Using TPM to Improve Boot Security at BIOS Layer

Kuan-Jen Lin and Chin-Yi Wang
Department of Electrical Engineering, Fu Jen Catholic University
{kjlin, 496505036}@mail.fju.edu.tw

***Abstract* –Boot security is fundamental to system security of PC and PC-based consumer products. Current BIOS exploits TPM to establish a trusted boot. However, it does not mean a secure boot. That is, the TPM does not prohibit booting into an insecure OS or using an insecure boot loader. In this work, we extend TPM BIOS interrupt calls to support (1) performing RAS encryption and decryption and (2) accessing the NVRAM. Furthermore, the following techniques were implemented in BIOS to improve boot security: (1) enhancing the security of the encrypted BIOS password using TPM RSA engine instead of conventional encryption techniques, (2) storing the encrypted password to TPM NVRAM such that an attacker can not clear it by removing the battery, (3) always authenticating BIOS password before passing control to OS boot-loader and (4) using TPM SHA-1 engine to verify data integrity of the full MBR and determine if the booting continues.**

Keywords: System security, TPM, BIOS, secure boot, trusted computing.

I. INTRODUCTION

TCG (Trusted Computing Group) uses the boot sequence to determine the trusted status of a platform [1, 2]. During the boot sequence, the digest of each executing program is recorded before it executes. TPM (Trusted Platform Module) is used to store all these records and then report on them securely. For example, before the BIOS hands over control to a boot loader, it hashes the boot loader and extends the resultant values into a PCR (platform Configuration Registers) in a TPM. However, the TPM does not prohibit booting into an insecure OS or using an insecure boot loader [2, 3].

A typical boot train in PC and PC-based consumer products is BIOS => bootloader (stored in MBR) => OS. BIOS maintain the platform configuration and store it in CMOS memory. Current BIOS use simple encryption technique such as CRC-16-IMB to encrypt the password. Some tools like CmosPwd [4] can break the password in a few minutes. Nevertheless, the password is stored in CMOS memory. If an attacker remove the battery of CMOS memory and leave it out for about 120 seconds, this will flush the CMOS memory which stores the BIOS password and all other configuration.

Preventing those programs executed in boot chain and related configuration data from being altered is very important to system security. Commercial pre-boot authentication technique [e.g. 5] and academic research [e.g. 6] have proposed methods to improve the boot security. However, they did not exploit the resource of the TPM at BIOS layer. Previous works exploited TPM in the EFI environment [e.g. 7]. This paper deals with legacy BIOS environment.

The main goal of this work is to improve boot security by exploiting TPM at BIOS layer. We extend TPM BIOS interrupt calls to support (1) performing RAS encryption and decryption and (2) accessing the NVRAM. Furthermore, the following techniques were implemented in BIOS to improve boot security: (1) enhancing the security of the encrypted BIOS password using TPM RSA engine instead of conventional encryption techniques, (2) storing the encrypted password to TPM NVRAM such that an attacker can not clear it by removing the battery, (3) always authenticating BIOS password before passing control to OS boot-loader and (4) using TPM SHA-1 engine to verify data integrity of the full MBR and determine if the booting continues.

II. EXPERIMENTAL PLATFORM

We use Intel Calpella platform as experimental platform, in which Phoenix BIOS is installed. Fig. 1 shows its block diagram. The TPM is linked through LPC (Low Pin Count) interface. The BIOS flash ROM is linked through SPI interface.

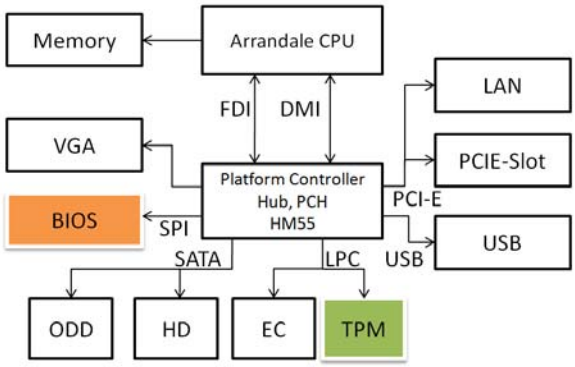

Fig.1: Experimental platform.

III. EXTENDED BIOS INTERRUPT CALLS

TCG specification defines a minimal set of BIOS interrupt calls that BIOS code and boot loader can use to talk directly to the TPM before even a TPM device driver interface is available or before the full TSS (TGG Software Stack) services are available [8]. The instruction is "int 1Ah" and the register AH must set BBh and AL set 00h~07h. The options AL=08h~7Fh are reserved. To enhance the security against breaking the BIOS password, we attempt to use RSA engine to encrypt the BIOS password. Hence, we extend the BIOS call to perform the same function as the TPM commands "TPM_seal" and "TPM_unseal" provided by TCG device driver library [2]. To avoid being cleared by removing the battery, we attempt to store the password to the TPM NVRAM. Hence, we extend the BIOS call to perform the same function as the TPM commands "Tspi_NV_WriteValue" and "Tspi_NV_ReadValue" [2]. The extended BIOS interrupt calls communicate with TPM following TPM MMIO (Memory-Mapped I/O) command specification. Table 1 lists the function and the interface parameters for the extended BIOS interrupt calls.

978-1-4577-0231-0/12/$26.00©2012 IEEE

Table 1: Extended BIOS Interrupt Calls.

AL=08h	; perform RSA encryption
ES:DI	password (at most 20 bytes)
DS:SI	encrypted password
AL=09h	; perform RSA decryption
ES:DI	encrypted password (at most 268 bytes)
DS:SI	password
AL=0Ah	; Store data to NVRAM
ES:DI	offest (4 bytes), DataLength (4 bytes), Data
AL=0Bh	;Load data from NVMM
ES:DI	offset (4 bytes), DataLength (4 bytes)
DS:SI	Data

III. Pre-OSBoot Authentication and MBR Integrity Check

After performing POST and storing platform configuration, a legacy BIOS will load the MBR to memory 0x7C00 and check if the final two bytes of the MBR are equal to AA55h. If they are equivalent, the BIOS will pass control to the code at 0x7C00, i.e. the entry point of boot loader. According to the disk partition in MBR, the OS will be executed. Current process does not prohibit booting into an insecure OS or using an insecure boot loader.

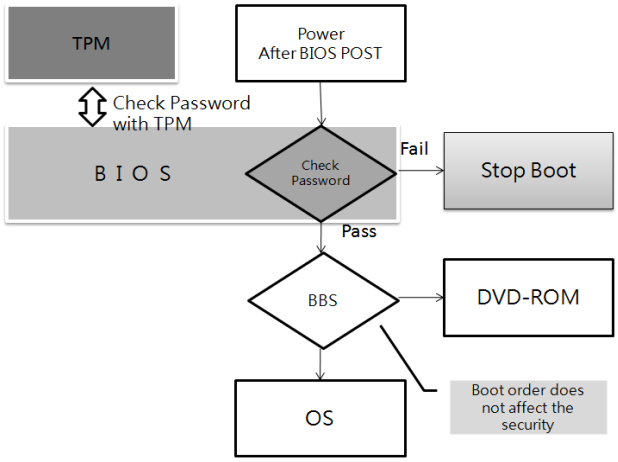

Fig. 2: The proposed pre-boot authentication flow using the extended BIOS interrupt calls.

We propose two techniques to improve the boot security. Firstly, the BIOS always execute BIOS-password authentication before reading the MBR. Fig. 2 shows the proposed flow to do the pre-boot authentication. If no such pre-boot authentication is performed, an attacker can use function key F7 to change boot priority sequence to run insecure OS in some device

Secondly, before passing control to the OS boot-loader, we verify the integrity of the full MBR using TMP Hashing function. The correct digest is stored in TPM MVRAM. If the MBR is altered, the boot process is stopped. Fig. 3 shows the proposed boot sequence. We use extended BIOS calls to store the digest of the full MBR to TPM NVRAM

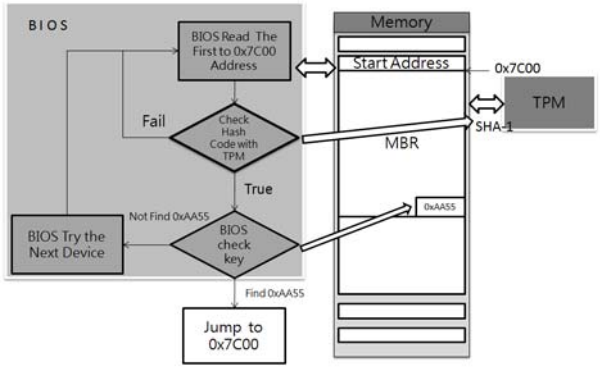

Fig.3: The proposed secure boot sequence before passing control to OS boot-loader.

IV. CONCLUSION

Boot security is fundamental to system security. Current BIOS uses TPM to establish a trusted boot rather than a secure boot. In this work, we attempt to exploit the TPM to improve boot security at BIOS layer. We have extended TPM BIOS interrupt calls to support the improvement. The proposed techniques enhance the security of BIOS password, force pre-boot authentication and verify data integrity of the full MBR to determine if the booting continues. Table 2 shows the source and binary size for the original BIOS and the version after our modification.

Table 2: Comparison of the source and binary size.

	LOC	Binary (bytes)
BIOS	3663401	1117621
BIOS after our modification	3665369 (+1966)	1118566 (+885)

REFERENCES

[1] Trusted Computing Group (TCG), *TPM Main Part 1.2.3 Design Principles Specification Version 1.2*, 2011.

[2] TD. Challener et al., *A Practical Guide to Trusted Computing*. IBM Press, 2008.

[3] Bernhard Kauer, "OSLO： Improving the security of Trusted Computing," *Proceedings of the 16th USENIX Security Symposium Proc.*, pp. 229-237, 2007.

[4] http://www.softpedia.com/get/Security/Decrypting-Decoding/CmosPwd.shtml.

[5] PhoenixTrustedCore http://www.phoenix.com/pages/pre-boot-authentication.

[6] W. A. Arbaugh, D. J. Farber and J. M. Smith, "A Secure and Reliable Bootstrap Architecture," *IEEE Symposium on Security and Privacy,* pp. 65-71. 1997.

[7] W. Fang, B. Yang, Z. Peng and Z. G. Tang, "Research and Realization of Trusted Computing Platform Based on EFI," *2009 Second International Symposium on Electronic Commerce and Security*, pp. 43-46, 2009.

[8] Trusted Computing Group (TCG), *TCG PC Client Specific Implementation Specification for Conventional BIOS0*, 2005.