Interested in learning more about security?

SANS Institute InfoSec Reading Room

Silicon Graphics IRIX Sanitization Overwrite Procedures

Maintaining the confidentiality of sensitive information is a fundamental mission of a computer security program. Ensuring that sensitive information is securely removed from computer media prior to release is a critical layer of security. This document references a United States Department of Defense three-pass overwrite standard and then describes procedures that are used to overwrite media according to that standard using the Silicon Graphics Incorporated IRIX operating system "FX" utility.

Global Information Assurance Certification (GIAC)
Security Essentials Certification (GSEC)

Practical Assignment: Version 1.4b (amended August 29, 2002) Option 1

Submitted by: Michael J. Davis

Date of submission: 7 April 2003

Title: Silicon Graphics IRIX Sanitization Overwrite Procedures

Local Mentor: 22 January 2003, Huntsville, AL, George Starcher

CERTIFICATION OF AUTHORSHIP: I certify that I am the author of this paper and that any assistance I received in its preparation is fully acknowledged and disclosed in the paper. I have also cited any sources from which I used data, ideas, words, either quoted directly or paraphrased. I also certify that this paper was prepared by me specifically for this course.

CONTENTS

FIGURES

ABSTRACT

Maintaining the confidentiality of sensitive information is a fundamental mission of a computer security program. Ensuring that sensitive information is securely removed from computer media prior to release is a critical layer of security.

This document references a United States Department of Defense three-pass overwrite standard and then describes procedures that are used to overwrite media according to that standard using the Silicon Graphics Incorporated IRIX operating system "FX" utility.

The basic operation of this low level disk exerciser utility is discussed, including alternatives and other ancillary risk management considerations. Then specific direction and visual aides are provided to instruct the reader how to set the correct pattern for each overwrite pass, how to write that pattern over all addressable locations, and how to verify that the overwrite performed correctly.

Silicon Graphics IRIX[1] Sanitization Overwrite Procedures

1. Introduction

1.1 Confidentiality

Maintaining the confidentiality of sensitive information is a fundamental mission of a computer security program. Regrettably, organizations and individuals often ignore (or may not understand) the risk associated with the release of improperly sanitized computer media.

For instance, how often do we observe organizations employ expensive technical gadgetry to protect the network perimeter (such as firewalls and intrusion detection systems),[2] but then simply release computer media without regard for the sensitivity of information contained on that media? Disregarding this critical layer of security occurs too often; for example, according to a recent study of 158 drives purchased on eBay or elsewhere, only 12 had been appropriately sanitized.[3]

1.2 Scope

With respect towards the intended audience, this paper will not elaborate nor justify the purpose for procedures that are used to overwrite magnetic computer media. This work will presume that the reader is familiar with the data remanence issue, as documented in many publications; perhaps most authoritative, the National Computer Security Center's ("Rainbow Series"), A Guide to Understanding Data Remanence in Automated Information Systems.[4] The SANS Info Sec Reading Room certainly includes additional information.[5]

The United States Department of Defense (DoD) defines sanitization as a "Process to remove information from media such that data recovery is not possible. It includes removing all classified labels, markings, and activity logs."[6] DoD standards[7] describe several methods to accomplish sanitization, in general: 1) degauss[8], 2) overwrite, or 3) destruction.

Be advised that residual risk may remain even after performing any of these actions because of inadequate training or procedures, malfunctioning or inadequate equipment, or other technical[9] or non-technical issues. Regarding overwrites, according to the Rainbow Series guide, "There are two primary levels of threat [...]: keyboard attack (information scavenging through system software capabilities) and laboratory attack (information scavenging through laboratory means)" (NCSC-TG-025 2). This paper addresses the threat from keyboard attack. The reader is reminded here that acceptance of residual risk is an organizational risk management decision. So, notice that the DoD does not authorize overwrites for sanitization of TOP SECRET material.

This document will describe a procedure that has been approved by certain defense agencies for overwrite sanitization of magnetic system media using the Silicon Graphics Incorporated (SGI) IRIX operating system “FX” utility.

The FX utility is menu-driven and includes a series of submenus for various disk maintenance functions. For the destructive write-read function, the user invokes the FX utility and sets appropriate test patterns to be written and compared sequentially over all addressable locations on the disk. The FX overwrite defaults to the entire disk (partition 10). The overwrite pattern is user definable and can be manipulated to write to all or any addressable locations regardless of the size of the hard disk drive. In addition, the user may increase the number of passes in which each pattern is written.

This document contains information released in good faith, but is otherwise without express or implied warranty.

1.2.1 Acknowledgements

SGI IRIX overwrite procedures have been used within the DoD industrial security community for many years and this document builds upon that earlier work. Sometime prior to August 2000 the author of this paper obtained, from a variety of sources within this community, a number of differing procedures that had been approved for use by various field officials.

Noticing significant differences within this rather narrow endeavor, the author of this paper proceeded to compile and verify the best practices,[10] using then current IRIX version 6.5.4. A training videotape is available from that earlier effort.[11] Unfortunately, too few artifacts remain from these early sources; however, the author of this document does indeed recognize and appreciate the overall contributions of the DoD industrial security community to the overwrite sanitization procedures that follow in this document.

1.2.2 Description

The SGI IRIX operating system conforms with UNIX System V Release 4 and provides the foundation for an array of high-performance computing visualization applications; used in such diverse sectors as defense,[12] aerospace, automotive, medical, energy, environmental, or entertainment (for example, Arnold Schwarzeneggar's Terminator 2: Judgment Day).

This paper will validate earlier work while providing a revised description of procedures for overwrite sanitization of primary (system) or secondary (data) magnetic media disks that are physically connected, through the small computer system interface (SCSI), to the SGI IRIX version 6.5.18 system. This procedure does not address Redundant Array of Inexpensive Disks (RAID) nor Fibre Channel fabric devices.

Specifically, the DoD standard addressed by this work is as follows:

"Overwrite all addressable locations with a character, then its complement. Verify 'complement' character was written successfully to all addressable locations, then overwrite all addressable locations with random characters; or verify third overwrite of random characters."[13]

1.2.3 Permission

Permission should be obtained from the legal owner of the media to be overwritten prior to conducting this procedure. This SGI IRIX sanitization overwrite procedure is destructive; after this procedure is completed all previously stored information will be permanently unrecoverable (without an extraordinary laboratory environment).[14] Backup valuable information and appropriately protect that media prior to executing this procedure.

Do not proceed without receiving the necessary permissions (Cole 2).

1.3 Bad Blocks

Bad blocks (sectors or tracks) are a portion of a disk that cannot be used because it is flawed. Almost all disks have sectors damaged during the manufacturing process and additional (or "grown") bad sectors may be identified during the life cycle of a disk. For SCSI disks,[15] according to the FX manual page (fx(1M)), "The list of bad blocks is maintained by the firmware on the disk drive itself." Sensitive information may have been written to a grown sector prior to its addition to the defect list; so, sanitization overwrite procedures should include methods for determining bad blocks.

Ideally, an overwrite utility should write to, and read from, the grown defect sectors;[16] otherwise the bad blocks should be mapped before initial use and remapped before sanitization (NCSC-TG-025 4.4). Differences in the comparison lists should be reviewed by the organization's risk management specialist. The IRIX FX utility does not write to, nor read from, grown defect sectors.

2. IRIX FX Utility

SGI includes the FX low level disk exerciser utility as a part of the IRIX maintenance function. Two versions of FX are used: a stand-alone version for system disk modifications and an IRIX command line version for non-system disk modifications. The FX utility may be invoked to perform a destructive surface analysis over the entire disk using either of these versions. The IRIX command line version of FX is run as the superuser (root) (Man fx(1M)).

2.1 FX Menu

FX menu items are selected by typing either the full option name or the least unambiguous prefix. A menu item can be an action (for example, exit) or the name of a submenu (for example, badblock). To return to a parent menu from a submenu, enter two dots (..) (Man fx(1M)).

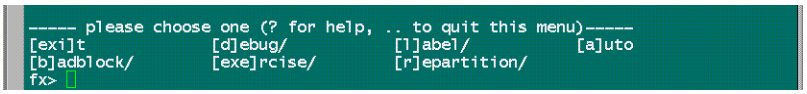

Figure 1. Example of the FX Menu.

2.1.1 Menu Hierarchy

The FX menus are organized as a hierarchy that can be navigated in a manner similar to UNIX file system navigation; for example, go up two levels by typing "../.." or use a command several levels down by separating each level by a slash. By typing a command pathname, such as "/badblock/showbb" a command can be executed from any point in the menu hierarchy. Similarly, typing the full pathname of any menu moves to that menu (Man fx(1M)).

2.1.2 FX Help

To obtain help for the items on the current menu, enter a question mark (?) at the prompt. Many of the functions have options to modify their actions; to obtain more information, enter "? Item" where item can be either the least unambiguous prefix, or the full name (Man fx(1M)).

2.2 FX Memory Buffer

FX provides a 524288 byte internal memory buffer as a source or destination for data (Man fx(1M)). Selected overwrite patterns are stored in this buffer and then flushed to the disk. The overwrite verification process will then read data patterns that were written to the disk into this buffer and then dump 512 byte blocks of this buffer to produce a three-column display.

Figure 2. FX Memory Buffer.

The first column is the hexidecimal address within the buffer of the data read from disk. The second column is the hexidecimal representation of the data pattern written to the disk. And the third column is the ASCII representation of the data pattern written to the disk.

2.3 Exiting FX

Once the main menu is reached, FX catches interrupts (i.e., Control-C), stopping any operation in progress, but does not terminate FX itself. To exit from FX, select "exit" at the main menu; a shorthand for exiting from any level is "/exit". Entering "/.." from any menu allows you to select a different disk without having to exit and restart (Man fx(1M)).

2.4 IRIX Hardware Inventory

The IRIX hardware inventory command should be executed prior to FX to determine the disk type, controller, and target.

Type: /bin/hinv

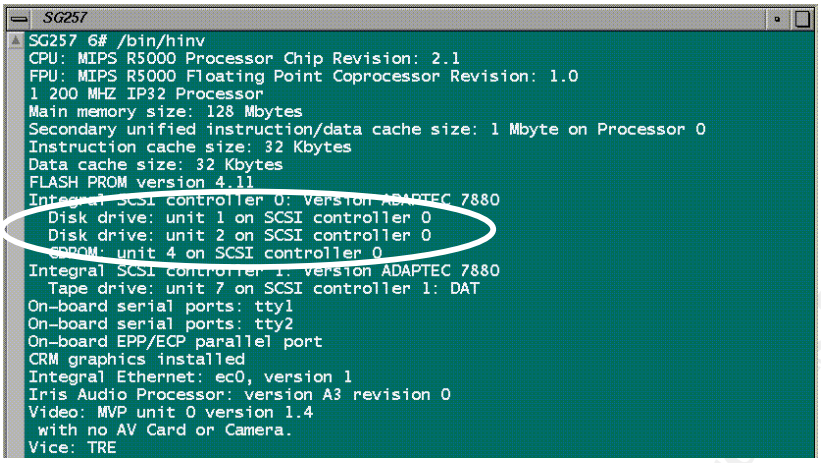

Figure 3. IRIX Hardware Inventory (HINV).

4. Command Line FX Overwrite Procedure

4.1 Start FX

Since the overwrite sanitization is a destructive procedure, FX must be run in "expert" mode (-x) by the superuser. “A mistake in expert mode can destroy all data on the disk” (Man fx(1M)). The command line version of FX is invoked as follows:

Type: /bin/fx -x

FX will prompt for disk type, controller, target, and lun numbers (see /bin/hinv). The controller type for SCSI disk drives is "dksc." FX next prompts for controller number, drive number (SCSI target ID), and lun (logical unit number). Logical unit numbers are used with RAID and have not been evaluated using this procedure; the lun default of 0 should be specified.

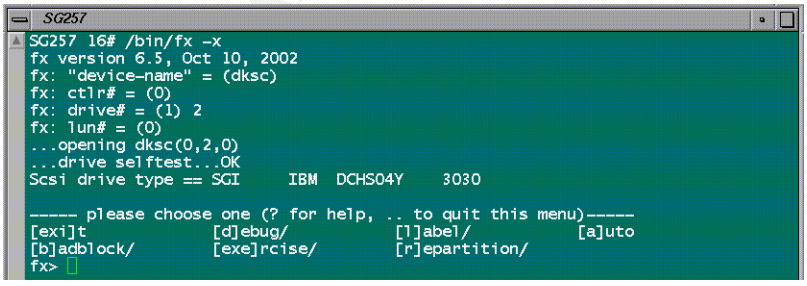

Figure 4. Starting Command Line FX.

After opening the specified drive, FX checks for in-use partitions, and warns the user if the target disk appears to have mounted filesystems (see Man mount(1M)). FX will then issue a diagnostic self-test command to the drive and return the drive information (for SCSI drives) (Man fx(1M)).

4.2 Verify Size of Disk

From the FX main menu, verify the size of the target disk by examining the partition table that is stored on the disk label within the volume header partition.

Type: repartition

```
----- please choose one (? for help, .. to quit this menu)-----
[exi]t             [d]ebug/           [l]abel/           [a]uto
[b]adblock/        [exe]rcise/        [r]epartition/
fx> repartition

----- partitions-----
part  type        blocks             Megabytes    (base+size)
  0: xfs     2101248 + 6787295      1026 + 3314
  1: raw        4096 + 2097152         2 + 1024
  8: volhdr        0 + 4096            0 + 2
 10: volume        0 + 8888543         0 + 4340

capacity is 8888543 blocks

----- please choose one (? for help, .. to quit this menu)-----
[ro]otdrive          [o]ptiondrive         [e]xpert
[u]srrootdrive       [re]size
fx/repartition>
```

Figure 5. Using FX Repartition Command to Verify Disk Size.

Note the size (hereafter referred as “nblocks”) of partition 10. In IRIX, partition 10 includes all addressable locations, or the entire disk. “One disk block equals 512 bytes” (Man df(1)). The size of partition 10 may be less than "capacity" because “Drives with variable geometry can have a partition layout that does not use all of the drive” (Man fx(1M)). Return to the FX main menu.

Type: ..

4.3 Verify Bad Blocks

From the FX main menu,[17] examine the grown defect list and compare to the list created prior to initial use of the disk.

Type: badblock/showbb -g

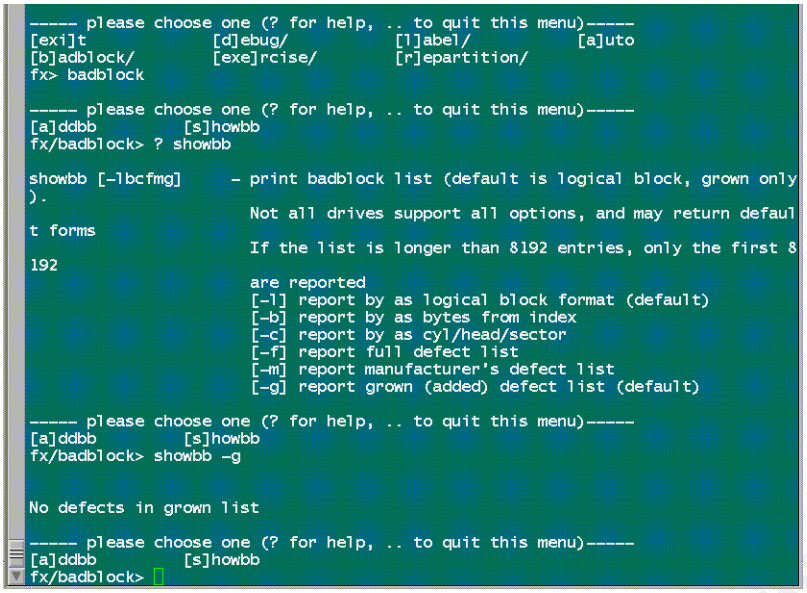

Figure 6. Using FX to Examine Grown Defect List.

Continue if FX reports “No defects in grown list,” or no differences are found with the comparison to the list created prior to initial use of the disk. Notice that “showbb –f” will display the full defect list--those mapped during manufacture (“showbb –m”) as well as grown.

4.4 Set First Overwrite Pattern

For the first overwrite pass, the standard requires an overwrite of all addressable locations with a character. From the FX main menu, set the first overwrite pattern to zeros.

Type: exercise/settestpat 0
Type: .. (to finish)

Verify the first overwrite pattern is set correctly.

Type: exercise/showtestpat

The hexidecimal pattern “00” should be displayed.

```
fx/exercise> ? settestpat

settestpat          - set data values (.. when done).   The pattern repeats
                      based the bytes entered to fill the entire test buffer
                      used for the write and write-cmp tests.

----- please choose one (? for help, .. to quit this menu)-----
[b]utterfly           [seq]uential          [set]testpat
[e]rrlog              [st]op_on_error       [sh]owtestpat
[r]andom              [m]iscompares         [c]omplete
fx/exercise> settestpat

Enter .. to finish; CR keeps current value
fx/exercise/settestpat: value = (219) 0
fx/exercise/settestpat: value = (109) ..

----- please choose one (? for help, .. to quit this menu)-----
[b]utterfly           [seq]uential          [set]testpat
[e]rrlog              [st]op_on_error       [sh]owtestpat
[r]andom              [m]iscompares         [c]omplete
fx/exercise> showtestpat

test pattern:
  0:  00
```

Figure 7. Using FX to Set the First Overwrite Pattern

FX allows up to 4K bytes of pattern to be set using the “settestpat” function, byte by byte, and that pattern is repeated as often as necessary to fill the 524288 byte input buffer. Each byte value is entered as either decimal or hexidecimal (with leading 0x) (Man fx(1M)).

4.5 Start First Overwrite Pass

From the FX main menu, start the first overwrite pass using the pattern set in the previous step.

Type: exercise/sequential
modifier = **wr-cmp**
starting block# = **0**
nblocks = (**size of partition 10**)
nscans = **1**

At the warning, about to destroy data on disk:

Type: yes

```
----- please choose one (? for help, .. to quit this menu)-----
[exi]t              [d]ebug/            [l]abel/            [a]uto
[b]adblock/         [exe]rcise/         [r]epartition/
fx> exercise

----- please choose one (? for help, .. to quit this menu)-----
[b]utterfly           [seq]uential          [set]testpat
[e]rrlog              [st]op_on_error       [sh]owtestpat
[r]andom              [m]iscompares         [c]omplete
fx/exercise> sequential

fx/exercise/sequential: modifier = (rd-only) ?
----- exercise modifiers-----
[rd]-only      [ro]-cmp       [s]eek         [wr-o]nly      [wr-c]mp
fx/exercise/sequential: modifier = (rd-only) wr-cmp
fx/exercise/sequential: starting block# = (0)
fx/exercise/sequential: nblocks = (8888543)
fx/exercise/sequential: nscans = (1)
 * * * * * W A R N I N G * * * * *
about to destroy data on disk dksc(0,2,0)! ok?
please enter a yes or no
about to destroy data on disk dksc(0,2,0)! ok? yes
sequential pass 1: scanning [0, 8888543] (8888543 blocks)
0%...
```

Figure 8. Using FX to Overwrite the First Pass.

The modifier “wr-cmp” is the significant function that performs the actual write, read, and compare operation. Notice that a block with a miscompare will be added to the bad block list (Man fx(1M)).

During the overwrite, FX will periodically refresh the display to provide status of the overwrite in progress. When completed (100%), FX returns to the menu.

```
about to destroy data on disk dksc(0,2,0)! ok? yes
sequential pass 1: scanning [0, 8888543] (8888543 blocks)
0%.........10%.........20%.........30%.........40%.........50%.........
60%.........70%.........80%.........90%......... 100%
```

Figure 9. FX Status Display

4.6 Set Second Overwrite Pattern

For the second overwrite pass, the standard requires an overwrite of all addressable locations with a complement character to that written during the first pass. From the FX main menu, set a second, complementary overwrite pattern to ones.

Type: exercise/settestpat 255
Type: .. (to finish)

```
----- please choose one (? for help, .. to quit this menu)-----
[b]utterfly            [seq]uential           [set]testpat
[e]rrlog               [st]op_on_error        [sh]owtestpat
[r]andom               [m]iscompares          [c]omplete
fx/exercise> settestpat

Enter .. to finish; CR keeps current value
fx/exercise/settestpat: value = (0) 255
fx/exercise/settestpat: value = (109) ..
```

Figure 10. Using FX to Set the Second Overwrite Pattern

Verify the second overwrite pattern is set correctly.

Type: exercise/showtestpat

```
fx/exercise> showtestpat

test pattern:
  0:  ff
```

Figure 11. Using FX to Verify Second Pattern

The hexidecimal pattern “ff” should be displayed. Notice in Figure 10 that the overwrite pattern is entered as a decimal value, 255. FX will also accept the same as a hexidecimal value, entered as 0xff. Either value converted to binary is equivalent to 11111111, a byte pattern that complements the first overwrite pattern of zeros.

4.7 Start Second Overwrite Pass

From the FX main menu, start the second overwrite pass using the pattern set in the previous step.

Type: exercise/sequential
modifier = **wr-cmp**
starting block# = **0**
nblocks = (**size of partition 10**)
nscans = **1**

At the warning, about to destroy data on disk:

Type: yes

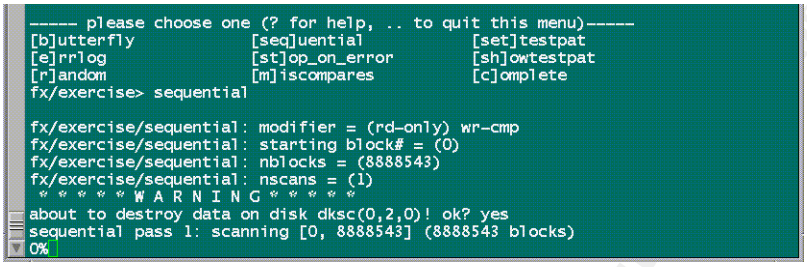

Figure 12. Using FX to Overwrite the Second Pass

4.8 Verify Second Pass Overwrite

After the second overwrite pass is completed, the standard requires verification that the complement character was written successfully to all addressable locations.[18] Three locations are read manually to confirm that the last pattern of data was indeed written; the first block (block number 0) is read, then a random block, and finally the last block (nblocks – 1).[19]

FX "debug" submenu items are used to perform these manual operations. For each of the three locations to be examined, the "seek" command will change the access position to the specified disk block, then the "readbuf" command will read one 512 byte block from that position into the buffer, and finally the "dumpbuf" command will display the contents of that buffer.

4.8.1 Read First Block

From the FX main menu:

Type: debug/seek 0
Type: debug/readbuf 0 1
Type: debug/dumpbuf b 0 512

The contents of the buffer should indicate that the pattern "FF" has been successfully written to the first block of the disk.

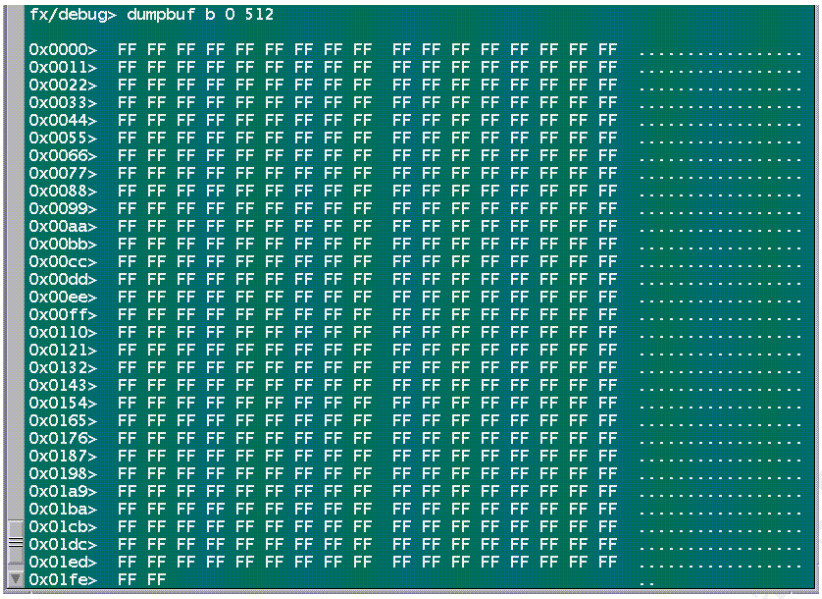

Figure 13. Using FX to Perform Manual Verification.

4.8.2 Read Random Block

From the FX main menu:

Type: debug/seek blocknum
(where 0 < blocknum < nblocks – 1)
Type: debug/readbuf 0 1
Type: debug/dumpbuf b 0 512

The contents of the buffer should indicate that the pattern “FF” has been successfully written to this block of the disk.

4.8.3 Read Last Block

From the FX main menu:

Type: debug/seek nblocks - 1
Type: debug/readbuf 0 1
Type: debug/dumpbuf b 0 512

The contents of the buffer should indicate that the pattern “FF” has been successfully written to the last block of the disk. Otherwise, the last block should be overwritten manually.

4.8.4 Manual Overwrite of Last Block, If Necessary

If FX has failed to write the last 512 byte block of the target disk, this block (or any other) should be overwritten manually, three times. The “fillbuf” and “writebuf” functions under the “debug” submenu are used to overwrite a specific block or blocks. For each

pass, the input buffer will be filled with a pattern and then that buffered pattern will be written to the specified block.

4.8.4.1 Special Considerations

Notice that the “fillbuf” function exhibits a regrettable program anomaly, syntax as follows:

Fillbuf [BO STR N] - fill buffer with string
BO = buffer offset in bytes
STR = source string
N = number of bytes to fill

In earlier versions of FX as well as the version considered here, testing has demonstrated that the value of “N” will need to be doubled to actually fill the buffer with the intended number of bytes. For example, specifying that “N” equals 512 will actually write just 256 bytes to the buffer.

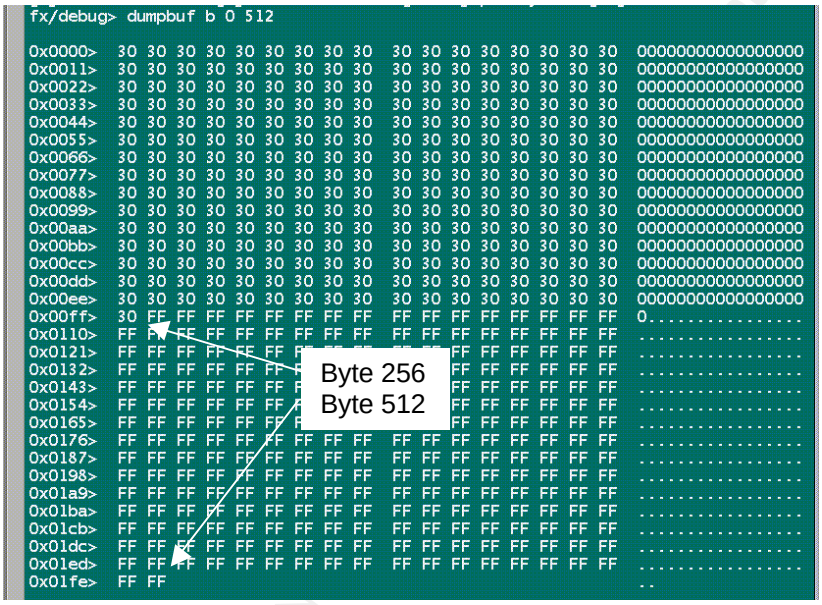

Figure 14. FX "fillbuf" Anomaly.

So in order to overwrite a 512 byte block of the disk, this procedure will accommodate the “fillbuf” anomaly by specifying that “N” equals 1024 bytes to fill.

Another unfortunate characteristic of “fillbuf” that requires a risk management decision is that the “source string” is indeed an ASCII character string, repeated here to form a 2-byte word. The “fillbuf” function does not accept hexidecimal values as does the “settestpat” function and that affects the preferred overwrite character values. The standard requires overwrite with a character and then its complement, but there are no printable complementary characters in the ASCII set. The strings chosen are therefore rather arbitrary in that regard.

STRING	HEXIDECIMAL	BINARY
00	30 30	00110000 00110000
11	31 31	00110001 00110001
aa	61 61	01100001 01100001

Figure 15. Character String Pattern Conversion.

4.8.4.2 Manual Overwrite Last Block Procedure

From the FX main menu:

Seek to the last block.
Type: debug/seek nblocks - 1

Fill buffer with first pattern and write the block.
Type: debug/fillbuf 0 00 1024
Type: debug/writebuf 0 1

Fill buffer with second pattern and write the block.
Type: debug/fillbuf 0 11 1024
Type: debug/writebuf 0 1

Fill buffer with third pattern and write the block.
Type: debug/fillbuf 0 aa 1024
Type: debug/writebuf 0 1

Verify.
Type: debug/readbuf 0 1
Type: debug/dumpbuf b 0 512

The last pattern written to this block should be "61."

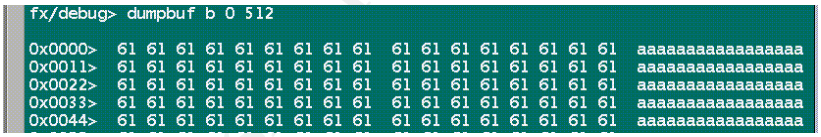

Figure 16. Using FX to Verify Last Block Written Manually.

4.9 Set Third Overwrite Pattern

For the third overwrite pass, the standard requires an overwrite of all addressable locations with random characters.

The easiest method to fill the overwrite buffer with random characters is to exit and restart FX, since the default pattern is “[…] a random pattern 1023 bytes long ensuring that few, if any sectors have the same data” (Man fx(1M)).

Type: /exit
Type: /bin/fx -x

FX will again prompt for disk type, controller, target, and lun numbers. Be certain to address the same target disk that has been overwritten twice preceding. Verify the third overwrite pattern is set correctly.

Type: exercise/showtestpat

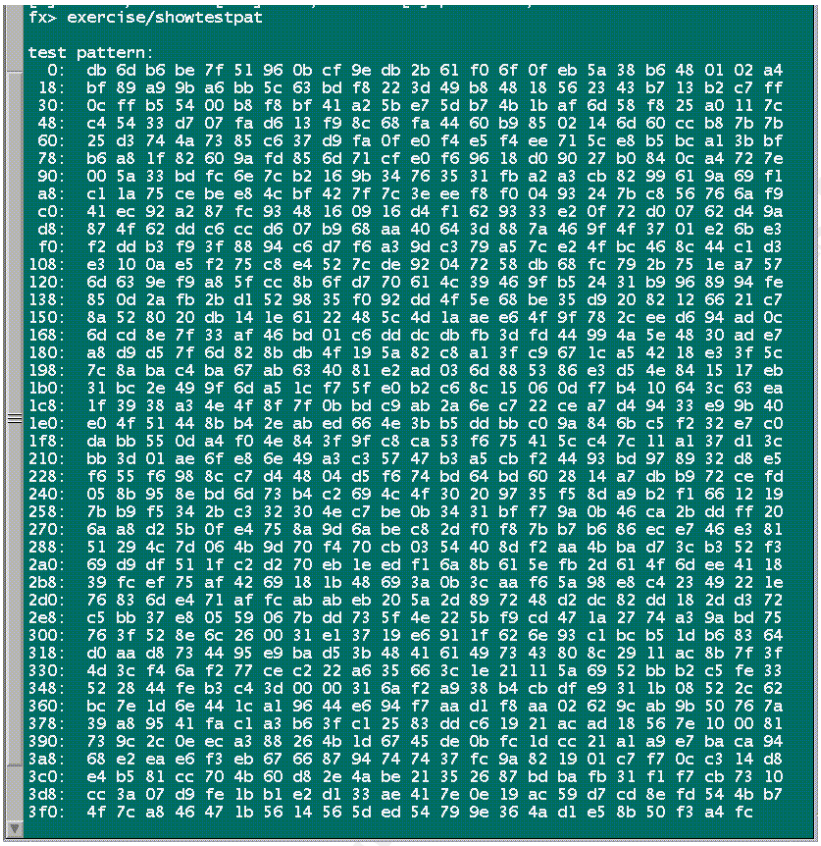

Figure 17. FX Buffer Filled With Random Characters.

4.10 Start Third and Final Overwrite Pass

From the FX main menu, start the third and final overwrite pass using the random pattern set in the previous step.

Type: exercise/sequential
modifier = **wr-cmp**
starting block# = **0**
nblocks = (**size of partition 10**)
nscans = **1**

At the warning, about to destroy data on disk:

Type: yes

When completed (100%), FX returns to the menu; prepare to complete the overwrite procedure. Notice that this procedure does not verify the third overwrite of random characters because verification has already occurred after the second overwrite pass, and is therefore not required by the standard. Additional verification could be accomplished, however, in a similar manner to that occurring after the second overwrite pass.

4.11 Completing the Overwrite Procedure

The overwrite sanitization procedure is nearly completed, so FX may be exited from any level of the menu system.[20]

Type: /exit

FX will exit with the following prompt:

"label info has changed for disk dksc(controller, target, lun) write out changes? (yes)"

Type: no

4.11.1 SGILABEL

If FX is not exited properly, the old “sgilabel” information may be rewritten to the disk. The “sgilabel” is innocuous, consisting of administrative information such as the type of disk drive and its serial number (Man fx(1M)). Nevertheless, the standard requires overwrite of all addressable locations, so rewriting that old label back to disk may contaminate the process. Strictly speaking, the system should be powered off and rebooted to clear volatile memory, then a new “sgilabel” may be created by simply restarting FX and selecting the same disk.

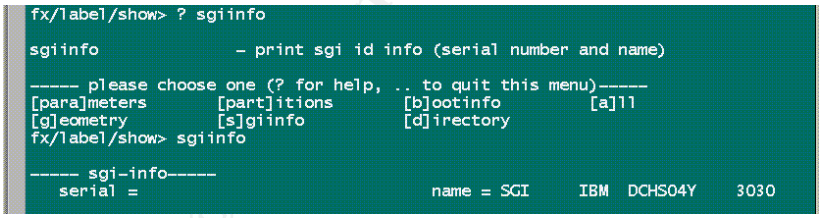

Figure 18. Using FX to Read SGILABEL.

4.12 Administrative Records

Best practice recommends that proper sanitization also includes an administrative action as well as the procedural aspect of the actual purging of the media and removing any sensitivity labels. The administrative aspect should require that a record be filed in the audit trail documenting the sanitization actions; at minimum this record should include the date, description of the event, and the person responsible.

5. Stand-alone FX Overwrite Procedure

There is no difference between the command line version of FX and the stand-alone version of FX except in the manner in which each is invoked. For obvious reasons, the stand-alone version of FX is used to sanitize a system disk. The following process describes how to start the stand-alone FX; afterwards the command line procedures discussed earlier remain valid.

5.1 Start FX

According to the manual page fx(1), a copy of the stand-alone version of FX is normally kept on the system disk in /stand/fx and can be invoked when the IRIX operating system is not running by giving the following command at the Command Monitor:

Type: boot stand/fx

A stand-alone fx is also provided in the /stand directory of IRIX installation CD-ROM discs, and can be invoked by the Command Monitor command.

For SGI systems with the 32 bit ARCS PROM (Indigo, Indigo2, Indy, Onyx, Challenge, and O2), use this command:

Type: boot -f dksc(ctlr,unit,8)sashARCS dksc(ctlr,unit,7)stand/fx.ARCS –x
(Where --x denotes expert mode, note 2 dashes, ctlr is the controller number (usually 0), unit is the SCSI id of the CD-ROM drive.)

For systems with 64-bit ARCS PROM (Power Challenge, Power Onyx, Power Indigo2, Indigo2 10000, Origin, Onyx2, and OCTANE) use this command:

Type: boot -f dksc(ctlr,unit,8)sash64 dksc(ctlr,unit,7)stand/fx.64 --x
(Where --x denotes expert mode, note 2 dashes, ctlr is the controller number (usually 0), unit is the SCSI id of the CD-ROM drive.)

6. The “dd” Alternative

Just like other UNIX operating systems, IRIX provides the “dd” command that might be used to overwrite computer media. This researcher experimented with this alternative using variations of the following command, both on the raw and block device.

dd conv=noerror,sync bs=16777216 if=/dev/zero of=/dev/dsk/dsk/dks0d2fvol

Using FX for verification, the “dd” command appears to succeed in overwriting all addressable locations; however, performance compared with FX was considerably slower. For example, using a 4.3 GB disk FX performed all three overwrite passes in approximately 78 minutes; while the “dd” command above performed just one pass in 162 minutes, then aborted after writing the last block. Notice, the large block size (“bs”)

used in attempt to optimize streaming to the block device. Additional work in that direction is beyond the scope of this paper, as is the means to generate the complement and random[21] pass input file, as well as verification procedures.

Likewise beyond scope, other overwrite alternatives include development of low-level control device functions (i.e., “ioctl”) or other similar programming efforts.

7. Additional Reading

Now that procedures are defined to sanitize the SGI IRIX computer media using FX, the reader should also investigate potential sources of information leakage related to other components within the system, such as nonvolatile memory located on system boards. For example, if a Command Monitor password is set, where is that sensitive information stored? Does the risk management policy permit that password to be released? Does the system administrator know how to clear that password?

For readers with such concerns, a proprietary document entitled SGI Statements of Volatility may be obtained directly from the manufacturer after executing a confidential disclosure agreement. This document describes in detail the volatility of each memory component for each board contained within the SGI product line; a valuable resource for informed risk management decisions.

8. Conclusion

This paper has asserted that maintaining the confidentiality of sensitive information is a fundamental mission of a computer security program and that ensuring sensitive information is securely removed from computer media prior to release is a critical layer of security. Important terms, definitions, references, figures, and standards have been provided so that the reader may understand the general concepts relating to the sanitization procedures presented. In addition to overwriting, alternative sanitization processes and other ancillary risk management considerations have been discussed as well.

The Silicon Graphics Incorporated IRIX "FX" low level disk exerciser utility has been thoroughly examined. Then stepwise operating and verification procedures have been described, including supporting visual aides. This paper has shown that the FX utility combined with validated procedures will provide a reasonably secure three-pass overwrite that meets the security standards in use by the United States Department of Defense.

APPENDIX A – AUTOMATED SCRIPT

```
#!/bin/sh
# $Date: 03/03/24 10:22:46 $
# $Revision: 1.0 $
# $State: Exp $
#
# NAME
#     overwrite - automate prompts to FX for three-pass overwrite.
#
# SYNOPSIS
#     overwrite.sh
#
# DESCRIPTION
#     This script prompts the user for the SCSI controller and disk
#     drive address to target for three-pass DoD 5220-22.M overwrite.
#     Then FX is called using input redirected from this script.
#
# OUTPUT
#     Session log and error files as defined below.
#
# DIAGNOSTICS
#     Error checks are weak.
#
# SEE ALSO
#     fx(1M)
#
# AUTHOR
#     Michael J. Davis

# Avoid Trojans, declare full pathnames.
AWK=/usr/bin/awk
BASENAME=/usr/bin/basename
DATE=/usr/bin/date
ECHO=/usr/bin/echo
EGREP=/usr/bin/egrep
FX=/bin/fx
HINV=/usr/bin/hinv
SORT=/usr/bin/sort


# Restrict permissions on log files.
umask 077
LOG=/tmp/log
ERRORS=/tmp/errors

# Is this an IRIX 6.5 system?
SYSCHK="`/usr/bin/uname -r`"
if [ "$SYSCHK" != "6.5" ]; then
    $ECHO "`$BASENAME $0`: `/usr/bsd/hostname` not an IRIX 6.5 system!"
1>&2
    exit
fi

# Destroy any command line arguments issued with this script.
[ $# -gt 0 ] && shift $#
```

```
# Must run as root.
/usr/bin/whoami | $EGREP -s "root"
if [ $? -ne 0 ]; then
    $ECHO "`$BASENAME $0`: must be run as root!" 1>&2
    exit
fi

# Ensure a SCSI disk is in the inventory.
$HINV | $EGREP Disk | $EGREP -s SCSI
if [ $? -ne 0 ]; then
    $ECHO "`$BASENAME $0`: no SCSI disk found!" 1>&2
    exit
fi
$ECHO "SCSI Disks Shown in Hardware Inventory:"
$HINV | $EGREP Disk | $EGREP SCSI

# Sanity checks.
t_CTLR="`$HINV | $EGREP Disk | $EGREP SCSI | $AWK '{print $NF}' | $SORT
-u`"
t_DRIVE="`$HINV | $EGREP Disk | $EGREP SCSI | $AWK '{print $4}' | $SORT
-u`"

# Prompt user for controller address, like FX.
$ECHO $0: "ctlr = (`$ECHO $t_CTLR | $AWK '{print $1}'`) \c"
read CTLR
if [ "$CTLR" == "" ]; then
    CTLR=`$ECHO $t_CTLR | $AWK '{print $1}'`
else
    $ECHO "$t_CTLR" | $EGREP -s "$CTLR"
    if [ $? -ne 0 ]; then
        $ECHO "`$BASENAME $0`: controller $CTLR not recognized!" 1>&2
        exit
    fi
fi

# Prompt user for drive address, like FX.
$ECHO $0: "drive = (`$ECHO $t_DRIVE | $AWK '{print $NF}'`) \c"
read DRIVE
if [ "$DRIVE" == "" ]; then
    DRIVE=`$ECHO $t_DRIVE | $AWK '{print $NF}'`
else
    $ECHO "$t_DRIVE" | $EGREP -s "$DRIVE"
    if [ $? -ne 0 ]; then
        $ECHO "`$BASENAME $0`: drive $DRIVE not recognized!" 1>&2
        exit
    fi
fi

# Confirm overwrite, must enter "yes", like FX.
$ECHO "about to destroy data on disk dksc($CTLR,$DRIVE,0)! ok? (no) \c"
read k
if [ "$k" != "yes" ]; then
    $ECHO "`$BASENAME $0`: terminated by user!" 1>&2
    exit
fi

# Silently find second & last seek for verifications after second pass.
```

```
(
$FX -x << EOF
dksc
$CTLR
$DRIVE
0
repartition
/exit
no
EOF
) >$LOG
LAST=`$EGREP '10: volume' $LOG | $AWK '{print int($5-1)}'`
# Second location should be "random".
RAND=`$ECHO $LAST | $AWK '{srand(); print int($0 * rand())}'`

# Call FX and perform the overwrites
(
$ECHO "BEGIN: `$DATE`"

$FX -x << EOF
dksc
$CTLR
$DRIVE
0
repartition
..
badblock/showbb -g
exercise/settestpat 0
..
exercise/showtestpat
exercise/sequential
wr-cmp
0

1
yes
exercise/settestpat 255
..
exercise/showtestpat
exercise/sequential
wr-cmp
0

1
yes
debug/seek 0
debug/readbuf 0 1
debug/dumpbuf b 0 512
debug/seek $RAND
debug/readbuf 0 1
debug/dumpbuf b 0 512
debug/seek $LAST
debug/readbuf 0 1
debug/dumpbuf b 0 512
/exit
no
EOF
```

```
$FX -x << EOF
dksc
$CTLR
$DRIVE
0
exercise/showtestpat
exercise/sequential
wr-cmp
0

1
yes
/exit
no
EOF

$ECHO "\nEND: `$DATE`"
) >$LOG 2>$ERRORS

# End
$ECHO "`$BASENAME $0`: destruction complete dksc($CTLR,$DRIVE,0), see
$LOG and $ERRORS"
```

WORKS CONSULTED

Cole, Eric, Mathew Newfield, and John H. Millican. GSEC Security Essentials Toolkit. Indianapolis: Que Publishing, 2002.

eTesting Labs Inc. "Hard Disk Data Erasure Product Functionality Test." Commissioned by Redemtech. April 2002. 1 March 2003 <http://www.redemtech.com/extaudit//news_research/eTesting_findings.pdf>.

Gutmann, Peter. "Secure Deletion of Data from Magnetic and Solid-State Memory." Sixth USENIX Security Symposium Proceedings. 22-25 July 1996. 1 March 2003 <http://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html>.

Infraworks. "Completely Scrub and Eliminate Data on Hard Drives." Advertisement. 9 March 2003 <http://www.infraworks.com/it_sanitizer.htm>.

Lyle, James. United States. Dept. of Commerce. National Institute of Standards and Technology. "Notes on dd and Odd Sized Disks." January 11, 2002. 25 March 2003 <http://www.cftt.nist.gov/Notes_on_dd_and_Odd_Sized_Disks4.doc>.

Mallery, John R. "Secure File Deletion, Fact or Fiction?" SANS Info Sec Reading Room. 16 July 2001. 1 March 2003 <http://www.sans.org/rr/incident/deletion.php>.

Newton, Stephen. "Security Doesn't Come in a Box." SANS NewsBites. Vol. 5 Num 10 March 12, 2003. 12 March 2003 <http://www.sans.org/newsletters/newsbites/vol5_10.php>.

---. "Proper Use of Our Trademarks." 2003. 9 March 2003 <http://www.sgi.com/company_info/trademarks/proper.html#pa>.

Rehman, Doug. "RE: Should forensic tools be certified?." Computer Forensics Tool Testing (cftt-subscribe@yahoogroups.com). 4 Feb 2003. 17 March 2003 <http://groups.yahoo.com/group/cftt>.

Roberts, Paul. "Study Shows Old Drives Not Adequately Cleaned." SANS NewsBites. Vol. 5 Num 3. January 22, 2003. 1 March 2003 <http://www.sans.org/newsletters/newsbites/vol5_3.php>.

Silicon Graphics Incorporated. Man Page dd(1M). "dd – convert and copy a file." dd version 6.5. n.d.

---. Man Page df(1). "df – report number of free disk blocks." df version 6.5. n.d.

---. Man Page fx(1M). "fx – disk utility." fx version 6.5. n.d.

---. Man Page mount(1M). "mount, umount – mount and unmount filesystems." version 6.5. n.d.

---. SGI Statements of Volatility. Confidential. Document Number 007-4200-003. Mountain View: Silicon Graphics, Inc., 1999-2000.

---. "SGI Wins Bulk of DoD High Performance Computing Modernization Program's Fiscal Year 2003 Supercomputer Procurement." 18 February 2003. 1 March 2003 <http://www.sgi.com/newsroom/press_releases/2003/february/dod.html>.

United States. Dept. of Defense. Assistant Secretary of Defense. "Disposition of Unclassified DoD Computer Hard Drives." 4 June 2001. 13 March 2003 <http://www.c3i.osd.mil/org/sio/ia/diap/documents/ASD_HD_Disposition_memo060401.pdf>.

---. ---. Defense Security Service. "Assessed Products List." 2 January 2002. 1 March 2003 <http://www.dss.mil/infoas/assessed_products_list.doc>.

---. ---. ---. "Clearing and Sanitization Matrix." n.d. 1 March 2003 <http://www.dss.mil/infoas/clearing_and_sanitization_matrix.doc>.

---.---. "Clearing and Sanitization Matrix." National Industrial Security Program Operating Manual. (NISPOM). DoD 5220.22-M. January 1995 (Original Version).

---. National Computer Security Center. A Guide to Understanding Data Remanence in Automated Information Systems. NCSC-TG-025 Version-2. September 1991. 1 March 2003 <http://www.radium.ncsc.mil/tpep/library/rainbow/NCSC-TG-025.2.html>.

---. National Security Telecommunications and Information Systems Security Committee. National Information Systems Security (INFOSEC) Glossary. NSTISSI No. 4009. September 2000. 1 March 2003 <http://www.nstissc.gov/Assets/pdf/4009.pdf>.

NOTES

[1] Silicon Graphics, SGI, and IRIX are registered trademarks of Silicon Graphics, Inc., in the United States and/or other countries worldwide.

[2] http://www.sans.org/newsletters/newsbites/vol5_10.php

[3] http://www.sans.org/newsletters/newsbites/vol5_3.php

[4] http://www.radium.ncsc.mil/tpep/library/rainbow/NCSC-TG-025.2.html

[5] http://www.sans.org/rr/incident/deletion.php

[6] http://www.nstissc.gov/Assets/pdf/4009.pdf

[7] http://www.c3i.osd.mil/org/sio/ia/diap/documents/ASD_HD_Disposition_memo060401.pdf

[8] A Degausser Products List (DPL) can be obtained by contacting the National Security Agency, Attn: S7 Media Technology Center, 9800 Savage Road, Ft. George Meade, MD 20755-6877. Tel: 1 (800) 688-6115.

[9] http://www.redemtech.com/extaudit//news_research/eTesting_findings.pdf

[10] http://www.dss.mil/infoas/assessed_products_list.doc

[11] Service charges may apply.

[12] http://www.sgi.com/newsroom/press_releases/2003/february/dod.html

[13] http://www.dss.mil/infoas/clearing_and_sanitization_matrix.doc

[14] http://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html

[15] For IDE disks, bad block lists are maintained by the driver.

[16] Odd as that may sound, some overwrite products that use proprietary means to access the disk at a low level will attempt to write to, or read from, grown defect sectors; and if unsuccessful may either dump the sector's contents for review (if feasible) or otherwise report the error.

[17] The FX main menu is a certain reference point, the operator may prefer to enter the submenu.

[18] Verification is performed at this point rather than after the third and final overwrite pass because the third pass consists of random characters. Notice that the precedence of this verification is different than the earlier work acknowledged. The NISPOM standard has been revised since that earlier work, previously stating "Overwrite all addressable locations with a character, its complement, then a random character and verify." The revised standard is better informed.

[19] Although not observed by this author, there is suggestion that an overwrite utility may fail to write the last block, so the last block (nblocks - 1) must be verified. This is perhaps an area for further research as there has been discussion within the Computer Forensics Tool Testing news group regarding "The odd sector acquisition problem [that] is a good example of an anomaly that

cuts across multiple tools" (Rehman). "Bottom line: The [sic] some operating systems or underlying drivers skip that last sector, but others see it all" (Lyle).

[20] A drive that has been cleared or sanitized by this procedure is not bootable, not even after installing an operating system. After completing the procedure the disk label contains only a partition table. To boot, SGI requires the appropriate version of sash to be written to the label. The SGI utility "dvhtool" is one way to copy sash into the disk label.

[21] IRIX does not provide /dev/random.

Last Updated: August 21st, 2017

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Tampa - Clearwater 2017	Clearwater, FLUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Network Security 2017	Las Vegas, NVUS	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, IE	Sep 11, 2017 - Sep 16, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MDUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, DK	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, GB	Sep 25, 2017 - Sep 30, 2017	Live Event
Data Breach Summit & Training	Chicago, ILUS	Sep 25, 2017 - Oct 02, 2017	Live Event
Rocky Mountain Fall 2017	Denver, COUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, NL	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Oslo Autumn 2017	Oslo, NO	Oct 02, 2017 - Oct 07, 2017	Live Event
SANS DFIR Prague 2017	Prague, CZ	Oct 02, 2017 - Oct 08, 2017	Live Event
SANS Phoenix-Mesa 2017	Mesa, AZUS	Oct 09, 2017 - Oct 14, 2017	Live Event
SANS October Singapore 2017	Singapore, SG	Oct 09, 2017 - Oct 28, 2017	Live Event
SANS AUD507 (GSNA) @ Canberra 2017	Canberra, AU	Oct 09, 2017 - Oct 14, 2017	Live Event
Secure DevOps Summit & Training	Denver, COUS	Oct 10, 2017 - Oct 17, 2017	Live Event
SANS Tysons Corner Fall 2017	McLean, VAUS	Oct 14, 2017 - Oct 21, 2017	Live Event
SANS Tokyo Autumn 2017	Tokyo, JP	Oct 16, 2017 - Oct 28, 2017	Live Event
SANS Brussels Autumn 2017	Brussels, BE	Oct 16, 2017 - Oct 21, 2017	Live Event
SANS Berlin 2017	Berlin, DE	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS San Diego 2017	San Diego, CAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Seattle 2017	Seattle, WAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, AE	Nov 04, 2017 - Nov 16, 2017	Live Event
SANS Miami 2017	Miami, FLUS	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Amsterdam 2017	Amsterdam, NL	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Milan November 2017	Milan, IT	Nov 06, 2017 - Nov 11, 2017	Live Event
Pen Test Hackfest Summit & Training 2017	Bethesda, MDUS	Nov 13, 2017 - Nov 20, 2017	Live Event
SANS Paris November 2017	Paris, FR	Nov 13, 2017 - Nov 18, 2017	Live Event
SANS Sydney 2017	Sydney, AU	Nov 13, 2017 - Nov 25, 2017	Live Event
SANS San Francisco Fall 2017	OnlineCAUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced