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. 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 designing test methodologies for write blockers, and describe the methodology developed for testing hardware write block devices.

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 adapter card. The most common physical interface is the ATA (AT Attachment) or IDE (Integrated Drive Electronics) interface. This interface includes variants such as EIDE (Enhanced IDE) or ATA-2, ATA-3, etc. Some other physical interfaces include SCSI (Small Computer System Interface), IEEE 1394 (also known as FireWire or i-Link), and USB (Universal Serial Bus).

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 0x21) or BIOS service interface (interrupt 0x13). 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.

3. Hardware Based Write Blockers

The primary goal of a hardware write blocking device is to prevent any change to stored data on a hard drive while allowing access to all data on a hard drive. Hardware write blockers 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 a 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 operationg 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, see Table 1, and an eight is in development.

Table 1 ATA Standards Publication History

Last Draft Standard	Approximate Publication Data
ATA-1 X3T10/791D Revision 4c	1994
ATA-2 X3T10/0948D Revision 4c	March 18, 1996
ATA-3 X3T13 2008D Revision 7b	January 27, 1997
ATA/ATAPI-4 T13/1153D Revision 18	August 19, 1998
ATA/ATAPI-5 T13/1321D Revision 3	February 29, 2000
ATA/ATAPI-6 T13/1410D Revision 3	October 30, 2001
ATA/ATAPI-7 V1 T13/1532D Revision 4b	April 21, 2004

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 commands codes that were defined in earlier standards. Consider the write commands listed in Table 2. Note that only four commands are defined in all seven standards. Also note that three standards introduced new write commands beyond the original commands and three standards discontinued six other write commands. The critical observation is that the command set changes over time.

Table 2 History of ATA Write Commands

1	2	3	4	5	6	7	Cmd	Name
N	N	N	N	N	N	S	3Ah	WRITE STREAM DMA EXT
N	N	N	N	N	N	S	CEh	WRITE MULTIPLE FUA EXT
N	N	N	N	N	N	S	3Eh	WRITE DMA QUEUED FUA EXT
N	N	N	N	N	N	S	3Dh	WRITE DMA FUA EXT
N	N	N	N	N	N	S	3Bh	WRITE STREAM EXT
N	N	N	N	N	S	S	34h	WRITE SECTOR(S) EXT
N	N	N	N	N	S	S	3Fh	WRITE LOG EXT
N	N	N	N	N	S	S	39h	WRITE MULTIPLE EXT
N	N	N	N	N	S	S	36h	WRITE DMA QUEUED EXT
N	N	N	N	N	S	S	35h	WRITE DMA EXT
N	N	N	S	S	S	S	CCh	WRITE DMA QUEUED
S	S	N	N	N	N	N	E9h	WRITE SAME
S	S	S	N	N	N	N	33h	WRITE LONG (w/o retry)
S	S	S	N	N	N	N	32h	WRITE LONG (w/ retry)
S	S	S	N	N	N	N	3Ch	WRITE VERIFY
S	S	S	S	N	N	N	31h	WRITE SECTOR(S)
S	S	S	S	N	N	N	CBh	WRITE DMA
S	S	S	S	S	S	S	E8h	WRITE BUFFER
S	S	S	S	S	S	S	30h	WRITE SECTOR(S)
S	S	S	S	S	S	S	C5h	WRITE MULTIPLE
S	S	S	S	S	S	S	CAh	WRITE DMA

We conducted a an experiment to observe the actual commands issued during startup on three different computers. A protocol analyzer[1] was used to capture ATA bus activity during startup and shutdown by three combinations of BIOS and computer. We observed the commands presented in Table 3 issued from the BIOS to drive 0 of the primary ATA channel. Note that for these systems, the BIOS did not issue any write commands to the hard drive.

Table 3 Commands Issued from BIOS During Startup

Host and BIOS	Cmd
Dell Phoenix 4.0 Rel 6.0	10=RECALIBRATE
Dell Phoenix 4.0 Rel 6.0	90=EXEC DRIVE DIAG

[1] Data Transit Corporation Bus Doctor Protocol Analyzer

Host and BIOS	Cmd
Micron Phoenix 4.0 Rel 6.0	90=EXEC DRIVE DIAG
Nexar Award V4.51PG	90=EXEC DRIVE DIAG
Dell Phoenix 4.0 Rel 6.0	91=INIT DRV PARAMS
Micron Phoenix 4.0 Rel 6.0	91=INIT DRV PARAMS
Nexar Award V4.51PG	91=INIT DRV PARAMS
Dell Phoenix 4.0 Rel 6.0	C6=SET MULTPLE MOD
Micron Phoenix 4.0 Rel 6.0	C6=SET MULTPLE MOD
Nexar Award V4.51PG	C6=SET MULTPLE MOD
Dell Phoenix 4.0 Rel 6.0	E3=IDLE
Micron Phoenix 4.0 Rel 6.0	E3=IDLE
Nexar Award V4.51PG	E3=IDLE
Dell Phoenix 4.0 Rel 6.0	EC=IDENTIFY DRIVE
Micron Phoenix 4.0 Rel 6.0	EC=IDENTIFY DRIVE
Nexar Award V4.51PG	EC=IDENTIFY DRIVE
Dell Phoenix 4.0 Rel 6.0	EF=SET FEATURES 03=Set Transfer Mode
Micron Phoenix 4.0 Rel 6.0	EF=SET FEATURES 03=Set Transfer Mode
Nexar Award V4.51PG	EF=SET FEATURES 03=Set Transfer Mode

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 (Table 4). 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.

Table 4 Write Commands Issued During Startup and Shutdown

Host/OS	Src	Count	Cmd
FreeBSD5.2.1	Boot	196	CA=Write DMA
FreeBSD5.2.1	Boot	1	30=WRITE W/ RETRY
FreeBSD5.2.1	Shutdown	104	CA=Write DMA
RH7.1	Boot	759	CA=Write DMA
RH7.1	Login	166	CA=Write DMA
RH7.1	Shutdown	297	CA=Write DMA
RH9PD.1	Boot	763	CA=Write DMA
RH9PD.1	Login	186	CA=Write DMA
RH9PD.1	Shutdown	402	CA=Write DMA
W98DS3	Boot	55	CA=Write DMA
W98DS3	Boot	58	30=WRITE W/ RETRY
W98DS3	Login	22	30=WRITE W/ RETRY
W98DS3	Shutdown	76	30=WRITE W/ RETRY

Host/OS	Src	Count	Cmd
W98dsbd	Boot	10	30=WRITE W/ RETRY
W98dsbd	Boot	48	CA=Write DMA
Win2KPro	Boot	424	CA=Write DMA
Win2KPro	Login	277	CA=Write DMA
Win2KPro	Shutdown	269	CA=Write DMA
Win98SE	Boot	65	30=WRITE W/ RETRY
Win98SE	Shutdown	90	30=WRITE W/ RETRY
WinNT4.0	Boot	452	C5=WRITE MULTIPLE
WinNT4.0	Login	520	C5=WRITE MULTIPLE
WinNT4.0	Shutdown	102	C5=WRITE MULTIPLE
WinXPPro	Boot	967	CA=Write DMA
WinXPPro	Shutdown	272	CA=Write DMA

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.

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*.

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
- **Read:** Any operation that requests data which is 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.

- **Information:** Any operation that requests data which is not stored on a storage device's medium and returns that data to the host.
- **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 leads to **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: host issues a command, blocker examines the command, 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 A 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.

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 A HWB, after receiving a *read category operation* from the host, shall return the data requested by the read operation.

HWB-RM-03 A 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.

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 is 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 a 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 HWB to assertions. Each assertion has one or more measurement methodologies defined. Each defined methodology depends on the combination of what must be measured and 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-5, 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 may be limited in the completeness of the command set generated.

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 a HWB under test, alternative measurement procedures are defined. The alternative measurement methodology may put some limitations on the test results. 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.

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 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 commends 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 not a protocol analyzer) 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, 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. Every possible command code is sent and the behavior of the blocking device is recorded by a protocol analyzer.

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.

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 was 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 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.

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 is 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 is 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 is 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 is acquired without modification) acquired.

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.

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.

The action that a 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. It is placed 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 a HWB then data stored on a protected device and data about the device can be obtained.

Nine test cases were defined.

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.

7. Conclusions

We have used this strategy with success to test a number of different write block devices. Test reports have been published at http://www.ojp.usdoj.gov/nij/topics/ecrime/cftt.htm for each tool tested. The test strategy continues to evolve and be refined.