National Institute of
Standards and Technology
U.S. Department of Commerce

Special Publication 800-147
(Draft)

BIOS Protection Guidelines (Draft)

Recommendations of the National Institute of Standards and Technology

David Cooper
William Polk
Andrew Regenscheid
Murugiah Souppaya

NIST Special Publication 800-147 (Draft)

BIOS Protection Guidelines

Recommendations of the National Institute of Standards and Technology

David Cooper
William Polk
Andrew Regenscheid
Murugiah Souppaya

C O M P U T E R   S E C U R I T Y

Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8930

February 2011

U.S. Department of Commerce

Gary Locke, Secretary

National Institute of Standards and Technology

Dr. Patrick D. Gallagher, Director

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analysis to advance the development and productive use of information technology. ITL's responsibilities include the development of technical, physical, administrative, and management standards and guidelines for the cost-effective security and privacy of sensitive unclassified information in Federal computer systems. This Special Publication 800-series reports on ITL's research, guidance, and outreach efforts in computer security and its collaborative activities with industry, government, and academic organizations.

National Institute of Standards and Technology Special Publication 800-147 (Draft)
Natl. Inst. Stand. Technol. Spec. Publ. 800-147, 25 pages (Feb. 2011)

Acknowledgments

The authors, David Cooper, William Polk, Andrew Regenscheid, and Murugiah Souppaya of the National Institute of Standards and Technology (NIST) wish to thank their colleagues who reviewed drafts of this document and contributed to its technical content.

Table of Contents

List of Appendices

Executive Summary

Modern computers rely on fundamental system firmware, commonly known as the system Basic Input/Output System (BIOS), to facilitate the hardware initialization process and transition control to the operating system. The system BIOS is typically developed by both original equipment manufacturers (OEMs) and independent BIOS vendors, and is distributed to end-users by motherboard or computer manufacturers. Manufacturers frequently update system firmware to fix bugs, patch vulnerabilities, and support new hardware. Malicious code in the system BIOS is a significant security threat because the BIOS executes very early in the boot process and initializes many key hardware and software components. While there are a variety of threats to the integrity of the system BIOS, this guide is focused on preventing update of the system BIOS by malicious software. The security controls and procedures specified in this document are oriented to desktops and laptops deployed in enterprise environments.

This document includes background material detailing the role of the system BIOS in current and emerging systems. While the recommendations provided in this document are independent of any particular system design, the background information focuses on the firmware on current and emerging x86-compatible desktop and laptop systems. Five processes for legitimate updates to the system BIOS are described:

- User-initiated updates;
- Management updates;
- Rollback;
- Manual recovery; and
- Automatic recovery.

The remaining background material provides further details regarding the importance of BIOS integrity, and an overview of threats to the system BIOS.

The final section of the document is focused on security requirements and management best practices mitigating a specific threat: preventing unauthorized modification of the system BIOS by potentially malicious software running on computer systems. The security requirements do not attempt to address installation of an unapproved system BIOS in the supply chain, manually by the user, or rollback to an authenticated but vulnerable system BIOS. However, the management best practices can be used to identify such systems after modification has occurred in most cases.

Security requirements are specified for four system BIOS features:

- The authenticated BIOS update mechanism involves using digital signatures to ensure the authenticity of the BIOS update image.
- A secure local update mechanism ensures the authenticity and integrity of the BIOS update image by requiring the administrator who is updating the BIOS to be physically present and by ensuring that no intermediary components can modify the BIOS update image that is presented by the administrator.
- Firmware Write Protection prevents unintended or malicious modification of the BIOS outside the authenticated BIOS update process with a write protection mechanism that cannot be overridden outside of an authenticated BIOS update.
- Non-Bypassability ensures that there are no mechanisms that allow the system processor or any other system component to bypass the authenticated update mechanism.

The management best practices are tightly coupled with the security requirements identified above. Five distinct phases are addressed:

- In the provisioning phase, an enterprise establishes configuration baselines identifying the approved BIOS version and configuration settings.

- In the Platform Deployment Phase, the secure local update process establishes or verifies the configuration baseline.
- The Operations and Maintenance Phase, systems are monitored for unexpected changes and planned BIOS updates are executed using the authenticated BIOS update mechanism.
- The recovery phase supports authorized rollback to an earlier BIOS version and recovery from a corrupted BIOS.
- The disposition phase restores the BIOS and configuration data to the original settings to prevent against accidental information leakage.

This section has two distinct subsections: the first specifies functional requirements for BIOS update mechanisms; the second specifies best practices for BIOS management. The functional requirements are intended primarily for BIOS developers. The best practices assume that the security controls are available,and is intended primarily for information system security professionals who are responsible for managing the endpoint and server platforms, secure boot process, and hardware security module.

Future revisions to this publication will look at the security of the system BIOS and boot process with a larger perspective.

1. Introduction

1.1 Authority

The National Institute of Standards and Technology (NIST) developed this document in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets; but such standards and guidelines shall not apply to national security systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), “Securing Agency Information Systems,” as analyzed in A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided in A-130, Appendix III.

This guideline has been prepared for use by Federal agencies. It may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright, though attribution is desired.

Nothing in this document should be taken to contradict standards and guidelines made mandatory and binding on Federal agencies by the Secretary of Commerce under statutory authority, nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other Federal official.

1.2 Purpose and Scope

This guide is intended to identify, prioritize, and mitigate threats to the integrity of fundamental system firmware, commonly known as the *Basic Input/Output System (BIOS).* The term BIOS is used in this publication to refer collectively to the conventional BIOS, the *Extensible Firmware Interface (EFI)*, and the *Unified Extensible Firmware Interface (UEFI).* This guide identifies the security controls and procedures required to mitigate a subset of these threats. Specifically, this guide includes requirements and guidelines for a secure BIOS update process targeting platform vendors, with additional recommendations for managing the BIOS in an operational environment. Future revisions to this publication will look at the security of the BIOS and boot process with a larger perspective.

While this document provides background information specific to the firmware on current and emerging x86 and x64 desktop and laptop systems to enhance both the readability and applicability of the document, the controls and procedures specified in this document are also independent of any particular system design. The security controls and procedures specified in this document are oriented to desktops and laptops deployed in an enterprise environment, with the hope that the necessary technologies will migrate to consumer-level products over time. Future efforts may look at boot firmware security for enterprise server platforms.

1.3 Audience

The intended audience for this document includes BIOS and platform vendors, and information system security professionals who are responsible for managing the endpoint platforms’ security, secure boot processes, and hardware security modules. The material may also be of use when developing enterprise-wide procurement strategies and deployment.

The material in this document is technically oriented, and it is assumed that readers have at least a basic understanding of system and network security. The document provides background information to help such readers understand the topics that are discussed. Readers are encouraged to take advantage of other resources (including those listed in this document) for more current and detailed information.

1.4 Document Structure

The remainder of this document is organized into the following major sections:

- Section 2 presents an overview of the BIOS, its role in the boot process, and identifies potential attacks against the BIOS in an operational environment.
- Section 3 examines how selected threats to the BIOS can be mitigated. Section 3.1 describes security controls for BIOS implementations that are required or recommended to mitigate these threats. Section 3.2 defines processes that leverage these controls to implement a secure BIOS update process within an enterprise as part of the platform management life cycle.

The document also contains appendices with supporting material:

- Appendix A defines terms used in this document.
- Appendix B contains a list of acronyms and abbreviations used in this document.
- Appendix C contains a list of references used in the development of this document.

2. Background

Modern computers such as desktop and laptop computers contain program code that facilitates the hardware initialization process. The code is stored in non-volatile memory and is commonly referred to as boot firmware. Firmware used to initialize the system is called the *Basic Input/Output System (BIOS)* or the *system BIOS*. This section provides background information on the system BIOS and its role in the boot process using the conventional BIOS and Unified Extensible Firmware Interface (UEFI) BIOS as examples. It identifies the primary methods used for updating the system BIOS, and security issues and threats to the system BIOS.

2.1 System BIOS

The system BIOS is the first piece of software executed on the main central processing unit (CPU) when a computer is powered on. On most modern computers, this firmware is known as the system BIOS. While the system BIOS was historically responsible for providing operating systems access to hardware, the primary role of the system BIOS, and other boot firmware, on modern machines is to initialize and test hardware components and load the operating system. In addition, the BIOS loads *System Management Interrupt (SMI)* handlers and initializes *Advanced Configuration and Power Interface (ACPI)* tables and code. These provide important system management functions while the computer system is in use, such as power and thermal management. The system BIOS may also load CPU microcode patches during the boot process.

There are several different types of BIOS firmware. While some computers still use a 16-bit conventional BIOS newer systems use boot firmware based on the UEFI specifications [UEFI]. In this document we will refer to all types of boot firmware as BIOS firmware, the system BIOS, or simply BIOS. When necessary, we will differentiate conventional BIOS firmware from UEFI firmware by calling them the conventional BIOS and UEFI BIOS, respectively.

System BIOS is typically developed by both original equipment manufacturers (OEMs) and independent BIOS vendors, and is distributed to end users by motherboard or computer manufacturers. Manufacturers frequently update system firmware to fix bugs, patch vulnerabilities, and support new hardware. The system BIOS is typically stored on electrically erasable programmable read-only memory (EEPROM) or other forms of flash memory, and is modifiable by end users. Typically, system BIOS firmware is updated using a utility or tool that has special knowledge of the non-volatile storage components in which the BIOS is stored.

A given computer system will have BIOS in several different locations. In addition to the BIOS on the motherboard, hard drive controllers, video cards, network cards and other add-on cards have their own BIOS firmware. This firmware is generally known as *Option ROMs* or *UEFI drivers*. These are loaded and executed by the system firmware during the boot process. Other system devices, such as hard drives and optical drives, may have their own microcontrollers and a different type of firmware that is used to allow the rest of the computer system to communicate with the device.

This document applies to system BIOS firmware (e.g., conventional BIOS or UEFI BIOS) stored in the system flash memory of computer systems. In many cases, the system BIOS firmware may include additional modules, in the form of Option ROMs or UEFI drivers, which extend the features of the system firmware. This document also applies to these types of Option ROMs or UEFI drivers when they are stored with the system BIOS firmware and updated by the same mechanism. However, it does not apply to other types of Option ROMs, UEFI drivers, or firmware stored elsewhere in a computer system.

2.2 Role of System BIOS in the Boot Process

The primary function of the system BIOS is to initialize important hardware components and load the operating system. This process is known as *booting*. The boot process for conventional x86-compatible systems is, at a high level, similar across different types of firmware. The boot process of the system BIOS typically executes in the following stages:

1. **Core Root of Trust:** The system BIOS may include a small core block of firmware that executes first and is capable of verifying the integrity of other firmware components. This has traditionally been called the *BIOS Boot Block*. For trusted computing applications, it may also contain the *Core Root of Trust for Measurement* (CRTM).
2. **Initialize and Test Low-Level Hardware:** Very early in the boot process the system BIOS initializes and tests key pieces of hardware on the computer system, including the motherboard, chipset, memory and CPU.
3. **Load and Execute Additional Firmware Modules:** The system BIOS will execute additional pieces of firmware that either extend the capabilities of the system BIOS or initialize other hardware components required to boot the system. These additional modules may be stored within the same flash memory as the system BIOS or they may be stored in the hardware devices they initialize (e.g., video card, local area network card).
4. **Boot Device Selection:** After system hardware has been configured, the system BIOS will search for a boot device (e.g., hard drive, optical drive, USB drive) and execute the boot loader stored on that device.
5. **Load Operating System**: While the system BIOS is still in control of the computer, the boot loader will begin to load and initialize the operating system kernel. Once the kernel is functional, primary control of the computer system will transfer from the system BIOS to the operating system.

In addition, the system BIOS loads SMI handlers (also known as System Management Mode (SMM) code) and initializes ACPI tables and code. These provide important system management functions while the computer system is in use, such as power and thermal management. SMM and ACPI code can run after the operating system takes primary control of the computer system.

This section will describe the boot process in conventional BIOS-based systems and the boot process in UEFI-based systems. While the conventional BIOS is still used in many desktop and laptop computers deployed today, the industry has already begun transitioning to the UEFI BIOS. This document focuses on these two types of BIOS firmware, but the guidelines may be applicable to other types of system firmware.

2.2.1 Conventional BIOS Boot Process

Figure 1 shows a typical boot process for x86 compatible systems running a conventional BIOS. Some conventional BIOS-based firmware have a small block of BIOS firmware, known as the BIOS boot block, that is logically separate from the rest of the BIOS. On these computer systems, the boot block is the first firmware executed on a computer system during the boot process. The boot block is responsible for checking the integrity of the remaining BIOS code and may provide mechanisms for recovery if the main system BIOS firmware is corrupted. On most trusted computing architectures, the BIOS boot block serves at the computer system's CRTM because this firmware is implicitly trusted to bootstrap the process of building a measurement chain for subsequent attestation of other firmware and software that is executed on the machine [TCG05].

The boot block will execute the part of the conventional BIOS that initializes most hardware components, the *Power-on-Self-Test* (POST) code. At this time, key low-level hardware on the computer system is initialized, including the chipset, the CPU, and memory. The system BIOS initializes the video card early in the boot process, which may load and execute its own BIOS in system memory to initialize graphics processors and memory.

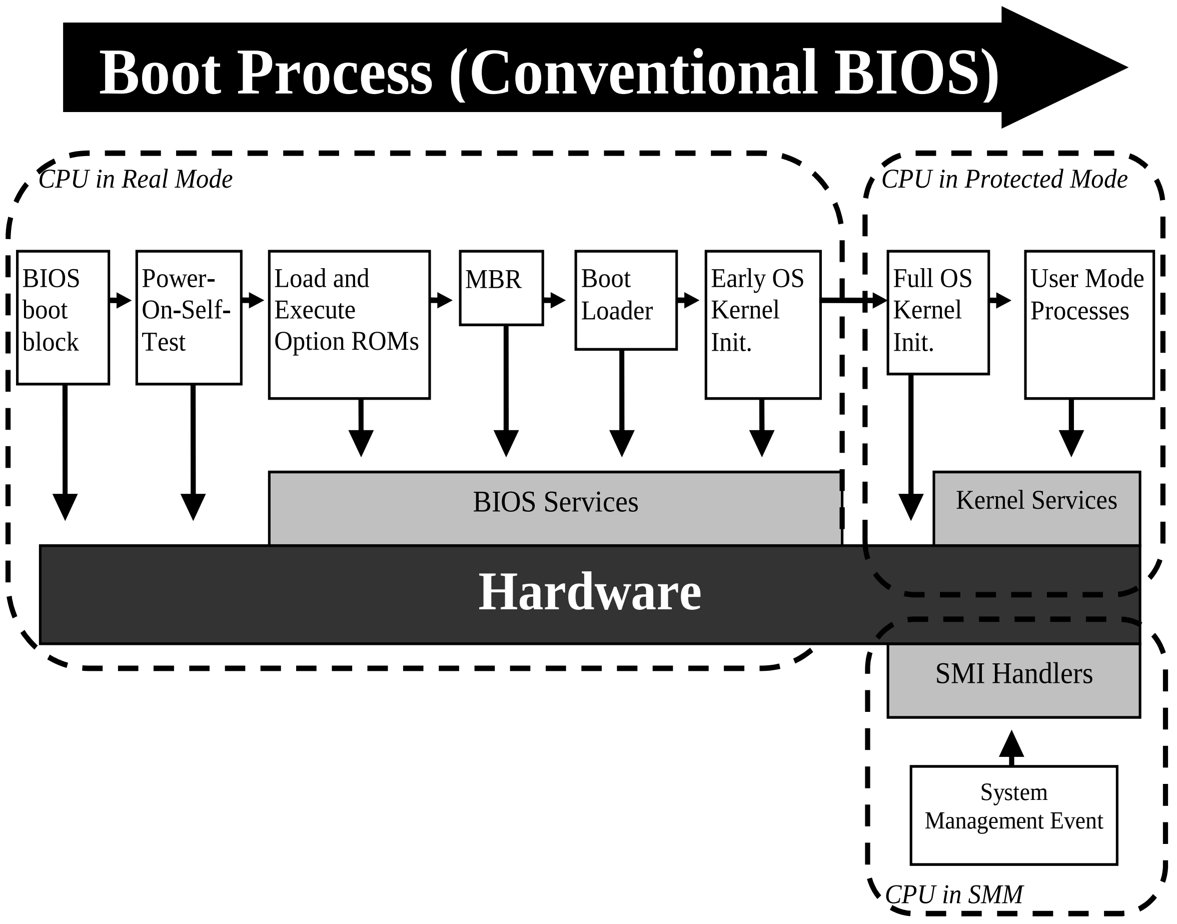

Figure 1: Conventional BIOS Boot Process[1]

Next the system BIOS searches for other peripherals and other microcontrollers that are connected to the computer system and executes any Option ROMs on these components that are necessary to initialize them. Option ROMs are one of the first opportunities to run unauthenticated code on the computer system, and they can add a variety of features to the boot process. For instance, Option ROM on a network adapter could load the Preboot Execution Environment (PXE), which allows a computer to boot over the network.

Then the system BIOS will scan the computer system for storage devices that have been identified as boot devices. In a typical case, the BIOS will attempt to boot from the first boot device with a valid master boot record. The master boot record (MBR) points to a boot loader stored on the hard drive, which in turn starts the process of loading the operating system. While the specific boot process will vary from system to system, depending on the boot loader and operating system, typically the kernel of the operating system will begin to load with the CPU still in 16-bit real mode, transitioning to 32/64-bit protected mode as more of the operating system kernel is brought online.

During the boot process the system BIOS loads SMI handlers and initializes ACPI tables and code. SMI handlers run in a special high-privilege mode on the CPU known as System Management Mode, a 32-bit mode that is capable of bypassing many of the hardware security mechanisms in protected mode, such as memory segmentation.

2.2.2 UEFI Boot Process

At a high level, the UEFI boot process follows a similar flow to the conventional BIOS boot process. One important difference is that UEFI code runs in 32 or 64-bit protected mode on the CPU, not 16-bit real mode as was the case with a conventional BIOS. Most UEFI-based platforms start with a small core block of code that has the primary responsibility of authenticating subsequent code executed on the computer system. This is very similar to the role of the boot block in computer systems with a conventional BIOS. This part of the boot process is known as the Security (SEC) phase, and it serves as the core root of trust in the computer system.

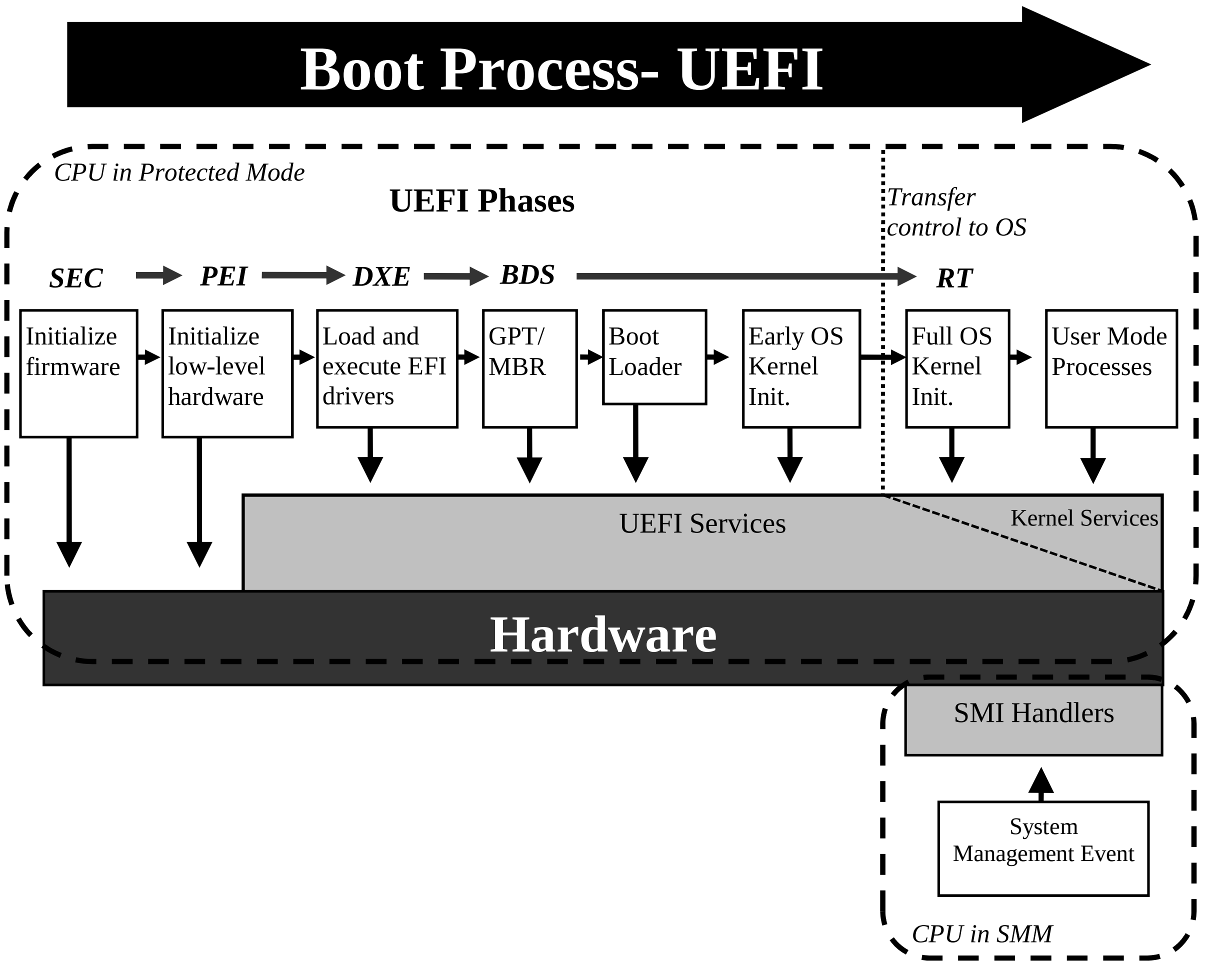

Figure 2: UEFI BIOS Boot Process

The next phase of the UEFI boot process is the Pre-EFI Initialization (PEI) Phase. The PEI phase is intended to initialize key system components, such as the processor, chipset and motherboard. In some cases, the code in the Security Phase and the PEI Phase make up the core root of trust in an UEFI system.

The purpose of the PEI Phase is to prepare the system for the Driver Execution Environment (DXE) phase of the boot process. The DXE phase is where most of the system initialization is performed. The firmware executed in this phase is responsible for searching for and executing drivers to provide device support during the boot process, or provide additional features. During this phase the UEFI BIOS may execute conventional option ROMs, which have a similar purpose.

The PEI and DXE phases of the UEFI boot process lay the foundation to load an operating system. The final tasks necessary to load an operating system are performed in the Boot Device Selection (BDS) phase. This phase initializes console devices for simple input/output operations on the system. These console devices include local text or graphical interfaces, as well as remote interfaces, such as Telnet or remote displays over HTTP. The BDS phase also loads any additional drivers necessary to manage console or boot devices. Finally, the firmware loads the boot loader from the first MBR or GUID Partition Table (GPT) formatted boot device, and loads the operating system.

The Run Time phase of the UEFI boot process begins when the operating system is ready to take control from the UEFI BIOS. UEFI runtime services are available to the operating system during this phase.

2.3 Updating the System BIOS

A system and its supporting management software and firmware may provide several authorized mechanisms for legitimately updating the system BIOS. These include:

1. **User-Initiated Updates:** System and motherboard manufacturers typically supply end users with utilities capable of updating the system BIOS. Historically, end users booted from external media to perform these updates, but today most manufacturers provide utilities that can update the system BIOS from the user’s normal operating system. Depending on the security mechanisms implemented on the system, these utilities might directly update the system BIOS or they may schedule an update for the next system reboot.
2. **Managed Updates:** A given computer system may have hardware and software-based agents that allow a system administrator to remotely update the system BIOS without direct involvement from the user.
3. **Rollback:** System BIOS implementations that authenticate updates before applying them may also check version numbers during the update process. In these cases, the system BIOS may have a special update process for rolling back the installed firmware to an earlier version. For instance, the rollback process may require the physical presence of the user to successfully update the firmware to an earlier version. This mechanism guards against attackers flashing old firmware with known vulnerabilities.
4. **Manual Recovery:** To recover from a corrupt or malfunctioning system BIOS, many computer systems provide mechanisms to allow a user with physical presence during the boot process to replace the current system BIOS with a known good version and configuration.
5. **Automatic Recovery:** Some computer systems are able to detect when the system BIOS has been corrupted and recover from a backup firmware image stored in a separate storage location from the primary system BIOS (e.g., a second flash memory chip, or a hidden partition on a hard drive).

2.4 Importance of BIOS Integrity

As the first code that is executed by the main CPU, the system BIOS is a critical security component of a computer system. While the system BIOS, possibly with the use of a Trusted Platform Module (TPM), can verify the integrity of firmware and software executed later in the boot process, typically all or part of the system BIOS is implicitly trusted.

The system BIOS is an attractive target for attack. Malicious code running at the BIOS level has a great deal of control over a computer system. It could be used to compromise any components that are loaded later in the boot process, including the SMM code, boot loader, hypervisor, or operating system. The BIOS is stored on non-volatile memory that persists between power cycles. Malware written into a BIOS could be used to reinfect machines even after new operating systems have been installed or hard drives replaced. Because the system BIOS runs early in the boot process with very high privileges on the machine, malware running at the BIOS level can be very difficult to detect. As the BIOS loads first, there is no opportunity for anti-malware products to authoritatively scan the BIOS.

However, the skills and resources necessary to develop a BIOS exploit are only possessed by a limited number of attackers. The computer system vulnerabilities needed to create a BIOS exploit have typically been highly system-specific, either impacting a specific version of a system BIOS or only certain hardware components (e.g., a particular motherboard chipset). Instead, most malware targets software executing at or above the operating system kernel, where it is easier to develop and where it can successfully attack larger classes of machines. While malware written at the BIOS level has the potential to be more powerful and difficult to detect than malware targeting operating systems or applications, the latter is likely to be the preferred attack vector when the goal is to infect a large number of users. Instead, BIOS-level malware may be more likely in highly targeted attacks on high-value computer systems. The move to UEFI-based BIOS may make it easier for malware to target the BIOS in a widespread fashion, as these BIOS implementations are based on a common specification.

For the reasons outlined above, there are few known instances of BIOS-level malware. At this time, the only publicly-known malware targeting the system BIOS that has infected a significant number of computers is the CIH virus, also known as the Chernobyl virus [Sym02], first discovered in 1998. One element of the payload of this virus attempted to overwrite the BIOS on systems using a specific chipset that was widely deployed at the time. This malware relied on several vulnerabilities that are not present in modern machines.

However, security researchers have demonstrated other potential attacks on conventional BIOS and EFI/UEFI firmware. Proof-of-concept attacks have been demonstrated which allow for the insertion of malicious code into conventional BIOS implementations that allow unsigned updates [SaOr09]. Other researchers discovered a buffer overflow vulnerability in the EFI BIOS on a modern platform. Although this EFI BIOS write-protects firmware early in the boot process and only flashes signed updates to firmware, the buffer overflow allowed the researchers to bypass the secure update process by executing an unsigned portion of the firmware update package before write-protections were applied [WoTe09].

Vulnerabilities such as these could allow attackers to create stealthy rootkits that operate with very high privileges on a system. The system BIOS loads SMI handlers before passing control of the computer to the operating system. Malicious code written into a BIOS could modify the SMI handlers to create a rootkit that would run in SMM [EmSp08]. This would give the rootkit unrestricted access to physical memory and peripherals connected to the host machine, and it would be very difficult for software running on the operating system to detect.

2.5 Threats to the System BIOS

The preceding section established the importance of maintaining the integrity of the system BIOS. This section describes some of the various ways that the integrity of the system BIOS can be attacked, and identifies the attacks considered within scope for the security controls and processes specified in Section 3.

The first threat to the integrity of the system BIOS comes while the system moves through the supply chain. Supply chain security techniques are out of scope for the security controls specified in this document. However, some of the procedures specified in Section 3.2 can be used to identify and remediate systems with an unapproved system BIOS.

Assuming that the system arrives with the manufacturer's intended system BIOS installed, without modification, there are a number of threats to the integrity of the system BIOS during the system's lifetime:

- One of the most difficult threats to prevent is a user-initiated installation of a malicious system BIOS. User-initiated BIOS update utilities are often the primary method for updating the system BIOS. The guidelines included in this document will not prevent users from installing unapproved BIOS images if they have physical access to the computer system. As with supply chain threats, security processes may be able to detect and remediate the unapproved system BIOS, such as initiating a recovery process to restore to an approved BIOS.

- Malware could leverage weak BIOS security controls or exploit vulnerabilities in the system BIOS itself to reflash or modify the system BIOS. While general-purpose malicious software is unlikely to include this functionality, a targeted attack on an organization could be directed towards an organization's standard system BIOS.

- Network-based system management tools could also be used to launch an organization-wide attack on system BIOSes. For example, consider an organization-maintained update server for the organization's deployed system BIOS; a compromised server could push a malicious system BIOS to computer systems across the organization. This is a high-impact attack, but requires either an insider or compromise of an organization's update process.

- Any of the preceding mechanisms could be used to rollback to an authentic but vulnerable system BIOS. This is a particularly insidious attack, since the "bad" BIOS was authentic (i.e., shipped by the manufacturer.) The security controls specified in the following section are primarily focused on verifying the source and integrity of the system BIOS. This attack would not violate those controls.

The controls described in the following section are primarily focused on preventing unauthorized modification of the system BIOS by potentially malicious software running on computer systems. Controlled manual installation is permitted as an institutional necessity. Installation of an unapproved system BIOS in the supply chain, manually by the user, or rollback to an authenticated but vulnerable system BIOS is not addressed by the controls in Section 3.1, but can be addressed using processes specified in Section 3.2.

3. Threat Mitigation

The system BIOS is a critical component of a secure system. As the first code executed during the boot process, the system BIOS is implicitly trusted by hardware and software components in a system. The previous section described the system BIOS's role in the boot process, the system BIOS's appeal to attackers, and potential threats resulting in the unauthorized modification of the BIOS. This section presents the security requirements for system BIOS implementations and recommended practices for managing BIOSes in an enterprise environment. The proposed recommendations are intended mitigate the threats introduced in the previous section.

3.1 Security Requirements for System BIOS Implementations

Each of the previously identified threats to the system BIOS involved making unauthorized modifications to the system BIOS. Maintaining the integrity of the system BIOS is key to mitigating those threats. This section provides guidelines intended to maintain the integrity of the system BIOS after it has been provisioned in a trustworthy state. By securing the mechanisms used for updating the system BIOS, these controls are intended to prevent remote attackers or malicious software running on a system from making unauthorized modifications to the system BIOS.

In particular, this section defines requirements for system BIOS implementations for a secure BIOS update mechanism. A secure BIOS update mechanism includes:

1. a process for ensuring the authenticity and integrity of BIOS updates; and
2. a mechanism for ensuring that the BIOS is protected from modification outside of the secure update process.

Authentication ensures that a BIOS update image is generated by an authorized source and is unaltered in transit to a client machine. All updates to the BIOS shall either go through an authenticated BIOS update process as described in Section 3.1.1 or use a secure local update process compliant with the guidelines in Section 3.1.2.

These guidelines for a secure BIOS update mechanism do not mitigate all risks associated with the system BIOS. Some threats to unauthorized modification of the system BIOS remain. For example, these guidelines do not prevent system administrators with physical access to systems from modifying the system BIOS. Nor do they guarantee the absence of vulnerabilities in the system BIOS implementations. The technical guidelines on the system BIOS should be used in conjunction with organizations' security policies and procedures.

3.1.1 BIOS Update Authentication

The authenticated BIOS update mechanism involves using digital signatures to ensure the authenticity of the BIOS update image. In order to update the system BIOS using the cryptographically protected update mechanism, there shall be a Root of Trust for Update (RTU) that contains a signature verification algorithm and a key store that includes the public key needed to verify the signature on the BIOS update image. The key store and the signature verification algorithm shall be stored in a protected fashion on the computer system and shall only be modifiable using an authenticated update mechanism or a secure local update mechanism as outlined in Section 3.1.2.

The key store in the RTU shall include the public keys that may be used to verify signatures on BIOS update images. If a copy of the public key is provided with the BIOS update image, then the key store may instead hold hashes [FIPS 180-3] of these public keys, in which case the update mechanism shall ensure that the hash of the public key provided with the BIOS update image appears in the key store before using the provided public key to verify the signature on the BIOS update image. In its simplest form, a key store may consist of a single public key, established by the platform manufacturer, which cannot be changed. However, in order to support enterprise management of the update process, there should be a key store update mechanism that allows for public keys to be added to or removed from the key store using an authenticated update mechanism or a secure local update mechanism. An enterprise may add one or more keys that it controls to the key store and may remove keys that were initially placed in the key store by the platform manufacturer in order to support enterprise management of the update process. Keys may also be removed from the key store if there is suspicion of compromise of the corresponding private key or if the enterprise is preparing to dispose of the system.

The RTU may include more than one key store. For example, the BIOS update process may be designed to only permit the installation of BIOS images that have been signed by the platform manufacturer while still allowing for enterprise control over the update process. In this scenario there may be one key store that only contains keys that are under the control of the platform manufacturer and a second key store whose contents are under the control of the enterprise. The update process could then have a configuration setting that only permits installation of BIOS images that have been signed by a platform manufacturer-controlled key and by an enterprise-controlled key.

BIOS images shall be signed in conformance with NIST SP 800-89, *Recommendation for Obtaining Assurances for Digital Signature Applications* [SP800-89], using an approved digital signature algorithm as specified in NIST FIPS 186-3, *Digital Signature Standard* [FIPS186-3], that provides at least 112 bits of security strength, in accordance with NIST SP 800-131A, *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths* [SP800-131A].

Except when the system BIOS is being updated using a secure local update mechanism, the update mechanism shall ensure that the BIOS update image has been digitally signed and that the digital signature can be verified using one of the keys in the key store in the RTU before updating the BIOS. Recovery mechanisms that are intended to install a known good BIOS recovery image shall also use this authenticated update mechanism unless the recovery process meets the guidelines for a secure local update.

The authenticated update mechanism should include a mechanism to prevent the unauthorized rollback of the BIOS to an earlier version, which may have known security weaknesses. This may be accomplished, for example, by verifying that the BIOS image to be installed has a newer version number than the currently installed BIOS image. Alternatively each BIOS image may be time stamped and the update process may verify that the time stamp on the update image is later than the time stamp[1] on the currently installed BIOS image.

In some cases, an enterprise may find it necessary to roll back a system BIOS to an earlier version. The update process should include a mechanism to confirm authorization for the authenticated replacement of a system BIOS with an older version. This may be accomplished through procedural mechanisms, such as assigning a newer version number to an earlier BIOS, or via other technical mechanisms.

3.1.2 Secure Local Update

A secure local update mechanism ensures the authenticity and integrity of the BIOS update image by requiring the administrator who is updating the system BIOS to be physically present and by ensuring that no intermediary components can modify the BIOS update image that is presented by the administrator. These requirements may be achieved by only enabling the secure local update mechanism during the

[1] In this context, the time stamp is simply a representation of the date and time (as opposed to a cryptographic time stamp).

initial phases of the boot process, before executing any firmware or software that can be modified without using an authenticated update mechanism or a secure local update mechanism. Further protections may be implemented in the secure local update mechanism by requiring the entry of an administrator password or the unlocking of a physical lock (e.g., a motherboard jumper) before permitting the system BIOS to be updated. The secure local update mechanism, if it is implemented, will typically only be used to load the first BIOS image and to recover from a corruption of a system BIOS that cannot be fixed using the authenticated update mechanism described in Section 3.1.1.

3.1.3 Firmware Write Protection

To prevent unintended or malicious modification of the system BIOS outside the authenticated BIOS update process, the system BIOS shall be write-protected with a protection mechanism that cannot be overridden outside of an authenticated BIOS update. The write-protection mechanism shall itself be protected from unauthorized modification. The write-protection requirement does not apply to configuration data used by the system BIOS that is stored in non-volatile memory.

The write-protection mechanism shall write-protect regions of the system flash memory containing the system BIOS prior to executing firmware or software that can be modified without using an authenticated update mechanism or a secure local update mechanism. For example, the motherboard may use system flash memory using hardware block locking registers. These registers could be set to disallow writes to the system flash memory very early in the boot process. The mechanisms for locking the system flash and firmware volume shall not be alterable except by an authorized mechanism.

The RTU shall be protected from modification using a locking mechanism that is at least as strong as that protecting the BIOS. For example, if the RTU is integrated with the system BIOS, then the RTU can be protected using the same protections as the BIOS. Alternatively, if an RTU can be isolated from the rest of the system BIOS, then a stronger locking mechanism may be implemented using physically protected storage.

3.1.4 Non-Bypassability

The authenticated BIOS update process shall be the exclusive mechanism for modifying the system BIOS absent physical user intervention. The design of the system and accompanying system components and firmware shall ensure that there are no mechanisms that allow the system processor or any other system component to bypass the authenticated update mechanism. Any such mechanisms capable of bypassing the authenticated update mechanism could create a vulnerability allowing malicious software to modify the system BIOS or overwrite the system flash with a BIOS image from an illegitimate source.

A modern desktop or laptop computer may include several design features that give system components direct access to the system BIOS. These may be consequences of performance improvements, such as shadowing the BIOS in RAM, or the result of system management features. While system components may have read access to BIOS flash memory, they shall not be able to directly modify the system BIOS except through the authenticated update process or by an authorized process requiring physical user intervention. This non-bypassability requirement does not apply to configuration data used by the system BIOS that is stored in non-volatile memory. Examples of potential bypass mechanisms that shall not directly modify the firmware include:

- Bus mastering that bypasses the main processor
- Memory mapping the RAM to allow shadow writes
- Direct Memory Access to the system flash by DMA-capable devices
- Microcontroller firmware execution

A recovery procedure may be implemented to restore the system to an operational state if the system recognizes that the system BIOS has been corrupted. The procedure can be invoked either manually or automatically. If a recovery mechanism is implemented, it shall use the authenticated update process specified in Section 2, or require a physically present user.

3.2 Recommended Practices for BIOS Management

This section introduces considerations for managing system BIOS in an enterprise operational environment leveraging the existing policy, process, and operations practices. It focuses on key activities revolving around provisioning, deploying, managing, and decommissioning the system BIOS as part of its overall platform life cycle. Activities performed in a recovery phase are also specified to handle exceptional conditions.

Provisioning Phase: It is crucial that the organization institute a mechanism for identifying, inventorying, and tracking the different computer systems to track the computers across the enterprise throughout their life cycle. Identifying and monitoring the BIOS image characteristics such as manufacturer name, version, or time stamp allows the organization to perform update, rollback, and recovery. The organization should maintain a "golden master image" for each approved system BIOS, including superseded versions, in secure offline storage.

If the platform has a configurable Root of Trust for Update (RTU), the organization needs to maintain a copy of the key store and verification code. Where the RTU is integrated into the system BIOS, this is achieved by maintaining the golden BIOS image. If they are managed separately, the security afforded the RTU should be consistent with or stronger than the security for the golden BIOS image.

Most organizations will rely upon the manufacturer as the source for the authenticated BIOS. In this case, the organization does not maintain any private keys, and the RTU will contain only public keys provided by the manufacturer. Where the organization prefers to participate actively in the BIOS authentication process by signing some or all approved system BIOS updates, the RTU will contain one or more public keys associated with the organization.[2] In this case, the organization must securely maintain the corresponding private key so that the next BIOS update can be signed. Private keys should be maintained under multi-party control to protect against insider attacks. For organizational keys, the corresponding public keys must also be maintained securely (to ensure authentication of origin).

In addition, a common configuration baseline for each platform must be created to conform to the organization's policy. The baseline should ensure that the firmware write protection and non-bypassability are enabled (if they are configurable), and organization policies for password policy and device boot order are enforced. Finally, the BIOS image information and associated baseline of settings for each platform should be documented in the configuration management plan.

Platform Deployment Phase: The secure local update process should be used to provision the approved BIOS for that platform from the golden master image, the corresponding RTU should be installed, and BIOS-related configuration parameters established before computer systems are deployed. This will help the organization maintain a consistent, known starting posture. The organization should periodically perform assessments to confirm that the organization's BIOS policies, processes, and procedures are being followed properly.

[2] A system could be configured to require both the manufacturer's and organization's signatures, or the organization's signature could authorize rollback or other exceptional updates.

Specifically, the procedures must ensure that the appropriate system BIOS is installed, the RTU contains all required keys and no others, and the firmware write protection and non-bypassability are enabled if they are configurable.

Operation and Maintenance Phase: This phase includes the operations and maintenance activities that are important for maintaining the BIOS security and reliability in the operational environment. System BIOS updates should be performed using a change management process and the new approved version should be documented in the configuration plan, noting the previous BIOS image has been superseded.

The BIOS image and configuration baseline should be continuously monitored. If an unapproved deviation from this baseline is detected, the event should be investigated, documented, and remediated as part of the incident response activities. The incident response plan should document the process and set of authorized tools that can be used to capture the evidence to help determine the root cause. The secure local update mechanism should be used to recover from a BIOS image compromise.

When a new BIOS image is required to extend system capabilities, improve system reliability, or remediate software vulnerabilities, BIOS updates should be performed using the authenticated update process. Where the organization participates actively in the update process, the private key needs to be retrieved from storage and a digital signature generated using the algorithm recorded in the inventory. The BIOS installation package should also be signed, and the digital signature should be verified before execution. Once the update has executed successfully, the configuration baseline should be validated to confirm that the computer system is still in compliance with the organization's defined policy.

Recovery Phase: In some cases, a BIOS update will be required that cannot be accomplished using the authenticated update process. Most commonly, a corrupted system BIOS or RTU may be unable to execute or invoke the authentication procedures. In this case, the appropriate system BIOS and/or RTU may be able to be reinstalled using the secure local update process. In other cases, a BIOS update may have unintended consequences, forcing the organization to roll back to an earlier version. Extra steps may be required for an authenticated update to authorize rollback (if versioning or timestamps are compared during the standard authentication process), or the secure local update process may be required to reestablish a secure baseline. As with the Operations and Maintenance phase, it is essential to reset BIOS configuration parameters after BIOS rollback or reinstallation.

Disposition Phase: Before the computer system is disposed and leaves the organization, the organization should remove or destroy any sensitive data from the system BIOS. The configuration baseline should be reset to the manufacturer's default profile; in particular, sensitive settings such as passwords should be deleted from the system and keys should also be removed from the key store. If the system BIOS includes any organization-specific customizations then a vendor-provided BIOS image should be installed. The fifth phase of the platform life cycle reduces chances for accidental information leakage and guarantees that sensitive data does not leave an organization.

Appendix A—Glossary

Selected terms used in the publication are defined below.

Basic Input/Output System (BIOS): BIOS is used in this publication to refer collectively to boot firmware based on the conventional BIOS, Extensible Firmware Interface (EFI), and the Unified Extensible Firmware Interface (UEFI).

Conventional BIOS: Legacy boot firmware used in many x86-compatible computer systems. Also known as the legacy BIOS.

Core Root of Trust for Measurement (CRTM): The first piece of BIOS code that executes on the main processor during the boot process. On a system with a Trusted Platform Module the CRTM is implicitly trusted to bootstrap the process of building a measurement chain for subsequent attestation of other firmware and software that is executed on the computer system.

Extensible Firmware Interface (EFI): A specification for the interface between the operating system and the platform firmware. Version 1.10 of the EFI specifications was the final version of the EFI specifications, and subsequent revisions made by the Unified EFI Forum are part of the UEFI specifications.

Firmware: Software that is included in read-only memory (ROM).

Option ROM: Firmware that is called by the system BIOS. Option ROMs include BIOS firmware on add-on cards (e.g., video card, hard drive controller, network card) as well as modules which extend the capabilities of the system BIOS.

Real Mode: A legacy high-privilege operating mode in x86-compatible processors.

System Management Mode (SMM): A high-privilege operating mode found in x86-compatible processors used for low-level system management functions. System Management Mode is only entered after the system generates a System Management Interrupt and only executes code from segregated block of memory.

System Flash Memory: The non-volatile storage location of system BIOS, typically in EEPROM flash memory on the motherboard. While system flash memory is a technology-specific term, guidelines in this document referring to the system flash memory are intended to apply to any non-volatile storage medium containing the system BIOS.

Trusted Platform Module (TPM): A tamper-resistant integrated circuit built into some computer motherboards that can perform cryptographic operations (including key generation) and protect small amounts of sensitive information, such as passwords and cryptographic keys.

Unified Extensible Firmware Interface (UEFI): A possible replacement for the conventional BIOS that is becoming widely deployed in new x86-based computer systems. The UEFI specifications were preceded by the EFI specifications.

Appendix B—Acronyms and Abbreviations

This appendix contains a list of selected acronyms and abbreviations used in the guide.

ACPI	Advanced Configuration and Power Interface
BDS	Boot Device Selection
BIOS	Basic Input/Output System
CPU	Central Processing Unit
CRTM	Core Root of Trust for Measurement
DXE	Driver Execution Environment
EEPROM	Electrically Erasable Programmable Read-Only Memory
EFI	Extensible Firmware Interface
FIPS	Federal Information Processing Standard
FISMA	Federal Information System Management Act
GPT	GUID Partition Table
GUID	Globally Unique Identifier
HTTP	Hypertext Transfer Protocol
IT	Information Technology
ITL	Information Technology Lab
MBR	Master Boot Record
NIST	National Institute of Standards and Technology
OEM	Original Equipment Manufacturer
OMB	Office of Management and Budget
OS	Operating System
PEI	Pre-EFI Initialization
POST	Power-on self-test
PXE	Preboot Execution Environment
ROM	Read-only Memory
RT	Runtime
RTU	Root of Trust for Update
SMI	System Management Interrupt
SMM	System Management Mode
SP	Special Publication
TPM	Trusted Platform Module
UEFI	Unified Extensible Firmware Interface

Appendix C—References

The list below provides references for this publication.

[Duarte08] G. Duarte. “How Computers Boot Up.” 5 June 2008. http://www.duartes.org/gustavo/blog/post/how-computers-boot-up

[EFI] *EFI 1.10 Specification*. Intel. 1 November 2003. http://www.intel.com/technology/efi/

[EmSp08] Shawn Embleton, Sherri Sparks, Cliff C. Zou. "SMM Rootkits: A New Breed of OS Independent Malware," *Proceedings of 4th International Conference on Security and Privacy in Communication Networks (SecureComm)*, Istanbul, Turkey, September 22-25, 2008.

[FIPS180-3] FIPS 180-3, *Secure Hash Standard*. October 2008.

[FIPS186-3] FIPS 186-3, *Digital Signature Standard.* June 2009.

[DuGr09] Loïc Duflot, Olivier Grumelard, Olivier Levillain and Benjamin Morin. “ACPI and SMI handlers: some limits to trusted computing.” *Journal in Computer Virology*. Volume 6, Number 4, 353-374.

[Graw09] D. Grawrock. *Dynamics of a Trusted Platform: A Building Block Approach*. Hillsboro, OR: Intel Press, 2009.

[Heas07a] J. Heasman. “Firmware Rootkits: A Threat to the Enterprise.” Black Hat DC. Washington, DC. 28 February 2007. http://www.nccgroup.com/Libraries/Document_Downloads/02_07_Firmware_Rootkits_The_Threat_to_the_Enterprise_Black_Hat_Washington_2007_sflb.sflb.ashx

[Heas07b] J. Heasman. “Hacking the Extensible Firmware Interface.” *Black Hat USA*. Las Vegas, NV. 2 August 2007. https://www.blackhat.com/presentations/bh-usa-07/Heasman/Presentation/bh-usa-07-heasman.pdf

[Intel03] *Intel Platform Innovation Framework for EFI- Architecture Specification v0.9*. Intel. September 2003. http://www.intel.com/technology/framework/

[KGH09] A. Kumar, G. Purushottam, and Y. Saint-Hilaire. *Active Platform Management Demystified*. Hillsboro, OR: Intel Press, 2009.

[Sal07] Salihun, Darmawan. *BIOS Disassembly Ninjutsu Uncovered*. Wayne, PA: A-LIST, 2007.

[SaOr09] A. Sacco, A. Ortéga. “Persistant BIOS Infection.” *Phrack*. Issue 66. 6 November 2009. http://www.phrack.com/issues.html?issue=66&id=7

[SP800-57] NIST SP 800-57, *Recommendation for Key Management – Part 1: General*. March 2007.

[SP800-89] NIST SP 800-89, *Recommendation for Obtaining Assurances for Digital Signature Applications*. November 2006.

[SP800-131A] NIST SP 800-131A, *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*. January 2011.

[Sym02] *W95.CIH Technical Details*. Symantec. 25 April 2002. http://www.symantec.com/security_response/writeup.jsp?docid=2000-122010-2655-99

[TCG05] *PC Client Work Group Specific Implementation Specification for Conventional Bios Specification, Version 1.2*. Trusted Computing Group. July 2005. http://www.trustedcomputinggroup.org/resources/pc_client_work_group_specific_implementation_specification_for_conventional_bios_specification_version_12

[UEFI] *UEFI Specification Version 2.3*. Unified EFI Forum. May 2009. http://www.uefi.org/specs/

[Wech09] F. Wecherowski. "A Real SMM Rootkit: Reversing and Hooking BIOS SMI Handlers." *Phrack*. Issue 66. 6 November 2009. http://www.phrack.com/issues.html?issue=66&id=11

[WoTe09] R. Wojtczuk, A. Tereshkin. "Attacking Intel BIOS." *Black Hat USA*. Las Vegas, NV. 30 July 2009. http://www.blackhat.com/presentations/bh-usa-09/WOJTCZUK/BHUSA09-Wojtczuk-AtkIntelBios-SLIDES.pdf