
# A Strategy for Testing Hardware Write Block Devices

*By*

**James Lyle**

*From the proceedings of*

The Digital Forensic Research Conference

**DFRWS 2006 USA**

Lafayette, IN (Aug 14th - 16th)

available at www.sciencedirect.com

SCIENCE DIRECT®

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

Digital Investigation

# A strategy for testing hardware write block devices☆

*James R. Lyle*

*National Institute of Standards and Technology (NIST), 100 Bureau Drive, Stop 8970, Gaithersburg, MD 20899-8970, United States*

*Keywords:*
Tool testing
Write blocking
Digital evidence
Computer forensics
Software testing

ABSTRACT

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, and test 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 tools' capabilities. Our approach for testing computer forensic tools is based on well-recognized international methodologies for conformance testing and quality testing. This paper describes requirements and test assertions that make up a strategy for testing hardware write block devices.



## 1. Introduction

The Computer Forensics Tool Testing (CFTT) Project at the National Institute of Standards and Technology is developing methodologies for testing software write block tools and hardware write block devices. A specification for write blocker behavior (HWB, 2004), a test plan (HWB, 2005), and test software are available on the CFTT web site, http://www.cftt.nist.gov/. The basic goal of a write blocker is to allow access to all digital data on a secondary storage device while not allowing any changes to the storage device. The basic strategy for implementing a write blocker is to place a filter between software executing on a host computer and a secondary storage device that is to be protected. The filter then monitors I/O commands sent from the application and only allows commands to the device that make no changes to the device. Such a filter can be implemented either in software or in hardware. The goal of this paper is to discuss our experience in designing test methodologies for testing hardware write block devices.

A hard drive is a device for the storage of digital data. The human user of a hard drive (or other digital storage media) usually views the drive as a place to store information as files. This simple view is not quite complete because, in reality, other information must be placed on the drive to enable retrieval of the information at a later time and describe properties of the stored information. In this paper, we refer to this as *file system meta-data*. These meta-data include objects such as partition tables, inodes, master file tables and so forth. The management of the meta-data objects is usually handled by an operating system running on a host computer with the drive attached. Both the user files and meta-data objects are located on an area of the hard drive called the *user area*. An attempt to directly access areas of the drive outside of the user area by the host computer results in an error. However,

☆ 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.

E-mail address: jlyle@nist.gov

1742-2876/$ – see front matter 
doi:10.1016/j.diin.2006.06.001

once again, there is another layer of data called *device meta-data*. The device meta-data can be accessed indirectly or with vendor defined commands (not usually publicly documented). Examples include device microcode (firmware), device serial number, and so forth.

A basic strategy for testing a hardware write block device is to simply try to write to a drive protected by the device under test. However, results from such a test may be misleading unless care is taken to ensure that the test is complete. A brief overview of hard drive operational details will help identify requirements for testing write block devices.

## 2. Background

Before a hard drive can be used it must be physically attached to a computer. A hard drive is attached to a computer by one of several available physical interfaces. A drive is usually connected by a cable to an interface controller located either on the system motherboard or on a separate adaptor card. The most common physical interface is the ATA (AT Attachment), also called IDE (integrated drive electronics) or EIDE (enhanced IDE) interface. Other common physical interfaces include SATA (Serial ATA), SCSI (small computer system interface), IEEE 1394 (also known as FireWire or i-Link), and USB (universal serial bus). Information on the ATA and SATA command sets can be found at http://www.t13.org/ and information of SCSI, IEEE 1394 and USB command sets for block devices (i.e., hard drives) can be found at http://www.t10.org/.

All access to a drive is accomplished by commands sent from a host computer to a drive through the interface controller. However, since the low level programming required for direct access through the interface controller is difficult and tedious, each operating system usually provides other access interfaces. For example, programs running in the DOS environment can, in addition to direct access via the drive controller, use two other interfaces: DOS service interface (interrupt $0 \times 21$) or BIOS service interface (interrupt $0 \times 13$). The DOS service operates at the logical level of files and records while the BIOS service operates at the physical drive sector level. More complex operating systems, for example, Windows XP or a UNIX variant (e.g., Linux), may disallow any low level interface (through the BIOS or the controller) and only allow user programs access to a hard drive through a device driver, a component of the operating system that manages all access to a device.

Note that changes to drive meta-data may originate from the drive without action by the host.

## 3. Hardware based write blockers

The primary goal of a hardware write blocking device is to prevent any change to data in the user area of a hard drive while allowing access to all data on a hard drive. The write blocker should in general preserve the configuration of a protected drive. Sometimes it may be desirable to change a drive configuration to obtain access to otherwise hidden sectors, for example, such as within an HPA (host protected area). Hardware write block devices usually work by breaking the bus used to attach a hard drive to a host computer into two segments. Instead of a single bus segment between a hard drive and a host there is a bus segment between the host and the blocking device and another bus segment from the blocking device to the hard drive. The two bus segments do not have to use the same protocol. One of the first blocking devices on the market used an SCSI connection to the host computer and an ATA connection the hard drive. Once the blocking device is connected it can intercept a command from the host and select a desired course of action for the command. The most common actions are the following:

- The device forwards the command to the hard drive.
- The blocking device substitutes a different command to the hard drive. This is the case if the blocking device uses different bus protocols for communication with the host and hard drive.
- The device simulates the command without actually forwarding the command to the hard drive. For example, the blocking device may already know the size of the hard drive and rather than asking the hard drive again if a request for the size of the hard drive is sent from the host, the device may just return the answer directly to the host.
- If a command is blocked, the device may return either *success* or *failure* for the blocked operation. However, returning *failure* may sometimes cause the host computer to lock up for some commands issued by some operating systems.

Hard drive standards are not static. The standards for the ATA drives are maintained at http://www.t13.org and continue to evolve. There have been seven releases of the ATA specification, and the eighth is in development.

- ATA-1 X3T10/791D Revision 4c 1994 (ATA-1, 1994).
- ATA-2 X3T10/0948D Revision 4c March 18, 1996 (ATA-2, 1996).
- ATA-3 X3T13 2008D Revision 7b January 27, 1997 (ATA-3, 1997).
- ATA/ATAPI-4 T13/1153D Revision 18 August 19, 1998 (ATA/ATAPI-4, 1998).
- ATA/ATAPI-5 T13/1321D Revision 3 February 29, 2000 (ATA/ATAPI-5, 2000).
- ATA/ATAPI-6 T13/1410D Revision 3 October 30, 2001 (ATA/ATAPI-6, 2002).
- ATA/ATAPI-7 V1 T13/1532D Revision 4b April 21, 2004 (ATA/ATAPI-7, 2004).
- ATA/ATAPI-8 ATA Command Set Rev 3b March 21, 2006 (ATA/ATAPI-8, 2006).

Of the 256 possible command codes in the ATA protocol, what action should a blocking device take for each code? In the ATA-7 standard, of the possible command codes, about 70 are defined as general use commands that are not reserved, retired, obsolete or vendor specific. In addition, there are more than 30 retired or obsolete command codes that were defined in earlier standards. There have been 21 distinct write commands defined in the first seven ATA standards. Only four commands are defined in all seven standards: WRITE BUFFER (E8h), WRITE SECTORS with retries (30h), WRITE MULTIPLE (C5h), and WRITE DMA (CAh). Three standards introduced

new write commands beyond the original commands and six other write commands have been discontinued. It is critical to observe that the ATA command set changes over time.

We conducted an experiment to observe the actual commands issued during startup on three different computer and BIOS combinations: Dell Phoenix 4.0 Rel 6.0, Micron Phoenix 4.0 Rel 6.0, and Nexar Award V4.51PG. A protocol analyzer[1] was used to capture ATA bus activity during startup and shutdown. We observed the following commands issued from the BIOS to drive 0 of the primary ATA channel: 10 = RECALIBRATE, 90 = EXEC DRIVE DIAG, 91 = INIT DRV PARAMS, C6 = SET MULTPLE MOD, E3 = IDLE, EC = IDENTIFY DRIVE, and EF = SET FEATURES 03 = Set Transfer Mode. Note that for these systems, the BIOS did not issue any write commands to the hard drive.

We again used the protocol analyzer in a second experiment to observe commands issued by several operating systems (DOS 6.22, PCDOS 6.3, FreeBSD 5.21, RedHat Linux 7.1, Red Hat Personal Desktop Linux 9.1, Windows 98, Windows NT 4.0, Windows 2000, and Windows XP Pro), during boot and shutdown. Windows 98 operating system used the WRITE (30), Windows NT used WRITE MULTIPLE (C5h), the UNIX variants (Linux and FreeBSD) used WRITE DMA (CA), and the later Windows operating systems also used WRITE DMA (CA). Neither PCDOS 6.3 nor DOS 6.22 issued any write commands as part of startup or shutdown. Note that the newer operating systems have shifted away from the WRITE (30) command to the faster WRITE DMA (CA) command.

The conclusion from the second experiment is that while there are a variety of write commands available to a device driver not all commands are used at any one time. This implies that any testing of a hardware write blocking device must be careful to cover all possible commands. For example, the WRITE DMA EXT (35h) command, used to access drives requiring 48 bit sector addressing, was not observed in the second experiment because the hard drive used in the experiment only contained 28 bit sector addresses.

## 4. Developing requirements

Development of a useful set of testable requirements is often an iterative process. This section describes how a set of requirements for write block devices evolved from a basic statement of what the device should do to a formal set of requirements.

### 4.1. *Proposal 1: a write blocker should block all write commands sent to a hard drive*

This proposal is simple and to the point. However, devices that satisfy proposal 1 may not be useful for forensic applications. For example, *write command* is not clearly defined. If *write command* is defined as any command with the word *write* in the command name then there are other commands that can change the contents of the hard drive, e.g., SECURITY ERASE UNIT.

[1] Data Transit Corporation Bus Doctor Protocol Analyzer.

To avoid ambiguity, we developed the following command classification scheme:

Each interface command represents one or more distinct operations. Every operation must exist in only one category. The commands of each interface and their associated operations can be partitioned into the following *command operation categories*:

- *Modifying*: any operation that:
  1. directly causes a modification
  2. could *potentially* cause a modification
  3. is a necessary pre-requisite for a modification
  4. is undefined in the interface specifications
  5. changes how the storage device is presented to the host
  6. changes any of the storage device's configurable parameters

  In other words, a modifying operation is a command that writes to the user area (WRITE DMA), is publicly undocumented (vendor specific), is part of a multi-command sequence that ultimately writes to the user area (SECURITY ERASE PREPARE), is undefined currently (but might become a write command later), changes the visibility of sectors in the user area (SET MAX ADDRESS), or modifies the device meta-data directly (DEVICE CONFIGURATION SET).

- *Read*: any operation that requests data, which are stored at specific locations on a storage device's medium and returns that data to the host. A read operation requests one or more blocks of data from the storage device's medium. Each block of data is specified by a location on the medium and a length.

  All sectors of the user area are accessible.
- *Information*: any operation that requests data which are not stored on a storage device's medium and returns that data to the host.

  These are commands such as IDENTIFY DEVICE and DEVICE CONFIGURATION IDENTIFY that return information about the drive.
- *Other non-modifying*: any operation not existing in any of the other operation categories that requests the storage device to perform a nondestructive action.

  This is a category for any commands that do not fit anywhere else, e.g., SEEK. This leads to proposal 2.

### 4.2. *Proposal 2: a write blocker should block all modifying commands sent to a hard drive*

Proposal 2 is an improvement. The definition of the commands to be blocked is less ambiguous. However, several other issues arise. The wording of proposal 2 implies a specific model for write block device operation: the host issues a command, the blocker examines the command, the blocker either returns to the host with no action (blocks the command) or passes the command on to the drive unchanged. Blockers that bridge between two bus protocols, e.g., USB from host to blocker and ATA from blocker to drive, use a different model. The blocker substitutes corresponding commands from the ATA protocol (sent to the drive) for the commands

issued from the host using the USB protocol. The requirement needs to allow for a blocker that substitutes one command for another.

The final published requirement was the following:

HWB-RM-01 An HWB shall not, after receiving an *operation of any category* from the host, nor at any time during its operation, transmit any *modifying category operation* to a protected storage device.

The final version of the requirement allows a write blocker to substitute a different command or commands or cache command results so long as no modifying commands are issued.

While devices that conform to HWB-RM-01 protect a drive from modification, for the device to be useful in a forensic application additional requirements are necessary. In brief, the blocker should allow reading of the entire drive, report the size of the drive correctly and report any drive errors (bad sectors). Three additional requirements are derived from these issues.

HWB-RM-02 An HWB, after receiving a *read category operation* from the host, shall return the data requested by the read operation.

HWB-RM-03 An HWB, after receiving an *information category operation* from the host, shall return a response to the host that shall not modify any access-significant information contained in the response.

HWB-RM-04 Any error condition reported by the storage device to the HWB shall be reported to the host.

The last three requirements ensure that the entire user area is accessible, that a blocker may modify some information returned by commands such as IDENTIFY DEVICE but nothing related to accessing the user area (e.g., size of the user area) and the blocker does not hide disk errors from the host.

Some issues were considered either beyond the scope of what should be addressed or did not warrant special consideration:

- The blocker is assumed to be installed correctly.
- The blocker is assumed to meet any electrical requirements for attached buses.
- There are no special considerations with RAID configurations.

## 5. Developing test cases

The test cases are developed in three stages. First the requirements are restated as *test assertions*. A *test assertion* is a testable atomic statement. Second one or more measurement methods are developed for each test assertion. Third, test cases are constructed that allow observation of blocker behavior under likely conditions.

HWB-AM-01. The HWB shall not transmit any modifying category operation to the protected storage device.

HWB-AM-02. If the host sends a read category operation to the HWB and no error is returned from the protected storage device to the HWB, then the data addressed by the original read operation are returned to the host.

HWB-AM-03. If the host sends an information category operation to the HWB and if there is no error on the protected storage device, then any returned access-significant information is returned to the host without modification.

HWB-AM-04. If the host sends an operation to the HWB and if the operation results in an unresolved error on the protected storage device, then the HWB shall return an error status code to the host.

HWB-AM-05. The action that an HWB device takes for any commands not assigned to the modifying, read or information categories is defined by the vendor.

Assertion HWB-AM-05 was created to allow for diversity of design among write block device vendors. For some commands there is difference of opinion about blocking or allowing the commands. This assertion allows each command to be tried and the results included in a test report.

### 5.1. *Measuring conformity to assertions*

This section describes the methodology for measurement of the conformity of an HWB device to assertions. Each assertion has one or more measurement methodologies defined. Each defined methodology depends on the combination of what must be measured and the measurement tools available for each test case. The complete measurement of conformity requires two critical components: a method for generating commands on the protected bus and a method for determining the action of the HWB.

Some assertions may be measured in more than one way. For example, measuring the assertion that the HWB does not send any modifying command to the protected storage device can be done in more than one way. A known sequence of commands can be sent from the host to the HWB protecting a storage device. Then either the commands sent from the HWB to the protected device can be monitored by a protocol analyzer or the protected device can be examined (either directly or by comparing a pre-test hash to a post-test hash) for changes. Both methods determine if the HWB protects the actual device used for the test, however, using the protocol analyzer records the HWB action for all commands sent. For example, if a storage device that only supports up through the ATA-4 protocol was used in a test and the HWB under test only blocked write commands defined up through the ATA-5 protocol then the HWB might (incorrectly) allow write commands defined in the ATA-6 and 7 protocols to be transmitted to the storage device with no detectable change occurring to the device. The protocol analyzer, however, if available, would report all commands transmitted by the HWB device.

Commands may be generated by a combination of operating system software, test harness software or by widely used

forensic software. Some methods for generating commands are limited in the completeness of the command set generated.

The most complete method of observing bus activity is a protocol analyzer. A protocol analyzer can capture all bus activity between the write block device and the protected storage device or between the test host and the HWB. If a protocol analyzer is not available for the input bus or output bus of an HWB under test, alternative measurement procedures are defined. The alternative measurement methodology puts some limitations on the test results. These limitations are usually due to the difficulty of generating all possible commands without special software.

If more complete command generation software or additional protocol analyzer components become available after a test report is issued for a device, the more complete tests can be executed and a supplement to the original report can be produced.

Four categories of measurement methodology are defined based on availability of command generators and protocol analyzers. The protocol analyzer may be required on either the host to blocker bus segment, the blocker to drive bus segment or both bus segments.

*Operational*: neither a command generator nor a protocol analyzer is required for operational tests. In this method, widely used forensic tools and operating system environments generate commands. The main advantage of this method is that commands are generated by the actual conditions under which the HWB device functions. There are two limitations to this method: commands tested are limited to ones generated by operating systems and selected forensic applications used in the test and it is unknown which commands are actually generated. This category represents the minimal level of testing required to provide assurance that a write block device provides adequate protection from undesired change to a storage device.

*Observational*: if only a protocol analyzer is available, then the observational methodology is used. This method runs the same tools to generate commands as the operational test but the protocol analyzer monitors the actual commands generated and records the behavior of the blocking device. This method documents the HWB behavior for all commands generated. The limitation of this method is the commands tested are limited to ones generated by operating systems and selected forensic applications used in the test. In other words, although the set of generated commands is known, the entire possible command set may not be generated.

*Indirect*: this methodology is used if only a command generator (and no protocol analyzer for the blocker to device bus segment) is available for the test case. This limits the scope of testing to commands that can produce an observable result on the storage device or return verifiable data to the host. For testing commands that write to a device or change the device configuration, this requires a sophisticated command generator that produces configuration and content changes that can be detected by examination of the storage device. For read and information commands, the returned data or information must be verifiable. If a protocol analyzer is available for the host to blocker bus segment, it may optionally be used to record the actual commands sent from the host.

*Detailed*: this methodology is used if both a command generator and a protocol analyzer are available. This category of testing is only needed for determining the exact set of commands blocked by the HWB. All possible command code including defined, retired, obsolete, reserved and vendor specific command codes, are sent from the host and a protocol analyzer records the behavior of the blocker.

### 5.2. *Measurement methodology*

This section describes the methodology for measuring conformity of the HWB device to each defined assertion. Not all measurement categories are required for every assertion.

HWB-AM-01 The HWB shall not transmit any modifying category operation to the protected storage device.

*Detailed*: the command generator sends all feasible command codes to the HWB device. The protocol analyzer records a trace of all command activity between the HWB device and the protected device. Any commands classified as modifying are reported.

*Indirect*: the command generator sends modifying commands designed to write specific information in known locations to the protected device. After a test run, the protected device is examined to determine if the data stored on the protected device were changed. Any changes are reported.

*Observational*: a variety of forensic tools running in commonly used operating system environments generate commands to do tasks that are known to write to a storage device, e.g., attempting to create an image file on the protected device, and a protocol analyzer records a trace of all command activity between the blocking device and the protected device. Any commands classified as modifying are reported along with a trace of all commands actually generated.

*Operational*: a variety of forensic tools running in commonly used operating system environments generate commands to do tasks that are known to write to a storage device. A pre-test hash matching a post-test hash verifies that no changes occurred to the protected device.

HWB-AM-02 If the host sends a read category operation to the HWB and no error is returned from the protected storage device to the HWB, then the data addressed by the original read operation are returned to the host.

*Detailed:* not applicable.

*Indirect*: the command generator sends all feasible read command codes to the blocking device to read known data from the protected device. The returned data are compared to known content already placed on the storage device. Any differences are reported.

*Observational*: a variety of forensic tools in commonly used operating system environments are used to generate commands to acquire a storage device. A protocol analyzer records a trace of all command activity between the HWB device and the protected device. A pre-test hash and a hash of data acquired through the HWB are used to verify that the protected device is accurately (the data on the storage device are acquired without modification) acquired. Either a second run allows the protocol analyzer to be attached between the host computer and the HWB to record a trace of commands generated or a second protocol analyzer records a trace of all commands actually generated for reporting.

*Operational*: a variety of forensic tools in commonly used operating system environments are used to generate commands to acquire a storage device. A pre-test hash and a hash of data acquired through the HWB are used to verify that the protected device was accurately (the data on the storage device are acquired without modification) acquired.

HWB-AM-03 If the host sends an information category operation to the HWB and if there is no error on the protected storage device, then any returned access-significant information is returned to the host without modification.

*Detailed*: not applicable.

*Indirect*: the command generator sends all information category commands to a protected device of known size and configuration. The access-significant information is checked against known values obtained without the HWB present.

*Observational*: forensic tools in commonly used operating system environments are used to acquire a storage device. If the storage device is completely (all user accessible sectors) acquired this implies that the size of the device and any other access-significant information is reported correctly to the host from the HWB. The protocol analyzer located between the host and the HWB records the actual commands generated.

*Operational*: forensic tools in commonly used operating system environments are used to acquire a storage device. If the storage device is completely (all user accessible sectors) acquired this implies that the size of the device and any other access-significant information is reported correctly to the host from the HWB.

HWB-AM-04 If the host sends an operation to the HWB and if the operation results in an unresolved error on the protected storage device, then the HWB shall return an error status code to the host.

*Detailed*:not applicable.

*Indirect*: a command generator attempts to read from an invalid sector and reports the result.

*Observational*: not applicable.

*Operational*: not applicable.

HWB-AM-05 The action that an HWB device takes for any commands not assigned to the modifying, read or information categories is defined by the vendor.

*Detailed*: the command generator sends all feasible command codes to the blocking device. The protocol analyzer records the behavior of the HWB for each command sent from the host. The protocol analyzer monitors the traffic between the host and the HWB.

*Indirect*: not applicable.

*Observational*: not applicable.

*Operational*: not applicable.

## 6. Test cases

This section describes nine test cases that use several methodologies to determine HWB device actions for commands that might change a storage device, and verify that if a storage device is protected with an HWB then data stored on a protected drive and data about the drive can be obtained.

Test cases HWB-01 through HWB-04 address the write blocker response to attempts at writing to the protected drive. For testing a given blocker, one or two of the four cases would be selected depending on available test tools.

Test cases HWB-05 through HWB-07 address the write blocker response to attempts to read from the protected drive.

Test case HWB-08 addresses information returned about a hard drive.

Test case HWB-09 addresses the blocker response to an error returned by the hard drive.

HWB-01 Identify commands blocked by the HWB. This case uses a protocol analyzer and a general command generator.

HWB-02 Identify modifying commands blocked by the HWB. This case uses a write command generator to try to write a unique message to a unique location for each defined write command.

HWB-03 Identify commands blocked by the HWB while attempting to modify a protected drive with forensic tools. This case uses a protocol analyzer to record the commands generated and blocked by attempting to write to a drive with either a forensic tool or an operating system command.

HWB-04 Attempt to modify a protected drive with forensic tools. This case attempts to write to a drive with either a forensic tool or an operating system command. Any modifications to the protected drive are detected by comparing a pre-test hash of the drive to a post-test hash of the drive.

HWB-05 Identify read commands allowed by the HWB. A read command generator is used to try to read known data from a drive using each defined read command.

HWB-06 Identify read and information commands used by forensic tools and allowed by the HWB. Use a forensic tool to read an entire drive with a protocol analyzer recording the actual commands generated by the forensic tool.

HWB-07 Read a protected drive with forensic tools. Use a forensic tool to read an entire drive.

HWB-08 Identify access-significant information unmodified by the HWB. Use a tool to generate a request for drive size and verify that the correct size is reported.

HWB-09 Determine if an error on the protected drive is returned to the host. Generate an error at the drive by attempting to read a sector beyond the end of the drive.

This strategy has been used to test several write blocker bus configurations. Most of the tested blockers use an ATA bus between the blocker and the drive. The connection is either over a cable or by directly plugging the blocker into the device. This makes a difference in selected test cases because the protocol analyzer cannot be attached if the drive and blocker plug into each other. Development of an adaptor is under investigation. The host to blocker connection is either an ATA bus or an *apparent SCSI bus*, i.e., a bus that appears to be an SCSI bus to the user at the programming level, but such a bus could be USB, firewire or an actual SCSI bus.

For a configuration where host to blocker bus is ATA and blocker to drive is ATA via cable, test cases HWB-01, HWB-03, HWB-06, HWB-08 and HWB-09 were used. An ATA

command generator program issues each of the possible ATA command codes. A protocol analyzer can then observe traffic on the blocker to drive bus segment to determine blocker action for each command code. Some notable behaviors were observed as follows:

1. Some blockers cached the results of the IDENTIFY DEVICE command so that after the first time the command was issued from the blocker to the drive, the command was never issued again to the drive. Whenever the host afterwards issued the command the cached result was returned. This had the side effect that if a SET MAX ADDRESS command was used to change the total number of sectors on the device, the value for number of sectors on the drive returned for the IDENTIFY DEVICE command was not updated to reflect the new value established by the SET MAX ADDRESS command.
2. Some blockers substituted the READ DMA command for the READ MULTIPLE command.
3. Some blockers allowed the FORMAT TRACK command. This command cannot modify drive contents with meaningful data but can erase the drive. The command has been dropped since the ATA-4 specification and is designated *obsolete*.
4. Some blockers allow the volatile SET MAX ADDRESS command to manipulate an HPA temporally while blocking the non-volatile variation.

For a configuration where the host to blocker bus is either USB or firewire and the blocker to drive is ATA via cable, test cases HWB-01, HWB-03, HWB-06, HWB-08 and HWB-09 were used. An SCSI command generator program issues each of the possible SCSI command codes. A protocol analyzer can then observe traffic on the blocker to drive bus segment to determine blocker action for each command code.

For a configuration where the host to blocker bus is either USB or firewire and the blocker to drive is ATA directly attached, test cases HWB-02, HWB-04, HWB-05, HWB-07, HWB-08 and HWB-09 were used. An SCSI command generator program issues each of the possible SCSI command codes. Because a protocol analyzer cannot then observe traffic on the blocker to drive bus segment the blocker action must be inferred.

Some of the newer blocker designs and firmware will require small revisions to the specification. For example, for some blockers using a bus other than ATA to connect to the host computer, recent firmware updates automatically do a volatile removal of an HPA if present on an ATA drive. This should be easy to accommodate with small revisions to the specification.

## 7. Conclusions

We have used this strategy with success to test a number of different write block devices. The tests are designed to be used by a variety of organizations and to accommodate diverse blocker designs. The specification is flexible and can easily evolve as the technology changes. Test reports have been published for each tool tested at http://www.ojp.usdoj.gov/nij/topics/ecrime/cftt.htm. The test strategy continues to evolve and be refined.

## REFERENCES

0791M AT Attachment Interface for Disk Drives (ATA-1); 1994.
0948D AT Attachment Interface with Extensions (ATA-2); 1996.
2008D AT Attachment-3 Interface (ATA-3); 1997.
1153D AT Attachment-4 with Packet Interface Extension (ATA/ATAPI-4); 1998.
1321D AT Attachment with Packet Interface-5 (ATA/ATAPI-5); 2000.
1410D AT Attachment with Packet Interface-6 (ATA/ATAPI-6); 2002.
1532D AT Attachment with Packet Interface-7 Volume 1-Register Delivered Command Set, Logical Register Set (ATA/ATAPI-7 V1); 2004.
1699-D AT Attachment 8-ATA/ATAPI Command Set (ATA8-ACS); 2006.
Hardware Write Blocker Device (HWB) Specification Version 2.0; 2004.
Hardware Write Blocker (HWB) Assertions and Test Plan; 2005.

**Dr. Lyle** is currently the project leader for the Computer Forensics Tool Testing (CFTT) project at The National Institute of Standards and Technology (NIST). He wrote his first FORTRAN program in 1968 and has been programming ever since. He received a B.S. in Mathematics (1972) and an M.S. in Mathematics (1975) from East Tennessee State University; from the University of Maryland at College Park, Dr. Lyle received an M.S. (1982) and PhD (1984) in Computer Science.

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.