Investigation and Vulnerability Analysis of ARM TrustZone

2018042733 / Junbeom Wi
College of Computing
Hanyang University ERICA
minkwns@hanyang.ac.kr

ABSTRACT

By the rapid growth of IoT technology, ARM proposed a security technology called *TrustZone* to protect important information against various software attacks. And many embedded systems currently use ARM architecture because of TrustZone.

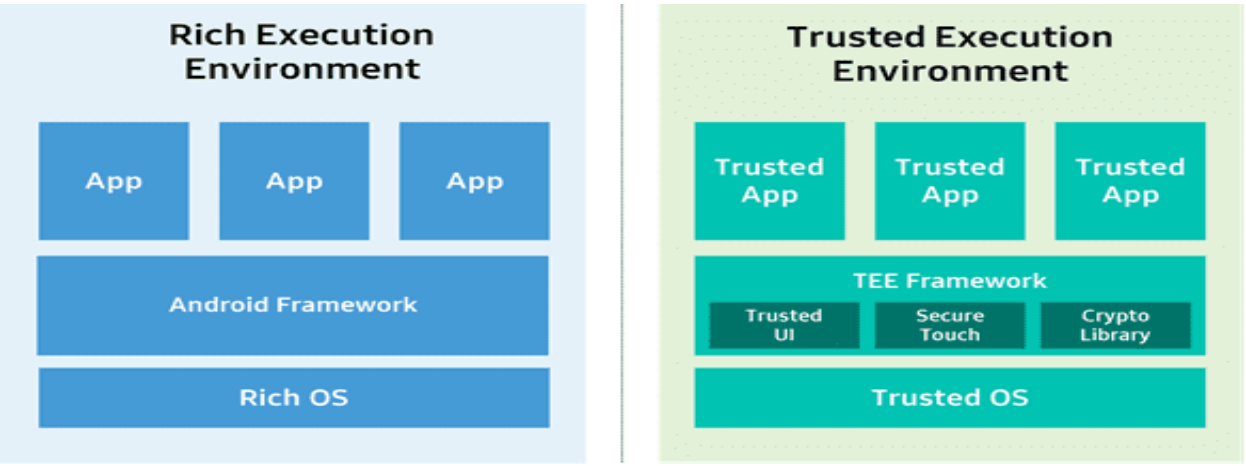
Several attacks on the TrustZone have been uncovered by Black Hat. Several vulnerabilities exist in TrustZone, but this report will focus on vulnerabilities caused by *System Monitor Call*. Cases of attacks by Qualcomm and Huawei were revealed in 2014 and 2015, respectively. TrustZone of the two manufacturers show a common vulnerability: *SMC's verification routines*. However, these cases can be protected by memory protection techniques, as they often cause problems with memory storing the return values of SMC. Eventually, the use of memory protection techniques and enhancing verification of Trusted Application could solve the problem.

1 INTRODUCTION

Although there have been security studies on embedded systems for a long time, Kernel attacks on operating systems still exist. By attacking the kernel, attackers access important system information, hide malicious behavior, or elevate the privileges of malicious programs. If the kernel is attacked, the attack on the operating system cannot be prevented. Therefore, security tools were placed in the kernel to block it in the existed operating system. However, since they have the same run level as the kernel, the security tools become useless if the kernel is attacked.

To protect the security of the OS and embedded system, supplementation of vulnerabilities in the kernel has always been researched. If the kernel is damaged in the process, it will be handled relatively easily to prevent incorrect code from being uploaded through *secure boot*. However, the problem is that the kernel is already loaded and corrupted. In order to protect the kernel in this system, integrity is prevented by a software solution by using a *hypervisor*. When the level of security tools for the kernel is raised through the hypervisor, runtime integrity is prevented. However, when the hypervisor code becomes too large, the hypervisor itself becomes large. Also, there are problems with the software itself, such as *vm escapes*. Therefore, a hardware-based security solution that can replace the traditional embedded system has been proposed.

2 WHAT IS TEE?



A software-based method is not a good answer for the attack and security of the kernel. Accordingly, a secure execution environment from external attacks is provided by placing security-related tools in the isolated execution space provided by hardware. This is called a *Trusted Execution Environment*, which is a hardware-based solution.

Trusted Execution Environment (TEE) is a solution to maintain security by controlling information exchange between the two areas by dividing the *Rich OS* and the *Secure OS* within the processor. By storing private and important information such as biometric information, payment information and encryption key in the secure world, information exchange with the normal world is controlled and security software can be safely executed. *Globalplatform*, a standardization organization, stipulated an international standard in 2010 for the TEE. Nowadays,

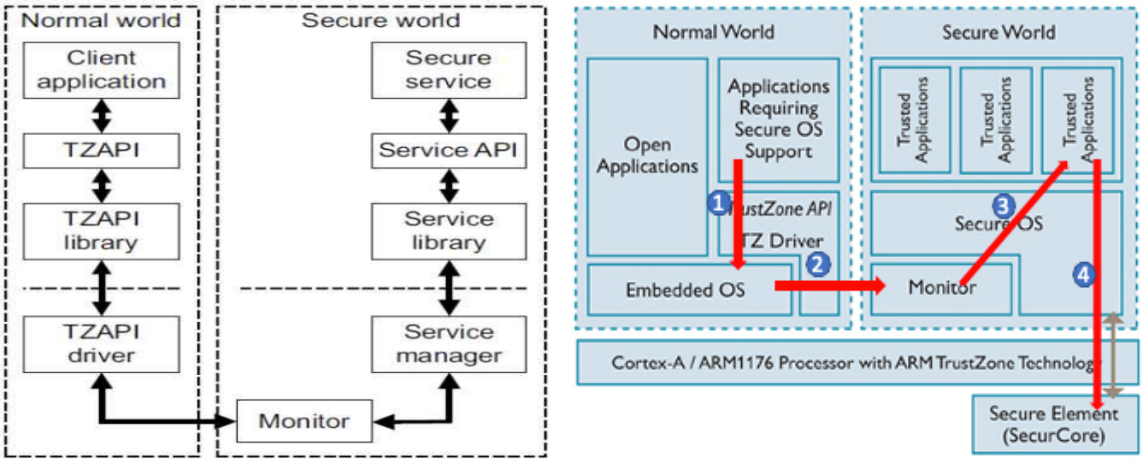
TEE is used as a data storage space for mobile services, public certificates, smartphone OTP, and biometric information. In addition, it is widely used in the IoT market about 95% share, problems in traditional situations can be blocked and by TEE.

3 ARM TRUSTZONE

Trustzone is a type of TEE, a hardware-based security technology and solution, and is an arm design method that guarantees confidentiality and integrity. The TrustZone is the name of the technology presented in the ARM, and follows the instructions presented by ARM. TrustZone has a hardware-based isolated execution space. In an architecture with TrustZone function, each physical processor provides two virtual cores, one treated as a *Normal World* as Rich OS and the other as a *Secure World* as Secure OS. Both worlds have separate memory, separate CPU registers, and separate page table-related registers. Also, there is memory area that only TrustZone can access. That is, software running in the normal world does not have privilege what access to resources in the secure world. The normal world and the secure world communicate using the *SMC (Secure Monitor Call) convention*.

4 SYSTEM MONITOR CALL

Currently, TrustZone uses *Systm Monitor Call (SMC)* to cross the normal world and the secure world. SMC is a software command that can cause a mode change from the normal world to the secure world and requests the execution of the secure world's information required in the normal world with SMC. SMC can be executed only in privileged code and passes the requestable service to the secure world as a parameter. The SMC instruction is used to generate a synchronous exception that is handled by Secure Monitor code, running in EL3(Exception Level 3: It is physical level where secure monitor is). Arguments and return values are passed in registers. After being handled by the *Secure Monitor*, calls that result from the instructions can be passed on to a Trusted OS or some other entity in the secure software stack.



Sequence of Request to secure world via SMC

1) Call TZAPI which is related to SMC.

2) Call SMC via TZAPI driver in the kernel.

3) Proceed SMC and call Trusted Application (TA)

4) Access to secure element by command from executed TA.

* Secure element: A hardware chip that is by design protected from unauthorized access and used to run a limited set of applications

5) When the SMC command is executed, the return value is stored in the general area kernel memory address passed as an argument. (This point will become vulnerability.)

5 USE CASE OF TRUSTZONE

Since there are various mobile systems or IoT using TrustZone, there are many use cases. Among them, examples used in KNOX and Samsung will be presented.

5.1 TRUSTED BOOT (KNOX, SAMSUNG)

Trusted boot is an enhancement to secure boot. Traditional secure boot is a security technique that makes it easy to defend against malicious code because it builds only authorized bootloader and kernel. However, the secure boot

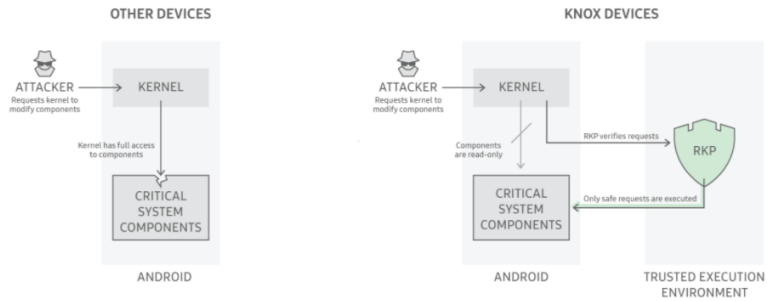
has a limitation that it does not preserve the markings of certified firmware after booting sequence. This means that an OS - level security program cannot always be applied on Android systems.

KNOX solved the limitations of secure boot using *trusted boot*. It checks whether firmware running on the device is authenticated and stores measurements safely within the TrustZone. At system runtime, it is checked on the knox platform and TA within trustzone uses these measurements to make important security decisions, such as disclosure of security keys.

5.2 TIMA (TZ BASED INTEGRITY MEASUREMENT ARCHITECTURE, KNOX)

KNOX is basically based on SE Android. However, SE has a problem with the hypothesis that the OS kernel is flawless. In other words, when the Linux kernel presents a threat or vulnerability, all devices using SE Android become potentially dangerous. Because of this, KNOX improves its vulnerability as a *TIMA*. TIMA continuously monitors the integrity of the Linux kernel using the design of the ARM TrustZone. SE Android's Linux kernel works in the normal world, while TIMA works in the secure world, which cannot be stopped. Eventually, when TIMA detects that the integrity of the kernel has been corrupted, it informs the person via Mobile Device Management (MDM) (using wireless networks to remotely control mobile devices such as smartphones). Through this process, a person in charge of a business can take measures such as deleting data remotely.

5.3 RKP (REALTIME KERNEL PROTECTION, KNOX)



RKP is a kernel security solution available in KNOX using TIMA. RKP uniquely employs a security monitor within an isolated execution environment. Running within an isolated execution environment would normally compromise a security mechanism's ability to see into the kernel and monitor activities at runtime. However, RKP

succeeds by utilizing patented techniques to control device memory management and by intercepting and inspecting critical kernel actions before allowing them to execute. RKP is thus able to prevent a compromised kernel from bypassing other security protections.

6 VULNERABILITIES ANALYSIS OF TRUSTZONE

As explained earlier, TrustZone has been used for a long time, and there have been several attacks by attackers because of its high share. Among them, the typical attacks are cases in which vulnerabilities were revealed during the monitor call process between Normal World and Secure World. That is, misvalidation of SMC call parameter values on monitors within the secure world is the main cause. A vulnerability occurs when verification of the memory address to store the return value fails.

Several security researchers presented an approach to common vulnerabilities at *blackhat* in 2014 and 2015. Although only memory addresses in the normal zone must be passed as parameters, a vulnerability in the address verification routine causes memory addresses in the secure world to be passed and SMC call returns written to the trust zone memory zone. As a result, the value of a specific memory location inside the secure world can be maliciously manipulated and the security world can be accessed.

6.1 EXPLOITATION CASES

Examples that will be used to analyze vulnerabilities are cases of *Huawei* and *Qualcomm*.

The vulnerability was exploited in three big steps.

1) First, elevate privileges to run kernel-level code.

→ It exploits general area or security area system level (kernel) vulnerabilities.

→ Privilege (root) privileges usually mean system file access and privileged command execution.

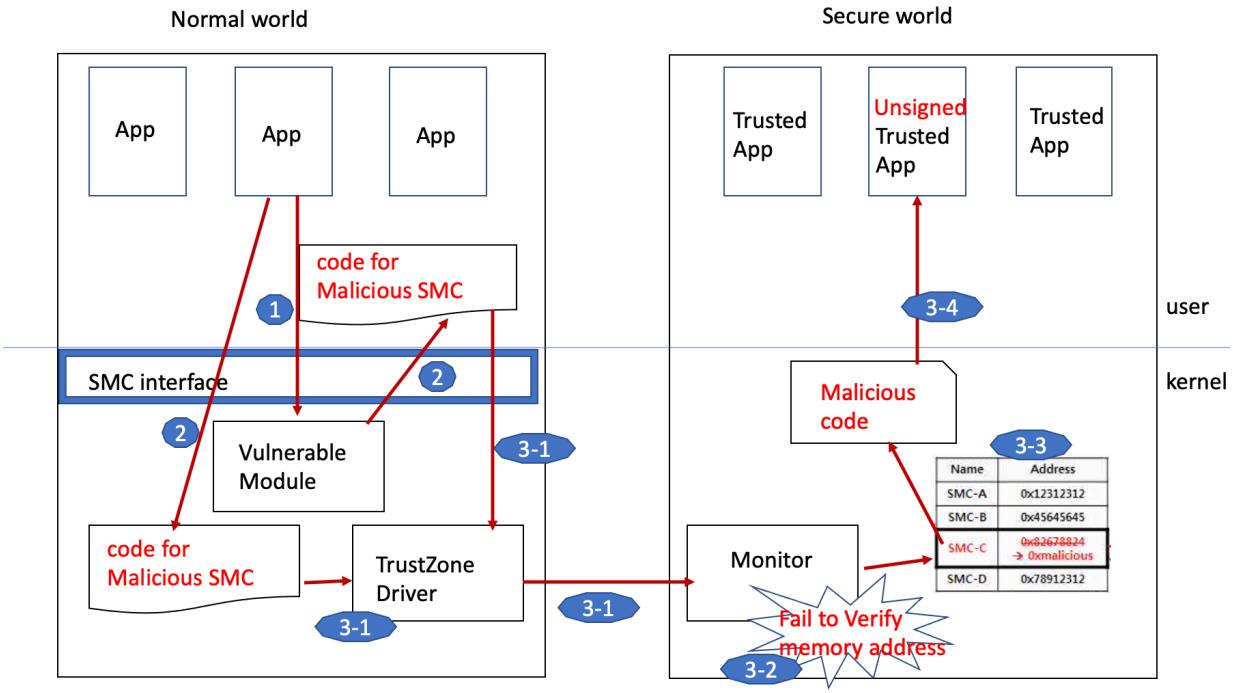
2) Second, run kernel-level code to manipulate SMC calls.

→ It is necessary for SMC calls and factor value manipulation necessary for exploitation of vulnerabilities.

3) and lastly, Load and run malicious code within the secure world.

→ It exploits corrupted SMC calls and TrustZone vulnerabilities (memory manipulation within a secure world to load and execute malicious code at the secure world.

More information is shown in the figure below.



1) Malicious SMC calls for exploitation of trust zone vulnerabilities require privileged privileges, which can exploit recently issued as *Android Stageflight vulnerabilities*, etc.

2) For malicious SMC command execution and manipulation, use RET2USER, LKM (Loadable Kernel Module) and ROP (Return Oriented Programming) techniques to load code and execute kernel-level code.

* RET2USER: A technique that exploits kernel vulnerabilities such as system calls to modulate temporary stored function return addresses on kernel memory (stacks) to arbitrary code addresses in the user space.

* LKM: This is a method of dynamically adding kernel modules (such as system calls and drivers) on Linux.

* ROP: This is a method of finding code fragments necessary for malicious behavior in kernel, etc., and executing them in combination and in succession.

3-1) This step analyzes SMC processing functions for security domain memory corruption.

Vulnerable Manufacturer	Malicious SMC call function
Qualcomm	tzbsp_prng_getdata_syscall() - Returns an arbitrary value generated through PRNG* H/W tzbsp_fver_get_version() - Returns the version code of the secure world
Huawei	get_sys_time() - return security zone kernel internal time

3-2) Tampering with secure area memory through weakness verifying routine.

Vulnerability Manufacturer	Location of corrupted memory
Qualcomm	Addresses of non-critical SMC functions in the SMC function address table present in secure zone memory.
Huawei	Command in any system call that is invoked by a particular SMC such as alloc_exception_mem()

3-3) Load the malicious code and execute it.

3-4) Access the TA and steal or modify information from the secure world.

7 COUNTERMEASURE

In the case of Huawei and Qualcomm, most of them have not been prevented from memory address protecting technology against *RET2USER, ROP* etc. To solve this, it must be prevented and mitigated through access control. The first way is to prevent certain memory address, using stack-based memory protection technology such as *ASLR*, *NX-bits*. Indeed, Huawei did not apply memory protection technologies such as ASLR to the TrustZone when a

vulnerability was presented in Black Hat in 2014. Furthermore, a method to supplement TA by separating general area privileges from SMC call command execution privileges through forced access control can also be used to prevent vulnerabilities.

8 REFERENCE

[1] Prashant Dewan, David Durham, Hormuzd Khosravi, Men Long, Gayathri Nagabhushan : *A Hypervisor-Based System for Protecting Software Runtime Memory and Persistent Storage ,* Communications Technology Lab, Intel Corporation. : 2008

[2] 최휘민, 장창복, 김주만 : *Efficient Security Method Using Mobile Virtualization Technology And Trustzone of ARM*: Pusan National University, R2Soft :2014, p299-302

[3] Rosenberg, D. *QSEE TrustZone Kernel Integer Overflow Vulnerability*. Black Hat USA 2014.

[4] Shen, D. *Exploiting Trustzone on Android*. Black Hat USA 2015.

[5] QuarksLab , *Breaking Samsung's ARM TrustZone*. Black Hat USA 2019.

[6] Yue Chen, Yulong Zhang, Zhi Wang, Tao Wei : *Downgrade Attack on TrustZone*: Florida State University, Baidu X-Lab: p1-9