ISSN:2231-0711

TPM: A More Trustworthy Solution to Computer Security

Angela Francis, Renu Mary Daniel, Vinodh Ewards S. E.

Department of Computer Science and Engineering, Karunya University
Coimbatore – 641114, India

***Abstract*— Ensuring the trustworthiness in cyber space has become an important and indispensable security challenge. Questions about trust in the physical space can be answered based on the factors namely closeness, time, analysing actions and body language. But in the cyber space these factors are not readily available correctly to ensure and verify trust. In this paper, we discuss the approaches used earlier for establishing trust and their limitations, and focus on the need for hardware-based root of trust as software-only solutions are inadequate to ensure complete trust.**

We discuss an emerging technology in the field of trusted computing, Trusted Platform Module (TPM) that provides a hardware-based root of trust. we also discuss about its scope, various applications, and the future work being done on it.

***Keywords*— Trust, Physical space, Cyber Space, Trusted Computing Base, Trusted Platform Module.**

I. INTRODUCTION

In recent years the general purpose computer has become pervasive and supports a large number of functions. This multiple functionality of a single computer interconnected with various other computers in the cyberspace has reduced the isolation needed to protect security-sensitive operations. The interconnection of multiple computers has also paved a way for easy propagation of malware. When these attacks are analysed and examined it often lead us to the weakness of the operating systems and even the kernel's growing complexity has become problematic. In this era of innovation, OS vendors are forced to implement new features, making the system more complex and prone to bugs, which makes the reliability of the operating system an ever growing concern. If an application runs on top of a compromised operating system then its correctness cannot be guaranteed. To thwart such threats, a system that allows applications to run on a minimal trusted computing base (TCB) [1] is needed, that provides stronger isolation functions.

In earlier days, most of the operating system kernels were implemented on a single entity. Communications between subsystems happened by means of procedure calls and shared memory. Subsystems had no practical boundaries between them, and such systems were said to be monolithic. There were severe problems concerning the robustness and security of these monolithic operating systems as the whole systems control can be gained by attacking any subsystem. As the isolation between applications is insufficient, an attacker can gain access to a single application and from there exploit the entire machine [2]. A monolithic kernel cannot ensure trust as all OS subsystems run in privileged mode without isolation from one another; instead architecture is needed that allows more fine grained access control and applications with a small TCB to be built.

The second generation of microkernels provided a suitable platform for secure systems to be built. But they have a slight overhead when compared to the monolithic systems as they communicate using IPC which requires switching from user privilege level to kernel privilege level and then switching back to user level.

Due to the inabilities of these current operating systems, virtualization has become popular. Virtualization can be used to run applications written for different operating systems concurrently. An efficient isolated duplicate called the virtual machine (VM) is run by a control program called a virtual machine monitor (VMM). As multiple VMs can exist in a time-shared environment a strong isolation is guaranteed where multiple users may own different VMs.

There is not much advantage in moving from a microkernel to a hypervisor as it doesn't contribute much in system trustworthiness. Each virtual machine of a hypervisor runs a complete operating system and does not isolate one application from the other. In essence, the TCB for each application i.e. the set of components on which a subsystem depends must be minimized.

II. BACKGROUND

A. Threat Model

Most of the commercial operating systems assume that the software running on behalf of a subject is trusted. However, these software may be under the control of an attacker and can result in the leakage of the user's data. For example, low level programs may get escalated to high level privileges thereby increasing the possibility of error and malice. This may allow untrusted user code to execute with full super user privilege and may also grant access to user's secrets. The overwhelming number of communications and computations occurring in the cyberspace has to maintain a high degree of trust for their validity. The concept of trusted platforms thus came into picture and many software and hardware based mechanisms were experimented. A purely software based trusted system is not impossible to implement, but they have to successfully overcome many unresolved challenges. Cryptographic keys stored in the hard disk can be accessed, code can be manipulated without leaving a clue and much vulnerability in the software can be exploited by a skilled attacker.

To ensure trust in a system, the TCB should be protected. This will ensure that the processes behave in the way they were intended to. But these processes which are

untrusted interact among each other, and the possibility of secrets being shared increases. As the secrets become available to attackers, they can be further used to compromise other systems which use the same data for authentication.

Once compromised, a system can prove to be a threat to the entire network when connected. To prevent such systems from connecting to the network, there must be a mechanism to identify them as compromised. Software-only authentication mechanisms use the computer name or MAC address which can be forged. As these vulnerabilities increase, there must be a way to detect when a system is compromised and prevent an attacker from exploiting it.

Hence, the concept of a hardware based root of trust was perceived and many promising attempts have already been made. One such attempt resulted in a secure coprocessor, which can be a smart card for simple applications or a completely independent execution environment provided by a separate computer like IBM 4758 [7]. Though it was successful in mitigating side-channel attacks and fault injection, these coprocessors were expensive and meant for securing specific applications. To reduce the expense, virtual coprocessors like ARM TrustZone were developed, which provided two operational modes for the processor called a secure mode and an insecure mode. Despite the fact that it was cheaper and had a separate memory, it still had to get code and data from the untrusted environment and could only run one application in the protected platform.

B. Earlier Approaches

Kernel is the cardinal part of an operating system; it has unrestricted access to all the resources in the system and forms a part of the trusted computing base of a system.

1) Monolithic Kernel: Initially, monolithic kernels were developed which executed the basic system services like interrupt handling, inter process communication, memory management, file system, etc. in the kernel space. Applications and libraries were included in the less privileged user space. Applications used well defined system call interface to interact with the kernel. Both operating system and device driver code share the same kernel memory space. It also provided rich and powerful hardware access. Since all the basic system operations were included in the kernel space, it was not extensible or maintainable and it inflated the kernel size. Kernel recompilation was required for bug fixing and addition of new features [2]. Also bugs in one part of the kernel could corrupt the data structures of another kernel part or of any other programs in execution.

2) Microkernel: By late 1980's the microkernels were introduced, in which only the most basic process communication and input-output control was included in kernel space, while other system services were made resident in the user space as normal processes. The OS was divided into separate processes or servers which implemented a set of services like file server, display server, memory server, process server, etc. Only the microkernel has access to the hardware resources and each application which requires a service from a server sends a message to the appropriate server which is intercepted and forwarded by the microkernel [3]. The result from the server to the application is also intercepted and conveyed by the microkernel. This provides a good level of memory protection and isolation among the user processes. This solved the issue of extensibility and maintainability, by reducing the TCB to the barest minimum system functionalities running in the kernel space, but introduced new difficulties. Due to the basic nature of the microkernel API, it alone is not adequate to run real-time tasks or applications. So system designers had to frequently provide standard APIs to their application developers, which increased the complexity of the operating systems. Because of the multiple interactions of the kernel components running at different privilege levels, communicating by IPC, the performance can also be affected.

3) Virtualization Technology: Soon virtualization technology was introduced. It is a framework that enables the resources of a computer to be shared among multiple execution environments. At the core of virtualization technology is the VMM which provides the foundation for virtualization management. It was used to consolidate workloads of under-utilized servers to lesser number of machines. It provides isolation between untrusted applications through one of the techniques known as sandboxing. It executes multiple operating systems or multiple instances of the same operating system in the same machine in complete isolation [3]. But still, it increases the TCB as the hypervisor code is also included in the trusted base. If it gets corrupted, then the isolation and partitioning cannot guarantee the code in execution.

4) Trusted Computing Base: The concept of TCB was first proposed in 1972 by James. P. Anderson in a study entitled 'computer security Technology and Planning'. TCB refers to the hardware, software and firmware components that provide a secure computing environment and enforce the security policies. If the TCB can be altered, influenced or compromised by any means, it lacks integrity and the system is in an untrusted state. All the security decisions must be made by the TCB using the predefined untampered set of security policies [9], [11]. The TCB must be small enough so that it can be tested and verified. But in most of the modern OS models, this last principle of minimizing the TCB is ignored which has led to serious security breaches. The trusted platform module is a cutting edge technology that aims in minimizing the TCB by providing a hardware-based root of trust for ensuring the integrity of systems.

III. TRUSTED PLATFORM MODULE

The harsh reality that even the most secure conventional operating systems fail to guarantee its genuineness to its users or to the remote entity that it is communicating with, marked the genesis of the Trusted Computing Group (TCG)[4], [9]. With the goal of enhancing the security of computing environment, TCG a not-for-profit industry-standards organization was formed by AMD, Hewlett-Packard, IBM, Intel and Microsoft in spring 2003. TCG has adopted the specifications developed by the Trusted Computing Platform Alliance (TCPA) for better industry

participation, transparent specification development process and documented IP policies. Its mission is to "develop and promote open, vendor-neutral, industry standard specifications for trusted computing building blocks and software interfaces across multiple platforms".

TCG specifications characterize trusted platform into three major components. The core component is a secure crypto processor called TPM, which if enabled, provides much functionality, including secure booting by computing the measurement of the boot sequence after the device is powered on. The Core Root of Trust for Measurement (CRTM) [7], [10], [11] which constitutes the second component is the first code that the trusted platforms execute during boot time and is responsible for initiating the chain of measurement. It forms the first part of BIOS that cannot be flashed or modified. The third component is the TCG Software Stack (TSS) which provides a platform independent software interface to utilize the TPM functions.

TCG developed TPM, a secure cryptographic integrated circuit (IC) which has been included in almost half a billion end products to enable trusted computing. TPM has progressed from its first level over 10 years ago to the TPM 1.2 version [10] today. It was designed to provide a high level of security to establish trustworthiness of the computing platform. Hardware-based credential storage, software integrity, attestation, as well as authentication, or proof of identity, are among the tasks enabled by the root of trust established by the TPM.

A. TPM Components

The TPM 1.2 implementations are stand-alone chips which are soldered on the motherboard of a computer on the LPC bus or integrated into a custom PCB for an embedded device, they communicate with the rest of the system by using a hardware bus. A TPM should support the following core functionalities: secure storage, platform integrity reporting and platform authentication.

TPM is the hardware realization of the TCG specifications. It enhances the level of trust in networks and computing devices. TPM is a passive (slave device that does not control or prohibit the normal execution flow of the system and does not have access to the system resources); opt-in device, which provides privacy-enabling functions when activated. According to the TCG specification [4] it is not mandatory for a TPM to be implemented as an IC. Developers are free to implement this functionality, either in hardware or software.

The I/O block allows data to be transported over virtually any bus or interconnect; it manages information flow between the components and between the TPM and external bus. The flags maintained by the Opt-In block determine the access rights. The non-volatile memory in the TPM stores two long-term keys i.e. the Endorsement Key (EK) and the Storage Root Key (SRK) which forms the basis of key hierarchy. It is also used to store the owner's authorization data (owner's password). The Endorsement Key (EK) which is unique to the TPM is embedded in it. More precisely, a TPM has an endorsement key pair, whose private key never leaves it. The EK pair is provided by TPM manufacturers and stored in the tamper resistant non-volatile memory before shipping the TPM.

The private EK is never used to generate signatures. The process of encrypting data sent to the TPM during the process of taking ownership and the process of creating AIK certificates uses the public EK. The Attestation Identity Key (AIK) regarded as an alias for the Endorsement Key may also be stored within the TPM. Multiple AIKs are supported by a TPM, this helps to maintain anonymity between different service providers who require proof of identity. To make the AIKs persistent, they should be stored in secure external storage. A volatile storage area in the TPM is provided where one or more AIKs can be loaded when in use. The Platform Configuration Registers (PCR) are used to store integrity metrics which measure the integrity of any code, from BIOS to applications, mainly before the execution of the code. These registers are reset on power-offs and restarts. They store 160-bit values which are SHA-1 digests. In TPM v1.1 there are 16 PCRs (0-15), but in the latest TPM v1.2 there are 24 or more PCRs, in v1.1 specification the PCR values will be reset only when the system is rebooted, registers 0-7 are reserved for TPM use and register 8-15 for operating system and application use. While in v1.2 specification there are static and dynamic PCRs. Specifically, PCR 0-16 (static PCRs) will be reset to 0 by a system reboot, thus providing a static root of trust for measurement (SRTM) and PCRs 17-22 (dynamic PCRs) can be reset to 0^{16} without a system reboot or to 1^{16} with a system reboot, providing dynamic root of trust for measurement (DRTM). The Programme Code is the "root of trust" for integrity measurements which is referred as the Core Root of Trust for Measurement (CRTM). The Execution Engine runs the programme code described above.

TPM chip contains a Random Number Generator (RNG) that can seed random numbers to induce randomness in key generation, nonce creation and to strengthen passphrases. SHA-1 Engine is used to generate AIK blobs, computing signatures and for other general purpose use. RSA Key Generation and RSA Engine is used to produce 2048-bit modulus storage and signing keys (SRK and AIKs) using the RSA algorithm. TPM chips will be in ready-to-be-owned state when the devices are shipped. Depending on the user discretion, its state can vary from disabled and deactivated to fully enabled. Opt-in facility maintains the physical state of the TPM and applies the disabling feature to all the TPM components as per the user directions.

B. TPM Functions

The major TPM security features are secure or protected storage in shielded locations, integrity measurement and remote attestation.

1) Secure Storage: A TPM can store secrets securely. As the TPM has limited storage space, it allows to store keys, and other data needs to be protected. This limited storage can be extended by exporting keys in encrypted form (encrypted using SRK or some other storage key), that are decrypted only when loaded back into the TPM. The private key of the SRK never leaves the TPM. Binding and sealing are the two mechanisms provided by TPM for secure storage [4], [5], [10]. Binding refers to the encryption of

data using a key managed by a particular TPM. The bound data inside the TPM can be decrypted using the private key (unbinding). Sealing refers to encrypting externally provided data with reference to a specific PCR state along with a nonce specific to a particular TPM using a storage key. It is a way to combine the measurements (PCR content) and external data. Unsealing refers to loading the key used for sealing into the TPM and decrypting the blob, if the nonce does not match the one of the TPM or if the specified PCR values do not match the platforms current PCR values, it returns error.

2) Integrity Measurement: The process of obtaining configuration parameters of a platform is known as integrity measurement [7], [8]. The goal of integrity measurement is to measure system state into the PCRs [5]. The steps involved in this process are to measure (compute the hash value of) the next entity, e.g. BIOS measures the integrity of the OS Loader, the OS Loader measures the integrity of the operating system and this process continues up to the user level applications. The measurement is made by creating a SHA-1 digest of the code to be loaded (SHA-1(data)) and extended (appends the new measurement to the old PCR value) into one of the PCRs. Measurements change with system updates and patches.

3) Remote Attestation: Attestation provides a current platform state stored in the Integrity Measurement Architecture (IMA) to the remote entity for platform authentication [6][11]. IMA contain the log of software events stored as measurements and extended to TPM's PCRs. Attestation involves a challenge-response protocol. A remote verifier (challenger) sends a challenge consisting of a nonce (to thwart replay attacks), and a list of PCR indices. The response consists of the current PCR values of the listed PCR indices, along with a quote. Quote is a digital signature computed within the TPM, over the aggregate of the list of PCR values and nonce received from the challenger, using private AIK [6]. The challenger upon reception of the response should verify the following:

- The value of nonce in the reply
- Decrypt the quote using the public AIK obtained through an authenticated channel
- Verify whether the list of PCR values matches those included in the quote.
- Verify whether the PCR values itself represent an acceptable and secure boot sequence.

IV. APPLICATIONS

There are a number of reasons why TPM chips are useful. For example, they permit online service providers to verify the platform authenticity through secure booting and integrity measurement, thereby reducing online fraud and identity theft. A website could also be verified by a consumer if it's a legitimate merchant site or not using TPM technology. Hardware-based security provided by the TPM can be used to encrypt emails, and for improving protection for VPN, wireless networks, file encryption and password/PIN/credentials' management. User authentication can be provided by augmenting the device with TPM features likes sealed storage along with normal security practices like finger-print biometrics [5], smart cards [5] and passphrases. Already a transition is well underway to use TPM-based security in mobile devices which access the restricted information in government and other networks. Attempts to incorporate TPMs in VANETs, cloud computing networks, grids are in progress and soon it will be included in security sensitive devices like electronic voting machines and biomedical equipment.

V. SCOPE AND FUTURE WORK

As mobile-phone embedded computers are gaining popularity, with the host of interesting services that they provide including Javascript and interactive web-services, TPM can be deployed for controlling and monitoring these devices. TPM can even be incorporated in tape drives and USB drives. Full disk encryption applications like TrueCrypt, BitLocker drive encryption can make use of TPM to protect the keys, encrypt hard disk and provide trusted boot through integrity measurement. TPM can be added to network devices for authentication of requests, before allowing access to resources. A future home network with Internet-capable devices can be safeguarded using a TPM this will prevent the stealing of the Home Key. The TPM also plays a vital role in the Windows 8 operating system by providing remote attestation by trusted third parties. It provides a trusted boot mechanism called the hardened UEFI BIOS standard. There is a future for TPM in protecting credentials for authentication, where TPM stores credentials of users for different services. Each user will just have to remember a unique access code for a TPM-enabled device, which can then provide the access credentials for all the required services in complete isolation with other user accounts. As TPM cannot mitigate some hardware attacks, plans are underway to release specialized hardware with more tightly integrated TPMs, as the TPM specification does not require it to be a separate chip.

VI. CONCLUSION

Cyberspace has entered the realm of reality; it is no longer a science fiction. Our increased dependency in this virtual world of networked information systems emphasizes the need to ensure trust in these devices. Any kind of disruption in their operation can put life, property and economy at stake. In this paper, we have surveyed the security threats in the cyberspace and have elaborated on the countermeasures to those threats. Having mentioned the countermeasures, we identified that these countermeasures were inadequate to ensure complete trust of an end system. As the underlying security mainly depends on the hardware, the TCG's TPM is capable of providing a hardware-based root of trust, which increases the overall security of the computing devices. Further we have provided a detailed account on the necessity of a trusted computing module and the various application areas it can be applied to. With various innovations taking place in the area of trusted computing TPM still has a long way to go.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support of the faculty in Karunya University towards this survey and analysis.

REFERENCES

[1] Nor Fatimah Bt Awang, "Trusted Computing- Opportunities & Risks," *Collaborative Computing: Networking, Applications and Worksharing*, IEEE, 2009.

[2] Arun Viswanathan, and B. C. Neuman, "A survey of isolation techniques," *University of Southern California, Information Sciences Institute*.

[3] François Armand, and Michel Gien, "A Practical Look at Micro-Kernels and Virtual Machine Monitors," *Consumer Communications and Networking Conference*, IEEE, 2009.

[4] Trusted Computing Group, Incorporated, "TCG specification architecture overview," 2007.

[5] Keith E. Mayes and Konstantinos Markantonakis, *Smart Cards, Tokens, Security and Applications*, ISBN-13: 978-0-387-72197-2, Springer Science + Business Media, LLC, 2008.

[6] Dries Schellekens, Brecht Wyseur, and Bart Preneel, "Remote Attestation on Legacy Operating Systems with Trusted Platform Module," *Electronic Notes in Theoretical Computer* Science 197, Elsevier, 2008.

[7] Bryan Parno, Jonathan M. McCune, and Adrian Perrig, *Bootstrapping Trust in Modern Computers*, ISBN 978-1-4614-1459-9, Springer, 2011.

[8] Junkai Gu, and Weiyong Ji, "A secure bootstrap based on trusted computing," *International Conference on New Trends in Information and Service Science*, IEEE, 2009.

[9] A. Sadeghi, M. Selhorst, C. Stuble, C. Wachsmann, and M. Winandy, "TCG inside?: A note on TPM specification compliance," *Proceedings of the first ACM workshop on Scalable trusted computing, ACM*, pp. 4756, 2006.

[10] Sundeep Bajikar, "Trusted Platform Module (TPM) based Security on Notebook PCs White Paper," *Intel Corporation*, 2002.

[11] Siani Pearson, "Trusted Computing Platforms, the Next Security Solution," *HP Laboratories*, Hewlett-Packard Company, 2002.