International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

A REVIEW PAPER ON UEFI SECURE BOOT

Basanagouda Venkanagoudar[1], Prof. P N Jayanthi[2]

[1]*Department of Electronics and Communication Engineering. RV College of Engineering. Bangalore Karnataka, India*
[2]*Asst Professor, Department of Electronics and Communication Engineering. RV College of Engineering. Bangalore, Karnataka, India*

---***---

Abstract - *Firmware is the first piece of code to be executed while turning on a device. The firmware attacks are nearly undetectable giving the attacker complete access to the system. Existing implementations based on the NIST guidelines provide some degree of protection by detecting the executed malicious code but it does not prohibit its execution. UEFI Safe Boot provides full protection for user by blocking the malicious firmware from execution through the digital signature verification.*

Key Words: BIOS, Firmware, Secure Boot, Trusted Platform Module. NIST, UEFI

1. INTRODUCTION

Booting refers to the process of initializing the hardware. Components of the system and also handing over the system functionalities from the firmware to the Operating Systems (OS). As the connectivity among the computers increased the attacks on the operating system level also increased which led to the development of cybersecurity domain. Due to the advancements in the security checks at the OS level the malicious attackers focused their interest on the vulnerabilities in the firmware [1]. The two most widely known attacks are bootkits and rootkits. The existing solution based on National Institute of Science and Technology (NIST) guidelines [1] provide protection to a certain extent on the running of compromised firmware code by informing the user about the execution of the tampered code. With the introduction of the Unified Extensible Firmware Interface (UEFI) Secure Boot these kinds of attacks can be mitigated effectively by not executing the unknown firmware [2].

1.1 Booting Process

Once the system is powered on, the CPU is self-initialized and searches for a small piece of code which is normally located on a motherboard chip. It generally loads BIOS or UEFI firmware depending on the PC. The BIOS or UEFI firmware initializes configurational settings from a specified location on the processor, usually it is in CMOS battery supported memory. Cold boot or reboot boot is verified by determining the. Value at the memory address 0000472. The BIOS skips the rest of POST. In cold boot it performs a POST where all the hardware of platform are checked for the proper functionality. The PC shows contents on the monitor exhibiting information about the boot phases. This contains information about the BIOS developer and version, specification of the processor, etc. It looks for a bootloader program in various disks like floppy, hard-disk or external disks depending on order set. This can be changed in BIOS menu. Once the bootloader program is found it loads the and runs the program in RAM giving access to it. BIOS initialises and executes Master Boot record bootstrap loader(MBR).

MBR: It is present in the first block of the bootable disk.. Size of MBR is generally lesser than 512B. It has 3 sections 1)bootloader information in initial 446 bytes 2) 64 bytes of information on partition table 3) MBR verification check in final 2 bytes. It has instructions regarding GRUB.

Grand Unified Bootloader(GRUB):If a system has more than one kernel images loaded, there is an option to choose which one to be executed. It is usually performed by GRUB. Which has info regarding Information about system file system. Is contained in GRUB. Generally loading and running of kernel and initrd images is performed by GRUB.

Initrd: Kernel makes use of initrd as a initerim root filesystem until the original is initialized. In addition it also has essential drivers, that are required to access the harddrive partition and other things. Initrd determines the default init level from /etc/init tab and uses that to load all other program.

1.2 Threats to Firmware

The increase in the security at the run level application has made it difficult for the malware developers to get access to the system. Malware developers found loopholes in the firmware of the system to get access into the system. Rootkits and bootkits are the prominent threat to the firmware of a system. Malware present in the firmware is highly untraceable by OS, until specific search target is located inside the firmware. Firmware provides full acces to the system hardware so an attacker can. Direct access to the hardware if he manages to inject malicious code into the firmware and can use the machine virtually without knowledge of the system admin. However, some attacks try to copy themselves in the system ROM allowing themselves present even if new OS is installed or new hard disk is used. This poses a great threat to the user data and compromising the integrity of the system. Firmware rootkits directly tamper with the firmware code by injecting the malicious codes in the form of patches or by exchanging the original firmware image itself. Bootkits on the other hand utilises the period when firmware handovers the platform functions to the OS usually by changing the OS loader. Bootkits and rootkitts uses many techniques to avoid detections targeting the firmware. A prominent example bootkit that tries steal the user credentials necessary for the process of booting are usually referred as "evil maid attack" indicating a situation in which an. Unauthorized person stores a bootkit into a system that is left freely in a hotel room, office or any location. These kinds of attacks are generally used for stealing credentials for the disk encryption system and third-party security tools. Although rootkits and rootkits are a problem for any framework, especially legacy BIOS environments, they pose the UEFI ecosystem with a specific challenge. The whole idea behind UEFI is to standardize the pre-boot interface and allow for extensibility. Although this standardization and transparency allows the device more prone than it would have made in legacy BIOS , BIOS has less support detection and stopping of faulty firmware code.

2. Existing Solutions

There are numerous existing software/hardware solutions and guidelines that provide protection in the pre-booting process. The TPM would be able to provide a provision to have precaution between and firmware and OS during entire booting, "Measured boot" is one of the solution which detects the alterations in the firmware code, boot paths and settings and collect enough information about these kind of activities to inform the user later once after the firmware loading is finished. These kind of solutions does not prevent malicious firmware from executing hence the threat is imminent. NIST of the United States government has recommended several guidelines pertaining to the safety in pre-boot area, some of them are "NIST 800-14-BIOS Protection Guidelines [3]", "NIST 800-147B. BIOS Protection Guidelines for Servers NIST 800-155 BIOS Integrity Measurement Guidelines"

3. UEFI Secure Boot Image Validation

In UEFI Secure Boot every firmware part is validated using digital signatures and keyring databases present in the device. When a CPU begins, it just executes asmall number of instruction from a specific location. Nothing has been initialised yet. Scarcity of the rsources makes it difficult to get stared to execute code. The system is at this point locating, validating, installing and running a small initial piece of firmware. This security step, as initial turn-on is called, forms the foundation for the Root of Trust (RoT) in the UEFI Stable Boot cycle. CPU vendors have now started testing signatures of this initial piece of firmware to ensure that it has not been inappropriately updated and to ensure a strong start to the trust chain. The Chain of Trust used in UEFI Secure Boot flows from the very stable basis. The trust is preserved via a public key cryptography. Hardware manufacturers put in the firmware e store the Platform Key (PK), reflecting the RoT. The relationship of confidence with operating system vendors and others is recorded by using the Platform Key to sign their keys. The flow of image validation in secure boot in shown in fig 1

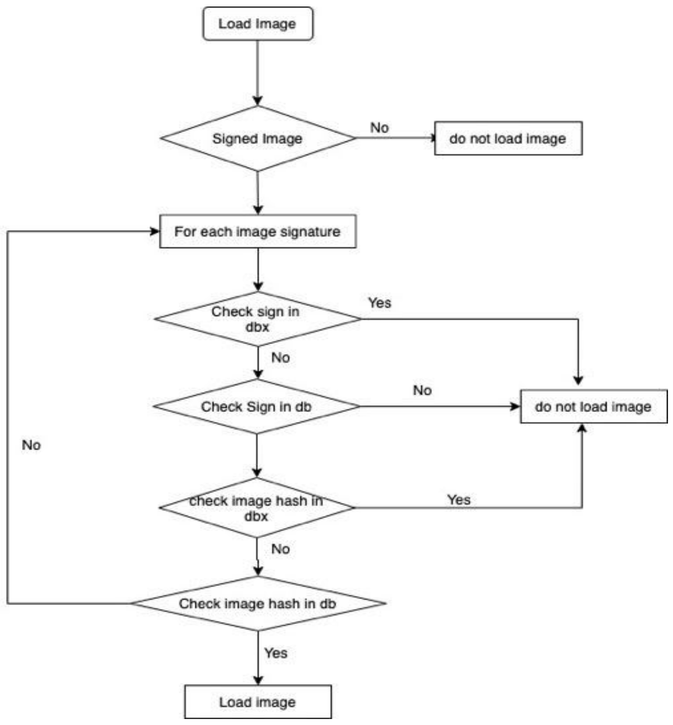

Fig -1: Image validation in secure boot.

4. Signature and Key Maintainance

A set of databses, keys are employed for the protection and maintenance of digital signature that are necessary to validate the code prior to its compilation. Platform key (PK) is the primary key. It is usually generated by the original equipment producer when a system is produced. It can be replaced by the end user if and only if he knows the original platform key. "Key Exchange Key(KEK)" ensures the protection of the key database db and dbx containing the digital keys from unintentional alterations. The signature databases only be modified if the private part of the key exchange key is known to the user. There can be more than one KEKs supplied by the OS and some of the third-party application developers. User with a correct key exchange key able to modify the signature databases. Key exchange key protects two databases namely, db which contains all the keys are allowed to authenticate the code and dbx that contains the blocked signatures that contains the signatures that forbidden to authenticate. As the boot phase continues from an initial secured firmware, additional parts of code and drivers given by peripheral suppliers will be loaded and executed for enabling their devices. This process finally results in loading and running the OS bootloader which sets up the OS. The digital key is verified by the firmware from its db, as each section of code is loaded and prior to execution, but also the digital key is absent in dbx. Including the boot loader of the OS itself. How the firmware does with signature matching details is a policy judgment, and is not decided by the specification. Unauthorized code is not allowed to be executed, and thus the program will not be able to finish the OS bootstrap phase.

5. Conclusion

The possible threat to device firmware security can be through rootkitts or bootkitts . They are in existence since the beginning of the PC's. Since other possible threats of attack were restricted and system firmware development became more versatile and reliable, the pre-boot arena became a prime objective for the malicious developers to get access to the system. UEFI Secure Boot was designed to make the systems less vulnerable to these attacks

REFERENCES

[1] V. Z. M. Krau. (2019). "Establishing the root of trust," [Online]. Available: https: //uefi.org/learning_center/papers/.

[2] B. R. Richard Wilkins. (2019). "Uefi secure boot in modern computers," [Online]. Available: https://uefi.org/learning_center/papers/.

[3] A. Regenscheid, "Platform firmware resiliency guidelines," Tech. Rep., May 2018. doi: 10.6028/nist.sp.800-193