Abstract

In this paper we introduce basic knowledge of Trusted Computing briefly. And then, we describe attack towards trusted computing system, and also discuss vulnerability of modern trusted computing system. We will exam two solutions, which can be named under using 'dynamic root of trust', for these problems. Generally, it will change the 'old' architecture of 'static root of trust'-based trusted computing system by leveraging new feature provided by new chips of AMD and Intel. We also summary these solutions with a discussion of their benefits and drawbacks compare with formal methods. Finally, for the drawbacks, we propose some methods to improve the design.

KEYWORDS: Trusted Computing, Dynamic Root of Trust, Minimal TCB

1 Introduction

1.1 Basic Concepts of Trusted Computing

Trusted Computing is a special listed part of the larger subject of computer security [3]. It is a relative new concept whose rudiment was formed around 2000. The main tasks of Trusted Computing are proving the statue that the computer is running in a secure way to a remote computer (Remote Attestation), and ensuring only authenticated entries can access the secret (Seal).

Firstly, we introduce Remote Attestation and Seal in brief:

- Remote Attestation:
  This is a main task of Trusted Computing. One side, such as VPN client on your computer, will prove that it is running in a secure environment to another side, such as VPN server in your corporation. It is always achieved by recording the hardware environment, boot sequence, and host O/S configuration; and then, sending it to the third party to attest that the statue of all hardware and software is secure (no malicious modify, and under control). [4] It will be encrypted by TPM (a part of microprocessor in charge of cryptographic operation, will be introduced later), and transferred with public-key encryption to ensure the correction of the data. The transportation bases on challenge-respond protocol.

- Seal:
  To encrypt a data in a way which it can only be decrypted under the same environment, from both hardware and software aspects. To achieve this goal, the data will be encrypted by the key, which is generated by TPM, binding with platform configuration information. This technique is wildly used in Digital Rights Management.

When these two tasks done, the remote computer can trust the local computer, and run security sensitive program on the local computer. So it is named Trusted Computing. Usually, the security sensitive program is about dealing with secret keys, such as encrypt and decrypt an authentication key in a server. For example, when a client of SSH want to send its secret key to a SSH server, it should confirm that the server side can be trusted. And this point can be achieved by Trusted Computing.

In practice, we generate signature of the state of current environment of computer, and then, send this signature to the remote computer. In this case, we can achieve the goal of Trusted Computing. So the hardware can handle this kind of operations is so important in Trusted Computing field. It is named Trusted Platform Module (TPM). On one hand, TPM is a special part of microprocessor which can handle cryptographic operation. On the other hand, TPM is the specification of Trusted Computing Group (TCG) to define the feature of microprocessor should have to achieve the aim of Trusted Computing. [12] So both these two concepts can be considered as TPM. To avoid confusion, we will take the first one by default, and mark the second one as TPM (specification).

Considering the importance of TPM, it is always an important integrity of Trusted Computing Base (TCB). TCB is "the totality of protection mechanisms within it, including hardware, firmware, and software, the combination of which is responsible for enforcing a computer security policy." [8] To be easy, TCB is a minimal integrity which take responsibility to ensure the security of the whole computer. So when we consider the whole environment of the computer, from booting process to using application finally, we can get components of TCB. Normally, it includes hardware security module (such as TPM), the boot loader, the BIOS, and the operating system. So TCB is relative large nowadays.

At last, we go back to the point of concept of Trusted Computing whit taking a look of the definer of it: TCG. [11] It is an alliance who takes responsibility to standardize the concept of Trusted Computing. TCG achieves this responsibility by developing TPM (specification). Because TPM plays an important role in Trusted Computing, to define the TPM is the best way of standardizing Trusted Computing.

In a summary, the main idea of Trusted Computing is ensuring the computer always works in a secure way and proving it to a remote computer. We have already done these by using Trusted Platform Module (TPM) to generate sig-
nature of the state of current environment. But the Trusted Computing Base (TCB) is still so big that the whole Trusted Computing system is polemics. This is the main problem we are facing now. So in the following sections, we will locate problems and provide some methods to deal with them.

1.2 Static and Dynamic root of trust

Static root of trust is a normal TCB including BIOS, Bootloader, and the whole Operation System (OS), which is considered so large for security request nowadays. So, to reduce the size of TCB is a big issue in Trusted Computing field. Some new designs has made progress by using new feature of processors which are shipped by AMD and Intel: AMD’s Secure Virtual Machine (SVM) architecture; Intel’s Trusted Execution Technology (TXT); and AMD’s processor instruction: SKINIT, TPM v1.2, and PCRs. These techniques enable some pieces of code be executed in an isolated environment. In this case, we can execute the security sensitive code in that isolated environment. So we do not need to trust the normal TCB, but only the code running in the isolated environment. In this condition, the TCB becomes the code running in the isolated environment. And it is always loaded dynamically. So it is named dynamic root of trust. In this paper, we will exam this kind of solutions.

1.3 Other Relative Concepts

1.3.1 Platform Configuration Registers (PCRs)

PCR is a 160 bit register that holds an SHA-1 hash. It only can be updated by using agetextendaf operation. A series of PCRs can record the environment of computer to fulfill the requirement of Remote Attestation. They work like this: The default values of PCRs are '0' for PCRs 0-16 and '-1' for PCR 17. The extend operation, which executed by TPM, will hash the information of loaded software with the old PCR value. In this case, an attacker can not modify the value of PCRs.

1.3.2 Virtualization

A technique to hidden the physical feature of computer resource, and make it appear as some other logical resource to the user. Virtual Machine is a technique to realize Virtualization. And one more step, the Virtualization provided by software layer is called Virtual Machine Monitor (VMM) which can run on top of an operation system. [10] The new security feature of AMD processor, which we will discuss later, utilizes the VMM technique.

2 Background

2.1 New Feature of New Chips

1. VMM and SK

AMD’s SVM and Intel’s TXT techniques can atomically detect and run a VMM or Security Kernel (SK) without rebooting. These two platforms can run some security sensitive code on a more isolated environment.

In this case, it reduce the size of TCB, cause some code outside this environment can be eliminate from TCB.

2. AMD’s processor instruction: SKINIT

It protect and control a Secure Loader Block (SLB) which is defined by a physical start address. It give the SLB to the TPM so that TPM can hash it into PCR 17 (achieve the resetting of PCRs), and then execute from SLB. SKINIT will disable interrupts, access to SLB, and any other access in this process.

By using these new features, we can control TPM and PCRs in order to make a piece of code to be executed in a totally isolated environment. In this case, the TCB of this operation is just these isolated codes. So the size of TCB is reduced dramatically. This is the main idea of designing the solutions which will be introduced in Sec. 3.1.2.

2.2 Boot Sequence

We know that when a computer is just powered on, no OS exists. So, we need to load it from hard disk to memory. The process is by the sequence of BIOS, BootLoader, and OS. After that, we can use applications. In a Trusted Computing System, each component will measure the next component. [13] And TPM will wake up firstly. It will hash the measurement, and store it into a set of PCRs. So PCRs will record all the boot statue of a computer. That is the reason that PCRs will be provided to the remote computer to make verification of Remote Attention and Seal. In a summary, the trust chain will be: TPM -> BIOS -> OptionROMs (Firmware on adapter card, not a big consideration in this paper) -> Boot-Loader -> OS -> Application.

3 Problem Define

3.1 Analysis the Problems

3.1.1 Some implementation bugs

As we discussed the boot sequence above, there are potential problems in the trust chain [2]:

1. Chip can be rested by TPM without restarting the whole system in TPM’s version 1.1.

We can set the reset bit in a PCR by v1.1 TPM. In this case, when the remote computer can not notice that the PCR has been reseted, it will receive a remote attestation information produced by PCRs. So the attacker can provide a designed remote attestation to the remote computer by resetting PCR first. Remote Attestation will completely failed. In the same way, seal also makes no sense. Attacker can generate the expected environment after reset the chip. The paper by Bernhard Kauer [2] describes an example of attacking towards this vulnerability.

2. BIOS is easy to attack by a feature that the CRTM can be exchanged easily.

Core Root of Trust for Measurement (CRTM) can extend PCR 0 initially. It is in the BIOS. It should only
be changed by the authenticated code for basic security requirement. Unfortunately, it is easy to be exchanged in many machines without checking authentication. So, with this vulnerability, attacker can erase attack recording by flash CRTM. It is impossible or very hard to recalculate the hash value after flashing, when a vendor want to check it. In [2], the author provides an attack towards this vulnerability by patching the TPM driver.

3. Bootloaders, which is commonly used, such as LILO and GRUB, are buggy.

A work based on LILO has a bug on it. [2] They use the Mater Boot Record to hash the rest part of LILO, and also hash the loaded Linux kernel image. But only the last part of it. So the problem is that the first part is missing. We can solve it with hash all the images of Linux kernel.

Another work, which is based on GRUB, from IMB Japan [2] also has bugs. The way of hash for this GRUB is to load file twice, first time for extraction while next time for hash. It leads to the problem that when GRUB loads the code for the second time to hash it into PCRs, attacker can provide another code to GRUB. So the PCRs will not be correct. And another GRUB based work Trusted GRUB [2] solves this problem with hashing the code into PCRs when the code is just called. But it still have two bugs. One for self-hash, the other for boot from LiveCDs. And no currently work could completely fix all bugs in Bootloaders so far.

3.1.2 Size of TCB

We just located some bugs from booting point of view. On the other hand, the reason of booting problems can be considered as dependent on a so big TCB, which is a long trust chain including buggy TPM reset function, BIOS, and Bootloader. So removing these bugs can be done by reducing the size of TCB. Now, we focus on this ’fundamental’ problem.

The TCB is relative big in commodity modern PC system, which includes TPM, BIOS, bootloader, the whole OS. Especially, the OS is getting larger and larger. And even the size of VMM or SK (reference to new feature of TC chips) is still not small enough. And also, there is potential thread from the OS itself. It can not make sure that millions of lines of code are bug-free. And with the TCB including OS, we also have a problem that it is hard to provide a exact attestation to remote part, because the attestation (often PCRs) will content so many non-security-sensitive information, and disrupt the exam of the true security-sensitive code. So the remote computer will find that it is so difficult to get the information they really need to make sure the other end is trust-worth. In this situation, the remote attestation makes no sence. At the same time, a large remote attestation will also leak information of rest part of the system. Maybe, it will leak your privacy information to an attacker who pretends to require a remote attestation.

So, to achieve secure level ordered by TCG, we could not still trust a big TCB including OS any more. When we aim to reduce the size of TCB, the most effective way is to remove the OS and VMM or SK from the ’old’ TCB (which is called static root of trust).

3.2 Goals

To handle the problems above, solutions should be with these features:

No bug allowed: To solve all bugs which had been defined.

Minimal TCB: To rely on the minimal amount of code to achieve security sensitive operation.

Well provable protection: To convince a remote computer that the security sensitive code is executing in a protected environment. And make it easy to be analyzed by the remote computer.

These goals lead the research of the following solutions.

4 Solutions

We will introduce two solutions provided by two research papers [2, 6]. First one deals with the booting problem, while the second one focuses on minimizing TCB. The booting problem can not cover the whole system. So we introduce another architecture [7] dealing with operation system and application to extend the research of booting problem.

4.1 OSLO

We have noticed that the resettable TPM, BIOS, and BootLoader in the trust chain of Trusted Computing system is not trustworthy. From the basic idea of reduce TCB, we could eliminate these process from trust chain. Fortunately, with the feature of new chips from AMD and Intel, we can achieve this goal. These features, which have been introduced in the frontwards paragraphs, includ processor instruction: SKINIT, SLB, and TPM v1.2, and PCRs are called Dynamic Root of Trust for Measurement (DRTM).

It solves the three problems we have mentioned:

1. Chip reseted

The manual resetting of the PCRs will get a default value ("0" for PCRs 0-16; "-1" for PCR 17), while DRTM will only set PCR 17 to "0", and then extend it with hash of SLB. In this case, the TPM can tell the difference between a malicious resetting and a DRTM request. And attacker can not hash PCRs, because SKINIT jumps directly to the SLB. So the attacker can not change the PCRs to fit their purposes both in the initial point and the normal process.

2. BIOS attack and Bootloaders attack

With the DRTM, we can remove the BIOS, Option-ROMs, and Bootloaders from the trust chain of booting. It will be like this: TRM -> OSLO (a little program which is implemented by using DRTM) -> OS -> Application. So we do not need to consider the risk of BIOS and BootLoader, although we will still take the System Management Mode (SMM) code and correct ACPI tables into this design. They will be hashed into PCRs.
The implementation of this work is described as following:

The implementation is name by Kauer with Open Secure L0ader (OSLO). With using OSLO, he disables BIOS call to remove BIOS from the trust chain. He designed the process as following: Firstly, OSLO initializes the TPM, and enable it to extend PCR; Secondly, OSLO will stop other processors; In the third step, OSLO will run SKINIT to take control of SLB and PCRs; At the same time, just before the new module is started, all modules have been loaded in the first step should be hashed, and stored into PCRs. In the whole process, we should not enable BIOS. So Kauer developed an own TPM driver to communicate with TPM, which is only 70 lines.

And there are two features are still missing in Kauer’s implementation: 1. Prevent Direct Memory Access (DMA) from this process. 2. extend the event log of TPM to make Remote Attestation. The first feature can be achieved by using SKINIT to manage the SLB and PCRs. It can prevent DMA access to the SLB. And the second feature can be considered as a help or backup of PCRs. We can make Remote Attestation by PCRs. And also, for the backup or further certification, we can use event log of TPM which record the hash of input which is used to extend PCRs. Because it is impossible for attacker to make Remote Attestation by event log. After this feature is implemented, remote user could check every input to TPM with this log. So it performs a good additional to the Trusted Computing system. But, by now, these two features have not been implemented so far. We can keep tracing for the publish page [1].

4.2 Nizza

We just introduced the architecture to deal with booting problem. But when a big OS is running, the problem of big TCB still exists. To reduce the size of TCB after booting process, we will introduce architecture: Nizza implemented by Singaravelu et al. [9].

Nizza includes architecture of OS and applications running on it. It is NOT based on the features of new chips. The main idea of this architecture is that extracting the core of application (AppCore) which is related with security operation, and executing it in the kernelized TCB (often done by extracting the kernel of OS).

To get the AppCores, Singaravelu et al. made three steps: 1. Analysis the application to locate what are the security-sensitive parts of this application; 2. Extract the security-sensitive parts, and compose them into an AppCore; 3. Modify the original application to make sure that it will only use AppCore to do the security-sensitive tasks. For detail of these three steps, please read [9].

For the secure kernel of TCB, Singaravelu et al. selected a ‘small kernel -> execution environment -> system and application level’ - architecture which they call it Nizza. In the first level, for the small kernel, they used L4 microkernel [5] which can make sure component isolation is in protection domains. In the second level, the execution environment refers to the services to the functioning of the system. It will contain a name server and resource management of memory, CPU, and I/O. It also could include other components, such as GUI, depending on the need of applications. And for the third level, the small kernel and execution environment can support AppCores as well as entrusted OS with legacy code.

In a summary, we can only run the security-sensitive code in a secure platform by this architecture. But this architecture still needs an insurance of the process of booting. So it is can be combined with OSLO to make sure the whole process of running a Trusted Computing system is secure.

4.3 SEA

McCune et al. published an architecture [6], which is regardless of the affect of booting process and OS, to reduce the size of TCB. It is called Secure Execution Architecture (SEA). It enables the security-sensitive code to execute in an isolated environment (from both software and hardware aspects). Fig. 1 So that the TCB is only the code running in the isolation environment. So the size of TCB is reduced heavily, comparing with the TCB which includes OS.

And then we can see how McCune et al. execute a small piece of code which is called Piece of Application Logic (PAL) in this architecture. In this architecture, the VMM or SK, which is provided by the new shipped chips, is still
too large to be trusted. So when the SKINIT is called, they will give it PAL as parameter, so that PAL will be loaded instead of VMM or SK. We can consider the SKINIT as an interrupt with the highest level. So this interrupt can not be interrupted by other process until it terminals. (SKINIT will disable interrupts) And then, we show the ‘interrupt server program’-like three steps to execute the PAL in an isolated environment.

1. Invoking the PAL

They store the state of the current environment in a location which is easy to find and handle. This is defined by [6] as follow: ‘the base address of the page tables, global and local descriptor tables (if present), interrupt descriptor tables, the task register contents, extend feature register (EFER) contents, and certain bits in the EFLAGS register.’ And for a multi-CPU system, bootstrap processor (BSP) is invoked to execute SKINIT. OS should deschedule all application processors (APs) and distribute an INIT inter-processor-interrupt (IPI) to each one so that they can be rescheduled later. And then, the the process of SKINIT can be executed. PAL is isolated from the OS, then.

2. The Secure Execution Environment

Following, the SKINIT will reset the PCRs 17-22 and extend PCR 17 whit PAL. And then, a tiny code, which is named shim, will manage PAL. It will extend PRC 18 with input parameters of the PAL and jump to PAL. When the PAL finishes, the process will jump back to the shim. The whole process will be protected by the feature of SKINIT described in Sec. 2.1.

3. Resuming the OS

After the execution of PAL, the shim will erase all traces of the PAL. And then, it will extend PCR 18 with output of PAL. The next step is to extend PCR 17 and 18 with the single of termination of PAL. By now, PCRs are suppose to have recorded all the states of PAL’s execution, including the shim (records load and exit process); so they can prove that PAL had run in an isolated environment. With this feature, they can be used to Remote Attestation. Finally, the shim restores the formal OS from the location of record file. And OS can reschedule the processors.

The whole process is like a normal interrupt. But this one can not be interrupted by any other process.

And there are some extensions to this design by McCune et al. to make further functionalities:

- **Attestation:**
  We can use PCRs to achieve Remote Attestation. And we will discuss the framework of the whole attestation providing system. The verifier sends a request of PCRs 17 and 18 with a nonce which provides freshness and replay prevention of verifier to the TPM. TPM signs the nonce and PCRs 17 and 18, and sends to the verifier. The verifier can check the authenticity by TPM’s Attestation Identity Key (AIK), and then compares the PCRs 17 and 18 with the expected value.

- **Multiple Invocations:**
  Some PALs tend to be invoked several times, such as creating a public key and establishing secure channel by this key. There are two ways to do so: 1. PAL secures data with seal it with value ofPCR 17. When this PAL is invoked next time, SKINIT will reset RPC 17 and extend it with PAL. So the PAL will get the same environment of the previous invocation. Then it can unseal the secret data. 2. To make different PAL (2) to access the secret value of PAL (1). We can reset thePCR 17 and extend it with hash value of 2. And then, we can seal the data of 1 whit PCR 17 so that 2 could unseal it when it is invoked.

- **Secure Communication:**
  When the remote party wants to communicate with the PAL, we still can do this in a secure environment by using the multiple invocation. It still isolates the OS and other components of computer. For detail, please check [6].

5 Analysis

5.1 OSLO and Nizza

Goals: OSLO removed the bugs successfully. And it also reduced the trusted chain in the Trusted Computing system, so it can be considered as reduced the TCB. And with the OSLO, Nizza can ensure the whole system running in a secure way. For the other point: attestation, I didn’t find this function in the log file of OSLO, so far. But it should be easy to achieve with the DRTM. SKINIT can handle it very well by providing PCRs to the remote computers. And with the unimplemented feature ‘event log of TPM’, it will be a good Remote Attestation metheod, too. And, basing on the reduced trust chain, the attestation should be exact. But the Remote Attestation is still limited in the booting procedure.

Compare with formal work: OSLO utilizes the DRTM to shorten the trust chain. Comparing with the vulnerable trust chain, it makes a good progress. And it could provide more exact Remote Attestation, comparing with hash the whole kernel image to do the Remote Attestation. But, on the other hand, we could lose some features, which are provided by the BIOS and BootLoaders, when using this security-orientated component to boot computer. So I don’t think this way to boot computer will be used in the real product until it develops to a multiple functional tool for booting. And for the Nizza, it reduced the size of TCB dynamically. The key point of this architecture is the metheod of extracting the AppCores. When it can be done effectively and correctly, the usability of the Nizza will be imporved.

5.2 SEA

Goals: SEA removed booting problem totally. And it reduced the TCB dramatically by removing OS from the TCB. In this implementation, the TCB are only the tiny shim and the PAL. It is an excellent design to fulfill the goal of minimal TCB. And it also provides exact Remote Attestation.
The PCRs only include the security-sensitive code. So, basically, all goals are achieve by the SEA.

Compare with formal work: SEA only takes PAL and the tiny shim code into the TCB, which is a big progress of Trusted Computing design. It can remove all the consideration of OS and application security issues from the Trusted Computing. By the supporting from SKINIT, it eliminates hardware affect from the system (still can not handle highly sophisticated hardware attack). But with such a small TCB, the privileges will be assigned to it totally. It will occupy CPU as long as its execution. So the PAL becomes the most important issue in this situation. Firstly, PALs can’t be so large that it holds CPU for a too long period. This problem can be solve by cut the long-running PAL into small pieces, and connect of them by applying the multiple invocation technique. Secondly, PAL becomes the aim of attackers, because PAL holds the ability to crash the whole system. It can use all of the CUP resource as it like. So it is really dangerous. From these two points of view, hard to control of PALs is the most significant drawback of this design, comparing with formal work.

5.3 Compare OSLO with Nizza and SEA

The same technique is used in different level of designs. Both of OSLO and SEA are using the series of the new feature of AMD’s processor instruction SKINIT to isolated execute code. But OSLO uses it with booting, while SEA uses it in the whole system design. OSLO handles booting problem very well, which has been eliminated by the SEA. So SEA can deal with more problems what OSLO can’t. But with the help of Nizza, the whole system can be trusted. But the size of TCB is still bigger than the SEA’s. But for Nizza, it can easily control the privilege of AppCores. So, comparing with SEA, Nizza takes less risk while SEA can’t control PALs very well. However, we still prefer to use SEA to provide a TCB of a Trusted Computing system. Because it utilizes more streamlining components to achieve the same goal. This can be considered as a progress of the normal architecture. At least, we still can combine these methods to achieve a better solution. Especially, the idea of "event log of TPM" from OSLO can be introduced into the SEA design so that SEA can provide more sufficient Remote Attestation by integrate PCRs and event log of TPM.

6 Problem and Solution

The most important problem of these techniques should be the privilege of PALs, which has been located before. There is a similar discuss in [6]. It provides some techniques to deal with the problem of trusted PALs. One is to analysis the size of PALs, another is that OS requires PALs to proof its safety, and also, we can dynamically control the privilege of the PALs. The method of analysising size is not so exact: a death cycle without so much code could easily crash the system, so it can not be a final solution. The second one will include OS into the TCB again. It will cause more problems. The third one can only limit the damage of PALs, and also introduces segmentation and/or page table permissions into the TCB. So it is not a good solution.

We need a solution that will not increase the size of TCB so significant. So we should better only adjust the component current in the TCB: shim and PALs. The point is on the shim. Firstly, the PALs always dealing cryptographic computation, we can make a rules of PALs, and use incremental shim code to exam it. Although it still introduces a set of rules into the TCB, the increment of TCB is not so dramatically. For the PAL which fails to pass the exam, we will not run it. But we can still run it with its original application, and make a notification to the system administrator. Of course, the Remote Attestation will not include the statue of the PAL running. And the remote computer will get this information.

7 Related Work

SEA can also be used in the client part to protect the sensitive information of user which is request by the applications, such like web browsers. And there also are other reports about application and operation system of trust [9]. This research can support the OSOL and Nizza architecture.

8 Conclusion

We exam two implements in Trusted Computing system using the new features available on the new chips which is shipped by AMD and Intel. Both of them reduced the TCB successfully. Our suggestion is combining these two method together. There still are problems in the combined solution. So it needs more work on it, both from Trusted Computing system design and hardware design.

References


