Chapter 11

DISK DRIVE I/O COMMANDS AND WRITE BLOCKING

James Lyle, Steven Mead and Kelsey Rider

Abstract A write blocker allows read-only access to digital data on a secondary storage device by placing a hardware or software filter between the host computer and the storage device. The filter monitors I/O commands sent from the application on the host computer, only allowing commands to the device that make no changes to its data. This paper examines the I/O commands used to access secondary storage devices and discusses their implications for BIOS-based and hardware-based write blockers.

Keywords: Data acquisition, forensic tool testing, write blockers

1. Introduction

A write blocker allows access to digital data on a secondary storage device while not allowing any changes to be made to the data on the device. This is implemented by placing a hardware or software filter between the software executing on a host computer and the secondary storage device that is to be protected. The filter monitors all the I/O commands sent from the application on the host machine and only allows commands to the device that make no changes to its data. This paper examines the I/O commands used to access secondary storage devices and discusses their implications for write blockers.

This research is part of the Computer Forensics Tool Testing (CFTT) Project at the National Institute of Standards and Technology (NIST), which is developing methodologies for testing forensic tools and devices. CFTT is a joint project of the Department of Justice's National Institute of Justice (NIJ) and NIST's Office of Law Enforcement Standards and the Information Technology Laboratory. CFTT is supported by other organizations, including the Federal Bureau of Investigation, Department of Defense Cyber Crime Center, Internal Revenue Service

Please use the following format when citing this chapter:

Lyle, J., Mead, S., Rider, K., 2007, in IFIP International Federation for Information Processing, Volume 242, Advances in Digital Forensics III; eds. P. Craiger and S Shenoi; (Boston: Springer), pp. 163-177.

Criminal Investigation Division, Bureau of Immigration and Customs Enforcement and U.S. Secret Service. CFTT's objective is to provide measurable assurance to practitioners, researchers and other users that digital forensic tools provide accurate results. Accomplishing this requires the development of specifications and test methods for forensic tools, and the subsequent testing of tools against the specifications.

Test results provide the information necessary for developers to improve forensic tools, for users to make informed choices, and for the legal community and others to understand the tools' capabilities. Our approach to testing forensic tools is based on well-recognized methodologies for conformance and quality testing. The specifications and test methods are posted on the CFTT website [4] for review and comment by the digital forensics community.

The next section presents a simplified description of command interfaces to hard drives. Section 3 discusses BIOS access to hard drives, BIOS commands used for hard drive access, and the behavior of several BIOS-based write blockers. Section 4 focuses on hardware-based write blockers and the commands used to access hard drives via the ATA interface. The final section, Section 5, summarizes our observations.

2. Background

A hard drive is usually attached to a computer by a cable to an interface controller located either on the system motherboard or on a separate adapter card. The most common physical interfaces are the ATA (AT Attachment) and IDE (Integrated Drive Electronics) interfaces, including variants such as ATA-2, ATA-3 and EIDE (Enhanced IDE). Other 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).

All access to a drive is accomplished by commands sent from the computer to the drive via 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 a DOS environment can use two additional interfaces: the DOS service interface (interrupt `0x21`) and the 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 such as Windows XP and UNIX variants (e.g., Linux and FreeBSD) may disallow any low level interface (through the BIOS

or the controller) and only permit user programs to access a hard drive via a device driver that manages all access to a device.

There are five basic strategies for ensuring that digital data on a secondary storage device is not modified during a forensic acquisition:

- **No Protection:** If it is not desirable to disrupt an active system, relevant data could be acquired without any explicit protection of the source. For example, if a web server cannot be shut down for business reasons, digital data of interest could be copied to removable media.
- **Trusted OS with Trusted Tools:** These include operating systems designed to meet forensic requirements that are used in conjunction with tools that do not modify secondary storage.
- **Trusted OS with Trusted Tools (Variation):** These include BIOS-based software write blockers running on DOS. Examples are HDL from the Royal Canadian Mounted Police (RCMP) and PDBLOCK (Physical Drive BLOCKer) from Digital Intelligence.
- **Driver-Based Software Write Blockers:** These blockers are designed for operating systems such as Microsoft Windows that use a driver stack model (a stack of driver filters). A write block filter inserted into the driver stack examines all the commands sent to a device through the stack. A command that could modify a protected drive is blocked, i.e., it is not passed on to lower layers of the stack.
- **Hardware-Based Write Blockers:** These devices break the usual one-segment bus between the host and the storage device into two segments with the blocker acting as a bridge between the two bus segments. The blocker intercepts all traffic from the host to the storage device and only issues safe commands to the storage device.

This paper focuses on BIOS-based write blockers and hardware-based write blockers that use the ATA interface.

3. BIOS-Based Write Blockers

A BIOS-based write blocker is intended for use only with a trusted operating system such as DOS. Such a write blocker is usually designed as a Terminate and Stay Resident (TSR) program that, once initiated, intercepts all attempts to access a hard drive through the interrupt `0x13` interface. Only a command that is considered to be "safe" is passed to

the hard drive; otherwise the blocker returns to the calling program without passing the command to the hard drive.

While write commands to a device should always be blocked, it is not clear how a write blocker should treat other commands. The `0x13` BIOS interface uses an eight-bit function code (with 256 possible commands). Beyond a core set of commands, most function codes are not used. However, as technology changes, new core commands are often introduced and other commands are no longer implemented. BIOS vendors may implement additional commands as desired. Vendors often introduce system components that implement new commands; also, vendors may enhance the functionality of existing commands. The most significant change to the core command set was the introduction of extended commands (`0x41--0x49`) for accessing hard drives larger than 8 GB.

Table 1. Phoenix BIOS 4.0 interrupt `0x13` command set.

Command	Code	Category
Reset	00	Control
Get Last Status	01	Information
Read Sectors	02	Read
Write Sectors	03	Write
Verify Sectors	04	Information
Format Cylinder	05	Configuration
Read Drive Parameters	08	Information
Initialize Drive Parameters	09	Configuration
Read Long Sector	0A	Read
Write Long Sector	0B	Write
Seek Drive	0C	Control
Alternate Drive Reset	0D	Control
Test Drive Ready	10	Information
Recalibrate Drive	11	Configuration
Controller Diagnostic	14	Configuration
Read Drive Type	15	Information
Check Extensions Present	41	Information
Extended Read	42	Read
Extended Write	43	Write
Verify Sectors	44	Information
Extended Seek	47	Control
Get Drive Parameters	48	Information

The command set for a typical BIOS [7] is presented in Table 1. Note that only 22 of the possible 256 command codes are defined. Each command belongs to one of the following six categories:

- **Read:** These commands read from a drive, and should never be blocked.
- **Write:** These commands write to a drive, and should always be blocked.
- **Information:** These commands help obtain information about a drive. They should be allowed or be accurately simulated by the write blocker so that applications can obtain information about the drive (e.g., size and drive capabilities).
- **Control:** These commands request a drive to execute an operation that does not change the drive configuration or contents. It should be safe to allow these commands, but a tool may block some of these commands with no ill effects.
- **Configuration:** These commands change the way a drive appears to the host. They include format and diagnostic commands. Format commands should always be blocked. Diagnostic commands are questionable: it is safer to block them as they are often vendor-specific and undocumented, but in practice they may not make any changes to the drive.
- **Miscellaneous:** These commands, e.g., undefined and discontinued commands, are not included in the other five categories, Since undefined and discontinued commands should not be issued, it usually does not matter if the commands are allowed or blocked as long the write blocker does not encounter these commands. However, if the BIOS command set were to be extended with a new write command, this command should, of course, be blocked.

An experiment was conducted to determine the commands that a write blocker might encounter during a forensic examination (drive preview and acquisition). A monitor was installed to track the interrupt `0x13` commands issued by several common programs doing the routine tasks specified in Table 2. Several different hosts were used to obtain a satisfactory coverage of BIOS implementations, including a host with a legacy BIOS (without the extended read and write commands). The results of the experiment are presented in Table 3.

The experimental results give rise to the following observations:

- Only a few (10) of the 22 possible Phoenix BIOS commands were observed.
- All the defined read commands were observed.

Table 2. Experimental tasks.

Task
Copy/Edit Tools:Copy *.* to image disk, DOS
Copy/Edit Tools:Edit High Sector, Norton Disk Editor
Copy/Edit Tools:Edit Low Sector, Norton Disk Editor
Imaging Tools:Drive(Entire), EnCase 3.22
Imaging Tools:Drive(Entire), EnCase 4.14
Imaging Tools:Drive(Entire), SafeBack 3.0
Imaging Tools:Partition-High, EnCase 3.22
Imaging Tools:Partition-High, EnCase 4.14
Imaging Tools:Partition-High, SafeBack 3.0
Imaging Tools:Partition-Low, EnCase 3.22
Imaging Tools:Partition-Low, EnCase 4.14
Imaging Tools:Partition-Low, SafeBack 3.0
Induced Drive Read Error:Drive(Entire), SafeBack 3.0

- One command (Read Long) only appeared when a read command failed. This was accomplished by simulating a single bad sector on the drive. A TSR program intercepted each disk read command and, when the designated sector was requested, returned a read error to the application. The application encountering the bad sector then tried the Read Long command to read the simulated bad sector.
- Both the regular write commands were observed, but the Write Long command was not observed. It is unlikely that the Write Long command would be issued by a forensic tool as it should only be used on rare occasions.
- Other commands, e.g., Format Cylinder, that could write to a hard drive should not be issued by forensic tools and were not observed.
- The fact that certain commands were not seen in the experiment does not mean that are never encountered. The missing commands likely manifest themselves with other programs and/or hardware.

Six software write blockers were tested against the NIST CFTT software write blocker specifications [1, 6]. The observation from testing interrupt `0x13` write blockers is that, although there is agreement on the treatment of common commands used by write blockers, there are minor differences in the treatment of the less frequently used commands. The results are presented in Table 4. The "Spec" column has an "A" for commands that should be allowed and a "B" for commands that should

Table 3. Observed Interrupt `0x13` commands observed.

Code	Command	Program	Sum
42	Ext Read	DOS Copy	36
43	Ext Write	DOS Copy	223
41	Check for Extensions	EnCase 3.22	14
42	Ext Read	EnCase 3.22	657722
43	Ext Write	EnCase 3.22	1280151
48	Get Drive Parms	EnCase 3.22	14
08	Read Drive Parms	EnCase 3.22	23
02	Read Sectors	EnCase 3.22	2148
00	Reset	EnCase 3.22	6
41	Check for Extensions	EnCase 4.14	14
42	Ext Read	EnCase 4.14	654989
43	Ext Write	EnCase 4.14	1274995
48	Get Drive Parms	EnCase 4.14	14
08	Read Drive Parms	EnCase 4.14	23
02	Read Sectors	EnCase 4.14	2020
00	Reset	EnCase 4.14	6
42	Ext Read	Norton Disk Editor	2
08	Read Drive Parms	Norton Disk Editor	5
02	Read Sectors	Norton Disk Editor	6
03	Write Sectors	Norton Disk Editor	6
41	Check for Extensions	SafeBack 3.0	16
42	Ext Read	SafeBack 3.0	939146
43	Ext Write	SafeBack 3.0	812666
48	Get Drive Parms	SafeBack 3.0	14
08	Read Drive Parms	SafeBack 3.0	34
0A	Read Long	SafeBack 3.0	1
02	Read Sectors	SafeBack 3.0	85368
00	Reset	SafeBack 3.0	21
04	Verify Sectors	SafeBack 3.0	14
03	Write Sectors	SafeBack 3.0	62416

be blocked according to the NIST specifications. The columns labeled "0.4" through "0.8" present the results obtained for different versions of the RCMP's HDL write blocker; the columns labeled PDB and PDL present the results for PDBLOCK and PDLITE. An "A" in a column indicates that the corresponding tool allowed the command, while a "B" indicates that it blocked the command. For the miscellaneous commands in the last row of Table 4: "A" means all commands were allowed, "A3" that all but three commands were allowed, and "A4" that all but four were allowed.

For the critical commands, all the tools (except for HDL 0.4) are in agreement with the NIST specifications. This is because the HDL

Table 4. HDL and PDBLOCK commands (blocked and allowed).

Command	Code	Category	Spec	0.4	0.5	0.7	0.8	PDB	PDL
Format Track	05	Config.	B	B	B	B	B	B	B
Format Track (with Bad Sectors)	06	Config.	B	B	B	B	B	B	B
Format Cylinder	07	Config.	B	B	B	B	B	B	B
Init. Drive Parms.	09	Config.	B	A	A	A	B	A	A
ESDI Diag. (PS/2)	0E	Config.	B	A	A	A	B	A	A
ESDI Diag. (PS/2)	0F	Config.	B	B	B	B	B	B	B
Cntrlr. RAM Diag.	12	Config.	B	A	A	B	B	A	A
Drive Diag.	13	Config.	B	B	B	B	B	A	A
Cntrlr. Diag.	14	Config.	B	A	A	B	B	A	A
Reset	00	Control	A	A	A	A	A	A	A
Seek Drive	0C	Control	A	A	A	A	A	A	A
Alt. Drive Reset	0D	Control	A	A	A	A	A	A	A
Recalib. Drive	11	Control	A	A	A	A	B	A	A
Extended Seek	47	Control	A	A	A	B	B	A	A
Get Last Status	01	Info.	A	A	A	A	A	A	A
Verify Sectors	04	Info.	A	A	A	A	A	A	A
Read Drive Parms.	08	Info.	A	A	A	A	A	A	A
Test Drive Ready	10	Info.	A	A	A	A	A	A	A
Read Drive Type	15	Info.	A	A	A	B	A	A	A
Chck. Extns. Prsnt.	41	Info.	A	A	A	A	A	A	A
Verify Sectors	44	Info.	A	A	A	A	A	A	A
Get Drive Parms.	48	Info.	A	A	A	A	A	A	A
Read Sectors	02	Read	A	A	A	A	A	A	A
Read Long Sector	0A	Read	A	A	A	A	A	A	A
Extended Read	42	Read	A	A	A	A	A	A	A
Write Sectors	03	Write	B	B	B	B	B	B	B
Write Long Sector	0B	Write	B	B	B	B	B	B	B
Extended Write	43	Write	B	A	B	B	B	B	B
Undefined	Other	Misc.	B	A	A4	B	B	A3	A3

0.4 tool was created before the extended BIOS commands were introduced. Table 4 also documents a change in the design criteria used by the RCMP. Versions 0.4 and 0.5 were designed to block known write commands and allow everything else. However, starting with version 0.7, the criteria changed to allow known safe commands and block everything else; this change occurred because of the uncertainty surrounding the actions of some of the more esoteric commands. As shown in Table 3, several commands were never issued by programs that would be commonly used during an acquisition or initial examination of a hard drive. Although the unused commands may not change data on the hard drive,

without adequate documentation about the behavior of these commands, it is safer to simply block them.

4. Hardware-Based Write Blockers

Hardware-based write blockers use a two-segment bus to connect a host computer and a hard drive, one segment between the host and the write blocker and the other between the blocker and the drive. The two bus segments do not necessarily use the same protocol; one of the first hardware blockers used a SCSI connection to the host computer and an ATA connection to the hard drive. A hardware-based write blocker intercepts each command from the host and selects a desired course of action. The most common actions are:

- The blocker forwards the command to the hard drive.
- The blocker substitutes a different command, which is sent to the hard drive. This is the case when the blocker uses different bus protocols to communicate with the host and the hard drive.
- The blocker simulates the command without actually forwarding the command to the hard drive. For example, the device may already know the size of the drive; upon receiving a request from the host, instead of re-querying the drive, it may return the answer directly to the host.
- If a command is blocked, the blocker may return either success or failure for the blocked operation. However, returning failure for certain commands may cause the host computer to lock up.

The remainder of this section focuses on write blockers that use the ATA protocol for communicating with the host and the hard drive. Table 5 lists the seven releases of the ATA standard [8]; an eighth standard is currently under development.

The ATA protocol has 256 possible command codes. In the ATA-7 standard, approximately 70 codes are defined as general use commands (commands that are not reserved, retired, obsolete or vendor-specific). In addition, there are more than 30 retired or obsolete codes that were defined in the earlier standards.

Table 6 lists the write commands specified in the seven ATA standards. An "S" indicates that the listed command (row) is supported for the given ATA standard (column). 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

Table 5. ATA standards.

Last Draft Standard	Publication Date
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

Table 6. History of ATA write commands.

1	2	3	4	5	6	7	Code	Name
N	N	N	N	N	N	S	3A	Write Stream DMA Ext
N	N	N	N	N	N	S	CE	Write Multiple FUA Ext
N	N	N	N	N	N	S	3E	Write DMA Queued FUA Ext
N	N	N	N	N	N	S	3D	Write DMA FUA Ext
N	N	N	N	N	N	S	3B	Write Stream Ext
N	N	N	N	N	S	S	34	Write Sector(s) Ext
N	N	N	N	N	S	S	3F	Write Log Ext
N	N	N	N	N	S	S	39	Write Multiple Ext
N	N	N	N	N	S	S	36	Write DMA Queued Ext
N	N	N	N	N	S	S	35	Write DMA Ext
N	N	N	S	S	S	S	CC	Write DMA Queued
S	S	N	N	N	N	N	E9	Write Same
S	S	S	N	N	N	N	33	Write Long (w/o Retry)
S	S	S	N	N	N	N	32	Write Long (w/ Retry)
S	S	S	N	N	N	N	3C	Write Verify
S	S	S	S	N	N	N	31	Write Sector(s)
S	S	S	S	N	N	N	CB	Write DMA (w/o Retry)
S	S	S	S	S	S	S	E8	Write Buffer
S	S	S	S	S	S	S	30	Write Sector(s)
S	S	S	S	S	S	S	C5	Write Multiple
S	S	S	S	S	S	S	CA	Write DMA (w/ Retry)

discontinued six other write commands. The critical observation is that the command set changes quite significantly every few years.

We conducted an experiment to observe the commands issued during startup by three different computers. A protocol analyzer (Data Transit Corporation Bus Doctor Protocol Analyzer) was used to capture ATA bus activity during the startup and shutdown of various computer-BIOS combinations. The commands listed in Table 7 were issued from the

Table 7. Commands issued from BIOS during startup.

Host and BIOS	Command
Dell Phoenix 4.0 Rel 6.0	10=Recalibrate
Dell Phoenix 4.0 Rel 6.0	90=Exec Drive Diag
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 Multiple Mod
Micron Phoenix 4.0 Rel 6.0	C6=Set Multiple Mod
Nexar Award V4.51PG	C6=Set Multiple 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

BIOS to drive 0 of the primary ATA channel. Note that the BIOS did not issue any write commands to the hard drive for all the systems examined.

We also used the protocol analyzer 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. The results of our investigation are presented in Table 8. Neither PCDOS 6.3 nor DOS 6.22 issued any write commands during startup and shutdown. Also, note that the newer operating systems use the faster Write DMA (`0xCA`) command instead of the slower Write (`0x30`) command.

Several test reports for hardware-based write block devices have been published by NIJ [1]. These blockers were tested against a specification [5] developed by the CFTT Project. Some interesting results are:

- One write blocker substitutes the Read DMA (`0xC8`) command for the Read Multiple (`0xC4`) command [3].
- One write blocker disallows certain read commands [3] (see Table 9).

Table 8. Write commands issued during startup and shutdown

Host/OS	Source	Count	Command
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)
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

- Another blocker allowed some commands that could modify a hard drive. However, these commands are not typically used without special software [2]:
  - Download Microcode (0x92): This command enables hard drive firmware to be reprogrammed. While the command could change drive behavior, information about the command is drive-model-specific and is generally not available.
  - Format Track (0x50): This command is not defined in the current ATA specifications; however, it was defined in the older specifications (ATA-1 through ATA-3). The command can be used to erase information on an older drive that supports the instruction, but it cannot be used to change any user or operating system data stored on the drive.

Table 9. Blocker commands.

Commands Sent to Blocker	Commands Allowed by Blocker
20=Read (w/ Retry)	20=Read (w/ Retry)
21=Read (w/o Retry)	24=Read Sector Ext
22=Read/L (w/ Retry)	25=Read DMA Ext
23=Read/L (w/o Retry)	27=Read Max Addr Ext
24=Read Sector Ext	25=Read DMA Ext
25=Read DMA Ext	C8=Read DMA
26=Read DMA Queue Ext	C8=Read DMA
27=Read Max Addr Ext	F8=Read Natv Max Addr
29=Read Multiple Ext	
2A=Read Stream DMA	
2B=Read Stream PIO	
2F=Read Log Ext	
40=Read/V (w/ Retry)	
41=Read/V (w/o Retry)	
42=Read/V (w/ Ext)	
B0=Smart D0=Smart Read Data	
B0=Smart D5=Smart Read Log	
C4=Read Multiple	
C7=Read DMA Queued	
C8=Read DMA	
C9=Read DMA (w/o Retry)	
E4=Read Buffer	
F8=Read Natv Max Addr	

- Smart Write (`0xB0`, `0xD6`): This command records information in a device maintenance log, which is stored in a different data area from the data files and operating system data.
- Vendor-Specific Commands: These commands, which are often undocumented, are specific to hard drive models.
- CFA Erase (`0xC0`): This command applies to compact flash devices, not hard drives.
- SATA Write FPDMA (`0x61`): This command was noted by the protocol analyzer in a parallel ATA test, but the command is only valid for Serial ATA (SATA) devices.

5. Conclusions

Our work has focused on BIOS-based and hardware-based write blockers. We have used monitoring software to record BIOS command usage by BIOS-based write blockers in a DOS environment, and a protocol

analyzer to record direct ATA command usage by hardware-based write blockers in a variety of environments.

Our experiments indicate that, in the DOS environment, only a small subset of possible BIOS commands are used for system startup, shutdown and routine forensic tool operations.

A similar situation is encountered in the case of ATA commands. While some I/O commands are present in all versions of the ATA specifications, the changes made to the specifications on a fairly regular basis lead to new commands being implemented and several old commands being dropped. For direct access to ATA drives during system startup and shutdown, only a small subset of possible commands are used for a given operating system and the command sets vary for different operating systems. For example, Windows 98 uses the Write (`0x30`) command, Windows NT uses the Write Multiple (`0xC5`) command, and Windows XP and Linux versions use the Write DMA (`0xCA`) command.

Other interesting observations are that write blockers also block certain read commands; some (software and hardware) write blockers disallow known write commands and allow everything else; others allow only known safe commands and block everything else. Furthermore, some write blockers filter commands that may write arbitrary user or operating system data, but they allow some unsupported or atypical commands that have the potential to hide or destroy data (e.g., Download Microcode and Format Track).

Future work related to write blockers will focus on other interfaces, including SATA, SCSI, USB and IEEE 1394.

Finally, although certain company names and products are mentioned in this work, in no case does the identification of these companies and/or products imply any recommendation or endorsement by NIST or the U.S. Government.

References

[1] National Institute of Justice (NIJ), Computer Forensic Tool Testing Project (www.ojp.usdoj.gov/nij/topics/ecrime/cftt.htm).

[2] National Institute of Justice (NIJ), *Test Results for Hardware Write Block Device: FastBloc IDE (Firmware Version 16)*, NIJ Report NCJ 212956, Washington, DC (www.ncjrs.gov/pdffiles1/nij/212956.pdf), 2006.

[3] National Institute of Justice (NIJ), *Test Results for Hardware Write Block Device: MyKey NoWrite (Firmware Version 1.05)*, NIJ Report NCJ 212958, Washington, DC (www.ncjrs.gov/pdffiles1/nij/212958.pdf), 2006.

[4] National Institute of Standards and Technology (NIST), Computer Forensic Tool Testing Project (www.cftt.nist.gov).

[5] National Institute of Standards and Technology (NIST), Hardware Write Block, (www.cftt.nist.gov/hardware_write_block.htm).

[6] National Institute of Standards and Technology (NIST), Software Write Block (www.cftt.nist.gov/software_write_block.htm).

[7] Phoenix Technologies, *PhoenixBIOS 4.0 Revision 6 User's Manual*, Phoenix Technologies, San Jose, California, 2000.

[8] Technical Committee T13 – International Committee on Information Technology Standards (T13–INCITS), AT Attachment (ATA) Storage (www.t13.org).