Research Report for Secure EL2

authors in alphabetical order

Abstract

Some sorts of documents need abstracts. Others do not.

1 Introduction & Background

Under data explosion adn data becaming treasures, the cloud computing is valued and being developed to store and dig the data mine. In order to meet the various needs of cloud computing, virtualization has become an important technology of cloud computing. In the meanwhile, the securtiy of cloud is also to be taken more serious. Lots of techniques is used, including TEE.

Trusted Execution Environment (TEE) is a security domian in CPU who is isolated from the other, i.e. REE. TEE is capable of providing confidentiality and intergity for the codes and data stored in it. All accepted code have been authorized by TEE so that it is trusted. Besides, TEE need all code and assets load and start in a expected way so that it won't be tempered. A TEE should also responsible for isolation between TEE and REE. In addition, each TA can only access it own assets. A TEE is a important composition of trusted computing. Famous TEE includes TrustZone(6), SGX (10), SEV(7), and etc.

1.1 TrustZone

TrustZone (6) is the first commercial hardware architecture to support TEE. After development in these years, TrustZone is widely used in mobile devices to provide trusted computing. TrustZone seperates the hardware and software resources, such as memories, I/O, and interrupts, into two parts, Secure world and Non-secure world (i.e. Normal world). Each world has their own kernel space and user space, associated with cache, memory, and etc. Non-seucre world is not capable of accessing the resources in Secure world, but Secure world has access rights to resources in Non-secure world. In addition, each world has its own privileges level, represented in exception levels. Note that there is only Non-secure world having exception level 2 (EL2). Correspondingly, processors have two state, Secure state and Non-secure state, to handle affairs in two different worlds.

1.2 Secure EL2

In Armv8.4-A, Secure EL2 (4) is introduced as an optional feature. Virtualization is used to be in Normal world for there is no Hpy mode, the same with EL2, in Secure world. When the Secure Virtualization enable, the platform firmware in EL3 can be move to seperate partitions in EL1, so that the code in

EL3 can be minimized. Secure Partion Manager (SPM) manages these Secure Partions (SPs). If EL2 in Secure world is supported, the processor needs use SCR_EL3.EEL2 bit to enable it.

1.3 Hafnium

Hafnium (3) is an open-source software for manage the Secure Partion.

1.4 Virtualization of TEE Technology

Virtualization is a technology to divide the hardware resouce and create virtual instances who act like a real system. Since 1960s, virtualization has been always popular in computer science. The trend of virtualization is more intense these year, especially after cloud computing becomes more and more popular.

However, TrustZone is not designed to be virtualizable in the past few years. All virtual machines managed by the same hypervisor shoudld trust the same Trusted kernel, while there are amount of vulnerabilities discovered in major venders' trusted kernel. Besides, some manufacturers would like to use their own Trusted OS instead of other vendors'.

Some TEE supports virtualization at the beginning, such as SEV and TDX (8). There are new kinds of TEE came out. confidentiality

The arrangement of the report. Firstly talk about the software-simulation base virtualization(vTZ). second hardwared-assisted base virtualization(not SEL2)

2 Hypervisor

2.1 KVM

vTZ is proposed by Christoffer Dall and Jason in 2014 on ASPLOS. As ARM CPUs become increasingly common in mobile devices and servers, there is a growing demand for providing the benefits of virtualization for ARM-based devices. At that time, ARM Xen has already been proposed, but there are some disadvantages on ARM Xen. For example, ARM Xen requires much maintenance effort since it is a hypervisor interacting with hardware by itself. So it needs to write its own code to be compatible with different ARM-based platform. However, KVM does not have this disadvantage. KVM is just a kernel module of Linux. It only provides hypervisor-related functions in KVM code and gives Linux full privilege to interact with hardware. Since Linux is implemented on all ARM-based platforms, KVM can be adapted to all ARM-based platforms by a good-defined API between KVM and Linux.

The reason we try to understand KVM design is that in S-EL2 project we need a hypervisor running in NWd to build the whole model. And the preferred choice is KVM. Understanding the design and implements can help us to build the model and the following modification on KVM.

2.2 HypSec

HypSec is proposed by Shih-Wei Li, John S. Koh, and Jason Nieh in 2019. HypSec is a new hypervisor design scheme. With HypSec, we can retrofit an existing hypervisor. After retrofitting the hypervisor, its TCB will decrease with

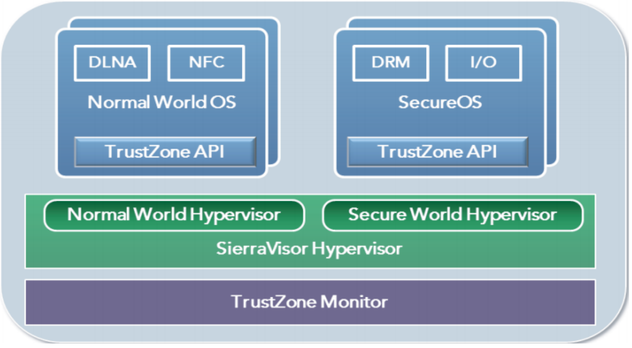

Figure 1: SierraVisor Architecture Overview

serveral orders of magnitude and its performance overhead is still acceptable. In conclusion, this is a hypervisor design pattern which can improve security of a hypervisor with a cost of small overhead.

The reason that we read this paper is that there are two hypervisors in our S-EL2 model. A normal world hypervisor like KVM in NWd and a secure world hypervisor like Hafnium in SWd. In future we need to build a cross-world communication channel between these two hypervisors. And then there is a problem, the TCB might be too large. We might need to include KVM, hafnium and hafnium code lines into TCB. The TCB can be over millions lines of code, which is not acceptable. With HypSec, we can reduce KVM and hafnium's TCB by a several orders of magnitude, so a much smaller TCB can be gained.

2.3 SierraVisor

Proposed by Sierraware, a company dedicated to deliver the virtualization technologies and security solutions, SierraVisor is a hypervisor for ARM which is integrated with TrustZone and Android. Specifically, SierraVisor supports TrustZone Monitor as VMM on Cortex A9 and ARM11, i.e. SierraVisor use TrustZone Monitor and extend it as a hypervisor. The guest can continue to work without modification. And kernel can continue to run in supervisor mode. Guest OSes can run in their inidividual containers.(1)

Above is the architecture of SierraVisor. At the bottom there are TrustZone Monitor and SierraVisor Hypervisor. The SierraVisor has two hypervisor, normal world hypervisor and secure world hypervisor running in each world. Both hypervisor supports multiple OS in Normal World and Secure World.

As a product, SierraVisor is quite old because the supported SoC presenting on the their website is up to Cortex A15, a processors implementing the Armv7-A architecture. The timestamp of most web pages about SierraVisor freezes in 2012. But the idea of SierraVisor and its contribution on supporting multiple TEE-kernel is on the right direction.

(advantages: high compatibility, type-2 hypervisor, as long as linux support, mine supports.)

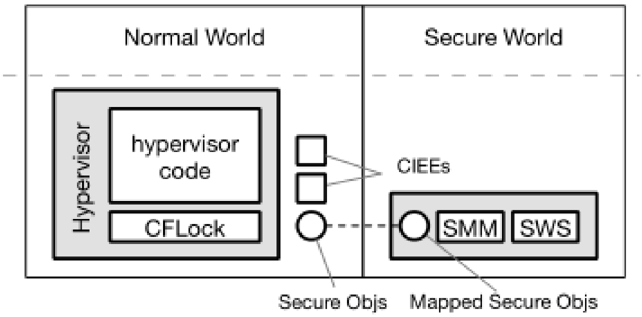

Figure 2: vTZ system designs

3 TEE Virtualization

The TEE virtualization is to create a virtual TEE and simulate a part of, even all of TEE functionalities in a software way. The important component of TEE, such as interrupt controller and memory controller, is achieve by the software. The represensative is vTZ.

3.1 vTZ

vTZ is proposed by Zhichao Hua in 2017 on usenix conference. TrustZone does not provide any hardware support for virtualization at that time. All VMs in normal world should shared the same TEE and it leads to a serious security issue for breaking the isolation of VMs. vTZ aims to virtualize TrustZone and provide each VM an independent TEE. Different from other jobs, it make use of the existed TEE architecture instead of design a new one. vTZ seperate the funcutionality and security by running another VM in normal world as guest TEE to serve the VM. The security is ensure by the TrustZone hardware isolation. In details, a tiny monitor in secure world control the memory mapping and world switching. A Constrained Isolated Execution Environment (CIEE) provides isolation.

The design of vTZ is shown in figure. There are four secure module in vTZ, Secure Memory Mapping (SMM), Secure World Switching (SWS), Control Flow Lock (CFLock) , and Constrained Isolated Execution Environment (CIEE). Running in secure world, SMM controls all memory mappings. To achieve that, the hypervisor is not allow to contain the sensitive instruction to access and control the translation table, not to mention the first level translation and second level translation. Instead, the sensitive instruction is replaced to the invocation to SMM. SMM will check whether the request is legal. SWS is responsible for the world switching and it is another module in secure world. It will check the VID of the VMs and restore the correct VM. As long as SWS ensure the correction of the switch from VMs to the hypervisor and the hypervisor to VMs, all switches in vTZ will be covered, because switches between VMs are composed of both switches. As a module in normal world hypervisor, CFLock is used to prevent the execption control flow from tempering by protecting exception vector table. CFlock can used to ensure SWS will eventually handle all switches. The last module is CIEE, a region in normal world EL2, running with the hypervisor, serves the guest-TEE. CIEEs contiain the software implementations of TrustZone. In other words, vTZ does not utilize all the hardware in considerations of the support of the virtualization. To protect the isolation and security of CIEE, CIEE is designed to execute atomically and independently. The CIEE itself is also verifiable. As a result of it, CIEE can't be falsified.

Base on these four modules, vTZ reaches the goal of secure boot, memory isolation, and privilege isolation.

The advantages of vTZ is high efficiency and high security. In vTZ evaluation, the performance overhead of applications is low compared to the native environment, TrustZone, and Xen hypervisor. For security, vTZ has a TCB of thousands lines of code, which is far less than one who putting a entire hypervisor and guest TEE into secure world. And vTZ utilizes TrustZone to protect itself. In details, the privilege mode and secure world of TrustZone make the isolation and security of CIEE, SWS and SMM. In a result, vTZ successfully defenses several kinds of attacks, such as code-reuse attacks, DMA attacks, debugging attacks and code tempering attacks.

However, vTZ also has some disadvantages. vTZ is not compatible to the exisiting commercial hypervisor. In vTZ, the hypervisor is not allow to have sensitive instructions related to address translations. In addition, the module, CFLock should run in hypervisor to support vTZ. So the hypervisor must modified their source code in order to run on vTZ. That is a huge disadvantage to the promotion and application.

In conclusion, vTZ is mostly close to our goals Secure EL2 project. vTZ is also a close competiter of our project. We want to compare the performance with vTZ and proof the advantage of our one.

3.2 TEEv

TEEv is another TEE virtualization architecture to provide restricted TEEs, i.e. vTEEs. These vTEEs are concurrent in runtime and isolated from each other. The vTEEs is not like traditional TEEs who can access the REE's resources. They can only access their own assest(More about vTEE). A tiny hypervisor, TEE-visor, running in Secure World to manage these vTEEs and to provide isolation between vTEEs. TEE-visor allows each vTEE from different manufacturers to host their own TA.

To address the challenge of the lack of TrustZone virtualization support, TEEv deploys the TEE-visor and vTEEs at the same privilege level, secure EL1. In addition, TEEv modified the commercial TEE by scanning vTEE's binary to remove the MMU-related instructions, so that TEE-visor can exclusively control the MMU. Controling the MMU, TEE-visor can protect itself in security and confidentiality, and isolate vTEEs by protecting each vTEE's entries. As a result of removing the sensitive instruction, TEE-visor is responsible for the communication between vTEE and corresponding client application (CA). In details, TEE-visor handles the request from CAs and verify the page access from TA to CA. For isolation of vTEEs and REEs, TEE-visor manage the page table exclusively in order that each vTEE and each REE can only access their own page. Similar to vTZ, TEE-visor protect the kernel of vTEEs by mapping their kernel's page as read only.

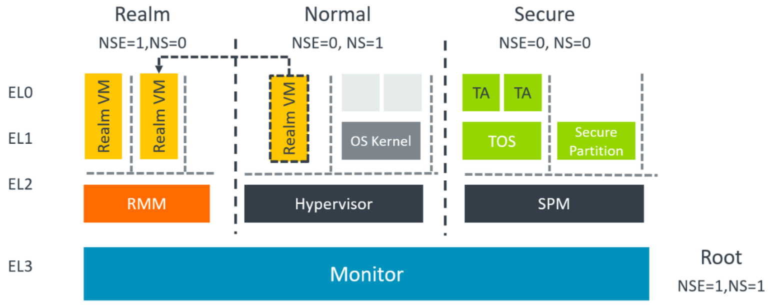

Figure 3: CCA System Overview

The advantage of TEEv is high security and high flexibility. The vTEE in TEEv is not a traditional TEE and it is restricted. If a vTEE is malicious, it can not affect the other vTEEs and REEs. Its TCB is also low. The TEE-visor only contians 3800 lines of C and assembly code. Each TEE should only trust their own code base and TEE-visor. (CVEs and attacks) For the flexibility, TEEv allows to install multiple vTEEs and support each own TA.

3.3 CCA

With the Armv9 architecture (2) came out, Confidential Computing Architecture (CCA) (5) is introduced to the people at the first time. The aim of CCA is create a execution environment, Realm, who can be created from Normal world without inherent the trust from the creators. Note that CCA could create Realm to run a virtual machine, Realm virtual Machine (RMM), in it, and this Realm virtual Machine can be manage by Normal world hypervisor. Although Realm Virtual Machine is scheduled by the hypervisor or kernel, CCA does not need to trust them to gain the confidentiality, even the trusted code in Secure world. CCA could stablishing the trust of Realm through Attestation.

CCA is different from TrustZone. The Trusted OS run on TrustZone is part of a chain of the turst. It depends on the high privilege firmware, such as SPM and Secure Monitor. There are two ways entering Trusted OS, either the yielding from Rich OS or the secure interrupt. Both may cause a world swithcing. However, the Realm Virtual Machine is managed by Normal world hypervisor. RMM just like normal VMs, being create and being allocated memory via the Host. Different from TOS, RMM may not be executed when the host is broken. In conlcusion, the software running in TrustZone is with confidentiality, intergity, and Authenticity. The software running in Realm is with confidentiality and intergity.

4 Design

Figure 4 describes the whole model of our project. Arm-trusted-firmware(ATF) is running in EL3, which handles secure booting and secure world switching. The cross-world communication channel is handled by ATF. In normal world, the layout is the same as the cloud server providers' layout. A normal world hypervisor, such as KVM, runs in EL2. The guest OSes are running in EL1, which

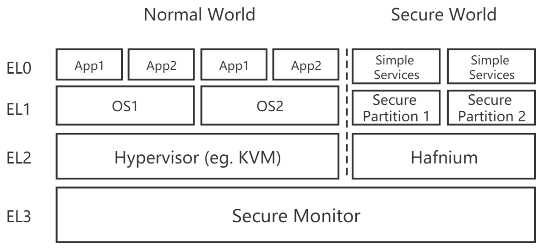

Figure 4: Whole Model

provides services hosted in EL0 to cloud users. In secure world, a hypervisor called Hafnium (9) is running in secure EL2, which supports multiple secure partitions (SPs) in secure EL1. Secure Partitions, running in secure EL1, are actually multiple bare-metal trusted guest OSes. Each of them occupies only a small block of memory, for example 1MB, providing trusted services for the normal world software. Since trusted services are always simple and their design purposes are always limited to encryption, secure storage and so on, such a small block of memory fulfills their requirements. There are two main challenges in our design:

- Large trusted computing base (TCB) if not handled carefully
- Binding of virtual machine with its associated SP

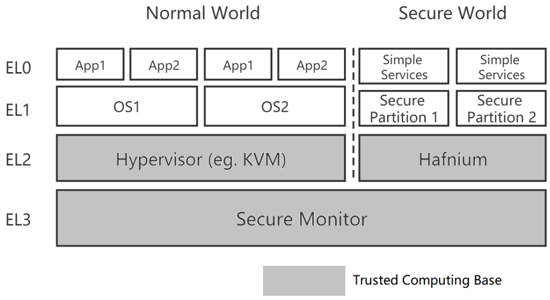

Figure 5: TCB before Modification

As for the first challenge, it is caused by the reason that the management of VMs is done by the two hypervisors, one in normal world and another one in secure world, in cooperation. Hafnium, the hypervisor in secure world, needs the information delivered by normal world hypervisor to deploy security policies. If the normal world hypervisor is compromised, it can cheat Hafnium to allow a virtual machine in normal world to ask for services from a SP which is not bound to itself. For example, assume there are two VMs in normal world and two SPs in secure world. VM1 is bound to SP1 and VM2 is bound to SP2. Now VM1 is going to ask for trusted services from SP1, so it sends message to normal world hypervisor, in this model KVM, that it wants to use trusted

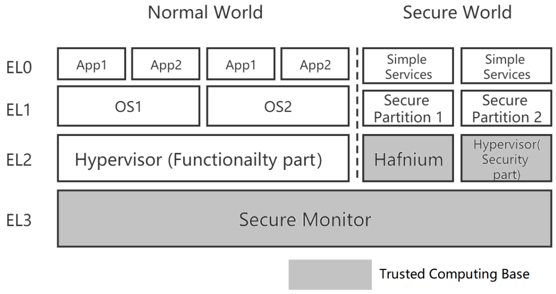

Figure 6: TCB after Modification

services hosted in SP1. However, a compromised KVM can forge the requirements as VM2 is asking for trusted services of SP2 and sends the request to Hafnium. Hafnium has no ability to verify whether the request is true, so it has to provide SP2 services to KVM. KVM can then provide the SP2 services to VM1, which completes an attack. Since the security of the whole system depends on the security of the normal world hypervisor, the code base of the normal world hypervisor must be considered when computing TCB as 5. To solve this problem, we refer to the idea mentioned in vTZ, which puts forward an idea called "Function-Protection-Separation Design Principle". It means, we separate the functionalities of the normal world hypervisor into function-related functionality and security-related functionality. The security-related functionality, like sending requests to secure world, should be moved to secure world as secure world is trusted in our threat model, while the function-related functionality remains in normal world to avoid a large TCB. Control flow locking (CFL) is added to the code in normal world hypervisor to make sure it must invoke the specified code residing in secure world. As described in Figure 6, after adopting the design principle introduced by vTZ, the TCB only includes Hafnium, SPs and the security-related functionality of the normal world hypervisor, TCB is decreased by several orders of magnitude. For the second challenge, it can be solved in a very simple and direct method. In our threat model, we assume that the normal world hypervisor is trustable during initialization process. In ARMv8 architecture, each guest OS will be assigned a VMID when initialization. We can make normal world hypervisor passed the VMIDs of the guest OS to Hafnium during initialization process. So Hafnium has the VMIDs of all guest OSes and SPs. Then every time a guest OS requests for trusted services from its associated SP. Hafnium will check whether the VMID of the guest OS and the SP matches. If matched, Hafnium can provide the services to the guest OS, and denies the request otherwise.

5 Discussion

we discussion the differences between S-EL2 and related works. List at least three novelties.

Our project Secure EL2 has flowing three novelties compare to existing jobs, performance, compatibility, security. Seucre El2 has a better performance com-

	vTZ	**TEEv**	**CCA**	**SecureEL2**
Architecture	Arm	Arm	Arm	Arm
modify code in TEE (EL0)	yes	yes	no	no
modify code in TEE (EL1)	no	no	no	no
modify normal world code	yes	yes	no	yes

pared to vTZ and TEEv. There are two main reasons. vTZ will have a world swtiching as long as hypervisor wants to access page table, becuase SMM in Secure World controls the exclusively controls memory mapping and the sensitive instruction to access page table in Normal World hypervisor is replaced to the invocation to SMM. It is not that works in Secure EL2. Secondly, the switch between VMs and VMs, vTEEs and vTEEs will also lead to a world switching, since the monitor, who is called SWS in vTZ and TEE-visor in TEEv, will verify the legality of switches. But Secure EL2 needn't do that. Becuase the switching between VMs and VMs is controlled by Normal World hypervisor, usually KVM, and the switching between vTEE and vTEE is controlled by SPM, usually Hafnium. Both are not related to world switching.

Another novelty is compatibility. A important advantages of Secure EL2 is that it make least impacts to the software running on Secure World. To meet the requirement of exclusively control the memory mapping, vTZ and TEEv make alterations to restrict code on EL1 and Secure EL1, manually or automatically. Although VMM in Secure EL2 may be modified to establish the communication to SPM, but it is much less than vTZ.

6 Conclusion

The conclusion about the research report.

References

[1] 2014. Sierraware Overview. `https://www.sierraware.com/sierraware_tee_hypervisor_overview.pdf`.

[2] Arm. 2021. *Arm Architecture Reference Manual Supplement Armv9, for Armv9-A architecture profile* (a.a ed.). Arm.

[3] Arm. 2021. Hafnium Architecture. `https://hafnium.googlesource.com/hafnium/+/HEAD/docs/Architecture.md`.

[4] Arm. 2021. *Learn the architecture: AArch64 Virtualization*. Learn Guide. Arm.

[5] Arm. 2021. *Learn the architecture: Introducing Arm Confidential Compute Architecture*. Learn Guide. Arm.

[6] Arm. 2021. *Learn the architecture: TrustZone for AArch64*. Learn Guide. Arm.

[7] Tom Woller David Kaplan, Jeremy Powell. 2016. AMD MEMORY ENCRYPTION. https://developer.amd.com/wordpress/media/2013/12/AMD_Memory_Encryption_Whitepaper_v7-Public.pdf.

[8] Intel. 2020. *White Paper — Intel® Trust Domain Extensions (Intel® TDX).* white paper. Intel.

[9] Linaro. [n.d.]. Hafnium. https://www.trustedfirmware.org/projects/hafnium/.

[10] John P Mechalas. 2016. Properly Detecting Intel® Software Guard Extensions (Intel® SGX) in Your Applications. https://software.intel.com/content/www/us/en/develop/articles/properly-detecting-intel-software-guard-extensions-in-your-applications.html.