April 1, 2004

Setup and Test Procedures

For Testing Interrupt 0x13 Based Software Write Block Tools

Version 1.0

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Abstract[†]

This document describes the procedures for testing of interrupt 0x13 based software write block tools using test cases described in *Software Write Block Tool Specification & Test Plan* Version 3.0.

The main objective of this document is to describe the test procedures that shall be followed to accomplish the testing of an interrupt 0x13 based software write block tool. This document also provides enough information about the testing process for either an independent evaluation of the process or independent replication of the results. The intended audience for this document should be familiar with the DOS operating system, Linux (or some UNIX like) operating system, computer operation, computer hardware components such as hard drives, hard drive interfaces (e.g., IDE or SCSI) and computer forensics.

[†] **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.**

Table of Contents

List of Tables

List of Figures

1 Introduction

The objective of the Computer Forensics Tool Testing project is to provide a measure of assurance that the tools used in computer forensics investigations produce accurate results. This is accomplished by developing specifications and test methods for computer forensics tools and then testing specific tools. The test 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 the legal community and others to understand the tools' capabilities. Our approach for testing computer forensic tools is based on well-recognized methodologies for conformance testing and quality testing.

The Computer Forensics Tool Testing (CFTT) program is a joint project of the National Institute of Justice (NIJ), the research and development organization of the U.S. Department of Justice; NIST's Office of Law Enforcement Standards (OLES) and Information Technology Laboratory (ITL); and is supported by other organizations, including the Federal Bureau of Investigation, the Department of Defense Cyber Crime Center, and the Department of Homeland Security's Bureau of Immigration and Customs Enforcement and U.S. Secret Service. The entire computer forensics community helps develop the specifications and test methods by commenting on drafts as they are published on the NIST website http://www.cftt.nist.gov/.

The main objective of this document is to describe the test procedures that shall be followed to accomplish the testing of an interrupt 0x13 based software write block tool. This document also provides enough information about the testing process for either an independent evaluation of the process or independent replication of the results. The intended audience for this document should be familiar with the DOS operating system, Linux (or some UNIX like) operating system, computer operation, computer hardware components such as hard drives, hard drive interfaces (e.g., IDE or SCSI) and computer forensics. To attempt an independent replication of the reported test results, an agency or lab other than NIST would require sufficient hardware and software resources to execute the test cases, this document, and *Software Write Block Tool Specification & Test Plan* Version 3.0. Since it is unlikely that the exact hardware used by NIST is present, adjustments and substitutions must be made to run the test cases.

1.1 Testing Overview

To accomplish the testing several items must be assembled and prepared. These include computers to execute the tests, hard disk drives, removable media, support software listed in Table 1-1 (SWBT Version 1.0 and FS-TST Version 1.0) and scripts to control the testing process listed in Table 1-2. The support software is available from the web site: http://www.cftt.nist.gov/.

A set of hard drives is selected and given an initial setup for the test cases. The drives are setup once and then used for multiple test cases. After all the components are prepared, the test cases are run. All the test cases follow a similar execution plan.

Table 1-1 Support Programs Used in Testing

Program	Package	Function
Tally13	SWBT 1.0	Monitor interrupt 0x13.
Test-hdl	SWBT 1.0	Sent selected interrupt 0x13 commands to the SWB tool.
T-off	SWBT 1.0	Turn off interrupt 0x13 monitoring.
Sig-log	SWBT 1.0	Log operator observations of A/V signals
Log-setup	FS-TST 1.0	Document drive setup
Diskwipe	FS-TST 1.0	Initialize a drive to known values
Diskhash	FS-TST 1.0	Compute a SHA-1 over an entire drive

Table 1-2 Scripts Used in Testing

Script	Function
XSWB.BAT	Run a typical protocol test case (1—36)
XSWBU.BAT	Run an uninstall protocol test case (39—40)
XBOOT.BAT	Run a boot protocol test case (37—38)
FAT-SRC.TXT	Partition Magic script to create FAT & EXT2 partitions
NT-SRC.TXT	Partition Magic script to create NTFS partitions
UDT-SET.BAT	Script to create deleted files
SETUP.BAT	Alternate script to start a typical protocol test case (1—36)
ZSWB.BAT	Alternate script to either finish a typical protocol test case or as an alternative to XBOOT.BAT
ZSWBU.BAT	Alternate script for an uninstall protocol test case (39—40)

In summary, the testing process involves the following steps:

1. Obtain the SWB tool to be tested.
2. Download this document, FS-TST and SWBT from http://www.cftt.nist.gov/.
3. Select test computers and disk drives.
4. Review tool documentation and use *Software Write Block Tool Specification & Test Plan* Version 3.0 to select the test cases to run.
5. Prepare a Windows 98 boot floppy (see Section 2.2).
6. Prepare and compute a reference hash of the source drives (see Section 2.1).
7. Run test cases (see Section 4).
8. Check each test case for problems, rerun test cases if required.
9. Rehash all source drives.
10. Extract relevant information from log files and create test report.

1.2 Document Overview

Section 2 describes procedures for creation or setup of source disks and DOS boot floppies. The script files for each test protocol are described in section 3. Section 4

describes the execution procedures for each test protocol. Section 5 describes alternative scripts and protocols for testing other SWB tools.

2 Media Setup

The test cases require several media components to be prepared before the test cases are executed.

1. Source hard disk drives for the test cases need to be created.
2. A DOS boot floppy that creates the run-time environment for the test case needs to be created.
3. Test support software (FS-TST and SWBT) needs to be copied to removable media. A CD-ROM needs to be created for FS-TST and **Partition Magic**. The SWBT tools need to be copied to the DOS boot floppy.

2.1 Source Disks

The source disks play the role of hard drives containing original digital evidence that must be preserved. There are too many possible disk layouts for all to be used in the tests. Four configurations have been selected that cover the most common partition types. The first configuration is a dual boot Red Hat Linux 7.1 and Windows Me. This configuration also includes FAT16, Linux EXT2, hidden partitions and deleted partitions (Table 2-1). The second configuration is a Windows 2000 system with both FAT32 and NTFS file systems (Table 2-2). The third configuration does not contain a valid partition table. The last selection is a single partition, either FAT32 or NTFS. The criterion for selection of a configuration for a drive is to have at least one drive of each configuration available for testing. All partitions are created with **partition magic Pro 6.0**.

Table 2-1 Windows Me/Linux Source Drive Layout

Type	Size (MB)	Comment
FAT16	600	Windows Me C drive, (set as active partition)
none	500	Unallocated Space
Extended	3500	Extended partition containing the next four partitions
EXT2	100	Linux EXT2 partition
FAT16	70	D drive for Windows
FAT16	2000	A FAT16 partition that has been deleted
FAT16	90	A FAT16 partition marked as *hidden*
EXT2	3000	Linux EXT2 partition with Red Hat 7.1
none	variable	Unallocated space up to the next partition
SWAP	200	Linux Swap partition

Table 2-2 Windows 2000 Source Drive Layout

Type	Size (MB)	Comment

Type	Size (MB)	Comment
FAT32	3000	Windows 2000 C drive, (set as active partition
none	1000	Unallocated space
Extended	variable	Remainder of disk space
NTFS	1000	Deleted NTFS partition
NTFS	600	D drive
FAT32	1000	Deleted FAT32 partition
NTFS	800	Hidden NTFS partition
none	variable	Unallocated space up to next partition
FAT32	600	Hidden FAT32 partition

The setup procedure for a source disk is as follows:

1. Select type of setup: Windows Me/Linux, Windows 2000 or none.
2. Select a hard drive.
3. Select computer, install drive, boot into PC DOS 6.3 from a boot floppy.
4. Run **LOGSETUP** to make a record of the setup.
5. Run **DISKWIPE** to initialize the drive contents.
6. If the setup type uses an operating system do steps 7-9
7. Run **partition magic** to partition the drive. For Windows Me/Linux source disk use the script in Figure 2-1 and for Windows 2000 source disk use the script in Figure 2-2.
8. Follow the installation instructions for each operating system that is installed. For a Windows Me/Linux configuration, first install Windows Me then install Linux. For a Windows 2000 configuration, install Windows 2000.
9. Create deleted files. This is accomplished by a script (DOS batch file) that creates a directory (X:\UDT, where X is a drive letter) with deleted files and a deleted subdirectory (Figure 2-3).
10. Run **DISKHASH** to create a reference SHA-1 hash for the source disk.
11. (optional) If practical, create a backup to another disk that can be used to restore the disk if it is modified. If a disk needs to be restored, the hash value can be recomputed to verify that the backup and restore were successful.

Figure 2-1 Partition Magic script for Windows Me/Linux Source (FAT-SRC.TXT)

```
Select Drive 1
Select Unallocated First
Create /FS=FAT /Size=600 /Label="P1FAT"
Select Unallocated First
Create /FS=Extended /Size=4000
Select Partition Extended
Resize Left Boundary Smaller 500
Select Unallocated 2
Create /FS=LINUXEXT2 /Size=100 /Label="X1Unix"
Select Unallocated 2
Create /FS=FAT /Size=70 /label="X1Fat"
Select Unallocated 2
Create /FS=FAT /Size=2000 /label="GONE"
Select Unallocated 2
```

```
Create /FS=FAT /Size=90 /label="GHOST"
Select Unallocated 3
Create /FS=LINUXEXT2 /Size=3000 /Label="Unix"
Select Unallocated 3
Create /FS=LINUXswap /Size=200  /Position=END
Select Partition "GHOST"
Hide
Select Partition "GONE"
Delete "GONE"
Select Partition "P1FAT"
Set Active
```

Figure 2-2 Partition Magic script for Windows 2000 Source (NT-SRC.TXT)

```
Select Drive 1
Select Unallocated First
Create /FS=FAT32 /Size=3000 /Label="FAT3GB"
Select Unallocated First
Create /FS=Extended
Select Partition Extended
Resize Left Boundary Smaller 1000
Select Unallocated 2
Create /FS=NTFS /Size=1000 /label="GONE1"
Select Unallocated 2
Create /FS=FAT32 /Size=600 /Label="GHOST32" /position=end
Select Unallocated 2
Create /FS=NTFS /Size=600 /label="X1NT"
Select Unallocated 2
Create /FS=FAT32 /Size=1000 /label="GONE2"
Select Unallocated 2
Create /FS=NTFS /Size=800 /label="GHOST4NT"
Select Partition "GHOST4NT"
Hide
Select Partition "GHOST32"
Hide
Select Partition "GONE2"
Delete "GONE2"
Select Partition "GONE1"
Delete "GONE1"
Select Partition "FAT3GB"
Set Active
```

Figure 2-3 Script to Create Deleted Files (UDT-SET.BAT)

```
echo undelete test setup
Rem Setup a directory with some deleted files and a deleted
subdirectory
date
time
:L1
Rem are we done?
      if "%1"=="" goto L1X
      echo "Set up drive %1:"
rem   create a directory for the deleted files
      mkdir %1:\udt
```

```
rem   create two files
      copy a:readme.txt %1:\udt
      copy a:back.txt %1:\udt
rem   delete one file
      del %1:\udt\back.txt
rem   undelete %1:\udt
rem   create a subdirectory
      mkdir %1:\udt\sub
Rem   create some files in the subdirectory
      copy a:missing.txt %1:\udt\sub
      copy a:gone.txt %1:\udt
rem   delete one file
      del %1:\udt\sub\missing.txt
rem   delete the directory
      rmdir %1:\udt\sub
rem   delete another file
      del %1:\udt\gone.txt
rem   shift cmd line, look for another drive
      shift
      goto L1
:L1X
echo Setup finished
```

2.2 Windows 98 Boot Floppy

The Windows 98 boot floppy disk provides an execution environment for the test harness and SWB tool execution. The steps to create the boot floppy are as follows:

1. Create the AUTOEXEC.BAT file defined in Figure 2-4.
2. Create the CONFIG.SYS file defined in Figure 2-5.
3. Execute the commands in Figure 2-6 to create the floppy.
4. Remove any references to the C: drive from the **io.sys** file and remove any references to the following programs: **DBLSPACE.BIN, DRVSPACE.BIN**, and **STACKER.** A log of changes to the **io.sys** file is presented in Figure 2-7.

After the Boot Floppy is setup, copy the following files to the boot floppy:

1. SWBT Version 1.0 programs (TALLY13.COM, TEST-HDL.EXE, T-OFF.EXE, and if needed SIG-LOG.EXE)
2. Test scripts (XSWB.BAT, XSWBU.BAT and XBOOT.BAT).
3. SWB tool to be tested.

Figure 2-4 Windows 98 Boot Floppy AUTOEXEC.BAT

```
@ECHO OFF
A:\drivers\mscdex.exe /D:mscd001 /L:Z
A:\drivers\findramd
set ramd=W:
```

```
if errorlevel 3 set ramd=C:
if errorlevel 4 set ramd=D:
if errorlevel 5 set ramd=E:
if errorlevel 6 set ramd=F:
if errorlevel 7 set ramd=G:
if errorlevel 8 set ramd=H:
echo RAM Drive is %ramd%
copy Z:\pm\*.* %ramd%
type A:\CMD-LOG.txt
echo "XSWB SWB-<CASE> <Version> <Host> <User> <Category> <Pattern>
<DriveList>"
```

The **CONFIG.SYS** file loads drivers for ASPI SCSI devices, CDROM drive and sets the last drive letter.

Figure 2-5 Windows 98 Boot Floppy CONFIG.SYS

```
ACCDATE=C- D- E- F- G- H- I- J- K- L- M- N- O- P- Q-
device=A:\drivers\himem.sys /testmem:off
device=A:\drivers\oakcdrom.sys /D:mscd001
device=A:\drivers\aspi8u2.sys /D /PD800 /Q9
device=A:\drivers\aspi8dos.sys
device=a:\drivers\aspicd.sys /D:mscd001
device=a:\drivers\ramdrive.sys 2000 /E
files=30
buffers=10
dos=high,umb
stacks=9,256
lastdrive=z
```

Figure 2-6 Windows 98 Boot Floppy Setup

From a Windows 98 System, insert a blank floppy disk and open a DOS command window

```
FORMAT A: /S
MKDIR A:\DRIVERS
COPY HIMEM.SYS A:\DRIVERS
COPY MSCDEX.EXE A:\DRIVERS
COPY ASPI2DOS.SYS A:\DRIVERS
COPY ASPI4DOS.SYS A:\DRIVERS
COPY ASPI8DOS.SYS A:\DRIVERS
COPY ASPI8U2.SYS A:\DRIVERS
COPY ASPICD.SYS A:\DRIVERS
COPY BTCDROM.SYS A:\DRIVERS
COPY BTDOSM.SYS A:\DRIVERS
COPY FINDRAMD.EXE A:\DRIVERS
COPY FLASHPT.SYS A:\DRIVERS
COPY OAKCDROM.SYS A:\DRIVERS
COPY RAMD.SYS A:\DRIVERS
setup AUTOEXEC.BAT and CONFIG.SYS
```

The **Scrub Log** (Figure 2-7) is a list of changes made to the **IO.SYS** file to ensure no references to the C: drive during boot.

Figure 2-7 Windows 98 Boot Floppy Scrub Log

```
A:\SCRUB.EXE compiled at 23:47:10 on Nov  7 2001
C:\ found at 17268, replaced by A:\
C:\ found at 17664, replaced by A:\
C:\ found at 20484, replaced by A:\
C:\ found at 20499, replaced by A:\
C:\ found at 42632, replaced by A:\
C:\ found at 42667, replaced by A:\
DBLSPACE.BIN found at 42720, replaced by NOOSPACE.BIN
DRVSPACE.BIN found at 42734, replaced by NOOSPACE.BIN
C:\ found at 60980, replaced by A:\
C:\ found at 60996, replaced by A:\
C:\ found at 66795, replaced by A:\
C:\ found at 66805, replaced by A:\
C:\ found at 69755, replaced by A:\
DBLSPACE.BIN found at 60999, replaced by NOOSPACE.BIN
DRVSPACE.BIN found at 60967, replaced by NOOSPACE.BIN
STACKER found at 61013, replaced by SLACKER
Starting Windows 98 found at 68272, replaced by Starting NIST Fboot
C:\ found at 220112, replaced by A:\
222390 bytes read from io.sys
```

2.3 FS-TST CD-ROM

The FS-TST CD-ROM is created by using any CD burning software to copy to a CD-ROM the FS-TST executable files and a copy of **Partition Magic 6.0** for DOS. The actual procedures followed for creating the CD-ROM depends on the CD-ROM creation software actually used.

3 Test Case Execution Scripts

There are three run protocols for testing SWB tools defined in *Software Write Block Tool Specification & Test Plan* Version 3.0. Each run protocol is implemented in a DOS batch file. The protocols and scripts are as follows: a typical case (1—36) uses the script `xswb.bat,` a tool uninstall case (39—40) uses the script `xswbu.bat,` and a case (37—38) activating the tool from the `autoexec.bat` file uses the `xboot.bat` script. These scripts are designed to execute several versions of the RCMP HDL[1] program. Minor alterations are required for other tools such as PDBLOCK[2]. Each script is described below.

Figure 3-1 Script to Execute a Typical Case (`XSWB.BAT`)

```
     1 @echo off
     2 REM Script to run a typical SWB case
     3 REM xswb CASE Ver HOST USR cat PAT 80 81 82 83 84
     4 set TheCase=%1
     5 set ver=%2
     6 set host=%3
```

[1] RCMP HDL is available to law enforcement only through the Royal Canadian Mounted Police Technological Crime Branch Technical Operations Directorate.

[2] PDBLOCK and PDB_LITE are available through Digital Intelligence, Inc. 1325 Pearl Street, Waukesha, WI 53186. http://www.digitalintel.com/index.htm

```
 7  set usr=%4
 8  set cat=%5
 9  set pat=%6
10  echo %usr% is running Case %TheCase% on %host%. Command set %cat%, pattern %pat%
11  shift
12  shift
13  shift
14  shift
15  shift
16  shift
17  If not "%1"=="" echo Drive 80 is %1
18  If not "%2"=="" echo Drive 81 is %2
19  If not "%3"=="" echo Drive 82 is %3
20  If not "%4"=="" echo Drive 83 is %4
21  If not "%5"=="" echo Drive 84 is %5
22
23  del A:\*.txt
24
25  echo xswb %theCase% %ver% %host% %usr% %cat% %pat% %1 %2 %3 %4 %5 > A:\CMD-LOG.TXT
26
27  echo Boot Test PC to > A:\x-log.txt
28  ver >> A:\x-log.txt
29  echo tally13 >> A:\x-log.txt
30  echo hdl%ver% %pat% >> A:\x-log.txt
31  echo test-hdl %theCase% %host% %usr% %cat% %1 %2 %3 %4 %5 >> A:\x-log.txt
32  echo Shutdown Test PC >> A:\x-log.txt
33
34
35
36  A:\tally13 > A:\t13-log.txt
37  A:\hdl%ver% %pat% > A:\hdl-log.txt
38  test-hdl %TheCase% %host% %usr% %cat% %1 %2 %3 %4 %5
39
40  mkdir A:\%TheCase%
41  copy A:\*.txt A:\%TheCase%
42  echo Test %TheCase% Finished, Reboot
43
44  rem parameters
45  rem xswb Case Version Host Usr category Pattern 80 81 82 83 84
```

The batch file script xswb.bat is presented in Figure 3-1. The general plan is to fetch command line parameters, execute the interrupt 13 monitor, execute the tool under test, send the set of commands being checked, and then save the log files for each test in their own directory on a floppy disk. After the test is finished the log files must be archived in a more permanent controlled-access location. The line by line description of `xswb.bat` is as follows:

Line 1: turn off command echo to the console.
Lines 2 & 3: comment.
Lines 4—9: set variables to values from the command line.
Line 10: feed back to user.
Lines 11—16: discard first six parameters used in lines 4—9 from the command line.
Lines 17—21: feedback drives specified to the used.
Line 23: delete all old log files.
Line 25: log to cmd-log.txt the script command line.
Lines 27—32: log the commands executed by the script to x-log.txt.
Line 36: start the interrupt 13 monitor program.
Line 37: start the SWB tool.
Line 38: run the test program to send the commands specified in `%cat%.`

Line 40: create a directory for the test case log files.
Line 41: copy the log files to the case directory.
Line 42: inform the user that the case is finished.

The general plan for the batch file script (xswbu.bat) used in the uninstall protocol is to also uninstall (or deactivate) the tool under test and then send all possible commands.

Figure 3-2 Script to Execute an Uninstall Case (`XSWBU.BAT`)

```
     1  @echo off
     2  REM Script to run a typical SWB case
     3  REM xswbu CASE Ver HOST USR cat PAT 80 81 82 83 84
     4  set TheCase=%1
     5  set ver=%2
     6  set host=%3
     7  set usr=%4
     8  set cat=%5
     9  set pat=%6
    10  echo %usr% is running Case %TheCase% on %host%. Command set %cat%, pattern %pat%
    11  shift
    12  shift
    13  shift
    14  shift
    15  shift
    16  shift
    17  If not "%1"=="" echo Drive 80 is %1
    18  If not "%2"=="" echo Drive 81 is %2
    19  If not "%3"=="" echo Drive 82 is %3
    20  If not "%4"=="" echo Drive 83 is %4
    21  If not "%5"=="" echo Drive 84 is %5
    22
    23
    24  del A:\*.txt
    25
    26  echo xswbu %theCase% %ver% %host% %usr% %cat% %pat% %1 %2 %3 %4 %5 > A:\CMD-
LOG.TXT
    27
    28  echo Boot Test PC > A:\x-log.txt
    29  ver >> A:\x-log.txt
    30  echo tally13 >> A:\x-log.txt
    31  echo hdl%ver% %pat% >> A:\x-log.txt
    32  echo test-hdl %theCase% %host% %usr% w %1 %2 %3 %4 %5 >> A:\x-log.txt
    33  echo ren a:\swb-log.txt wt-log.txt
    34  echo t-off %TheCase% %host% %usr% >> A:\x-log.txt
    35  echo hdl%ver% R >> A:\x-log.txt
    36  echo test-hdl %theCase% %host% %usr% a %1 %2 %3 %4 %5 >> A:\x-log.txt
    37  echo Shutdown Test PC >> A:\x-log.txt
    38
    39  A:\tally13 > A:\t13-log.txt
    40  A:\hdl%ver% %pat% > A:\hdl-log.txt
    41  a:\test-hdl %TheCase% %host% %usr% w %1 %2 %3 %4 %5
    42  ren a:\swb-log.txt wt-log.txt
    43  A:\t-off %TheCase% %host% %usr%
    44  A:\hdl%ver% R > A:\hdlr-log.txt
    45  a:\test-hdl %TheCase% %host% %usr% a %1 %2 %3 %4 %5
    46
    47  mkdir A:\%TheCase%
    48  copy A:\*.txt A:\%TheCase%
    49  echo Test %TheCase% Finished
    50
    51  rem parameters
    52  rem xswbu Case Ver Host Usr category Pattern 80 81 82 83 84
```

The line by line description of `xswbu.bat` (Figure 3-2) is as follows:

Line 1: turn off command echo to the console.
Lines 2 & 3: comment
Lines 4—9: set variables to values from the command line.
Line 10: feed back to user.
Lines 11—16: discard first six parameters used in lines 4—9 from the command line.
Lines 17—21: feedback drives specified to the used.
Line 24: delete all old log files.
Line 26: log to cmd-log.txt the script command line.
Lines 28—37: log the commands executed by the script to x-log.txt.
Line 39: start the interrupt 13 monitor program.
Line 40: start the SWB tool (first time).
Line 41: send write commands to verify the SWB is active.
Line 42: rename the log file.
Line 43: turn off the monitor.
Line 44: request the SWB tool to deactivate.
Line 45: send each possible interrupt 13 command and log results.
Line 47: create a directory for the test case log files.
Line 48: copy the log files to the case directory.
Line 49: inform the user that the case is finished.

The `autoexec.bat` file in Figure 3-3 is used for test cases using the boot protocol. Lines 2—5 delete any old log files, execute the interrupt 13 monitor and the software write blocking tool and are added to the usual autoexec.bat file to start the test case from the `autoexec.bat` file. The file must be adjusted for each tool to execute the actual tool being tested.

Figure 3-3 AUTOEXEC.BAT File Used with Boot Test Protocol

```
1.  @ECHO OFF
2.  del *.txt
3.  A:\tally13 > t13-log.txt
4.  A:\hdl8 13 >hdl-log.txt
5.  echo pattern is 13
6.  A:\drivers\mscdex.exe /D:mscd001 /L:Z
7.  A:\drivers\findramd
8.  set ramd=W:
9.  if errorlevel 3 set ramd=C:
10. if errorlevel 4 set ramd=D:
11. if errorlevel 5 set ramd=E:
12. if errorlevel 6 set ramd=F:
13. if errorlevel 7 set ramd=G:
14. if errorlevel 8 set ramd=H:
15. echo RAM Drive is %ramd%
16. copy Z:\pm\*.* %ramd%
```

The boot protocol script (Figure 3-4) is the xswb.bat with lines removed that execute the interrupt 13 monitor and the tool under test. For the boot protocol, these two programs are executed from the autoexec.bat file in the boot protocol tests.

Figure 3-4 Script to Execute a Boot Protocol Test Case (`XBOOT.BAT`)

```
1.  @echo off
2.  REM Script to run a boot SWB case
3.  REM xboot CASE Ver HOST USR PAT 80 81 82 83 84
4.  set TheCase=%1
```

```
5.  set ver=%2
6.  set host=%3
7.  set usr=%4
8.  set pat=%5
9.  echo %usr% is running version %ver% for Case %TheCase% on %host%. Command set w,
    pattern %pat%
10. shift
11. shift
12. shift
13. shift
14. shift
15. If not "%1"=="" echo Drive 80 is %1
16. If not "%2"=="" echo Drive 81 is %2
17. If not "%3"=="" echo Drive 82 is %3
18. If not "%4"=="" echo Drive 83 is %4
19. If not "%5"=="" echo Drive 84 is %5
20.
21. echo xboot %TheCase% %ver% %host% %usr% %pat% %1 %2 %3 %4 %5 >cmd-log.txt
22.
23. echo Boot Host (HDL in AUTOEXEC.BAT) > x-log.txt
24. ver >> x-log.txt
25. echo tally13 >> x-log.txt
26. echo hdl%ver% %pat% >> x-log.txt
27. echo test-hdl %theCase% %host% %usr% w %1 %2 %3 %4 %5 >> x-log.txt
28. echo Shutdown Host >> x-log.txt
29.
30.
31. test-hdl %TheCase% %host% %usr% w %1 %2 %3 %4 %5
32.
33. mkdir A:\%TheCase%
34. copy a:\*.txt A:\%TheCase%
35. copy a:\autoexec.bat A:\%TheCase%
36. echo Test %TheCase% Finished
37.
38. rem parameters -- note: no category
39. rem xboot Case Ver Host Usr Pattern 80 81 82 83 84
40.
```

The line by line description of `xboot.bat` is as follows:

Line 1: turn off command echo to the console.
Lines 2 & 3: comment
Lines 4—8: set variables to values from the command line.
Line 9: feed back to user.
Lines 10—14: discard first five parameters used in lines 4—8 from the command line.
Lines 15—19: feedback drives specified to the used.
Line 21: log to cmd-log.txt the script command line.
Lines 23—29: log the commands executed by the script to x-log.txt.
Line 31: run the test program to send the write commands to the tool.
Line 33: create a directory for the test case log files.
Line 34: copy the log files to the case directory.
Line 35: copy the autoexec.bat file to the case directory.
Line 36: inform the user that the case is finished.

4 Test Case Execution Procedures

This section presents the procedures for running a test case. It is assumed that the reader is familiar with basic computer operation.

Before any test cases are run, each test computer is checked for suitability as a test platform and the FS-TST Version 1.0 CD-ROM is installed. The procedure is as follows:

1. Select a host computer for check out.
2. If the computer is on, shutdown the computer and turn it off
3. Remove any hard drives.
4. Insert the DOS boot floppy.
5. Turn the computer on and boot to DOS.
6. Verify that the computer BIOS boot order specifies floppy before hard drive.
7. If the boot or Power on Self Test (POST) fails, an attempt is made to diagnose the problem. If the problem cannot be diagnosed and repaired, then the computer is not suitable for testing and is not used for testing. A computer that is not suitable for testing shall have a sticky label with the words *DO NOT USE* placed over the power switch to indicate that the computer should not be used.
8. If the computer is available for testing, continue.
9. Insert the FS-TST Version 1.0 CD-ROM.
10. Shutdown the computer.

As specified in Section 2.1, A SHA-1 is computed for each source drive used in testing before any test cases are executed. After all the test cases are executed, each drive is rehashed to verify that no changes were made to any drive during testing. The following procedures are used to hash a selected drive after testing is completed:

1. Select a drive to hash.
2. Select a host computer to execute the hash.
3. If the computer is on, shutdown the computer and turn the computer off
4. Remove any hard drives.
5. Install the drive to be hashed.
6. Verify that the correct DOS boot floppy is in the machine. If not, insert the correct DOS boot floppy.
7. Turn the computer on to boot to DOS.
8. Verify a successful boot and POST.
9. Hash the drive with the following command:
10. `diskhash label host drive /after /new_log /comment user`
11. When the hash program exits, remove the floppy disk.
12. Shutdown the computer and power off.
13. Copy the hash log files from the floppy to a permanent read only location subject to regular back up.

The parameters needed to execute a **SWB** test case are found in Table 9-2 of *Software Write Block Tool Specification & Test Plan* Version 3.0. The test case parameters are discussed in Section 8.2 of the same document. There are three test case execution protocols: typical, boot and uninstall. The typical protocol applies to cases 01-36, the boot protocol applies to test cases 37 and 38, and the uninstall protocol applies to cases 39 and 40. Each of the three test protocols uses a corresponding execution script. The

typical protocol uses XSWB.BAT, the boot protocol uses a custom boot floppy with the script XBOOT.BAT and the uninstall protocol uses the XSWBU.BAT script.

The procedure to execute a test case is as follows:

1. Select the test case to run. For guidance in test case selection, see section 9.4 (**Test Case Selection Guide)** in *Software Write Block Tool Specification & Test Plan* Version 3.0.
2. Collect removable media: forensic DOS Boot floppy (see Section 2.2), and FS-TST CD-ROM.
3. Select a test machine in the test lab. The selection is arbitrary since all the test machines are similar and only basic capabilities of the computers are required for the test runs.
4. Install the number of hard drives required for the test specified by the N Drives test parameter. A parameter of all indicates that as many IDE and SCSI drives as possible are to be installed. For the current lab computers this is either four or five drives.
5. Ensure that the host computer is off. Install forensic DOS boot disk, and CD-ROM.
6. Turn on the host computer to boot from the forensic DOS floppy.
7. Execute the selected test case using the required boot floppy and script.
8. Shutdown the computer. (Reboot via CTRL-ALT-DEL to run another test case with the same setup).
9. After the script finishes, the log files created on the floppy disk are copied to a permanent read only location subject to regular back up.

The parameters for XSWB and XSWBU are as follows:

```
XSWB  CASE VER HOST USR CAT PAT DRIVE_LIST
XSWBU CASE VER HOST USR a   PAT DRIVE_LIST
```

Table 4-1 Parameters for XSWB.BAT & XSWBU.BAT Scripts

Parameter	Description
CASE	The test case ID. SWB-xx where xx is the case number. SWB is in capital letters.
VER	The version number of the tool. E.g., 4, 5, 7 or 8.
HOST	Host computer name, e.g., Cadfael, Rumpole, etc.
USR	User initials, e.g., JRL, SN, SM, BLive, etc.
CAT	Command category, w for write, r for read, c for control, x for configure, i for information, m for miscellaneous, and a for all. Note that the value is always *a* for XSWBU.
PAT	Protection pattern from test case list. If the test case requires a return value of success, the pattern must begin with a capital *S*.
DRIVE_LIST	List the external label of each drive installed in order by drive number.

The parameters for XBOOT are as follows:

```
XBOOT CASE VER HOST USR PAT DRIVE_LIST
```

Table 4-2 Parameters for XBOOT.BAT

Parameter	Description
CASE	The test case ID. SWB-xx where xx is the case number. SWB is in capital letters.
VER	The version number of the tool. E.g., 4, 5, 7 or 8.
HOST	Host computer name, e.g., Cadfael, Rumpole, etc.
USR	User initials, e.g., JRL, SN, SM, BLive, etc.
PAT	Protection pattern from test case list. If the test case requires a return value of success, the pattern must begin with a capital *S*. Note that XBOOT does not use the CAT (category) parameter.
DRIVE_LIST	List the external label of each drive installed in order by drive number.

5 Alternative Scripts and Run Procedures

The following scripts are slightly modified versions of XSWB.BAT and XSWBU.BAT that are to be used with the tools PDBLOCK and PDB_LITE. The typical protocol script, xswb, is replaced by two scripts (startup and zswb). The tool is executed directly after running the startup script and before running zswb. The uninstall protocol script, xswbu, is replaced by zswbu. The boot protocol script, xboot, is replaced by zswb. The autoexec.bat file for the boot protocol tests is adjusted to execute the tool under test. Since PDBLOCK and PDB_LITE generate an audio signal on blocked commands, both zswb and zswbu run the sig-log program to document any audio signals observed by the operator.

The command line parameters for zswb.bat and zswbu.bat are slightly different from xswb.bat.

Table 5-1 Parameters for ZSWB.BAT Script

Parameter	Description
CASE	The test case ID. SWB-xx where xx is the case number. SWB is in capital letters.
VER	The version of the tool. f indicates the full version, PDBLOCK and l indicates the lite version, PDB_LITE.
HOST	Host computer name, e.g., Cadfael, Rumpole, etc.
USR	User initials, e.g., JRL, SN, SM, BLive, etc.
CAT	Command category, w for write, r for read, c for control, x for configure, i for information, m for miscellaneous, and a for all. Note that the value is always *a* for XSWBU.
RET	The return value for a blocked command. An s indicates success, an f indicates failure (or error).
PAT	Protection pattern from test case list.
DRIVE_LIST	List the external label of each drive installed in order by drive number.

Figure 5-1 Alternate Script to Start a Typical Case (STARTUP.BAT)

```
@echo off
echo Startup a case
rem Delete all log files and start int 13 monitor

del a:\cmd-log.old
ren a:\cmd-log.txt cmd-log.old

del A:\*.txt

A:\tally13 > A:\t13-log.txt
```

Figure 5-2 Alternate Script to Finish a Typical Case (ZSWB.BAT)

```
@echo off
REM Script to run a typical SWB case
REM zswb CASE Ver HOST USR cat ret PAT 80 81 82 83 84
set TheCase=%1
set ver=%2
set host=%3
set usr=%4
set cat=%5
set ret=%6
set pat=%7
set pat1=%7
echo %usr% is running Case %TheCase% on %host%. Command set %cat%,
pattern %pat%
shift
shift
shift
shift
shift
shift
shift
If not "%1"=="" echo Drive 80 is %1
If not "%2"=="" echo Drive 81 is %2
If not "%3"=="" echo Drive 82 is %3
If not "%4"=="" echo Drive 83 is %4
If not "%5"=="" echo Drive 84 is %5


set cmd=pdblock
if "%ver%"=="f" goto dofull
set cmd=pdb_lite
set pat=
set rc=
goto done
:dofull
set rc=/fail
if "%ret%"=="s" set rc=
set cmd=pdblock
:done

echo
```

```
zswb %theCase% %ver% %host% %usr% %cat% %ret% %pat1% %1 %2 %3 %4 %5 >
A:\CMD-LOG.TXT

echo Boot Test PC to > A:\x-log.txt
ver >> A:\x-log.txt
echo tally13 >> A:\x-log.txt
echo A:\%cmd% %pat% %rc% >> A:\x-log.txt
echo test-hdl %theCase% %host% %usr% %cat% %1 %2 %3 %4 %5 >> A:\x-
log.txt
echo A:\sig-log %theCase% %host% %usr%
echo Shutdown Test PC >> A:\x-log.txt

echo Command should have been A:\%cmd% %pat% %rc%
test-hdl %TheCase% %host% %usr% %cat% %1 %2 %3 %4 %5
A:\sig-log %theCase% %host% %usr%

mkdir A:\%TheCase%
copy A:\*.txt A:\%TheCase%
echo Test %TheCase% Finished, Reboot

rem parameters
rem zswb Case Version Host Usr category Pattern 80 81 82 83 84
```

Figure 5-3 Alternate Script to Run an Uninstall Case (ZSWBU.BAT)

```
@echo off
REM Script to start an uninstall SWB case
REM zswb CASE Ver HOST USR cat ret PAT 80 81 82 83 84
set TheCase=%1
set ver=%2
set host=%3
set usr=%4
set cat=%5
set ret=%6
set pat=%7
set pat1=%7
echo %usr% is running Case %TheCase% on %host%. Command set %cat%,
pattern %pat%
shift
shift
shift
shift
shift
shift
shift
If not "%1"=="" echo Drive 80 is %1
If not "%2"=="" echo Drive 81 is %2
If not "%3"=="" echo Drive 82 is %3
If not "%4"=="" echo Drive 83 is %4
If not "%5"=="" echo Drive 84 is %5


set cmd=pdblock
if "%ver%"=="f" goto dofull
set cmd=pdb_lite
```

```
set pat=
set rc=
set none=
goto done
:dofull
set rc=/fail
set none=none
if "%ret%"=="s" set rc=
set cmd=pdblock
:done

del A:\*.txt
A:\tally13 > A:\t13-log.txt
echo
zswbu %theCase% %ver% %host% %usr% %cat% %ret% %pat1% %1 %2 %3 %4 %5 >
A:\CMD-LOG.TXT

echo Boot Test PC to > A:\x-log.txt
ver >> A:\x-log.txt
echo A:\tally13 >> A:\x-log.txt
echo A:\%cmd% %pat% %rc% >> A:\x-log.txt
echo A:\test-hdl %theCase% %host% %usr% w %1 %2 %3 %4 %5 >> A:\x-
log.txt
echo A:\sig-log %theCase% %host% %usr% >> A:\x-log.txt
echo A:\t-off %TheCase% %host% %usr% >> A:\x-log.txt
echo ren A:\swb-log.txt wt-log.txt >> A:\x-log.txt
echo ren A:\sig-log.txt sg-wt.txt >> A:\x-log.txt
echo A:\%cmd% %none% >> A:\x-log.txt
echo A:\test-hdl %theCase% %host% %usr% a %1 %2 %3 %4 %5 >> A:\x-
log.txt
echo A:\sig-log %theCase% %host% %usr% >> A:\x-log.txt
echo Shutdown Test PC >> A:\x-log.txt

if "%ver%"=="l" goto xlite
A:\%cmd% %pat% %rc%
:xlite
A:\%cmd% > pdb-log.txt
A:\test-hdl %TheCase% %host% %usr% w %1 %2 %3 %4 %5
A:\sig-log %theCase% %host% %usr%
A:\t-off %TheCase% %host% %usr%
ren A:\swb-log.txt wt-log.txt
ren A:\sig-log.txt sg-wt.txt
A:\%cmd% %none% > a:\pdr-log.txt
A:\test-hdl %TheCase% %host% %usr% a %1 %2 %3 %4 %5
A:\sig-log %theCase% %host% %usr%

mkdir A:\%TheCase%
copy A:\*.txt A:\%TheCase%
echo Test %TheCase% Finished, Reboot

rem parameters
rem zswb Case Version Host Usr category Pattern 80 81 82 83 84
```