV**)** wrote his first FORTRAN program in 1968 and has been programming ever since. He received a BS in Mathematics (1972) and an MS in Mathematics (1975) from East Tennessee State University; from the University of Maryland at College Park, Dr. Lyle received an MS (1982) and PhD (1984) in Computer Science. Currently he is the project leader for the Computer Forensics Tool Testing project at the National Institute of Standards and Technology. Before joining NIST full time in 1993, Dr. Lyle was a Faculty Associate at NIST and an Assistant Professor at the University of Maryland Baltimore County.