Ten years of computer forensic tool testing[1]

By

Barbara Guttman, James R. Lyle and Richard Ayers

The Computer Forensic Tool Testing (CFTT) project at the National Institute of Standards and Technology (NIST) has been active since 2000. The project develops methodologies for testing computer forensic software tools by the creation of general tool specifications, test procedures, test criteria, and test data sets. 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 capabilities of such tools. Our approach for testing computer forensic tools is based on well-recognized international methodologies for conformance testing.

Introduction [Heading type A]

In 1999, members of US law enforcement involved in investigating computer crime approached NIST about the verification of the tools used to acquire and analyze digital evidence. This was partially motivated by the US Supreme Court in *Daubert v. Merrell Dow Pharmaceuticals*, 509 U.S. 579 (1993). The decision established four criteria that a trial judge may use to assess the admissibility of expert witnesses' scientific testimony. One of the four criteria involved testing the theory or technique used by the expert witness. The CFTT project was created to provide independent testing of the software tools used in digital forensics and thereby address the need for the testing of tools. The National Institute of Justice (NIJ) published the first test report in August 2002. Since then, as of August 2010, a total of 75 test reports have been published.

[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.

The Computer Forensics Tool Testing (CFTT) program is a joint project of the National Institute of Justice (NIJ), the Department of Homeland Security (DHS), and the National Institute of Standards and Technology's (NIST's) Law Enforcement Standards Office (OLES) and Information Technology Laboratory (ITL). CFTT is supported by other organizations, including the Federal Bureau of Investigation, the U.S. Department of Defense Cyber Crime Center, U.S. Internal Revenue Service Criminal Investigation Division Electronic Crimes Program, the Bureau of Immigration and Customs Enforcement and U.S. Secret Service. Representatives from each agency form a steering committee that provides project guidance, technical support, and selects tools for testing. The objective of the CFTT program is to provide measurable assurance to practitioners, researchers, and other applicable users that the tools used in computer forensics investigations provide accurate results. Accomplishing this requires the development of specifications and test methods for computer forensics tools and subsequent testing of specific tools against those specifications. A secondary objective is to provide assistance and resources to forensic practitioners for doing their own testing of forensic tools. This is accomplished by making test requirements, test plans, test tools and test data available for general use.

Test results provide the information necessary for developers to improve tools, users to make informed choices, and the legal community and others to understand the tools' capabilities. The CFTT approach to testing computer forensic tools is based on well-recognized methodologies for conformance and quality testing. Specifications and test methods are posted on the CFTT Web site for review and comment by the computer forensics community.[2]

Test methodology [Heading type A]

There are several significant challenges to testing forensic tools. These include the lack of agreement on what capabilities a tool should offer, the lack of agreement on what should be done in some situations, and the rapid evolution of technology. Some tools are designed to do a single task, such as acquire all data on a device. Other tools are designed

[2] http://www.cftt.nist.gov/.

like a Swiss army knife with a multitude of features. Write block devices protect secondary storage from modification by allowing some commands to be passed from the host computer to the storage device while blocking other commands. Different blockers do not agree on what commands should be allowed or blocked. When the CFTT project began there were two widely used interfaces to secondary storage: ATA and SCSI. Ten years later, as technology evolves, testing of tools needs to also accommodate SATA, USB, Firewire and Thunderbolt interfaces.

CFTT organizes testing by forensic function rather than by the specific type of tool. This allows test development to focus on a group of related issues together. Some forensic functions are critical to the integrity of any acquired evidence. If the acquired data is not accurate, then any analysis result is suspect. Initially, CFTT began with two functions: disk imaging and write blocking.

Disk imaging is the acquisition of all digital data present on a secondary storage device. Write blocking is used to control access to secondary storage such that no changes are written to the storage device. Write blocking can be accomplished either in software or with a hardware device. Other forensic functions identified by CFTT for tool testing include mobile device examination, forensic media preparation (erasure for reuse), string searching, live memory acquisition, metadata based deleted file recovery, file carving (signature based deleted file recovery) and first responder triage tools. This is not a complete list of possible functions, just the candidates under active development for testing by CFTT.

The CFTT testing process is directed by a steering committee composed of representatives of the law enforcement community. The steering committee selects tool functions for test method development and the specific tools for testing by members of staff at CFTT. After a forensic function is selected, CFTT takes several steps in preparation for testing:

1. CFTT obtains some of the more widely used tools that offer the forensic function that has been selected.
2. CFTT develops a list tool features related to the selected function offered by the tools. Some features may be designated optional.

3. CFTT, with input from forensic practitioners, develops a set of requirements for each related tool feature, describing the desired tool behavior. A requirement is a statement about something a tool must do. Requirement development becomes challenging if there is no widely held agreement on tool behavior. In this case, the requirement is written such that multiple behaviors are allowed, with the actual behavior noted in the tool test report. Another important consideration when writing requirements is the ability to test each and every requirement.
4. A draft set of requirements is published on the CFTT web site for public comment. After the closure of a comment period, the comments are addressed and the requirements are finalized.
5. CFTT develops a test plan. This is often the most complex and demanding step in the process. Several components need to be developed for the test plan, including test assertions, method for measuring the test assertion, tools to measure conformance of a test run to test assertions, test data sets, tools for the creation of test data and test cases. A test assertion is a simple statement about one condition that must be true after running a test case. Every test assertion must trace back to at least one requirement and every requirement must generate at least one test assertion. If a requirement does not generate any assertions, it cannot be tested and should be removed. A test case usually bundles several test assertions together for a single test run.
6. A draft test plan is published on the CFTT web site for public comment. After the closure of a comment period, the comments are addressed and the test plan is finalized.

Once a test plan has been created, tool testing can begin for tools selected by the steering committee.

1. CFTT obtains the latest version of the selected tool.
2. CFTT compares the features offered by the tool to the features listed in the test plan and selects test cases for each of the features offered by the tool.
3. CFTT develops a set of tool specific procedures for executing each selected test case.

4. CFTT executes the test cases. Any unexpected results are checked and possibly rerun. In some cases the vendor of the tool is contacted to ensure that the tool was used as designed by the vendor.
5. CFTT writes a draft test report documenting the test results.
6. The steering committee reviews the draft test report.
7. The draft test report is sent to the tool vendor for review. This allows the vendor to catch any errors CFTT has made in testing and allows the vendor time to fix any problems found by CFTT before the test report is published.
8. CFTT submits the report to NIJ. NIJ reviews the test report and publishes the report on the NIJ web site.

The final tool test report is intended to 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. A test report is generally divided into five sections. The first section is a summary of the results from the test runs. This section is sufficient for most readers to assess the suitability of the tool for the intended use. The remaining sections of the report describe how the tests were conducted, discuss any anomalies that were encountered and provide documentation of test case run details that support the report summary. Section 2 gives justification for the selection of test cases from the set of possible cases defined in the test plan for the tested tool. The test cases are selected, in general, based on features offered by the tool. Section 3 describes in more depth any anomalies summarized in the first section. Section 4 lists the hardware and software used to run the test cases with links to additional information about the items used. Section 5 contains a description of each test case run. The description of each test run lists all the test assertions used in the test case, the expected result and the actual result.

Table 1 summarizes the number of test reports published for each type of tool tested by year of publication as of July 2011. Seven other test reports are in final editing steps before final publication later in 2011. Six additional tools are currently undergoing testing with test reports to follow either in 2011 or 2012.

Table 1 Number of test reports published by year and tool type

Year	Disk imaging	Write block	Media prep	Mobil device	Total
2002	1				1
2003	3				3
2004	1	4			5
2005		3			3
2006		13			13
2007	1	7			8
2008	6	4		4	14
2009	2	2	1	4	9
2010	2		6	9	17
2011	1			1	2
Total	17	33	7	18	75

Lessons learned and test results by forensic function [Heading type A]

This section describes lessons learned developing requirements, test assertions and test plans along with notable test results, presented by forensic function.

Disk imaging [Heading type B]

Disk imaging is the process of acquiring digital data from a secondary storage device, referred to as the source. The basic process is simple. Read the source and make a copy on a destination device. To make a copy in a forensically sound manner is another story. A typical acquisition usually proceeds as follows:

1. A source storage device is obtained as evidence. This is usually a hard drive or a removable storage device. All devices are accessed through a hardware interface. On older devices this is usually ATA or SCSI. On more recent devices the interface is usually SATA or Thunderbolt. Removable devices are usually USB,

Firewire or eSATA. Other interfaces are possible, but CFTT has not tested any tools using them.

2. The source device must be attached to a computer for copying. The computer may be a single purpose device just for imaging or a general-purpose computer, either a desktop or a laptop. Some protection needs to be in place to prevent any changes to the source. The protection may be in the form of a write block device. The device must be attached through some interface and the host must use some interface to obtain access to the device. The interfaces may be different if there is a write blocker between the storage device and the computer. In this case the write blocker must translate commands from one interface to another.
3. The imaging tool executes to produce a copy of the source. The copy may be in one of two forms, either a clone on to another device, or an image file in some format. Often an image file is compressed to occupy much less space than the source.

Disk imaging testing [Heading type B]

The two fundamental requirements are to acquire the source completely and accurately. Additional requirements include making no changes to the source, making a record of any deviations from completeness and accuracy, and having a method to validate the contents of the acquisition.

The disk imaging requirements and test plan document, called collectively the specification, has gone through three versions. The first version was used to test a single tool and then significantly revised. The second version of requirements and test plan was used for testing four tools. At this point, several shortcomings became apparent and the requirements and test plan were again revised. One of the reasons for the revision was the pace at which the technology changed. The first two specifications were tied to the ATA and SCSI interfaces. Each test case specified the interface required for the test. With the introduction of USB and with Firewire, it became clear that a more flexible approach was needed. The original specification would have a set of test cases such as the following:

01 Make a clone of an ATA drive.

02 Make a clone of a SCSI drive.

03 Make an image of an ATA drive.

04 Make an image of a SCSI drive.

05 Make an image of a FAT partition.

06 Make an image of an NTFS partition.

Etc.

When USB was introduced, there was no test case specified to use that interface. The test plan at least needed to be revised to include cases for the USB interface. However, it would be inconvenient to make frequent revisions to such fundamental documents. Instead, CFTT decided on a more flexible approach by parameterizing the test cases. The new test cases are as follows:

01-X Make a clone of a drive using interface X.

02-X Make an image of a drive using interface X.

03-Y Make an image of a type Y partition.

Etc.

Now test case 01-X can be executed once for each interface the tested tool supports. New interfaces can be introduced with no effect on the requirements and test plan.

Test results [Heading type B]

The following is a list of some of the tool behaviors we have seen. Some behaviors are serious errors, but most are just quirks that the forensic practitioner should take note.

1. Source drive is changed. This can be quite serious, because it calls into question the evidence that is acquired. In an ideal case, a write blocker should always be used. However, the specific situation of the acquisition may preclude such use. Some tools are designed to function without a write block device. These tools are often designed to execute from a Linux boot CD that can be configured to not modify any storage devices that are attached. The configuration of the boot CD is complicated, and if not done correctly, subtle changes can occur. One possibility is if there is a swap partition on the source device. The Linux run environment

will use a swap partition unless the boot CD is configured to ignore any swap partition. Another problem is with NTFS partitions; some Linux device drivers may change NTFS metadata regardless of the mount status of the partition and may make changes even if the partition is not mounted. Note that the tool per se is not the cause of the problem, but rather the run time environment.

2. Acquisition fails to be complete or accurate over a specific interface. The acquisition may be successful over most interfaces, but fail to be accurate or complete over some specific interface or set of conditions. Examples are a tool that imaged a laptop drive in place over a PCMCIA interface, and an older tool imaging a SCSI drive with a specific type of SCSI card installed in the imaging computer.
3. Acquisition is incomplete. Older versions of Linux used a block size of 1024 bytes and did not acquire the last sector of a device or partition if it contained an odd number of sectors. Another situation where an acquisition is often incomplete is when there are hidden sectors present on a storage device. There are two methods for hiding sectors on ATA and SATA drives: host protected area (HPA) and device configuration overlay (DCO). Manipulation of DCO and HPA areas is easy to do, but the knowledge of how to do it is obscure and arcane.
4. NTFS partition acquisition is incomplete and inaccurate. Some tools do not bother to acquire the last few sectors of an NTFS partition because those sectors are not used to contain user data. In one case, a block of sectors that had already been acquired replaced these unused sectors.
5. Readable sectors are omitted from an acquisition if bad sectors are present. If a sector is faulty or unreadable, a tool should notify the user and replace the unreadable data with benign data such as zeros. There are several variations seen on this theme. For drives connected via an ATA interface and imaged from a Linux platform, seven readable sectors surrounding a bad sector are not acquired. Reading a device is done within the operating system in blocks of eight sectors. If one sector within the block is unreadable, the entire block is treated as unreadable and replaced with zeros. If another interface is used instead, a larger, variable size

block is replaced with zeros. In other environments, the data not acquired is not replaced with zeros, but with something of undetermined origin.

6. A restore is incomplete. A common feature of most imaging tools is the capability to restore a previously acquired image file to a secondary storage destination. Some tools that use a Microsoft Windows run time environment may not restore all of the data acquired from the source to the destination. This can happen if the source device and the destination device are identical in size.
7. There are changes to the restored data. If a partition (also known as a logical drive) is restored from a Microsoft Windows run time environment, some of the file system metadata is modified during the restore. These changes can, in some cases, be avoided by removing the destination device as soon as the restore finishes. This prevents the operating system from making the changes to metadata because the actual changes are often not written until the normal shutdown is initiated.

Write blocking [Heading type B]

The basic idea of write blocking is to put a filter between the storage device and any program that might try to write to the storage device. The filter can be either software that monitors access to the interface connected to the protected device, or a hardware device that sits between the computer and the storage device.

The normal way a computer reads and writes to a storage device is to issue a command over an interface connected to the device. There are several interface types, the main ones are ATA and SCSI. Some interfaces, such as USB, use a restricted set of SCSI commands to communicate. Both ATA and SCSI interface command sets have several commands that can be used to write to a storage device, and several read commands. Different operating systems may use different commands to obtain access to secondary storage. This means trying to test a write blocker by using the normal system programs available to a user to try to write to a protected drive is not likely to send more than one or two write commands over the interface. Such a test is incomplete.

Write blocker testing [Heading type B]

The CFTT approach to testing hardware write blockers is to develop a tool that can issue all possible command codes for both the ATA and the SCSI interface. In addition, a special hardware device, a protocol analyzer, is located between the computer and the blocker along with a second protocol analyzer located between the write blocker and the storage device. The protocol analyzers monitor the command traffic both going in and coming out of the write blocker. This gives a clear indication of the commands that are allowed and the commands that are blocked.

A similar approach was used to test software write blockers that monitored the BIOS interface. The early BIOS interface had two write commands: WRITE and WRITE LONG. The software write blocker was able to insert itself just in front of the BIOS to intercept all commands that obtained access to the secondary storage. The write blocker would examine each command and pass along to the device any command that was not a write command; write commands would then be ignored. Later versions of the BIOS interface added a third write command for addressing larger drives.

There were two basic requirements in the first draft of the specification:

01 Allow all read commands.

02 Block all write commands.

The second draft of the specification revised the first requirement as follows:

01 Allow at least one read commands.

Test results [Heading type B]

Some read commands are blocked. The first hardware write blocker tested blocked not only all write commands, but any read command that was dropped from the current interface specification. For example, the ATA command specification has evolved over eight versions, adding new commands and dropping others.

The write blocker replaces some read commands with a different command.

A write command is allowed. This was seen in a BIOS interface software write blocker. When a new write command was added to the BIOS of newer computers, the old write blocker did not know that the new command should be blocked. This highlights a fundamental design decision for write blockers. There are more than just read and write commands implemented on a given interface. Some command codes are not assigned any function currently, but may be assigned to new read or write commands in the future. One design is block writes, allow everything else; the other is allow reads, block anything else. There can be problems with both designs.

Forensic media preparation [Heading type B]

Forensic media preparation is the erasing of a drive for reuse within the organization. This is not to be confused with erasing a drive for disposal or transfer to another organization. The idea is to remove information to prevent cross contamination from one case to another, not try to prevent extraction of previous content by extraordinary means.

Drive wipe testing [Heading type B]

ATA and SATA drives have a special instruction, SECURE ERASE, defined for wiping drives. The implementation is optional. If SECURE ERASE is not implemented, the drive can be overwritten with other content. Some tools offer both methods. Some tools also offer multiple overwrites with user selectable patterns and wipe verification. CFTT did not find either of these options testable in a practical way for most implementations. To test multiple overwrites, the state of the drive would need to be examined at the end of each pass. This could be done, but at the expense of significant effort. Given that a single pass is adequate for the purpose at hand (reuse within the same organization), it would not be cost effective to develop test procedures to test this feature. As for testing post wipe verification, the process would need to be stopped after the wipe phase so that the wipe could be corrupted before starting the verification phase. Otherwise, there is no verification that wipe failure would be detected. As for a tool that writes random data to a drive, it should be pointed out that random data is difficult to distinguish from encrypted data. A tool that claims to write random data might just encrypt existing data unless the tool has a mechanism to provide assurance that the data really is random.

Test results [Heading type B]

SECURE ERASE fails to function. The implementation of SECURE ERASE in particular drives is sometimes not what the tool developer expected and the invocation of the command fails.

Hidden sectors (HPA or DCO) are not erased. Sometimes this is a design decision to ignore hidden sectors. Sometimes the tool would attempt to remove the hidden sectors, but fail to remove the sectors. Sometimes the tool will remove the hidden sectors, but not erase them.

Mobile devices [Heading type B]

The development of mobile device forensic tools and acquisition techniques continues to develop within the field of digital forensics. Mobile subscribers far out number personal computer owners, and studies have shown an increase of mobile device personal data storage compared to personal computers. At the time of writing, four times the number of mobile subscribers exists compared to the owners of personal computers. Higher-end mobile devices present users with advanced features and capabilities similar to those of a personal computer. Mobile devices provide users with the ability to maintain contact information, future appointments, day to day activities, inform us of important news events and provide us with the ability to correspond with friends and family via text message, e-mail, chat and social networking sites. Over time, mobile devices can accumulate a sizeable amount of information about their owner. Data acquired from these devices may be useful in criminal cases or civil disputes.

While the use and sophistication of mobile devices continues, so does the need for the validation of tools. In order for the information acquired from such devices to be admissible in legal proceedings, verification of a tools behavior and strict forensic acquisition methods are paramount. Potentially, one piece of data acquired from a mobile device may play a critical role in shedding light on an incident or possibly criminal activity. The need for rigorous testing conducted on a combination of forensic tools and specific families of mobile devices is critical for providing law enforcement and forensic examiners informative test results yielding known expectations of the behaviour, capabilities and limitations of a tool. Over the past three years, the CFTT project at NIST has tested numerous mobile device forensic tools that are capable of acquiring data from mobile devices operating over Global System for Mobile (GSM) communications and Code Division Multiple Access (CDMA) networks.

Test results [Heading type B]

The CFTT project has currently tested many versions of thirteen mobile device forensic tools capable of acquiring data from SIMs, the internal memory of GSM and CDMA devices. As of 2011, this testing has resulted in a total of nineteen documents providing an overview of the test results.[3] The test reports describe how the tests were conducted and provide documentation of test case run details that support the report summary.

Mobile device forensic tools have continued to improve, providing forensic examiners with acquisition solutions for multiple devices operating over various cellular networks. As new mobile devices continue to enter the market, tool updates often introduce problematic areas in software and hardware. Over the years of testing mobile device forensic tools, anomalies within tools tend to reoccur, such as the following:

1. Lack of Unicode support – Address book entries and/or text messages containing non-ASCII characters are not displayed in their native format.
2. Truncated entries – Long address book entries or text messages over 160 characters are partially acquired.
3. Connectivity issues – Connectivity between the mobile device and the forensic workstation or the mobile device forensic tool is not established.
4. Acquisitions ending in errors – Acquisitions abruptly ending due to connectivity errors between the mobile device and mobile device forensic tool resulting in an unsuccessful acquire.
5. Incorrect date/time stamps – Incorrect date/time stamps for call logs and text messages.
6. Inconsistencies between preview-pane data and generated reports – Data inconsistencies between what is presented in the preview-pane compared to the generated report.

[3] Hyperlinks for each test report are listed at www.cftt.nist.gov/mobile_devices.htm.

7. Subscriber related data not reported (IMEI, MSISDN) – Subscriber data either incorrectly reported or not acquired.
8. Deleted data acquisition – Unsuccessful recovery of recoverable deleted data objects.
9. Internet related data – Unsuccessful acquisition of Internet related data.
10. Application related data – Unsuccessful acquisition of application related data.

Recovery of deleted files [[Heading type B]

The specification for metadata based deleted file recovery tools is currently under development. The recovery tool uses file system metadata that remains after a file is deleted to recover allocated blocks and file attributes. Each file system leaves different pieces of metadata behind, so the tool needs to be tested on each file system and significant versions of the file system. CFTT is focused on the following file systems: FAT12, FAT16, FAT32, NTFS, Ext2, Ext3 and Ext4. The HFS+ file system is ignored because when a file is deleted from HSF+ all metadata is deleted. No formal testing has been completed, but we have some partial results. For example, consider a fragmented file that has been allocated two blocks, X and Y. Suppose the blocks are separated by an allocated file A. The block layout of just those three blocks would be XAY. Three tools have given three different results. Tool 1 returns just block A. Tool 2 returns both blocks X and Y. Tool 3 returns blocks X and A. It can also be shown that if the layout is more complicated, tool 2 is not always correct. If the layout adds block Z from another deleted file such that Z is between A and Y, XAZY, then tool 2 returns XZ as the recovered file.

Errors and Daubert [Heading type A]

In addition to testing as a criterion for admissibility of scientific testimony, Daubert includes an error rate for the technique. At first glance, it would seem useful to try to derive error rates from the test results. However, it is not obvious how to describe the error rates for digital forensic tools. The Daubert decision motivates attempts to establish error rates for digital forensic tools. The legal decision in 1993 by the Supreme Court of the United States indicates four criteria that a trial judge may use to assess the admissibility of expert witnesses' scientific testimony during federal legal proceedings

1. Whether the theory or technique has been tested,

2. Whether the theory or technique has been subjected to peer review and publication,
3. Whether there is a known or potential rate of error and whether standards exist to control the technique's operation, and
4. Whether the technique has general acceptance within the relevant scientific community.

Establishment of an error rate for using digital forensic tools is complicated by the difference between the underlying algorithm or technique and its implementation in software and process. For example, consider picking up a deck of cards that have been dropped in a room. The theoretical error rate for picking up the cards is zero because there is no reason why all the cards cannot be picked up. However, the dealer may introduce an implementation error into the process by not checking under furniture. Providing none of the cards fall under the furniture, there are no errors. The condition of cards under furniture allows the error (not systematically checking under furniture) to be manifest.

For digital forensic software tools there are three broad sources of error that can occur in using a tool:

1. The algorithm intended for the process.
2. The software implementation of the algorithm.
3. The performance of the process by a person in a specific situation.

There are two issues (not one) related to error rates of forensic evidence: (1) does the technique work? (2) Did the expert do it right?

For the first question, the issue is scientific validity of the technique. The intent is to filter out testimony based on bad science and pseudoscience. No astrology, water witching, cold fusion or contact with the spirit world is allowed into court. The error rate for a Daubert hearing is about the science. For example, what is the error rate for using a hash or checksum to determine if two files are identical? If an expert wants to offer an opinion about two files based on a 4-bit CRC checksum, the court should not allow the testimony to be admitted, since the error rate is two high (1 in 8). If an expert wants to offer an opinion about two files having the same content based his own hash algorithm that almost no one else has seen, but some of the people that have seen it claim that it is easy to generate files that have a given hash value, then this too is not admissible (untested, not reviewed, not accepted). On the other hand, the SHA1 algorithm is open for independent review and has a known error rate that is vanishingly small for determining if two files are the same.

The second question is about the specific expert opinion at hand. Was the technique implemented correctly? An error rate can be calculated here too, but with less certainty due to the number of independent variables. For digital evidence, the relevant parameters that make for systematic error have too much variation for a meaningful error rate to be constructed based on empirical testing.

For many forensics applications, the error rate for the algorithm is zero. For example, the copying techniques and algorithms that underlie disk acquisition have an error rate so small that for all practical purposes it is zero. The software implementation may have an error rate as well as the performance of the examiner. For example, the software may not correctly handle drives with an odd number of sectors, or the examiner may fail to follow correct procedure.

Software is more likely to contain systematic errors rather than random errors that can be described using an error rate. Systematic errors are normally seen in software errors (cases where the code is incorrect) and in cases where software is asked to do something it was not programmed to do. The software might appear to act randomly in these situations, but, most likely, the underlying cause can be explained as a software flaw and the behavior is logical; that is, it follows the logic of the (poorly written or misused) software. It is therefore, systematic and does not have an associated error rate.

The main issue is whether the software algorithm is implemented correctly and is appropriate for the given situation. For example, software to copy a drive needs to be able not only to copy, but to be able to handle the type of drive being used, to know what to do if the drive contains unreadable sectors, to know what to do if the drive being copied to is smaller than the source, and many other factors. Within the constraint that the software can handle the situation, the error rate is that of the physical attributes of the hardware and creates an error rate that is infinitesimal and not useful. To address this, the tool must be tested. Therefore, requiring an error rate is not relevant – what is needed is a test report.

Summary and conclusions [Heading type A]

The CFTT project has been writing tool requirements, test plans and testing forensic tools for the last 10 years. In general, the results are correct for the critical functions the tools are asked to do. Some serious errors have been found, and the tool vendors have been able to correct the problems quickly. Most of the problems are just quirky behaviors that can be avoided if the practitioner is aware of them.

Over the next few years CFTT will expand into testing other functional areas such as string searching, live memory acquisition, triage tools and e-mail extraction. Another area will be to try to make the CFTT testing methodology available to forensic laboratories in a form that is easy for a laboratory to test forensic tools in a common manner and to facilitate sharing of test results and test materials.

Barbara Guttman is the manager of the Software Components Group, Software and Systems Division, Information Technology Laboratory at the National Institute of Standards and Technology in the US. She is involved in the Computer Forensics Tool Testing Project, National Software Reference Library, Software Assurance Metrics and Tool Evaluation, and Voting Technology Team.

Dr James R. Lyle is a computer scientist. Before he joined NIST full time in 1993, Dr Lyle was a Faculty Associate at NIST and an Assistant Professor at the University of Maryland Baltimore County. Dr Lyle’s interests include Software Engineering, Computer Graphics, Human Factors, Digital Forensics and Computer Science Education.

Mr Rick Ayers, a graduate of the University of Tulsa (BS and MS in computer science), is a computer scientist in the Information Technology Laboratory at the National Institute of Standards and Technology. A participant in the Cyber Corps program, his current research focuses on mobile device forensics tools and proper acquisition techniques.