Trusted Firmware Services Based on TPM

Zhenlong Du[1,4], Xiaoli Li[1], and Kangkang Shen[2,3]

[1] College of Electronics and Information Engineering of Nanjing University of Technology
[2] High Tech Research Institute of Nanjing University of Technology
[3] Nanjing Byosoft Co. Ltd, Nanjing
[4] State Key Laboratory of Novel Software Technology, Nanjing University
{Duzhlcad,lixlorchid}@gmail.com, kshen@byosoft.com.cn

Abstract. How to build trusted firmware platform has been a research hot in computer security community. In this paper, a novel entrusted firmware services under UEFI framework are proposed, which exploits the high safety of Trusted Platform Module (TPM) root and uneditability of firmware for building the trusted platform in code modification checking, user identity authenticating, hard disk attestation and real time security alert. The experiment showed that the presented schemas are feasible, and could efficiently construct a trusted firmware platform.

Keywords: firmware, TPM, UEFI.

1 Introduction

Nowadays, such security problem as the virus incidence, insider abuse, internet fraud, etc [1, 7] are increased, for people to utilize a secure computer is desired. To build the secure computer is a comprehensive project, and it involves firmware, hardware, operating system, authentication, communication, and so on. Untill now, some tools as virus detection, digital signature, intrusion detection have been developed, but few methods on firmware are proposed. In this paper, how to construct the trusted firmware platform under UEFI framework is discussed.

The Trusted Computing Group (TCG) [2] proposed hardware-rooted TPM approach for PC security. TPM is a hardware-based security and cryptography chip and it has been widely adopted and installed on more than 100 million PC. TPM can augment PC with a secure hardware repository for safeguarding digital certificates, passwords and digital keys etc. TPM keeps the incidence of trusted computing on client and sends the information to a verifier which evaluates the validness of hardware and software [13].

Hardware is directly driven by firmware, which is prevailingly developed by UEFI (Unified Extensible Firmware Interface) [3, 4]. UEFI is an interface specification between hardware platform and OS (Operating System), it is independent of both hardware and OS. UEFI framework is comprised of such modules as driver, protocol and application. UEFI specification is contributed by many hardware vendors, OS developer and IBVs, it has become the actually industrial firmware standard. BIOS

L. Chen and M. Yung (Eds.): INTRUST 2009, LNCS 6163, pp. 227–235, 2010.

vendors like AMI, Insyde and Phoenix [6] are leading UEFI promoters, and nowadays UEFI has been the popular firmware developing platform.

Firmware abstracts the hardware interface, provides some basic services to operating system, and has higher security. Nowadays few developers are familiar with the firmware mechanism; secure strategy imposed on firmware is more invulnerable than one on the software. Essentially speaking, trusted firmware services is dependent on the combination of software and hardware. Firmware are, in general, much safer than OS, application and communication software. But the new generation of firmware might have security holes through API interfaces and extensive modules. In this paper, we proposed a method to use TPM to enhance the security of PC firmware.

The organization of the paper is as follows, UEFI framework and popular TPM functions are brief reviewed in section two. The third section is the main part of the paper, which covers three issues, multi-factor user authentication, hard disk protection and platform attestation alert. The experiments are discussed in section four, which is followed by conclusion.

2 Related Works

The goal of trusted firmware services based on TPM is to build the safe computer, it involves UEFI based firmware development and security services provided by TPM. We will briefly review these two issues in this section.

2.1 UEFI Based Firmware

In the traditional PC the firmware is called BIOS (Basic Input and Output System) and IBM defined its interface with OS, the interrupt services. This interface has not been changed very much until Intel proposed and implemented EFI (Extensible Firmware Interface) at the beginning of this decade. The Unified EFI Forum is a non-profit collaborative trade organization formed to promote and manage the UEFI standard, which is originally based on EFI specification. Today UEFI has close to 150 members and its standards have been widely used in today's electronic devices.

UEFI provides a program interface [5, 6] to the hardware platform which includes the motherboard, CPU, and other components. UEFI based implementation allows for executing the pre-operating system agents, such as OS loader, diagnostics, driver and application. Figure 1 illustrates UEFI framework is an ensemble of EFI system table, handles, protocols, EFI images, events, devices, drivers, and EFI based firmware. Meanwhile, UEFI framework is a complex body which balances the OS requirement and hardware initialization.

UEFI logically locates between OS and Hardware (as Figure 2 illustration), it manipulates the devices by EFI handle, driver, protocol, image, etc. Not only do EFI manage the existed hardware in computer, but also the extended devices. EFI only provides the basic functionality, such as the driver to keyboard, monitor, etc, and the driver to specific motherboard need to be additionally developed. Moreover, EFI has strong flexibility, for keeping the less space and higher efficiency, some EFI images and protocols could be dynamically loaded or unloaded. For supplying more compatibility to vendors and OEMs, EFI abstracts the interface to various hardwires with which vendors, IBVs and OEM could tightly incorporate with.

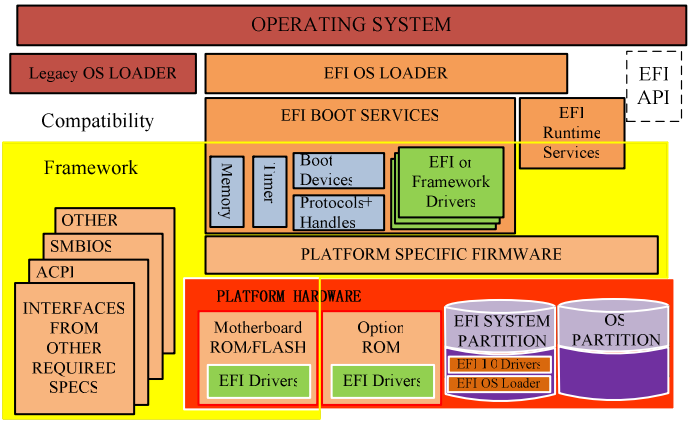

Fig. 1. EFI Framework

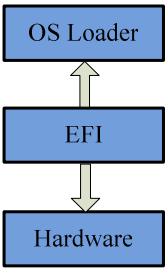

Fig. 2. EFI Architecture

As Figure 3 illustration, the common boot flow covers platform initialization, OS boot and shutdown three stages, and it includes SEC, PEI, DXE, BDS, TSL, RT and AL 8 steps. SEC is the first step and it diagnoses the integrality of firmware. Additionally, if network is available, SEC updates the latest strategy to secure leaks. PEI, initializes CPU, chip, and board, and constructs the prerequisite environment for successive step. DXE performs the device enumeration and initialization. After DXE is accomplished, some drivers reside in memory for OS access. BDS selects the appropriate device to start. TSL provides alternative boot option for added values such as maintenance. RT and AL separately refer to the states after OS running and shutdown.

2.2 TPM Based Security

TPM [11] is a computer chip (microcontroller) that can securely store artifacts used to authenticate the platform (PC or laptop). These artifacts generally include passwords, certificates, or encryption keys. TPM can also be used to store platform measurements for ensuring that the platform remains trustworthy.

The hardware-based cryptography ensures that the information stored in hardware, has better ability for protecting from the external attacks. Based on TPM, the

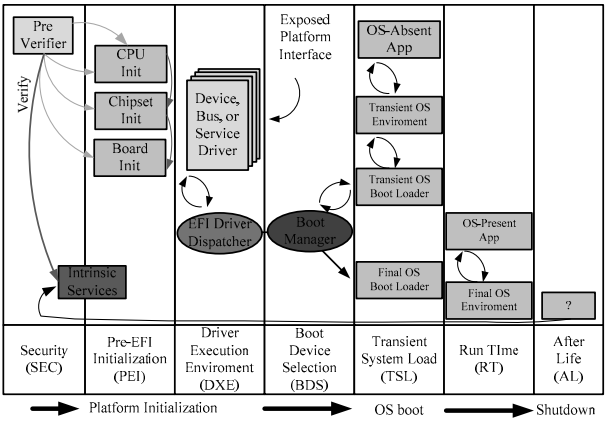

Fig. 3. Boot Flow

firmware-level applications storing secrets on hardware can be developed. These applications make it much harder to access information on computing devices without proper authorization. If the configuration of platform has changed by unauthorized activities, access to data and secrets can be denied and sealed off by firmware-level applications.

UEFI provides some classic encryption algorithms, including SHA (Secure Hash Algorithm)-1, SHA-224, SHA-256, SHA-384, SHA-512, MD5 etc [6, 12]. SHA condenses a message whose length is less than 2^{64}, produces a message digest whose length is 160-bit. The message digest is then input into the DSA (Digital Signature Algorithm) which generates or verifies the signature [10] for the message. Any change in the message digest would lead to the failure to verification. Because the message digest is usually much smaller in size than the message, SHA signs the message digest rather than the message for improving the efficiency. The same hash algorithm must be used by the verifier of a digital signature as was used by the creator of the digital signature. SHA-224 and SHA-256 extend the hash function by NIST (National Institute of Standards and Technology) for future hash algorithm requirements. SHA-384 and SHA-512 are the latest 64-bit secure hash algorithm, they could provide the higher security than SHA-1.

Trusted firmware attestation checks PCRs for deciding if the firmware is modified. PCRs store the hash value within TPM, which is employed for deciding the accordance with running firmware version. If the hash value of current firmware is not accorded with the PCRs, the current firmware assures to having been modified. Then, some operations should be taken to protect the computer from attacking. Building the trust chain basically serves as the firmware verification, additionally, as the authentication service to operating system.

Figure 4 illustrates the TPM necessary components, which include TPM storage, integrity storage, TPM authentication, cryptographic functions and platform attestation. The firmware authentication depends on these components.

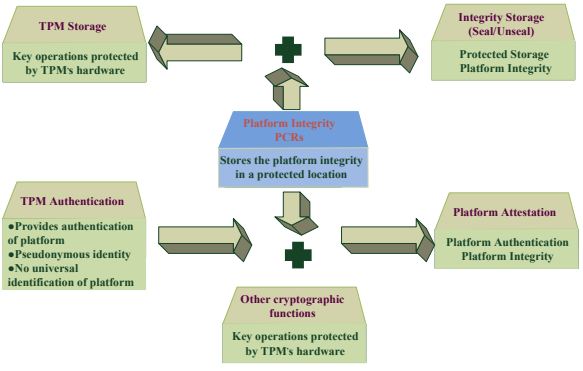

Fig. 4. TPM Features and Functions

3 Trusted Firmware Services

Attacks on software have been shown to be quite popular, but attacks on firmware and hardware have been less prolific. However, attacks on firmware are increasing common for UEFI source code is exposed to programmers, and firmware attack could continue to overwhelm the operating system. For instance, many DVD players have hacked firmware to support DVDs from any region [6].

Trusted firmware service is mainly for providing the secure firmware services which could protect computer from attack.

3.1 Biometric Password Protection

User authentication has always be the most important issue in computing security. The most common way to protect a PC is to use a password. Later more secure approach like USB token, fingerprint device are also used. Relatively speaking the biometric method is more convenient and more secure. But in a PC most of the password check is during the period of booing to OS. This gives a intruder the chance to boot to an alternative device [8, 9] with loaded tools to attack the system. In this paper we demonstrated the use of fingerprint device in the firmware level well before the boot device is selected. Since the biometric password protection is burned into the flash chip on the PC platform. It will require hardware specific and proprietary information to update the flash and bypass the password checking.

Before the UEFI firmware was introduced the implementation of fingerprint device in Firmware level were relatively complicated since it requires full source code disclosure to include a fingerprint device support. This presented major problem when different combination of CPU, Chipset and fingerprint device have to be chosen. In UEFI firmware, each device is supported by separated driver as long as it complies with the specification. In this case, we chose a USB fingerprint device. The UEFI driver supporting the device can be easily converted from an existing OS driver. What we did is to specify a UEFI protocol definition and help the fingerprint device provide to modify their drive according to our definition.

Since the fingerprint data is stored inside the flash chip, it might be easily accessed and copied if someone knows where to look for it. To further enhance the security, we adopted biometric-TPM multi-factor authentication. We use TPM to complement the fingerprint reader by encrypt the data and store the associated key in TPM.

3.2 Sensitive Data Protection

One of the key issues of computer security is to protect data such that it can not be accessed by an unauthorized person. There are many ways to accomplish this task at OS level. But at the level of firmware we do not have the computation power to do data encryption without affecting the computer performance. One might use complicated virtualization technology or proprietary hard disk firmware to achieve this goal, but these solutions will either increase cost or deteriorate performance. Here, we proposed an easy way to protect data on a personal disk at the firmware level. The method is easy to implement, transparent to the user and very secure.

In fact the ATA specification has been implemented security hard disk command to protect the contents on a hard disk. This feature is not commonly used because the data will be lost if one forget the password. If the password can be securely reproduced we have no reason not to use this security feature.

In our implementation we have used a public memorable user ID to generate a 32 byte password for the hard disk security password. Since the password is dynamically generated in a proprietary way before the computer is fully powered on, it ensured that only authorized person can access the data. With this implementation it is also possible that different people will have different access right based on how the passwords are set in different hard disks.

We have implemented a setup item to ask user's permission to use the password feature and input a user ID. The whole process of setting the password is done automatically and no one knows what exactly the password is. When the computer is turned on the next time, the UEFI firmware will retrieve the password and send an unlock command to hard disk. The whole process is automatically completed as part of the POST and the user is even not aware that the password is used.

To associate the hard disk with the particular PC, we used the TPM chip as a complementary tool. In this case we used platform identity keys and made the hard disk password generation process dependent on the platform. With this added protection the password can not be recovered once the hard disk is separated from the platform. In another words, the hard disk is not only associated with the user ID but also bound with the platform where the TPM chip is mounted.

3.3 UEFI Intrusion Reporting

During the system power on process, our secure firmware will check the Integrity of loaded software modules, verify user ID and passwords. In most of the cases a secure firmware will block the execution of an un-trusted software module or deny the access for an unauthorized person. This kind of passive action still leaves the intruder repeated chance to break into the system. One may also use event log feature in the computer. But this will only help to track the cause and is not help to prevent it from happening.

In UEFI the firmware, features like network communication can be easily enabled. We have utilized the network communication protocols to report intrusion actions to corporation IT service on real time. In this way we can effectively catch the intruder and prevent a disaster from happening.

Below is a diagram illustrating the major software components

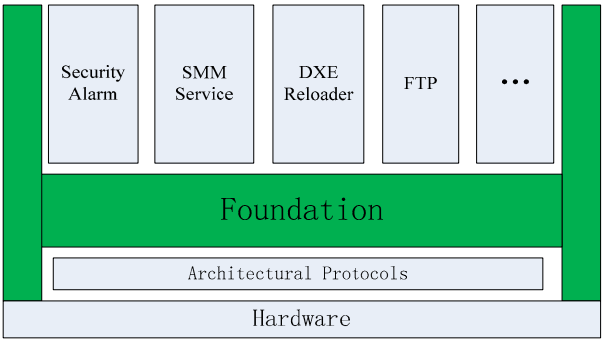

Fig. 5. Intrusion Detection Components

The security alarm module is the first module triggered after a security validation is detected. This module is responsible to collect the user ID, time and date and the type of violation. This module will set a flag and let the normal POST continue until a proper time to trigger a SMI. Once the SMI is trigged the program will display a warning message and remind the user a security validation of certain type has been detected and the system will soon be reset. In the background we will reload a new single purpose DXE environment where the network FTP module is enabled. It will take only a second to send the security violation message to the corporate IT services desk before the system reset.

4 Experiments

The whole experiment is carried out on an Intel 945G motherboard. The BIOS source code is licensed from Intel directly. In order to do the mentioned experiments We have developed critical components such as compatibility support module. Due to the limited capability we only removed the original flash chip and reprogrammed with our own BIOS.

The hard disk used in the experiment is standard SATA hard disk. The fingerprint adapter is an USB connected device. With the help of the fingerprint manufacture we are lucky to get the EFI drivers comply with the interfaced protocol we defined.

In order to verify that we have successfully implemented the TPM features suggested by the TCG group, we have installed Microsoft operating system VISTA and verified with Bitlocker feature installable and functioning.

We had to setup a temporary super password before a proper fingerprint password is recorded. This super password is only good for setting up the fingerprint device

and it can not be used to access the protected data on hard disk which is bound to a specific user. We have also verified that once the protect hard disk is removed from the system no other system can read the hard disk.

5 Conclusion

In our UEFI firmware services we successfully established the trust chain from the root of trust which made TPM services to OS and applications. We developed the digital signature at the firmware level such that our secure firmware will only execute those extension modules with recognized signature.

We implemented the biometric computer password at firmware level. A unauthorized person is even not able to boot the computer. The way, we have implemented the password of the hard disk, makes the hard disk data not only bounded to the user but also to the platform.

Under the relatively safe firmware environment, we have demonstrated with two security applications. One is mounting the anti-virus engine inside the firmware and lunch the anti-virus program as an pre-OS application. This guarantee we can remove virus infection even when the OS/Hard disk is not bootable. The second is error reporting to specified corporation IT center. In this way the owner can catch any intrusion, or attempt of loading unauthorized modules.

Acknowledgements

The research is supported by Natural science fund for colleges and universities in Jiangsu Province (09KJB52006), by State Key Laboratory of Novel Software Technology at Nanjing University (KFKT2008b15), and by preliminary research plan of Nanjing University of Technology.

References

1. CSI Computer Scrime&Security Survey (2008)
2. Trusted Computing Group, `http://www.trustedcomputinggroup.org/`
3. Zhang, X., Zhang, S., Deng, Z.: Virtual Disk Monitor Based on Multi-core EFI. In: Xu, M., Zhan, Y.-W., Cao, J., Liu, Y. (eds.) APPT 2007. LNCS, vol. 4847, pp. 60–69. Springer, Heidelberg (2007)
4. Zimmer, V., Rothman, M., Hale, R.: Beyond BIOS: Implementing the Unified Extensible Firmware Interface with Intel's Framework. Intel Press, Hillsboro (2006)
5. Intel MultiProcessor Specification, Version 1.4 (May 1997)
6. Intel Unified Extensible Firmware Interface Specification, Version 2.1 (January 2007)
7. Hendricks, J., Doon, L.: Secure Bootstrap is Not Enough: Shoring up the Trusted Computing Base. In: Proceedings of the Eleventh SIGOPS European Workshop, ACM SIGOPS. ACM Press, New York (2004)
8. Ball, T., Bounimova, E., Byron, C., Levin, V., et al.: Thorough static analysis of device drivers. ACM SIGOPS Operating Systems Review 40(4), 73–85 (2006)

9. Spear, M.F., Roeder, T., Hodson, O., Hunt, G.C., et al.: Solving the starting problem: device drivers as self-describing artifacts. ACM SIGOPS Operating Systems Review 40(4), 45–57 (2006)
10. Wang, G., Bao, F., Zhou, J.: The Fairness of Perfect Concurrent Signatures. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 435–451. Springer, Heidelberg (2006)
11. TCG EFI Protocol, Version 1.20 Final, Revision 1.00, June 9 (2006)
12. Brickell, E., Chen, L., Li, J.: A new direct anonymous attestation scheme from bilinear maps. In: Lipp, P., Sadeghi, A.-R., Koch, K.-M. (eds.) Trust 2008. LNCS, vol. 4968, pp. 166–178. Springer, Heidelberg (2008)
13. Suzaki, K., Iijima, K., Yagi, T., Quynh, N.A.: Trusted Boot and Platform Trust Services on 1CD Linux. In: Proc. of Third Asia-Pacific Trusted Infrastructure Technologies Conf. (October 2008)