Chapter 18

EXTRACTING CONCEALED DATA FROM BIOS CHIPS

P. Gershteyn, M. Davis, G. Manes and S. Shenoi

Abstract The practice of digital forensics demands thorough, meticulous examinations of all data storage media seized in investigations. However, BIOS chips and other firmware are largely overlooked in forensic investigations. No forensically sound procedures exist for imaging BIOS chips and no tools are available specifically for analyzing BIOS image files. Yet, significant amounts of data may be stored on BIOS chips without hindering machine performance.

This paper describes robust techniques for concealing data in BIOS freespace, BIOS modules, and throughout a BIOS chip. Also, it discusses how flashing utilities and traditional digital forensic tools can be used to detect and recover concealed data.

Keywords: BIOS chips, firmware, data concealment, evidence acquisition

1. Introduction

The Basic Input/Output System (BIOS) of a computer is an interface that enables its hardware and software to interact with each other [7, 16, 18]. BIOS chips provide diagnostics and utilities necessary for loading operating systems. No computer – from the smallest embedded device to the largest supercomputer – can function without a BIOS.

BIOS chips typically contain 128 to 512K of flash memory, which can be used to conceal data. A BIOS writing technique was exploited by the 1998 Win95/CIH virus that wiped out hard drives. Computer game enthusiasts often use BIOS editing to "mod" computers with personalized graphics [3]. We were able to store 40 pages of *The Jolly Roger's Cookbook* [11] on a functioning BIOS chip. Criminals can adopt similar techniques to conceal information: drug contacts, financial records, digital photographs, or cryptographic keys that encrypt child pornography

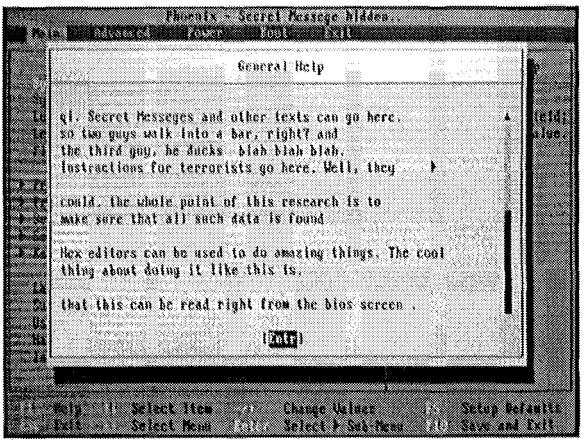

Figure 1. BIOS setup screen displaying hidden data.

stored on hard drives. However, BIOS chips are largely overlooked in forensic investigations. No forensically sound procedures exist for imaging BIOS chips and no tools are available specifically for analyzing BIOS image files.

This paper describes robust techniques for concealing data in BIOS freespace, BIOS modules, and throughout a BIOS chip. Also, it discusses how flashing utilities and traditional digital forensic tools can be used to detect and recover concealed data.

2. BIOS Overview

The Basic Input/Output System (BIOS) of a computer is an interface that enables its hardware and software to interact with each other [7, 16, 18]. Typically located on the motherboard, the BIOS contains software necessary for the computer to start, including instructions for performing a Power-On Self-Test (POST) and reading hard drive boot sectors [2, 18]. BIOS chips also offer basic diagnostics utilities and provide low-level routines that operating systems may use for communicating with hardware. BIOS configurations are stored on a CMOS chip and powered by a small lithium or nickel-cadmium battery that allows the CMOS to store data for several years. Modern BIOS chips use flash memory that enables them to be modified, updated and erased. BIOS chips on most modern computers have storage capacities between 128 and 512K.

Executable code within a BIOS is typically active only during the boot process and until operating system hardware drivers are loaded into memory [5]. From this point on, operating system commands are used to interact with hardware devices and the BIOS maintains only limited

control over low-level hardware features such as power management and certain software interrupts.

BIOS software is modular, enabling it to perform separate functions at startup depending on the computer's hardware configuration. The software consists of two main parts: compressed modules and an area called the "bootblock," which is responsible for decompressing and executing the modules. Hardware that requires initialization can be recognized and loaded by the BIOS. The BIOS checks reserved memory locations for specific byte sequences and headers that indicate the location and size of the hardware initialization code. Modern BIOSs check memory from 0x0C0000 to 0x0EFFFF at 2K boundaries, sidestepping memory ranges used by video hardware and the system BIOS itself. This behavior provides for plug-and-play functionality of hardware devices at the BIOS level [7].

2.1 BIOS Boot Process

When a computer is powered on, the processor first accesses a predetermined area of the system BIOS ROM to access the BIOS boot program. This location is at memory offset 0xFFFF0, sixteen bytes below the top of real mode memory [12]. Real mode is the default operating mode for modern Intel-architecture processors, allowing the processor to mimic the 8088 chip. Offset 0xFFFF0 is not located in RAM, but actually on the BIOS ROM chip. This location contains a jump instruction to a memory offset that contains the BIOS startup code, typically 0xFE05B.

At this stage of the process, most flash BIOS chips copy themselves to RAM in a process known as "shadowing," which allows faster access to the BIOS code while also decompressing and defragmenting noncontiguous BIOS code. The BIOS code then performs the Power-On Self-Test (POST). POST checks if all necessary hardware is present and functioning normally. Since the video card is not active at this point, POST uses beep codes to alert the user to fatal errors encountered during the boot process [2].

Next, the BIOS runs the video card's built-in BIOS program, which is normally found at location 0xC000. The video card's BIOS initializes the video card and displays the confirmation on the screen, providing the first visual indication that the computer is booting. The system BIOS then looks for and executes all other BIOS-enabled devices, displays the startup screen, and conducts more system tests, including a memory count. All fatal errors encountered by the BIOS at this point are displayed on the screen.

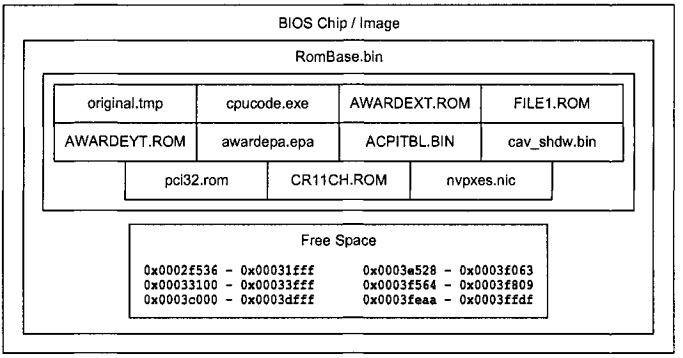

Figure 2. Schematic diagram of BIOS storage.

The BIOS then configures system hardware, determines memory timing and dynamically sets hardware parameters [12]. The BIOS stores hardware parameters in a 256 byte portion of reserved RAM between locations 0x000400 and 0x0004FF, which allows the operating system and other programs to reference hardware information. Once the BIOS completes its inventory of computer hardware, it displays a summary and begins searching for a boot device. After the BIOS finds a device containing boot information it executes code from the first sector of that device.

2.2 BIOS Storage

This work focuses on an ASUS A7N266-VM motherboard with a socketed BIOS chip [1]. The BIOS version is the original ASUS A7N266-VM ACPI BIOS Rev. 1004/AA with build date 08/23/02.

Figure 2 shows the storage structure of the ASUS A7N266-VM BIOS. It consists of eleven modules stored at the start of the file, followed by BIOS data interspersed with freespace.

Data may be stored at several sites on a BIOS chip's flash memory. Depending on the location and quantity of data stored, the BIOS could remain functional or it could become corrupted, rendering the computer inoperable. BIOS chips are designed to be expandable. Consequently, they have large portions of freespace that can be overwritten with data without adversely affecting the operation of the chip. Figure 3 shows a schematic diagram of a BIOS chip with data hidden in its freespace.

Data may also be stored within BIOS modules. Modules often contain text strings that are displayed as messages, e.g., obscure error messages and hardware data. These text strings can be overwritten with data without affecting the operation of the chip. Figure 4 shows a BIOS chip

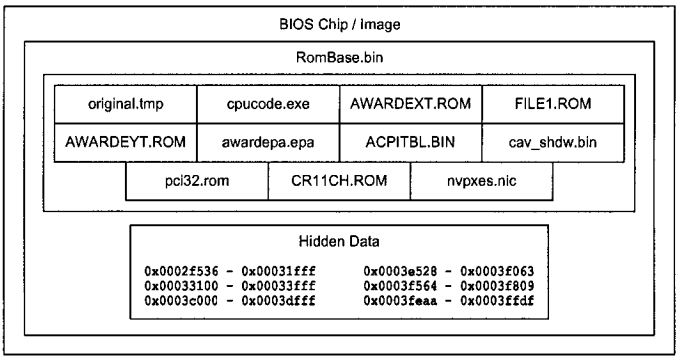

Figure 3. BIOS with hidden data in freespace.

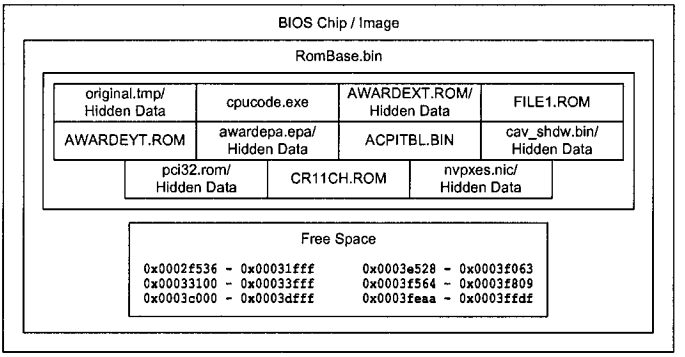

Figure 4. BIOS with hidden data in modules.

with hidden data in six modules. A procedure for concealing data in two of these modules is described in Section 3.2.

If BIOS functionality is not a concern, the entire BIOS chip memory (256K in the case of ASUS) can be used for data storage. However, this makes the recovery of the data problematic, as the computer cannot be booted when the BIOS has been corrupted.

2.3 BIOS Flashing

Computer manufacturers often release updated BIOS software (images or ROMs) for their motherboards [6]. A BIOS is upgraded using a "flashing" program, which erases the chip and replaces its software with the new BIOS image. Sometimes, manufacturers package a flashing program and BIOS image as a single executable file. Alternatively, a third party flashing program (Uniflash [17]), which is compatible with most motherboards and BIOSs can be used to upgrade a BIOS. These flashing programs can be used to conceal data on BIOS chips.

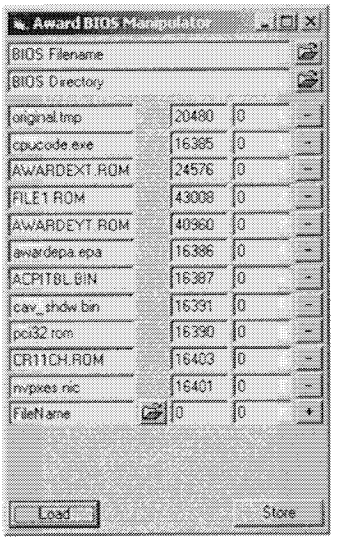

Figure 5. AwardMod screen during extraction of BIOS modules.

Most flashing programs run from the command prompt and require the computer to be running in the DOS mode with no other programs, drivers or services running. Therefore, an MS-DOS boot disk must be modified to create an environment for running a flashing program. Appropriate boot disks, e.g., Caldera Dr-DOS [4], may also be downloaded from the Internet. Newer motherboards now support BIOS flashing from Windows using special software; this makes it possible to quickly read and write BIOS chips.

A BIOS utility, e.g., AwardMod [8], can be used to extract, delete and add modules to a BIOS image. Figure 5 shows an AwardMod screen during the process of extracting ASUS BIOS modules. Hex editors may also be used to read and modify BIOS modules, except for those containing graphics, e.g., the BIOS boot logo, which is encoded in EPA format. A separate program, such as EPACoder [15], facilitates the editing process by converting between EPA and bitmap graphics. Figure 6 shows EPACoder being used to replace the standard BIOS logo with a skull and crossbones.

Editing BIOS modules with AwardMod can corrupt the chip. To recover from this failed flashing attempt, it is necessary to boot the computer in order to re-flash the BIOS. Since a computer with a corrupt BIOS will not boot, the "hotflashing" technique [9] must be used. Hotflashing involves replacing the corrupt BIOS chip with a working chip, booting the computer to a state that allows flashing, and then switching

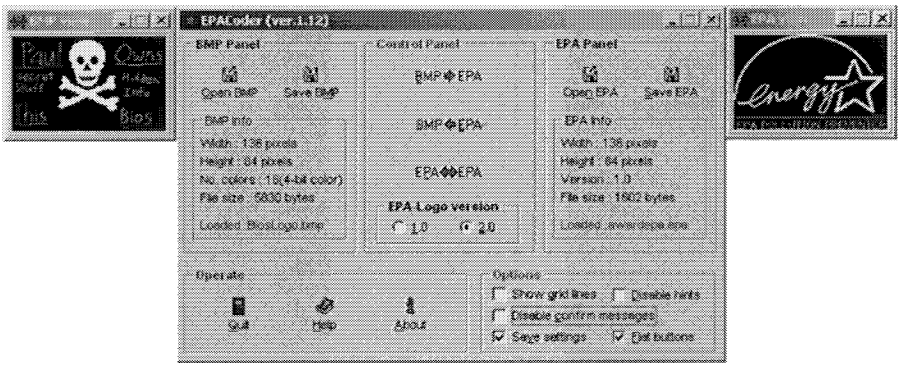

Figure 6. EPACoder screen during bitmap to BIOS image conversion.

the working chip with the corrupt chip while the computer is running. This permits the corrupt chip to be re-flashed.

Special hardware tools, e.g., BIOS Savior [10], simplify the hotflashing process. BIOS Savior interfaces the motherboard and BIOS chip, and provides a backup BIOS chip. A switch allows the user to choose between the original BIOS and the BIOS Savior backup chip. Thus, the user can hotswap BIOS chips with the flip of a switch rather than having to physically remove and insert chips into a running computer.

3. Data Concealment

This section describes techniques for concealing data in: (i) BIOS freespace, (ii) BIOS modules, and (iii) throughout the BIOS chip. The first two techniques produce a usable BIOS chip with hidden data. The third technique can hide a substantial amount of data, but it renders the chip unusable. Nevertheless, the hidden data can be extracted using special techniques (see Section 4).

The BIOS Savior tool [10] is used for hotflashing [9]. Caldera Dr-DOS [4] is used to boot the computer into a state where flashing the BIOS chip is possible. An ASUS flashing program (`aflash.exe` [1]) is used to read and write to the BIOS chip. AwardMod [8] is used to extract and replace BIOS modules. A hex editor (Hex Workshop) is used to edit BIOS data. EPACoder [15] is used to convert graphical images to a usable format. A separate workstation is used to manage the process of flashing the BIOS.

3.1 Overwriting BIOS Freespace

Because BIOS software is designed to be upgradeable, BIOS chips typically have significant amounts of freespace. The following procedure

describes how data may be hidden in BIOS freespace without affecting its operation. Note, however, that prior research is necessary to verify which blocks of freespace can be used without affecting a BIOS chip.

BIOS Freespace Overwriting Procedure

1. Procure an MS-DOS compatible boot disk approved for BIOS flashing (e.g., Caldera Dr-DOS). The disk should not execute any terminate and stay resident (TSR) programs.

2. Copy the ASUS flashing program (`aflash.exe`) to the boot disk.

3. Boot the ASUS machine using the boot disk. This may require altering the drive boot order in the CMOS settings. After Caldera Dr-DOS has booted, execute `aflash.exe`. Backup the original BIOS to the floppy disk and save the file on the boot disk as `asback.bin`.

4. Place the boot disk in the workstation and copy `asback.bin` to the hard drive.

5. Find all 8 blocks of null characters in `asback.bin`. Null blocks are long strings of either `0`s or `F`s. Since these blocks represent free space, data may be written to them without corrupting the BIOS. The null blocks are present at the following locations:

```
Block 1:  FFFFs at 0x0002F536--0x00031FFF
Block 2:  0000s at 0x00032A26--0x00032FFD
Block 3:  0000s at 0x00033100--0x00033FFF
Block 4:  FFFFs at 0x0003B6A0--0x0003BFFF
Block 5:  0000s at 0x0003C000--0x0003DFFF
Block 6:  0000s at 0x0003E528--0x0003F063
Block 7:  0000s at 0x0003F564--0x0003F809
Block 8:  0000s at 0x0003FEAA--0x0003FFDF
```

   Note that editing any part of Block 2 corrupts the BIOS. Also, while editing Block 4 will not corrupt the BIOS, the stored data cannot be recovered. The remaining blocks permit both data storage and retrieval.

6. Select a file (`evidence.rar`) to be hidden that can fit within the null blocks, which in this case is 26,850 bytes. Compression may be used to store more data than would otherwise be possible. Multiple files may also be stored as long as they do not exceed a total of 26,850 bytes.

7. Write `evidence.rar` across the empty blocks. Blocks that are not filled with data should be padded with zeros. A file that is too large for a block can be split using a hex editor and written to multiple blocks.

8. After the null bytes of `asback.bin` are overwritten by the data in `evidence.rar`, save `asback.bin` and rename it `asedited.bin`.

9. To complete the process of hiding `evidence.rar`, copy `asedited.bin` to the boot disk, boot the ASUS using the boot disk, and flash the BIOS. This is done by typing `aflash /boot /auto asedited.bin` at the command prompt.

10. Restart the computer to verify that it still functions properly.

3.2 Editing BIOS Modules

Significant amounts of data may be hidden within BIOS modules without affecting the operation of the chip. Modules often contain text strings that are displayed as messages (e.g., obscure error messages and hardware data); these text strings can be overwritten with data. Likewise, modules contain considerable amounts of freespace that can also be used to hide data.

The procedure for concealing data in two modules, `awardepa.epa` and `awardext.rom`, is described below. The module `awardepa.epa` stores the BIOS boot logo; `awardext.rom` contains text that is displayed at the BIOS setup screen. Data hidden in these modules can be read from the screen without having to image the BIOS. Note that the boot logo is stored as `awardepa.epa` in one of two formats, EPA 1 and EPA 2. EPA 1 is a limited format: a picture is divided into cells, each cell limited to two colors. Despite this restriction, it is possible to store information in this graphic. The ASUS BIOS only supports EPA 1.

BIOS Module Editing Procedure

1 Procure an MS-DOS compatible boot disk (Caldera Dr-DOS) approved for BIOS flashing.

2 Copy the ASUS flashing program (`aflash.exe`) to the boot disk.

3 Boot the ASUS machine using the boot disk. After Caldera Dr-DOS has booted, execute `aflash.exe`. Backup the original BIOS to the floppy disk as `asback.bin`. Place the boot disk in the workstation and copy `asback.bin` to the hard drive.

4 Copy `asback.bin` to the workstation from the boot disk and run AwardMod. AwardMod can be used to load and store a binary BIOS file or a directory containing the extracted modules from the binary BIOS file. Use AwardMod to extract the modules in `asback.bin` and store them in a directory.

5 Use a hex editor to edit the extracted module `awardext.rom`. The strings in this file are the same strings that are visible on the BIOS setup screen. Since the help text is scrollable and relatively long, this is the best place to write data. Save the edited version of `awardext.rom` as `zawardext.rom`.

6 Use EPACoder to convert `awardepa.epa` to a bitmap. Edit the bitmap using a graphics program and convert it back to the EPA 1 format using EPACoder.

7 Use AwardMod to open `asback.bin`. Copy the module number associated with `awardext.rom`. Then, delete `awardext.rom`. Next, enter the location of `zawardext.rom` in the new module (file name) prompt in AwardMod. Also, paste `awardext.rom`'s module number into the box associated with the new module, and add this module to the BIOS.

8 Similarly, delete `awardepa.epa` and add `zawardepa.epa`, making sure that `zawardepa.epa` has the same module number as `awardepa.epa`.

9 Enter the new file name `asedited.bin` and its location in the BIOS file name prompt of AwardMod to write this file on the hard drive. Flash this file to the ASUS BIOS chip using the boot disk and the command `aflash.exe /boot /auto asedited.bin`.

10 Restart the computer to verify that it still functions properly. Also verify that the changes made to the modules are reflected in the BIOS startup and setup screens.

3.3 Overwriting the Entire BIOS

As discussed earlier, the entire BIOS chip memory (256K in the case of ASUS) can be used to hide data. However, this makes the recovery of the data problematic, as the computer cannot be booted with a corrupted BIOS. Hotflashing is one solution. Alternatively, the motherboard must have two BIOS chips or a BIOS backup device, e.g., BIOS Savior, must be used to recover the hidden data. The procedure described below makes use of BIOS Savior. The user can flip a switch on the computer to choose whether the original chip or the BIOS Savior chip should be used.

Entire BIOS Overwriting Procedure

1 Procure an MS-DOS compatible boot disk (Caldera Dr-DOS) approved for BIOS flashing.

2 Copy the ASUS flashing program (`aflash.exe`) to the boot disk.

3 Create a RAR archive with the files to be hidden. The files should not be compressed to ensure that the size of the RAR archive is predictable.

4 When the RAR archive is close to (but less than) 262,144 bytes save and close the archive. Then, open it in a hex editor and pad with zeros at the end until the total size of the RAR archive is exactly 262,144 bytes. Name the RAR archive `evidence.rar`.

5 Copy `evidence.rar` to the boot floppy. Boot the ASUS with the boot floppy and flash `evidence.rar` to the BIOS chip using the command `aflash /boot /auto evidence.rar`. Note that the ASUS will no longer be able to boot with its BIOS chip.

4. Concealed Data Recovery

This section describes procedures for detecting and extracting data that has been hidden using the procedures described in Section 3. Data hidden in the freespace cannot be detected without imaging the BIOS chip. However, it may be possible to detect if data is hidden within BIOS modules before imaging the BIOS, depending on which modules contain hidden data. It is always possible to detect whether or not the

entire BIOS chip has been overwritten with data simply by turning on the computer.

The following procedure should be followed for investigating a seized computer that may have data hidden in its BIOS chip.

Initial Investigative Procedure

1 Turn on the seized computer after its hard drives have been removed. If the computer does not boot, it is possible that the entire BIOS chip has been overwritten with data.

2 Examine the BIOS startup and setup screens for any unusual text or graphics. The existence of anomalies indicates that the BIOS modules have been edited. Note, however, that the absence of anomalies does not guarantee that the BIOS modules are free of hidden data.

3 Search the seized storage media for BIOS modification tools and flashing programs. The presence of such software may provide clues with regard to the type of data hidden in the BIOS as well as the technique used.

The following three subsections describe procedures for detecting and extracting hidden data from various locations in a BIOS chip.

4.1 Recovering Data from BIOS Freespace

The following procedure should be used if the investigator suspects that data is hidden on the BIOS chip, although there is no visible evidence of it on the BIOS startup and setup screens. Note that BIOS modules should also be checked for hidden data.

BIOS Module Freespace Recovery Procedure

1 Procure an MS-DOS compatible boot disk (Caldera Dr-DOS) approved for BIOS flashing.

2 Copy the ASUS flashing program `aflash.exe` to the boot disk.

3 Boot the ASUS machine using the boot disk. After Caldera Dr-DOS has booted, execute `aflash.exe`. Backup the original BIOS to the floppy disk as `asback.bin`.

4 Place the boot disk in the workstation and copy `asback.bin` to the hard drive.

5 Use forensic utilities (e.g., Foremost, Encase, Forensic Tool Kit, ILook [13, 14]) to examine the BIOS image for file headers and regular expressions, and preserve all data of interest.

6 If the hidden data cannot be found using the forensic utilities, use a hex editor to compare the seized BIOS image with a clean copy of the BIOS image from the motherboard manufacturer's website. This comparison assists in locating hidden data.

7 If a clean copy of the BIOS image is not available, examine the seized BIOS's image with a hex editor and look for suspicious text strings.

8 Use forensically sound procedures to copy and preserve all data of interest.

4.2 Recovering Data from BIOS Modules

If hidden data is found upon examining all the BIOS setup screens, the BIOS modules must be processed to recover the evidence. Even if no hidden data is found within the BIOS startup and setup screens, it is still possible for data to be hidden within the modules. The following procedure should be performed along with the procedure to recover hidden data from BIOS freespace (Section 4.1).

BIOS Module Data Recovery Procedure

1 Procure an MS-DOS compatible boot disk (Caldera Dr-DOS) approved for BIOS flashing.

2 Copy the ASUS flashing program `aflash.exe` to the boot disk.

3 Boot the ASUS machine using the boot disk. After Caldera Dr-DOS has booted, execute `aflash.exe`. Backup the original BIOS to the floppy disk as `asback.bin`.

4 Place the boot disk into the workstation and copy `asback.bin` to the hard drive.

5 Use AwardMod to extract all modules from `asback.bin`.

6 Use forensic utilities (e.g., Foremost, Encase, Forensic Tool Kit, ILook [13, 14]) to examine the BIOS modules for file headers and regular expressions, and preserve all data of interest.

7 If the hidden data cannot be found using the forensic utilities, use a hex editor to compare the seized BIOS's modules with those in a clean copy of the BIOS image from the motherboard manufacturer's website. This comparison assists in locating hidden data.

8 If a clean copy of the BIOS image is not available, examine the seized BIOS's modules with a hex editor and look for suspicious text strings.

9 Use forensically sound procedures to copy and preserve all data of interest.

4.3 Recovering Data from Entire BIOS Chip

Recovering data from a BIOS chip that has been overwritten completely requires the chip to be physically removed. The hotflashing technique or a BIOS Savior may be used to image the seized BIOS chip. Alternatively, a chip programmer can be used.

Data of any type may be hidden on a BIOS chip. Therefore, it is recommended that forensic tools be used to conduct extensive examinations of the BIOS image. Careful manual examination of the hex code must also be performed. Forensically sound procedures must be used to copy and preserve all data of interest.

5. Conclusions

Modern BIOS chips can hold substantial amounts of hidden data without affecting their performance. This paper shows how data may be hidden in BIOS freespace, BIOS modules, and throughout a BIOS chip. Also, it presents forensically sound techniques for detecting and recovering concealed data. The work is intended to raise awareness about the ability of malicious individuals to store secret information on BIOS chips and other firmware. Moreover, it should stimulate new research in the area of firmware forensics.

References

[1] ASUS, A7N266-VM/AA motherboard support (support.asus.com), 2003.

[2] BIOS Central (www.bioscentral.com).

[3] BIOSMods (www.biosmods.com).

[4] Bootdisk.com (bootdisk.com).

[5] P. Croucher, *The BIOS Companion*, Electrocution Technical Publishers, Calgary, Alberta, Canada, 1998.

[6] W. Gatliff, Implementing downloadable firmware with flash memory, in *The Firmware Handbook*, J. Ganssle (Ed.), Elsevier, Burlington, Massachusetts, pp. 285-297, 2004.

[7] Gen-X-PC, BIOS info (www.gen-x-pc.com/BIOS_info.htm).

[8] J. Hill, AwardMod (sourceforge.net/projects/awardmod/), 2002.

[9] K. Hindistan, BIOS flashing and hotflashing (www.onlamp.com/pub/a/onlamp/2004/03/11/bios_hotflash.html), 2004.

[10] IOSS, RD1 BIOS Savior (www.ioss.com.tw), 2000.

[11] Jolly Roger, *The Jolly Roger's Cookbook* (www.textfiles.com), 1990.

[12] C. Kozierok, System BIOS (www.pcguide.com), 2001.

[13] K. Mandia, C. Prosise and M. Pepe, *Incident Response and Computer Forensics*, McGraw-Hill/Osborne, Emeryville, California, 2003.

[14] G. Mohay, A. Anderson, B. Collie, O. de Vel and R. McKemmish, *Computer and Intrusion Forensics*, Artech, Norwood, Massachusetts, 2003.

[15] S. Nikolayev and A. Prokopiuk, EPACoder (shareware.pcmag.com /product.php[id]38610[cid]301[SiteID]pcmag), 2000.

[16] Phoenix Technologies, *System BIOS for IBM PCs, Compatibles and EISA Computers (2nd Edition)*, Addison-Wesley Longman, Boston, Massachusetts, 1991.

[17] Rainbow Software, Uniflash (www.uniflash.org), 2005.

[18] A. Wong, *Breaking Through the BIOS Barrier: The Definitive BIOS Optimization Guide for PCs*, Prentice Hall, Indianapolis, Indiana, 2004.