Chapter 24

FORENSIC ANALYSIS OF BIOS CHIPS

Pavel Gershteyn, Mark Davis and Sujeet Shenoi

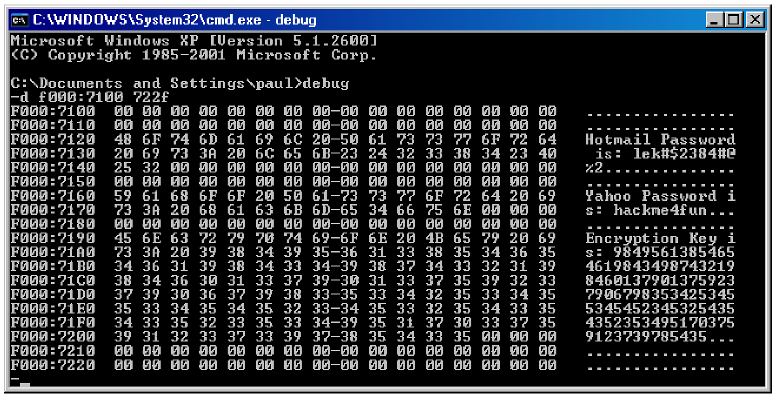
Abstract Data can be hidden in BIOS chips without hindering computer performance. This feature has been exploited by virus writers and computer game enthusiasts. Unused BIOS storage can also be used by criminals, terrorists and intelligence agents to conceal secrets. However, BIOS chips are largely ignored in digital forensic investigations. Few techniques exist for imaging BIOS chips and no tools are available specifically for analyzing BIOS data.

This paper focuses on the Award BIOS chip, which is commonly used in IBM compatible machines. It demonstrates how data may be concealed within BIOS free space and modules in a manner that makes it accessible using operating system commands. Furthermore, forensically sound techniques are described for detecting and recovering concealed data from BIOS chips.

Keywords: BIOS chips, Award BIOS, data concealment, evidence recovery

1. Introduction

The Basic Input/Output System (BIOS) is the lowest level of software in any embedded device or computer [4, 13, 16, 20]. A BIOS typically resides on the motherboard within a read/write flash memory chip [8] of capacity 128K to 512K. It interfaces the hardware with the operating system, which is critical during the booting process. Also, it provides diagnostics and utilities for the computer system. A BIOS is motherboard-specific, allowing the operating system to load from and use specific hardware configurations. It maintains several system settings, e.g., drive boot order and boot-up password protection. The BIOS settings are stored separately in CMOS memory (not flash memory), which requires a small battery to maintain its integrity [20]. After the operating system loads, the BIOS passes control to the operating system.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\paul>debug
-d f000:7100 722f
F000:7100  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
F000:7110  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
F000:7120  48 6F 74 6D 61 69 6C 20-50 61 73 73 77 6F 72 64   Hotmail Password
F000:7130  20 69 73 3A 20 6C 65 6B-23 24 32 33 38 34 23 40    is: lek#$2384#@
F000:7140  25 32 00 00 00 00 00 00-00 00 00 00 00 00 00 00   %2..............
F000:7150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
F000:7160  59 61 68 6F 6F 20 50 61-73 73 77 6F 72 64 20 69   Yahoo Password i
F000:7170  73 3A 20 68 61 63 6B 6D-65 34 66 75 6E 00 00 00   s: hackme4fun...
F000:7180  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
F000:7190  45 6E 63 72 79 70 74 69-6F 6E 20 4B 65 79 20 69   Encryption Key i
F000:71A0  73 3A 20 39 38 34 39 35-36 31 33 38 35 34 36 35   s: 9849561385465
F000:71B0  34 36 31 39 38 34 33 34-39 38 37 34 33 32 31 39   4619843498743219
F000:71C0  38 34 36 30 31 33 37 39-30 31 33 37 35 39 32 33   8460137901375923
F000:71D0  37 39 30 36 37 39 38 33-35 33 34 32 35 33 34 35   7906798353425345
F000:71E0  35 33 34 35 34 35 32 33-34 35 33 32 35 34 33 35   5345452345325435
F000:71F0  34 33 35 32 33 35 33 34-39 35 31 37 30 33 37 35   4352353495170375
F000:7200  39 31 32 33 37 33 39 37-38 35 34 33 35 00 00 00   9123739785435...
F000:7210  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
F000:7220  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

Figure 1. Hidden data in a BIOS viewed using Windows XP.

BIOS chips may contain 25K to 100K or more of unused space that can be used to store data without hindering computer performance. Unused BIOS storage space was exploited by the 1998 Win95/CIH virus that wiped out hard drives. Computer game enthusiasts often overwrite BIOS data to create personalized graphics. The same BIOS storage techniques can be used by criminals, terrorists and intelligence agents to conceal secrets, e.g., address books, financial data, incriminating photographs and cryptographic keys.

Data concealed in a BIOS chip can be accessed relatively easily. Figure 1 shows hidden data stored in the `D3VA1323.BIN` module of an Award BIOS chip viewed using the Windows XP command prompt. This is possible because most of the data in `D3VA1323.BIN` is copied to RAM during the boot process. Thus, the hidden data is discernible in a memory dump produced by the `debug` tool.

Although BIOS chips can conceal significant amounts of data, they are largely ignored in digital forensic investigations. Indeed, very few techniques exist for imaging BIOS chips [10] and no tools are available specifically for analyzing BIOS storage.

This paper focuses on the Award BIOS chip, which is commonly used in IBM compatible machines. It demonstrates how data may be concealed within BIOS free space and modules in a manner that makes it accessible using the operating system. Also, the paper suggests modifications to standard digital forensic procedures to include BIOS (and other firmware) chips.

The following section provides an overview of the Award BIOS chip, including the boot process and storage organization. Next, procedures are described for concealing data in various locations within an Award

BIOS chip. Finally, forensically sound techniques are specified for detecting and recovering hidden data from Award BIOS chips.

2. Award BIOS Overview

This paper focuses on the Award Version 6 BIOS chip (EPoX EP-D3VA with BIOS Version EPoX EP-D3VA ID# 03/23/2001-694X-596B-977-6A6LJPABC), which is representative of the family of IBM PC compatible BIOS chips. The BIOS chip contains modular software that facilitates communication between a specific motherboard and the operating system. The BIOS software runs from the BIOS chip at power-up and performs all the tasks necessary for the operating system to load successfully. The software also provides diagnostic and configuration tools for the user and low-level hardware routines for the operating system [9]. BIOS software is stored in flash memory, which allows the software to be upgraded. BIOS configuration data is stored on a separate CMOS chip. Award BIOS software consists of compressed modules along with executable code, which decompresses the modules and also provides for error recovery.

BIOS executable code usually becomes inactive after the operating system's hardware drivers are initialized. However, the BIOS may retain limited control over low-level functions such as power management.

The following subsections describe the boot process and storage organization of the Award BIOS chip.

2.1 BIOS Boot Process

A BIOS chip is critical to booting a computer. When a computer is turned on, the processor reads instructions from memory location `0xFFFF0` [13], which contains a jump call to the start of the BIOS program on the BIOS chip. When the BIOS is invoked, it executes a Power-On Self-Test (POST) that systematically checks that the necessary hardware is present and is in working order. At this point the video card is not yet initialized, so errors are communicated to the user through a series of beeps known as "beep codes" [1]. The BIOS also copies itself into system RAM for faster access, decompressing its modules in the process [7].

After the POST is completed, the system BIOS program finds and executes the video-card's own built-in BIOS program that initializes the video card. The system BIOS then locates and executes the BIOS programs of other devices.

Following these invocations, the system BIOS performs additional hardware tests and conducts a system inventory; this establishes hard-

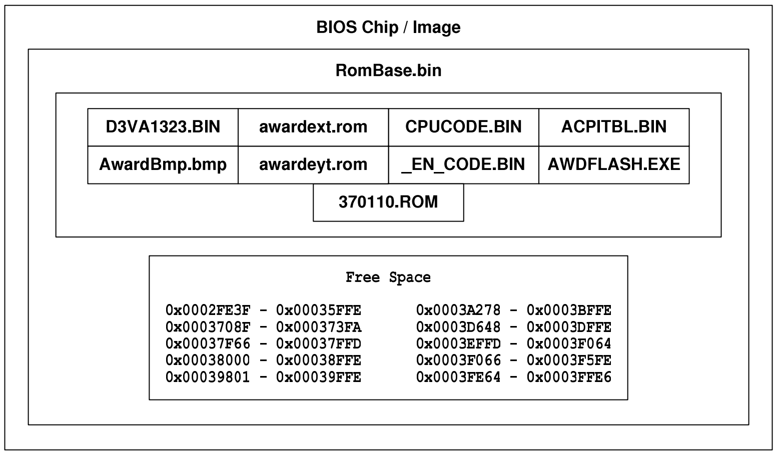


Figure 2. Award BIOS storage organization.

ware parameters. Next, the BIOS detects the memory size and identifies the drives. A summary of the system configuration is then displayed to the user.

The BIOS subsequently identifies its target boot drive, which is determined by the BIOS settings. Next, it searches for a master boot record and, upon finding it, starts loading the operating system. After the operating system is loaded, control passes from the BIOS to the operating system.

2.2 BIOS Storage Organization

The storage organization of an Award BIOS chip is shown in Figure 2. The BIOS has nine modules (`D3VA1323.BIN`, `awardext.rom`, `CPUCODE.BIN`, `ACPITBL.BIN`, `AwardBmp.bmp`, `awardeyt.rom`, `_EN_CODE.BIN`, `AWDFLASH.EXE` and `370110.ROM`), which are stored at the start of the flash memory. The modules are followed by executable BIOS code and data interspersed with free space. Also, the BIOS has ten sections of consecutive free space (Figure 2). Some of these sections comprise several consecutive "blocks" of free space; each block only contains hex strings of `00`s and `FF`s.

The integrity of all the modules as well as the first and second blocks of free space is protected by an 8-bit checksum. The integrity of the third block of free space is protected by a second checksum. However, all the free space following the third block is unprotected.

The checksum data are stored after the third block of free space at memory addresses `0x37FFE - 0x37FFF`. All the data stored after address

`0x37FFF` is not protected in any way. A checksum mismatch is treated as a fatal error by the BIOS, which halts the booting process.

All the BIOS modules and most of the executable code and data are stored in a compressed format using the LHA/LH5 algorithm [5]. The BIOS chip incorporates code that decompresses the data during the booting process.

Data may be concealed within BIOS free space and modules. Depending on the location and amount of data concealed, the BIOS could remain functional or it could become corrupted, which prevents the computer from booting. BIOS chips are designed to be expandable. Consequently, they have large amounts of free space that can be overwritten with data without affecting BIOS operation.

BIOS modules contain text strings that are displayed as messages, e.g., error messages and hardware data. These text strings can be overwritten with data without affecting the BIOS.

Of course, the entire BIOS memory can be used to conceal data. This makes the recovery of the data problematic, as the computer cannot be booted with a corrupt BIOS. However, special devices and techniques exist for booting such a computer and recovering hidden data [10].

3. Corrupted BIOS Recovery Technique

Editing or "flashing" BIOS modules, free space or any other code/data can corrupt a BIOS. Therefore, a BIOS data hiding strategy must incorporate a technique for recovering from BIOS flashing errors.

Upon detecting an error, a BIOS chip attempts to re-flash itself using a backup BIOS file from the floppy drive. However, automatic re-flashing is unreliable because errors sometimes go undetected. Also, the data hiding process can corrupt the flashing code (`AWDFLASH.EXE`) itself.

A BIOS Savior device can be used to recover from a data hiding attempt that results in a corrupted BIOS. This device provides a backup BIOS chip and a hardware switch that enables the user to select whether the computer will use the BIOS Savior chip or the original BIOS chip for the booting process. The BIOS Savior plugs into the BIOS chip's socket, and the original BIOS chip plugs into the BIOS Savior. Therefore, if the BIOS on the original chip is corrupted, the user can boot the computer using the BIOS Savior, switch back to the original chip after the computer is operational, and then flash the original chip.

Before a BIOS is edited, a backup copy of the BIOS must be preserved within the BIOS Savior. This backup copy enables the computer to boot successfully regardless of the changes made to its BIOS. Note that the booting process involves three main steps: POST, hardware initialization and master boot record access.

After data is written to the BIOS during the process of data hiding (see Section 4), two possibilities exist. The first is that the overwritten BIOS will boot the computer successfully. If the BIOS checksum computed during the POST is incorrect, the BIOS will attempt to re-flash itself. If the POST is unsuccessful and the BIOS checksum is correct, a fatal error occurs and the system halts. Regardless of the situation, the backup BIOS maintained in the BIOS Savior can be used to boot the computer. This is accomplished simply by flipping the switch on the BIOS Savior.

4. Hiding Data in BIOS Chips

This section describes procedures for hiding data in: (i) BIOS free space, (ii) BIOS modules, and (iii) free space within BIOS modules. In all three cases, substantial amounts of data can be concealed within the BIOS without hindering computer performance.

Hiding data on a BIOS chip requires a separate workstation to edit and store BIOS image files and to prepare the boot disk. Caldera Dr-DOS [2, 3] is used to create the boot disk and to facilitate BIOS flashing because it does not contain any TSR (terminate and stay resident) programs. A BIOS Savior device [12] is used to recover from BIOS flashing errors (Section 3). A hex editor is used to modify BIOS images and modules. AwardMod software [11] is used to load and store binary BIOS files and directories containing extracted BIOS modules. Also, Uniflash [17], a universal BIOS flashing utility, is used for BIOS read/writes instead of the standard Award BIOS program (`AWDFLASH.EXE`), which leaves portions of the chip unflashed.

4.1 Hiding Data in BIOS Free Space

The Award BIOS has 38,020 bytes of free space (located after the block of compressed modules) that can be used for storing data. In reality, 44,163 bytes of free space exist, but 6,143 bytes cannot be used because data stored in certain locations is not retained.

The Award BIOS has 12 blocks of free space; the specific locations of the blocks are shown in Table 1. Note that the first and last null bytes (`00` or `FF`) of each free space block are assumed to belong to the code preceding/following the free space, and are therefore not counted.

Free space blocks are null blocks containing long strings of `00`s or `FF`s (Table 1). These free space blocks can be overwritten without corrupting the BIOS. However, it is important to ensure that data written to

Table 1. Award BIOS free space blocks.

Block	Pattern	Range	Comments
Block 1	`FF`	`0x2FE3F–0x35FFE`	Protected by Checksum 1
Block 2	`00`	`0x3708F–0x373FA`	Protected by Checksum 2
Block 3	`FF`	`0x37F66–0x37FFD`	Protected by Checksum 3
—	—	`0x37FFE–0x37FFF`	
Block 4	`FF`	`0x38000–0x38FFE`	
Block 5	`FF`	`0x39801–0x39FFE`	
Block 6	`FF`	`0x3A278–0x3A744`	
Block 7	`00`	`0x3A745–0x3AFFF`	`0x3A800–0x3AFFF` not recoverable
Block 8	`FF`	`0x3B000–0x3BFFE`	Entire block not recoverable
Block 9	`00`	`0x3D648–0x3DFFE`	
Block 10	`00`	`0x3EFFD–0x3F064`	
Block 11	`00`	`0x3F066–0x3F5FE`	
Block 12	`00`	`0x3FE64–0x3FFE6`	

Blocks 1, 2 and 3 does not alter the checksums [19]. This is accomplished by reserving one byte each in Block 2 and Block 3 to balance the checksums. First, the 8-bit checksum of `BIOSback.bin` is computed before any changes are made (the checksum value for the Award BIOS is `EA`). Next, one byte in Block 2 is reserved by changing it to `00`. Then, data is written to Blocks 1 and 2, and checksum is re-calculated. Finally, the reserved byte in Block 2 is changed to a value that makes the checksum equal to `EA`. This "balancing value" is computed as $[(\textit{Original Value}) - (\textit{Current Value}) + \texttt{0x100}] \bmod \texttt{0x100}$. Block 3 is overwritten with data and the corresponding Checksum 2 is balanced in a similar manner.

The following procedure specifies the steps involved in hiding data in BIOS free space.

BIOS Free Space Overwriting Procedure

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

2. Copy the Uniflash program (`UNIFLASH.EXE`) and all its required components to the boot disk.

3. Boot the Award BIOS machine using the boot disk. After Caldera Dr-DOS has booted, invoke `UNIFLASH.EXE`. Backup the original BIOS to the boot disk as `BIOSback.bin`. Copy `BIOSback.bin` to the workstation hard drive.

4. Use the hex editor to write data to free space in `BIOSback.bin`, making sure that Checksums 1 and 2 are preserved using balancing values. Save the changes in a new file called `BIOSedited.bin`.

5. Complete the process of hiding data by copying `BIOSedited.bin` to the boot disk. Boot the Award BIOS machine using the boot disk, and flash the BIOS chip by invoking `UNIFLASH.EXE`.

6. Restart the computer to verify that it functions properly.

4.2 Hiding Data in BIOS Modules

The following procedure lists the steps involved in hiding data in BIOS modules.

BIOS Module Overwriting Procedure

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

2. Copy the Uniflash program (`UNIFLASH.EXE`) and all its required components to the boot disk.

3. Boot the Award BIOS machine using the boot disk. After Caldera Dr-DOS has booted, invoke `UNIFLASH.EXE`. Backup the original BIOS to the boot disk as `BIOSback.bin`. Copy `BIOSback.bin` to the workstation hard drive.

4. Use AwardMod to extract modules in `BIOSback.bin` and store them in a directory named `BIOSBackup`.

5. Use the hex editor to overwrite module data that is not critical to the operation of the BIOS (e.g., text strings). This data can be overwritten with text or binary data.

6. After one or more modules are overwritten, use AwardMod to load all the files in the `BIOSBackup` directory and store them in a new BIOS image called `BIOSedited.bin`.

7. Preserve Checksum 1 in `BIOSedited.bin` using a balancing value in Block 1 or 2.

8. Complete the process of hiding data by copying `BIOSedited.bin` to the boot disk. Boot the Award BIOS machine using the boot disk, and flash the BIOS chip by invoking `UNIFLASH.EXE`.

9. Restart the computer to verify that it functions properly.

4.3 Hiding Data in BIOS Module Free Space

This section describes how data may be stored in the BIOS module `D3VA1323.BIN` in a manner that makes it accessible from Windows using the `debug` command.

Module `D3VA1323.BIN`, which contains hardware-specific settings and routines, has 5,057 bytes of free space that can be accessed from Windows using the `debug` command. The `debug` memory dump of locations

0xF0000 - 0xFFFFF contains some data from D3VA1323.BIN. Note that debug uses the *segment:offset* memory notation [18]; the corresponding absolute memory address notation is 16**segment* + *offset*. For example, the *segment:offset* address F000:027E in RAM is equivalent to the absolute address 0xF027E in RAM. The following are the mappings of memory addresses in debug notation to absolute addresses in the D3VA1323.BIN module (when viewed as a file):

```
F000:027E - F000:13FF <=> 0x1027E - 0x113FF
F000:1514 - F000:1BFF <=> 0x11514 - 0x11BFF
F000:1C9F - F000:FFFF <=> 0x11C9F - 0x1FFFF
```

The last memory mapping contains two blocks of free space that constitute 5,057 bytes of 00s. Note that the last four digits of each pair of starting and ending address are identical, which simplifies the task of determining the memory addresses to be dumped to obtain the contents of a certain location in D3VA1323.BIN.

BIOS Module Free Space Overwriting Procedure

1. Run a telnet server on the Award BIOS machine. Use a telnet client to connect to the telnet server (this is needed to capture the screen output). Execute the command debug. At the debug prompt, type: `d F000:0000 FFFF` and press enter. Save the output in a new text document: originalBIOS.txt.

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

3. Copy the Uniflash program (UNIFLASH.EXE) and all its required components to the boot disk.

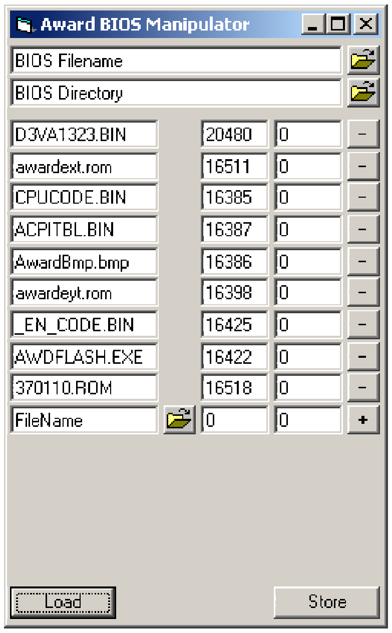
4. Boot the Award BIOS machine using the boot disk. After Caldera Dr-DOS has booted, invoke UNIFLASH.EXE. Backup the original BIOS to the boot disk as BIOSback.bin. Copy BIOSback.bin to the workstation hard drive.

5. Use AwardMod to extract modules in BIOSback.bin (see Figure 3) and store them in a directory named BIOSBackup.

6. Use the hex editor to view the module D3VA1323.BIN. Compare D3VA1323.BIN with originalBIOS.txt. Note that three hex segments of each of the two files are identical because they map to each other (see the discussion immediately preceding this procedure). Therefore, data hidden in these locations in D3VA1323.BIN can be viewed using the debug command.

7. The third segment of D3VA1323.BIN (i.e., 0x11C9F - 0x1FFFF), which can be viewed using the debug command, contains two blocks of 00s (0x16FFB - 0x17FFE, 0x1DC42 - 0x1DFFE) that can be overwritten without corrupting the BIOS. Overwrite these blocks with data that is to be hidden.

8. After the module is edited, use AwardMod to load all the files in the BIOSBackup directory and store them in a new BIOS image called BIOSedited.bin.



BIOS Filename

BIOS Directory

D3VA1323.BIN	20480	0	-
awardext.rom	16511	0	-
CPUCODE.BIN	16385	0	-
ACPITBL.BIN	16387	0	-
AwardBmp.bmp	16386	0	-
awardeyt.rom	16398	0	-
_EN_CODE.BIN	16425	0	-
AWDFLASH.EXE	16422	0	-
370110.ROM	16518	0	-
FileName	0	0	+

Figure 3. AwardMod screen during extraction of BIOS modules.

9. Preserve Checksum 1 in `BIOSedited.bin` using a balancing value in Block 1 or 2.

10. Complete the process of hiding data by copying `BIOSedited.bin` to the boot disk. Boot the Award BIOS machine using the boot disk, and flash the BIOS chip by invoking `UNIFLASH.EXE`.

11. Restart the computer to verify that it functions properly. Verify that the data hidden in `D3VA1323.BIN` can be viewed using the `debug` command.

5. Forensic Examination of BIOS Chips

A BIOS chip is a convenient location for hiding secrets because significant amounts of data are easily stored and retrieved. Law enforcement agents generally overlook BIOS chips during investigations. Moreover, at this time, no established forensic procedures exist for imaging and analyzing BIOS chips.

This section describes how common forensic tools can be used to examine BIOS chips. In particular, searches based on regular expressions and file headers (e.g., file carving) can be used to identify and extract data concealed in a BIOS.

<u>BIOS Data Recovery Procedure</u>

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

2. Copy the Uniflash program (`UNIFLASH.EXE`) and all its required components to the boot disk.

3. Boot the Award BIOS machine using the boot disk. After Caldera Dr-DOS has booted, invoke `UNIFLASH.EXE`. Backup the seized BIOS to the boot disk as `BIOSevidence.bin`; this creates a forensic image of the seized BIOS. Copy `BIOSevidence.bin` to the workstation hard drive.

4. Use AwardMod to extract modules in `BIOSevidence.bin` and store them in a directory named `BIOSEvidence`.

5. Use forensic tools (e.g., Foremost, EnCase, Forensic Tool Kit, ILook) to examine `BIOSevidence.bin` and the extracted modules, especially `D3VA1323.BIN`, for text, file headers and regular expressions, and preserve all data of interest. Also, examine `D3VA1323.BIN`, which is 128K in size, manually using the hex editor to detect all hidden text.

6. If hidden data cannot be found using the forensic tools, use the hex editor to compare modules from the seized BIOS with those from a clean copy of the BIOS image (e.g., one obtained from the motherboard manufacturer). This assists in locating hidden data.

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

6. Modifications to Forensic Procedures

Traditional digital forensic investigations involve three main steps: initial response, media duplication (imaging) and imaged media analysis. Investigations are jeopardized when important evidence is stored in media that are not seized by investigators or when the media are seized but, for a variety of reasons, evidence is not recovered from the media.

The initial response step is typically executed on a live computer system that contains volatile information. This volatile information, e.g., current users, open sockets and running processes, is captured and saved for further investigation. Code and data – including concealed information – stored on a BIOS chip are not lost when a computer system or embedded device is powered down. Therefore, no action specific to the BIOS chip is necessary during the initial response step. Of course, initial responders must be aware that importance evidence may be hidden in the chip.

It is important that forensic examiners image a BIOS chip just as they image other media (e.g., hard drives and flash memory) during the media duplication step. The procedure for imaging a BIOS chip has been described in Section 5.

Some authors (e.g., [15]) recommend that digital forensic examiners view drive geometry data in the system BIOS configuration – before the media duplication step – to obtain drive parameters that might aid in media duplication. An analysis of the system BIOS configuration may reveal that data is hidden in the BIOS. However, the examiner must be alert to the fact that the BIOS may contain hidden data.

It is possible that the BIOS in a seized computer may be intentionally corrupted, e.g., when the BIOS contains secret information or when the owner has overwritten the BIOS to hinder the forensic investigation. Such a computer will not boot. Therefore, the examiner may use a chip programmer [16] to image the BIOS or the BIOS Savior device [12] to boot the computer and image the BIOS. The latter technique has been described in Section 3. Note that some BIOS chips are soldered directly to their motherboards, which renders the BIOS Savior technique useless and the chip programming technique risky at best.

During the imaged media analysis step, a forensic examiner would analyze a BIOS image using standard forensic tools as described in Section 5. Once again, the examiner should be aware of where data might be concealed and should conduct a thorough search of the BIOS image. The locations where data might be hidden in a BIOS chip have been described in Section 4.

7. Conclusions

Modern hardware components, such as dual-BIOS motherboards and replaceable BIOS chips, simplify the task of concealing secret information in BIOS chips. However, digital forensic practice has not kept up with advances in BIOS technology. As a result, few, if any, recognized techniques exist for detecting and extracting hidden data from BIOS chips. This paper has shown that even BIOS chips with checksum-based integrity protection can be used to conceal data. The other main contributions of this paper include a technique for detecting and extracting hidden data, and suggestions for modifying forensic examination procedures to accommodate BIOS chips.

"BIOS forensics" is an interesting area of digital forensic research. A library of known good hashes of BIOS chips would make it trivial to verify whether or not BIOS chips have been tampered. Note, however, that in modern computers, the extended system configuration data (ESCD) is typically stored on the BIOS chip, so the hash value computations would have to omit certain areas of the BIOS. Boot disks and CDs that automate the process of imaging BIOS chips would greatly benefit forensic investigators. Likewise, forensic tools for heuristically analyzing BIOS images and detecting hidden data would be very valuable.

References

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

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

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

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

[5] M. Darmawan, Award BIOS reverse engineering (www.codebreakers-journal.com/viewarticle.php?id=38), 2004.

[6] M. Darmawan, Award BIOS code injection (www.codebreakers-journal.com/viewarticle.php?id=58), 2005.

[7] D. Dunn, BIOS basics (freepctech.com/articles/articles.php?ArticleId=122), 2002.

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

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

[10] P. Gershteyn, M. Davis, G. Manes and S. Shenoi, Extracting concealed data from BIOS chips, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, New York, pp. 217-230, 2005.

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

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

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

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

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

[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] D. Sedory, Removing the mystery from segment:offset addressing (thestarman.dan123.com/asm/debug/Segments.html), 2004.

[19] R. Sevko, Editing the BIOS (www.winsov.ru/sios002.php), 2003.

[20] J. Tyson, How BIOS works (computer.howstuffworks.com/bios.htm).

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