上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

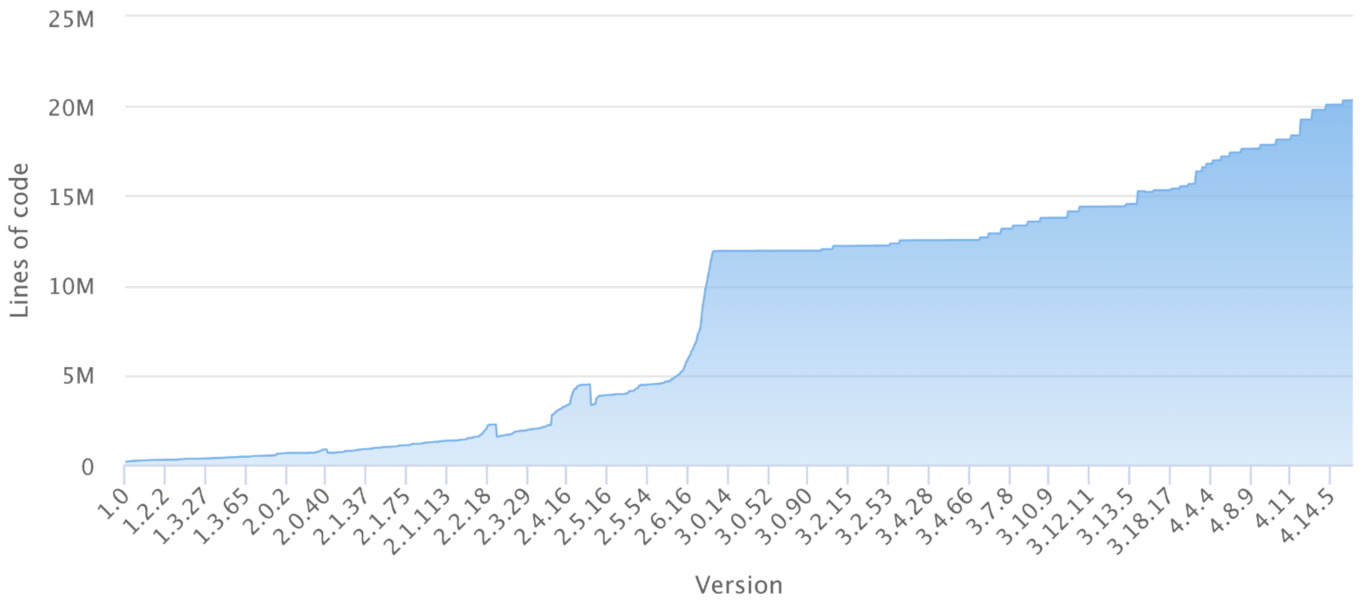
Hardware Enclave

Live with bugs

Yubin Xia

Background

- OS/Hypervisor has too much code as TCB



Lines of code in different versions of Linux

Background

- OS/Hypervisor has too much code as TCB
  - More and more bugs are founded and make OS/hypervisor vulnerable
    - CVE-2016-5195: user application can write read-only memory via the vulnerability in copy-on-write mechanism and thus get root privilege
    - CVE-2018-16597: user application can modify underlying database he is not supposed to access via incorrect permission check logic in kernel
    - Etc.
  - Malicious application can compromise OS/hypervisor via its vulnerability to attack other application
    - Things get worse in cloud environment where applications of multiple clients execute on the same machine

Background

- Moreover, cloud vendors may not be trusted by very security-sensitive clients
    - Security-sensitive clients want to use cloud vendors' computing resource
    - Whereas they do not allow cloud vendor to get the plain text of their data(confidentiality)
    - Neither do they allow cloud vendor to casually modify their data (integrity)

Background

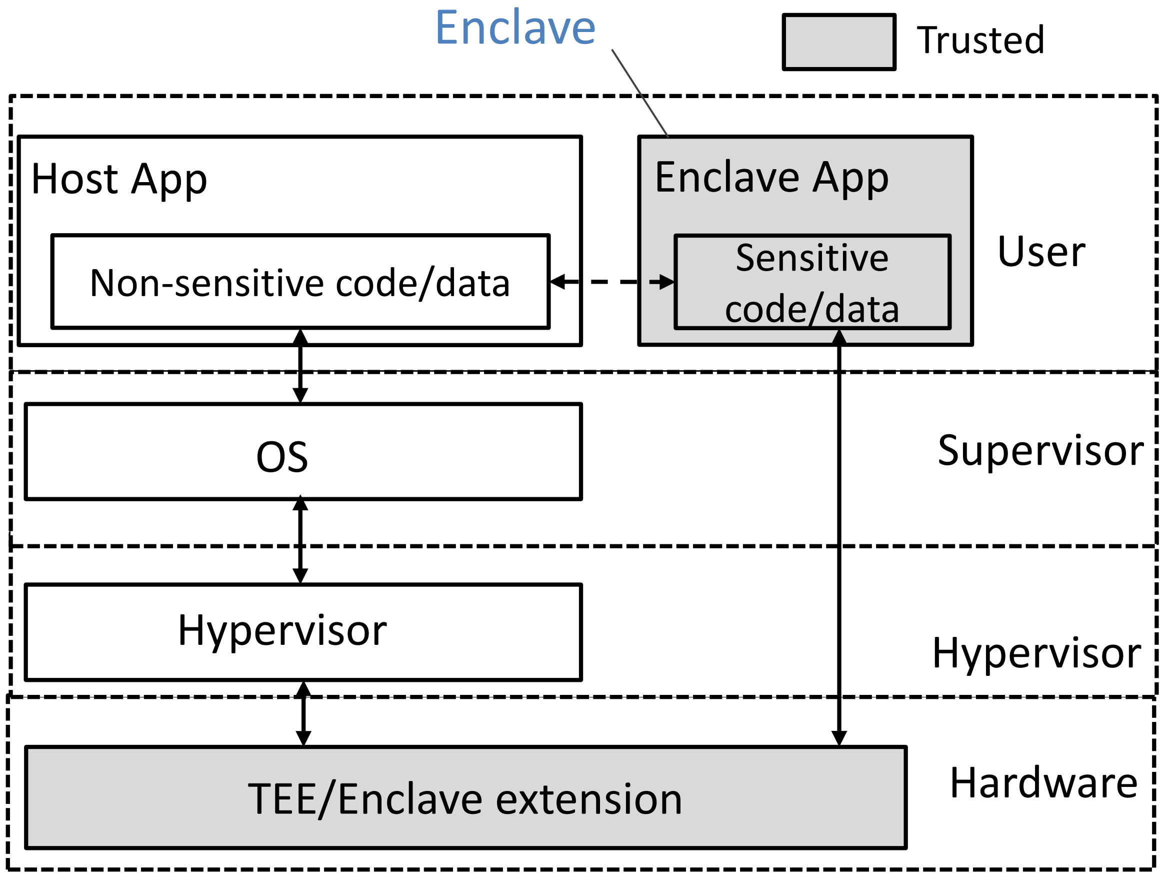
- Thus we have *enclave*
    - Technology to provide secure execution environment (aka enclave)
    - Only have a small TCB (hardware feature, CPU microcode, or small privileged software)
    - Use TCB to provide confidentiality, integrity and freshness guarantees for application executing in enclave environment

Background

- Attacks need to be considered
  - Privileged software attack from compromised OS/hypervisor
  - Side channel attacks applied by malicious application executing on the same machine
  - Physical attacks applied by cloud vendor
    - Memory bus snooping
    - Cold boot attack / NVM
    - Etc.

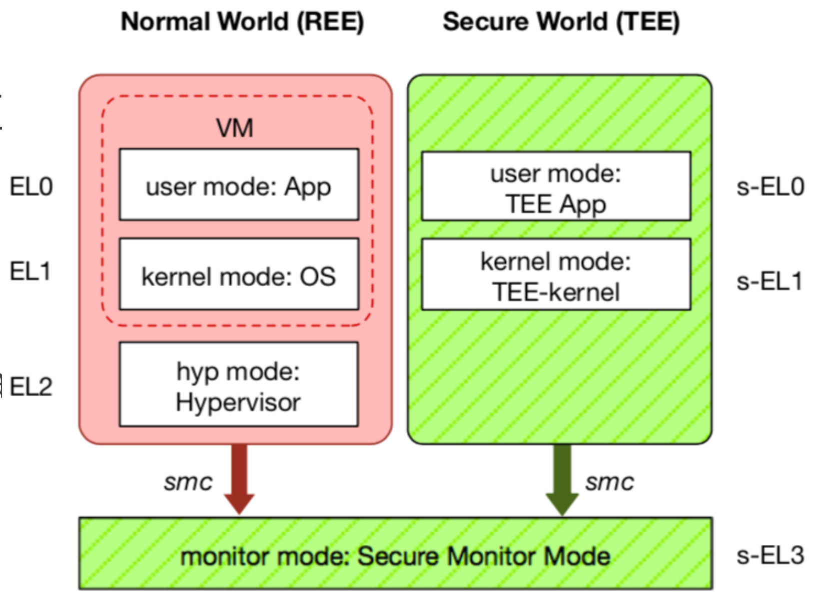
What is an Enclave?

Hardware-assisted Trusted Execution Environment (TEE)



TEE: Trusted Execution Environment

- Widely deployed on mobile devices
    - Every Android device must deploy TE
    - Mostly based on ARM TrustZone
- An *isolated* & *privileged* environment
    - TEE can access all DRAM & periphera
    - REE cannot access TEE's resources
- Designed to be secured and small
    - For security-critical tasks (e.g., fingerprint checker, mobile payment, etc.)



A Brief History of TEE

- **Stage-1 (before 2012):** A **fixed** piece of code by venders
  - Used for secure boot and firmware lock
- **Stage-2 (2012 ~ 2017):** Some **pre-installed** trusted apps (TAs) by venders
  - New applications: Biometric authentication (fingerprint, iris, 3D face, etc.), mobile payment, Digital Rights Management (DRM), etc.
  - New functionalities: Runtime protection (e.g., Samsung KNOX)
- **Stage-3 (2017 ~ now):** Support **dynamic installation** of third party TAs
  - A trend for openness and flexibility
  - Various TEEs and trusted apps are being deployed

Current Eco-system of TEE

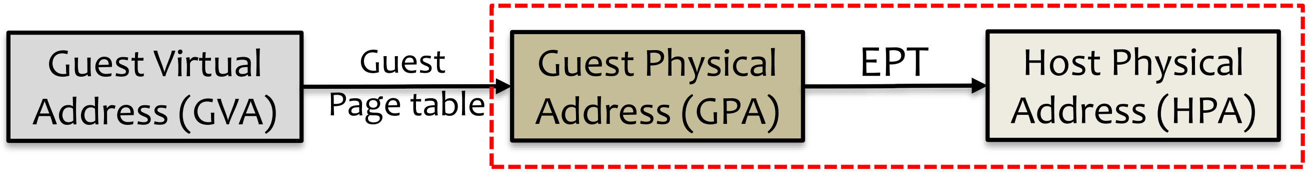
- Fragmentation of TEE
  - From chip venders: *QualComm, Spectrum, etc.*
  - From phone venders: *Apple, Huawei, etc.*
  - TEE OS venders: *TrustKernel, Trustonic, Google, Linaro, etc.*
  - Many other implementations based on OP-TEE
- Trusted applications:

  1. must be ported to each TEE OS, and,
  2. have to trust the underlying TEE OS

VM-BASED ENCLAVES

Review: Extended Page Table (EPT)

- Translate guest physical addr to host physical addr
  - The two-level translation are all done by hardware

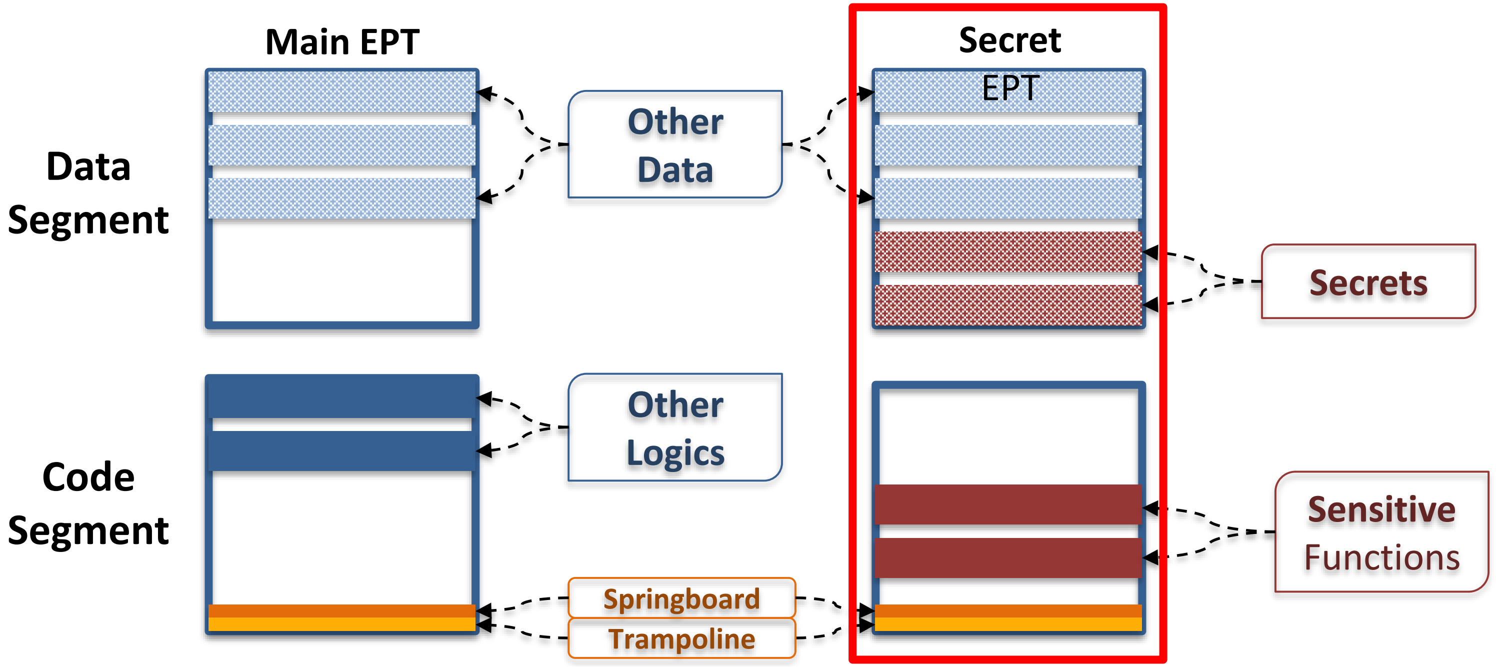


Guest Virtual Address (GVA)
Guest Page table
Guest Physical Address (GPA)
EPT
Host Physical Address (HPA)

- EPT is manipulated and maintained by hypervisor
  - Hypervisor controls how guest accesses physical address
  - Any EPT violation triggers VMExit to hypervisor

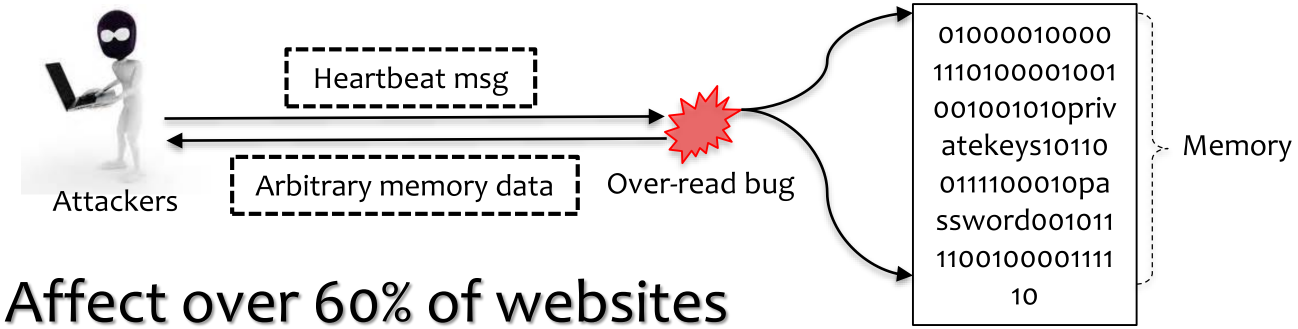
Memory Isolation using EPT Mechanism

- Leverage EPT mechanism to shadow secret memory
  - Data segment: secret memory is removed from main EPT
  - Code segment: sensitive functions only exist in secret EPT



HeartBleed Attack

- In-application memory disclosure attack
  - One over-read bug discloses the whole memory data

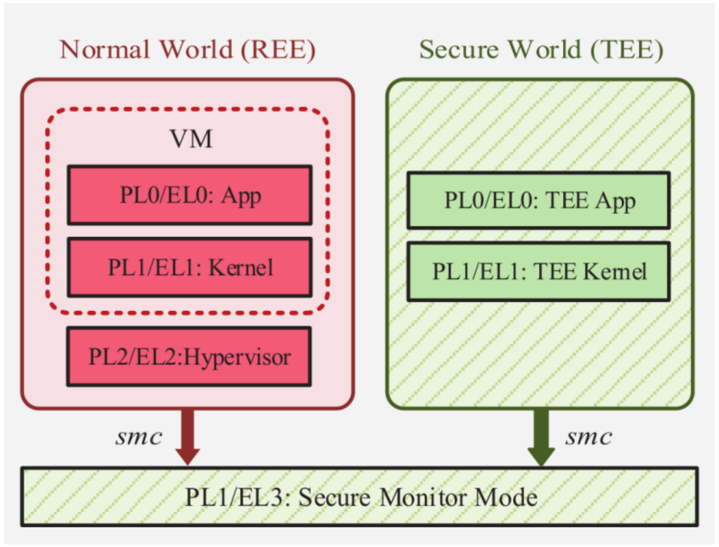


- Affect over 60% of websites
  - Attackers can steal private keys and user accounts

ARM TRUSTZONE

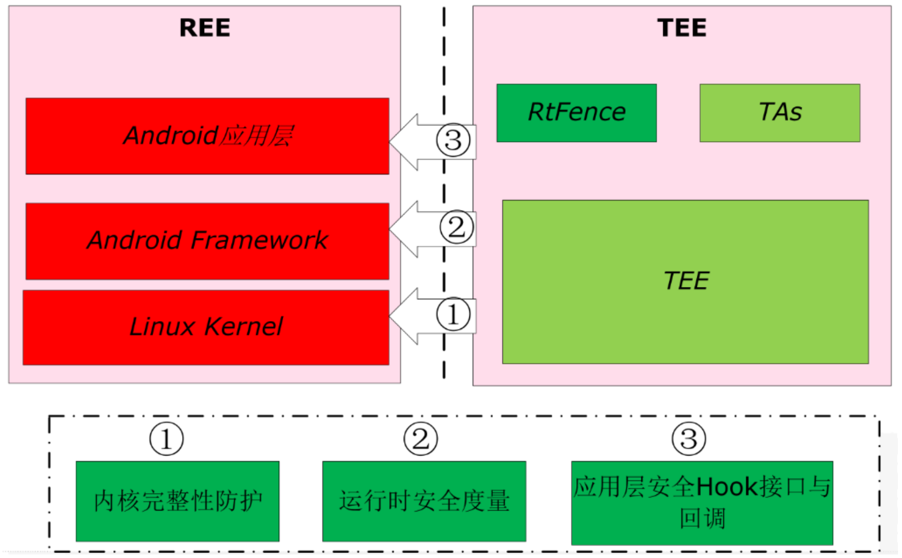
TrustZone and TEE

- Two execution environments
    - REE: Normal world
    - TEE: Secure world
- REE (Rich Execution Environment)
    - Run rich OS and apps, e.g., Android
- TEE (Trusted Execution Environment)
    - Run small OS and apps
    - Has higher privilege



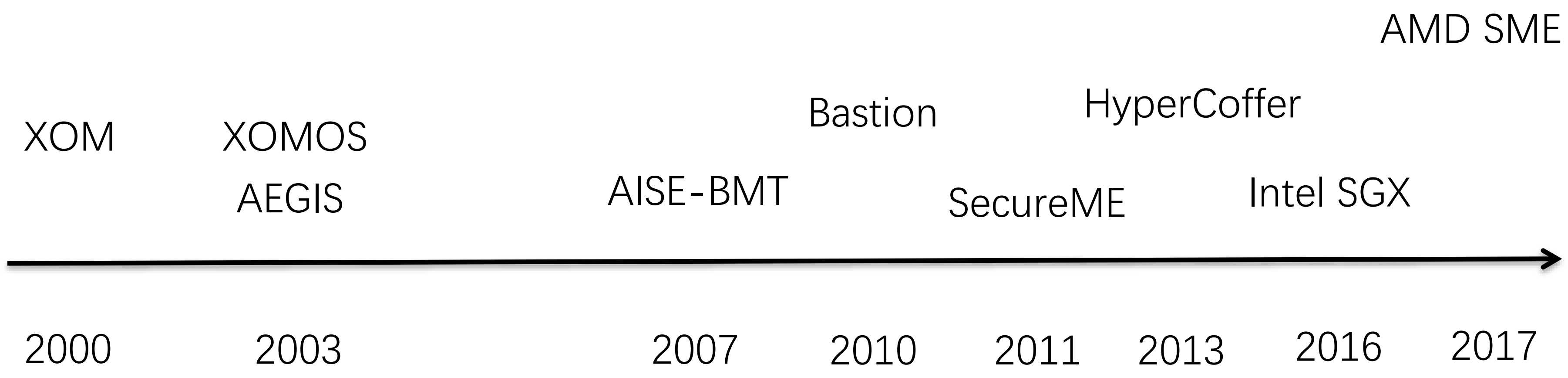
Real-time Kernel Protection on TEE

- Kernel Integrity Monitoring
  - Enforce the integrity of kernel's critical data and code
- Runtime Measurement
  - Protect the lifetime of critical data
- Hook/callback of Secure Service
  - Offer interface for up-level software



XOM

History of Memory Encryption



Counter-mode encryption

Merkle-tree based integrity check

eXecute-Only Memory (XOM)

- Architectural Support for Copy and Tamper Resistant Software
  - David Lie, Chandramohan Thekkath, Mark Mitchell, Patrick Lincoln, et al. from Stanford university
- ASPLOS 2000

eXecute-Only Memory (XOM)

- Threat model
  - Do not trust OS and external memory
- Propose execute-only code
  - Provides copy protection and tamper-resistance functions
  - Code and data are encrypted in memory
  - Store hash of encrypted value and its virtual address
- Abstract XOM machine
  - Compartment for each applications
  - Decode the compartment key using an asymmetric cipher
  - Decode the compartment content using the compartment key
  - Instruction for entering and exiting the compartment
  - Data tagging scheme for prevents cross-compartment access

Counter-Mode Encryption

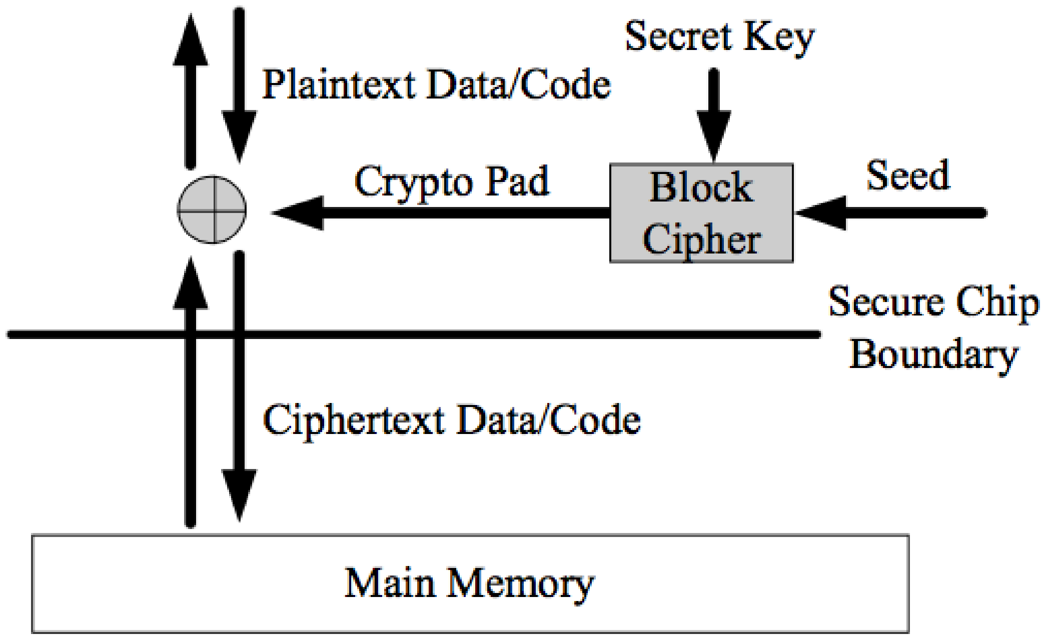


Figure 1. Counter-mode based memory encryption.

Security: Pad is unique

Approaches:

- global counter
- block address

Counter-Mode Encryption Problems

- Seed with global counter
  - Counter reuse
  - Large on-chip counter cache storage
- Seed with block address
  - Physical address may change due to page swap
  - Virtual address is vulnerable to pad reuse

Address Independent Seed Encryption

- For each chunk in the memory, its seed is the concatenation of:
  - a *Logical Page IDentifier* (LPID)
  - the page offset of the chunk's block
  - the block's counter value
  - the chunk id.

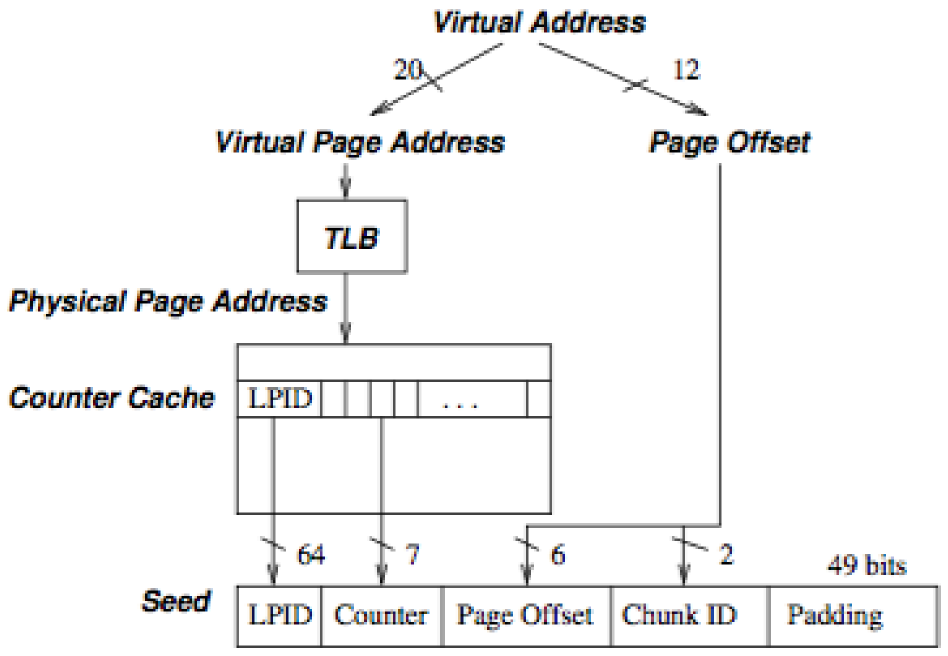


Figure 3. Organization of logical page identifiers.

Memory Integrity Verification

- Replay attacks for simply computing MAC
- Merkle tree verification
  - Keeps hierarchical MACs organized as a tree
  - The root of the tree is stored on-chip
  - Verification by checking chain of MAC values up to the root MAC

INTEL SGX

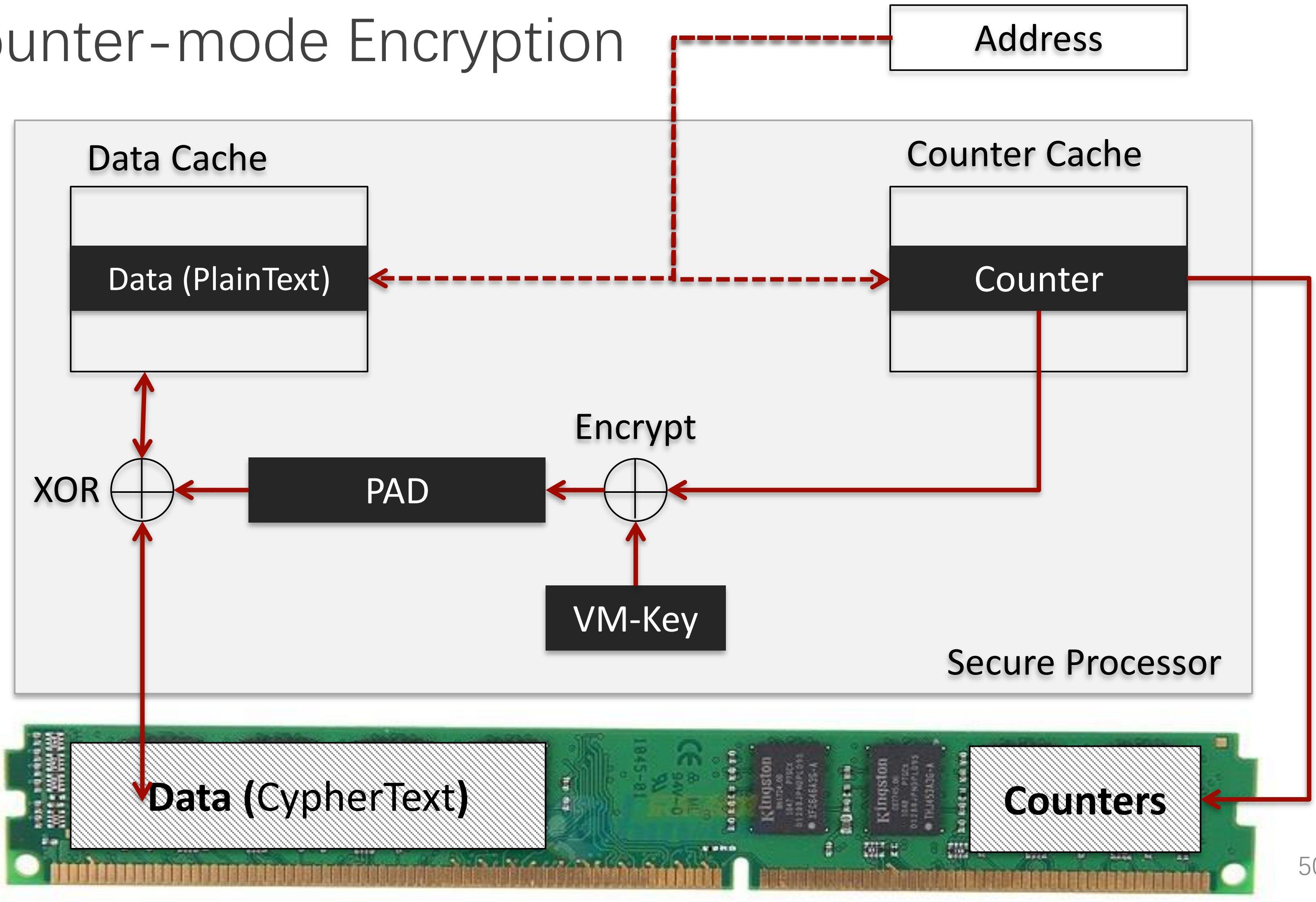
Why Intel SGX?

- Motivation: untrusted privileged software
  - E.g., protect application from **untrusted OS**
- What if the OS direct accesses application's memory?
  - Data are **encrypted** in memory
  - Data can only be accessed by the app within CPU boundary
  - The TCB contains only the CPU and app, no OS

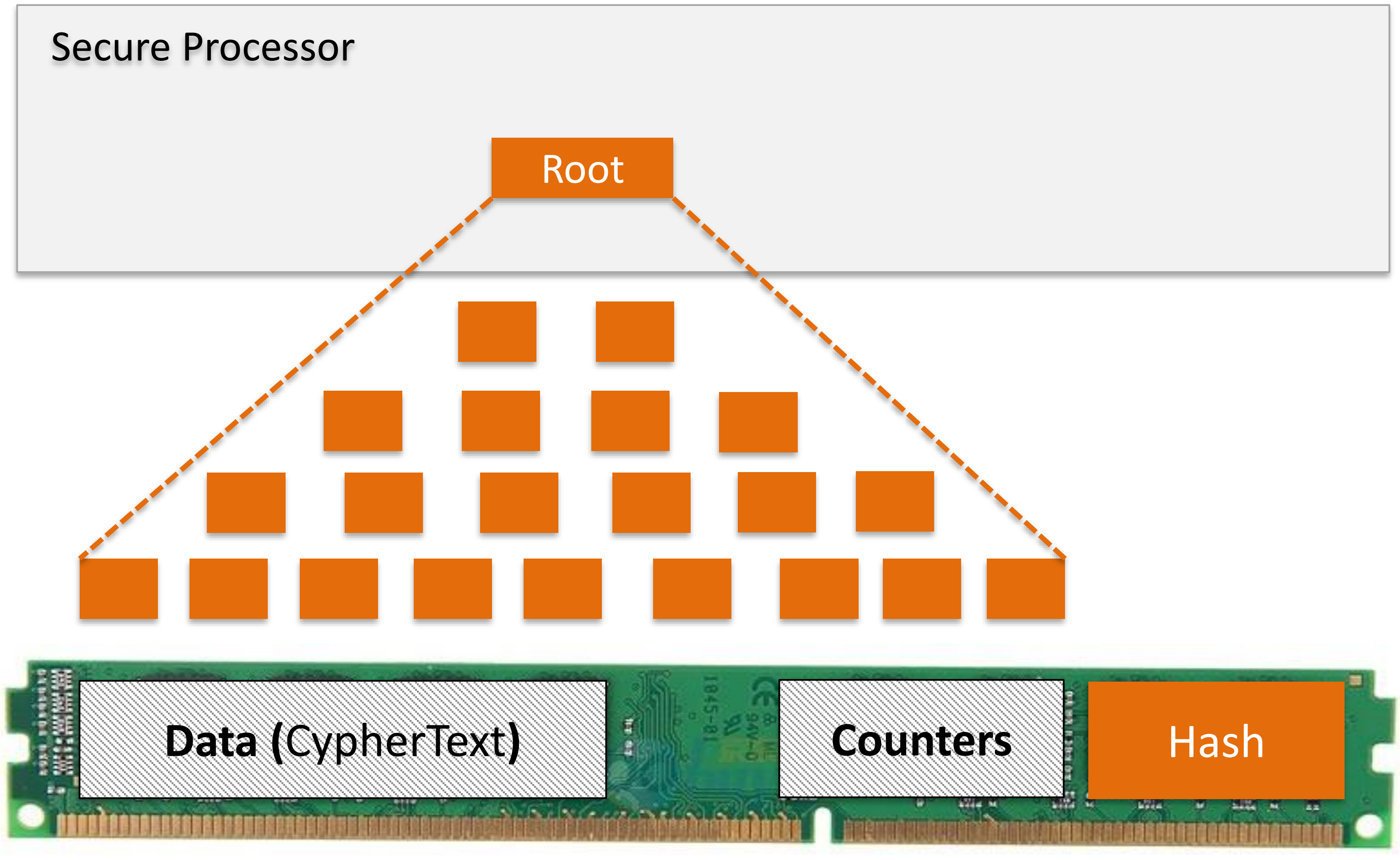
How can Memory Always be Encrypted?

- **Question**: data will **eventually be decrypted** when using
    - Then, what if an attacker steal data when it is being used?
- **Solution**: only decrypt data **inside CPU (in cache)**
    - The attacker now has to steal data directly from CPU

Counter-mode Encryption

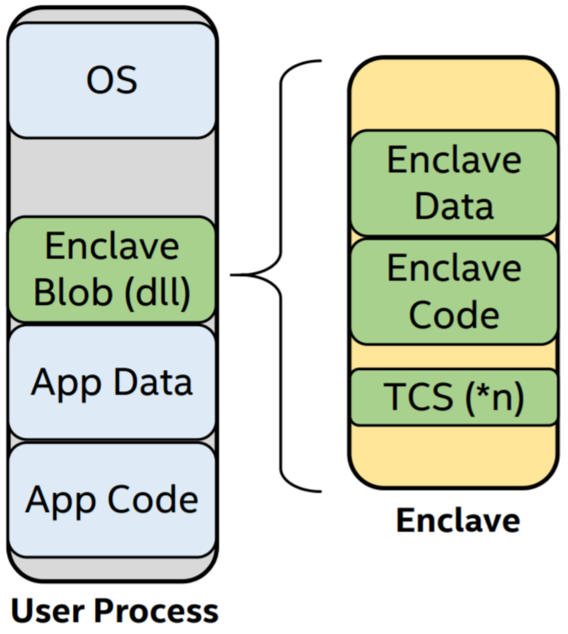


Merkel Tree for Data Integrity



Process View

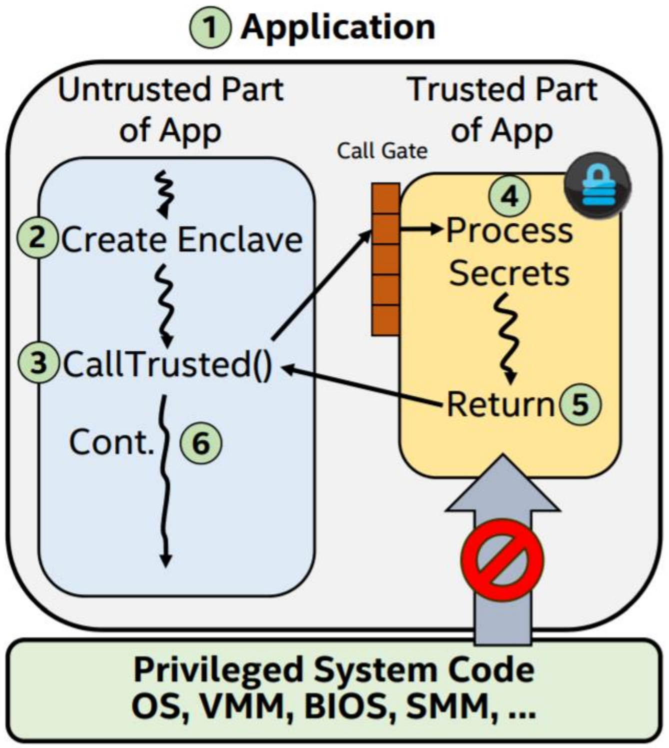
- With its own code and data
- Providing Confidentiality & Integrity
- Controlled entry points
- Multi-thread support
- Full access to app memory and processor performance



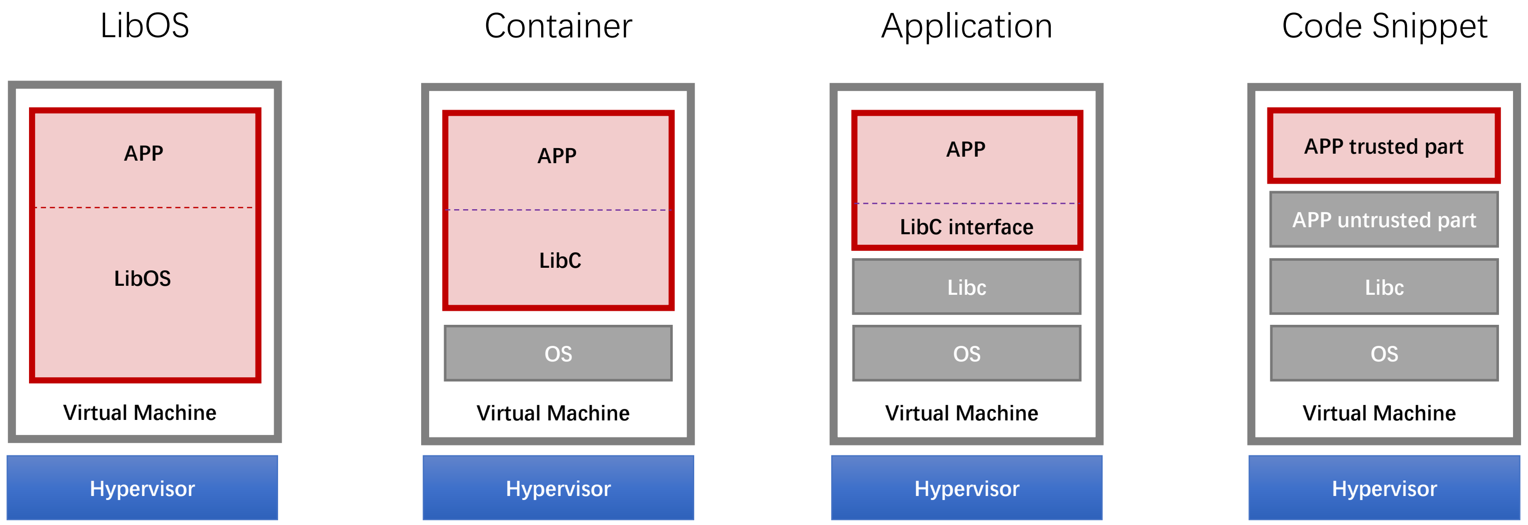
Protected execution environment embedded in a process

SGX Execution Flow

1. App built with trusted and untrusted parts
2. App runs & creates the enclave which is placed in trusted memory
3. Trusted function is called, execution transitioned to the enclave
4. Enclave sees all process data in clear; external access to enclave data is denied
5. Trusted function returns; enclave data remains in trusted memory
6. Application continues normal execution



Software Architectures of SGX



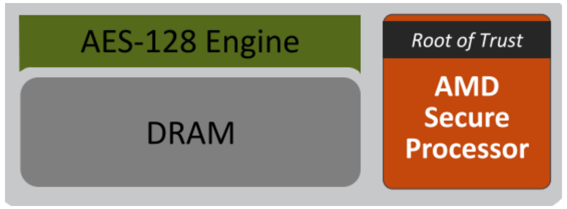
	Compatibility	TCB Size	Ocall Num	Attack Surface	Protect OS
LibOS	Part	Large	Few	Small	✓
Container	✓	Mid	Mid	Mid	✗
Application	✓	Mid	Many	Large	✗
Code Snippet	✗	Small	Few	Small	✗

SME: Secure Memory Encryption / SEV: Secure Encrypted Virtualization

AMD'S SME/SEV

2016: AMD x86 Memory Encryption Technologies

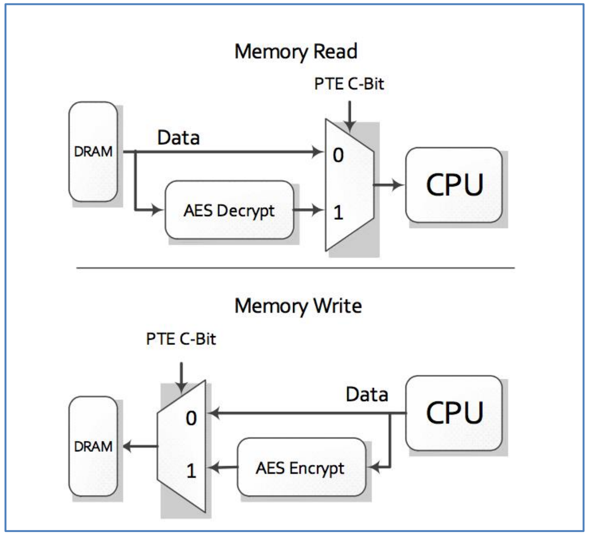
- Two Technologies
  - AMD Secure Memory Encryption (SME) & AMD Secure Encrypted Virtualization (SEV)
- Features
  - Hardware AES engine located in the memory controller performs inline encryption and decryption of DRAM
  - Minimal performance impact: Extra latency only taken for encrypted pages
  - No application changes required
  - Encryption keys are managed by the AMD Secure Processor and are hardware isolated
    - Not known to any software on the CPU

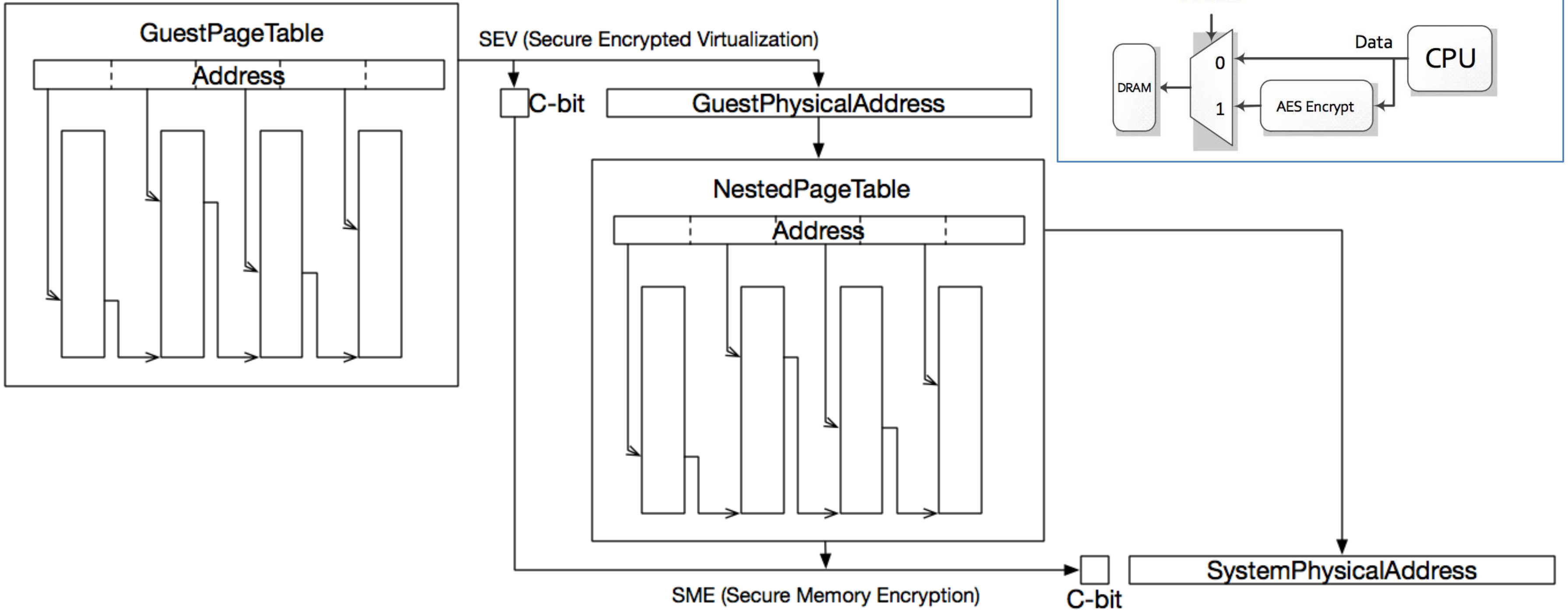


Comparing with Intel SGX

- The SME approach is different
    - It will not protect memory from an attacker who has compromised the kernel
    - It is intended to protect against cold-boot attacks, snooping on the memory bus, and the disclosure of transient data stored in persistent-memory arrays
- SEV focuses on virtualization
    - it can protect virtual machines from each other
    - keeping their contents secure even if one of them manages to compromise the host system
    - It should be able to protect virtual machines from the hypervisor itself

Usage of SEV


Memory Read
PTE C-Bit
DRAM
Data
0
1
AES Decrypt
CPU
Memory Write
PTE C-Bit
Data
CPU
DRAM
0
1
AES Encrypt

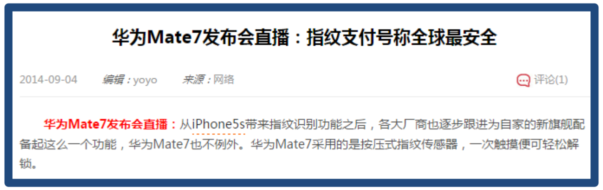

GuestPageTable
Address
SEV (Secure Encrypted Virtualization)
C-bit
GuestPhysicalAddress
NestedPageTable
Address
SME (Secure Memory Encryption)
C-bit
SystemPhysicalAddress

TEE SECURITY

More TEE Applications

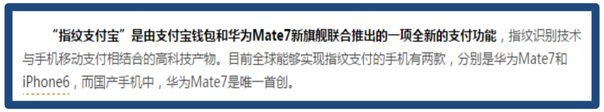
- TEE Malware Detection
- REE Security Enhancement
- More Applications
  - FIDO、Trusted Mail, Trusted SMS, Trusted Call…

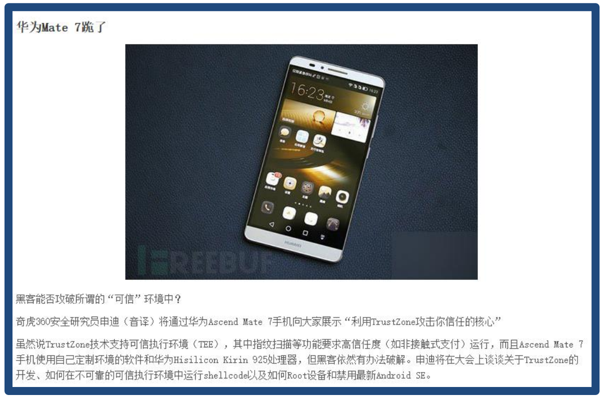
More and More TEE Vulnerabilities


华为Mate7发布会直播：指纹支付号称全球最安全

2014-09-04　*编辑*：yoyo　*来源*：网络　评论(1)

华为Mate7发布会直播：从iPhone5s带来指纹识别功能之后，各大厂商也逐步跟进为自家的新旗舰配备起这么一个功能，华为Mate7也不例外。华为Mate7采用的是按压式指纹传感器，一次触摸便可轻松解锁。

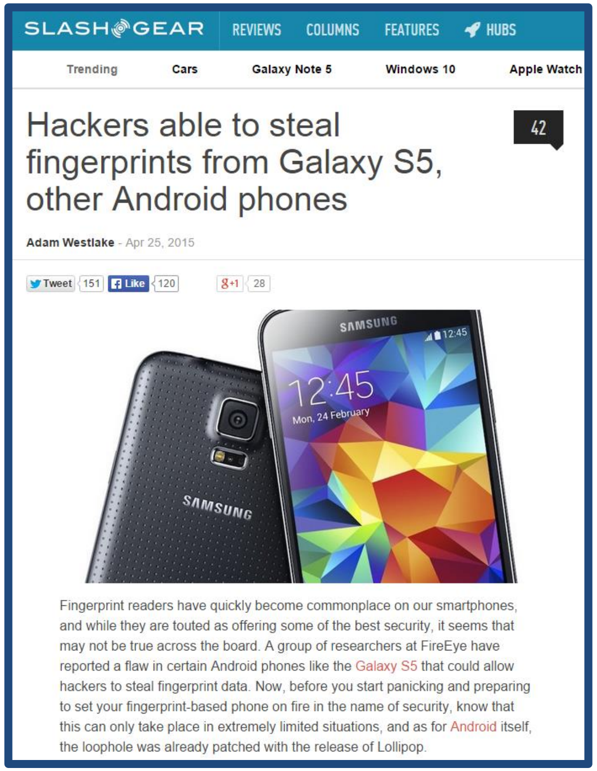

“指纹支付宝”是由支付宝钱包和华为Mate7新旗舰联合推出的一项全新的支付功能，指纹识别技术与手机移动支付相结合的高科技产物。目前全球能够实现指纹支付的手机有两款，分别是华为Mate7和iPhone6，而国产手机中，华为Mate7是唯一首创。


华为Mate 7跪了

黑客能否攻破所谓的“可信”环境中？

奇虎360安全研究员申迪（音译）将通过华为Ascend Mate 7手机向大家展示“利用TrustZone攻击你信任的核心”

虽然说TrustZone技术支持可信执行环境（TEE），其中指纹扫描等功能要求高信任度（如非接触式支付）运行，而且Ascend Mate 7手机使用自己定制环境的软件和华为Hisilicon Kirin 925处理器，但黑客依然有办法破解。申迪将在大会上谈谈关于TrustZone的开发、如何在不可靠的可信执行环境中运行shellcode以及如何Root设备和禁用最新Android SE。

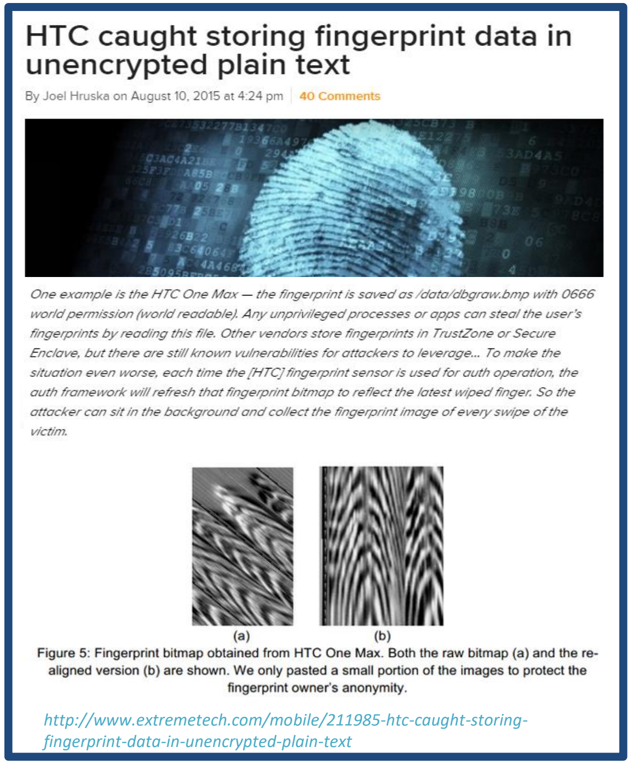


Hackers able to steal fingerprints from Galaxy S5, other Android phones

42

Adam Westlake - Apr 25, 2015

Fingerprint readers have quickly become commonplace on our smartphones, and while they are touted as offering some of the best security, it seems that may not be true across the board. A group of researchers at FireEye have reported a flaw in certain Android phones like the Galaxy S5 that could allow hackers to steal fingerprint data. Now, before you start panicking and preparing to set your fingerprint-based phone on fire in the name of security, know that this can only take place in extremely limited situations, and as for Android itself, the loophole was already patched with the release of Lollipop.


HTC caught storing fingerprint data in unencrypted plain text

By Joel Hruska on August 10, 2015 at 4:24 pm | 40 Comments

One example is the HTC One Max — the fingerprint is saved as /data/dbgraw.bmp with 0666 world permission (world readable). Any unprivileged processes or apps can steal the user's fingerprints by reading this file. Other vendors store fingerprints in TrustZone or Secure Enclave, but there are still known vulnerabilities for attackers to leverage... To make the situation even worse, each time the [HTC] fingerprint sensor is used for auth operation, the auth framework will refresh that fingerprint bitmap to reflect the latest wiped finger. So the attacker can sit in the background and collect the fingerprint image of every swipe of the victim.

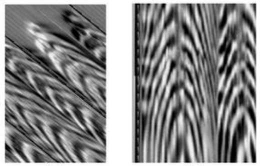
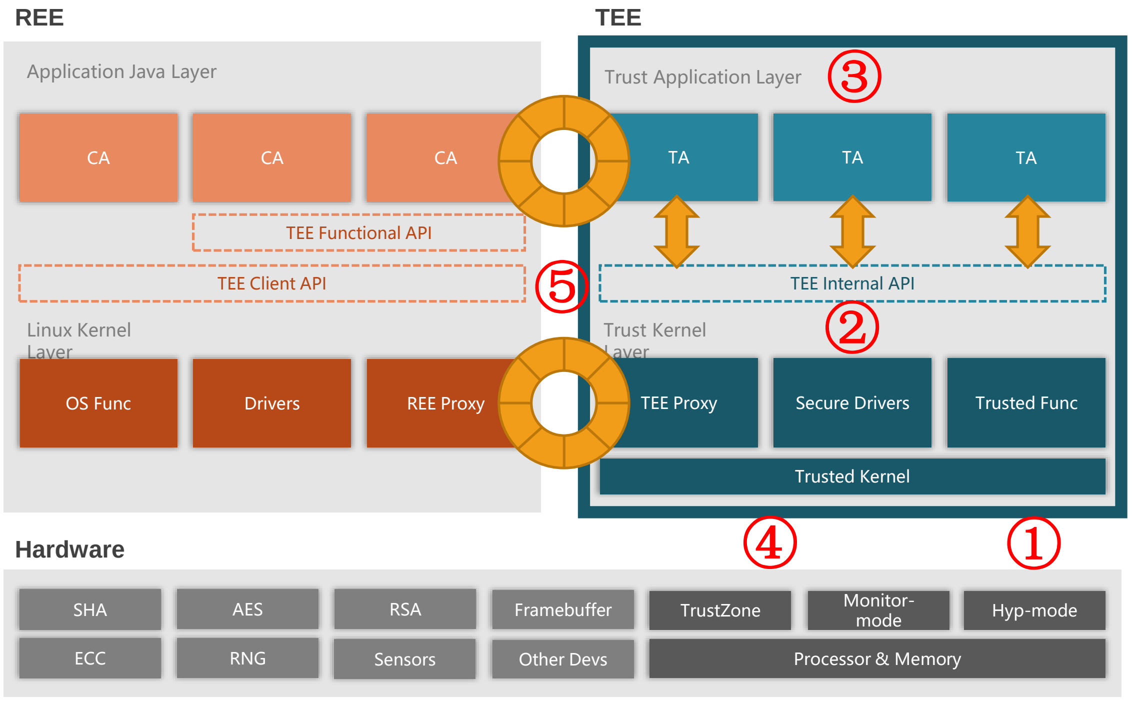


Figure 5: Fingerprint bitmap obtained from HTC One Max. Both the raw bitmap (a) and the re-aligned version (b) are shown. We only pasted a small portion of the images to protect the fingerprint owner's anonymity.

http://www.extremetech.com/mobile/211985-htc-caught-storing-fingerprint-data-in-unencrypted-plain-text

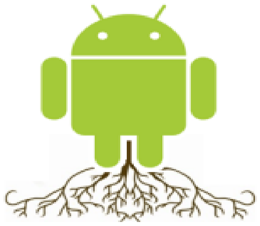
TEE Attacking Surface

- ①Secure boot
- ②TEE kernel
- ③TA security
- ④Hardware operation
- ⑤TEE/REE interaction



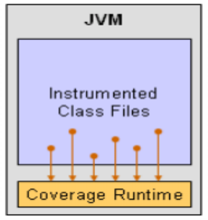
Attacker's Toolkit

Tools in REE


Root tool (e.g., KingRoot)

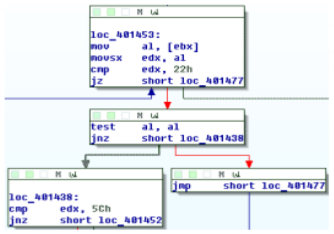
Decompiler（e.g., JEB）

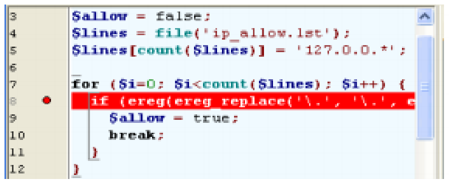
Introspection (e.g., Androguard)



Instrumenting（e.g., ADBI）

Tools in TEE


```
loc_401453:
mov     al, [ebx]
movsx   edx, al
cmp     edx, 22h
jz      short loc_401477

test    al, al
jnz     short loc_401438

loc_401438:
cmp     edx, 5Ch
jnz     short loc_401452

jmp     short loc_401477
```
Disassembler (e.g., IDA)

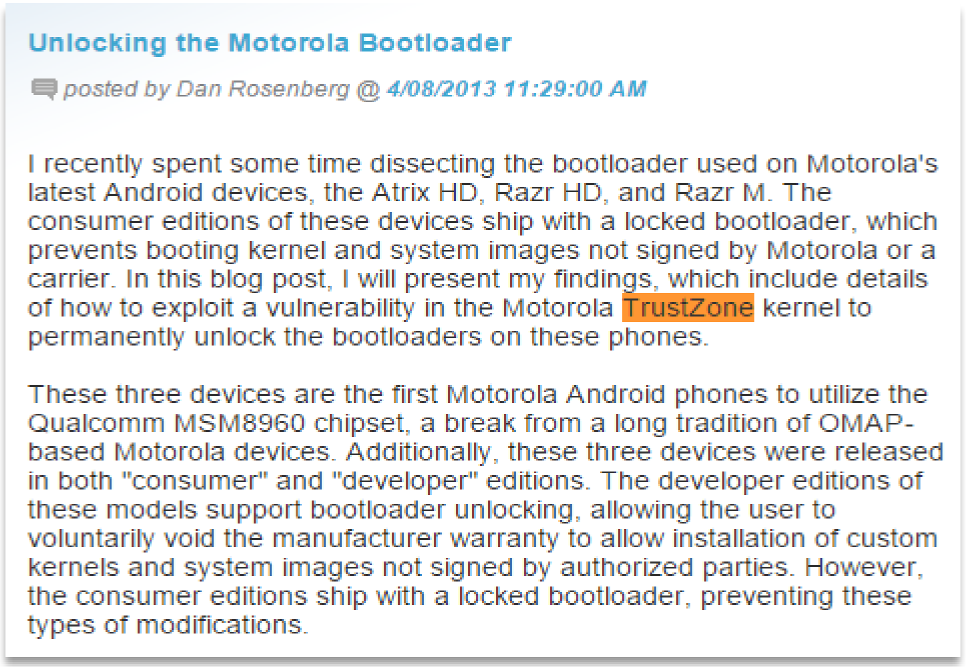

```
$allow = false;
$lines = file('ip_allow.lst');
$lines[count($lines)] = '127.0.0.*';

for ($i=0; $i<count($lines); $i++) {
  if (ereg(ereg_replace('\.', '\.', e
    $allow = true;
    break;
  }
}
```
Debugger Tool (e.g., GDB)

TEE SDK

Side Channel, Snooping

Attack on Secure Boot

- 2013, Many Moto and Samsung Phone
  - Exploit TrustZone Kernel
- Unlock bootloader
  - Can load any OS kernel



Unlocking the Motorola Bootloader

posted by Dan Rosenberg @ 4/08/2013 11:29:00 AM

I recently spent some time dissecting the bootloader used on Motorola's latest Android devices, the Atrix HD, Razr HD, and Razr M. The consumer editions of these devices ship with a locked bootloader, which prevents booting kernel and system images not signed by Motorola or a carrier. In this blog post, I will present my findings, which include details of how to exploit a vulnerability in the Motorola TrustZone kernel to permanently unlock the bootloaders on these phones.

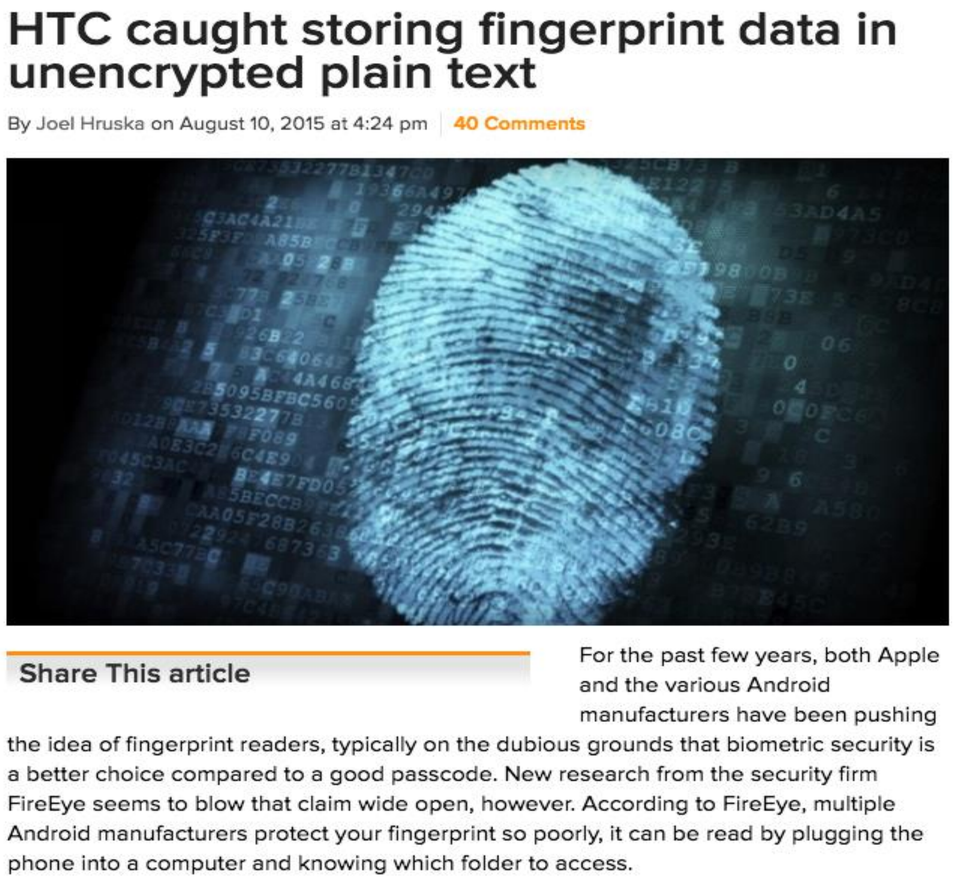
These three devices are the first Motorola Android phones to utilize the Qualcomm MSM8960 chipset, a break from a long tradition of OMAP-based Motorola devices. Additionally, these three devices were released in both "consumer" and "developer" editions. The developer editions of these models support bootloader unlocking, allowing the user to voluntarily void the manufacturer warranty to allow installation of custom kernels and system images not signed by authorized parties. However, the consumer editions ship with a locked bootloader, preventing these types of modifications.

Source: http://blog.azimuthsecurity.com/2013/04/unlocking-motorola-bootloader.html

Attacks on TEE

- QualComm QSEE Bug: CVE-2014-4322
  - drivers/misc/qseecom.c
  - No sanity check on offset and length
  - May lead to privilege escalation or DoS
- Huawei RTOSck Bug: CVE-2015-4422
  - Lead to any code execution
  - No ASLR or NX enabled

Attacks on TA



HTC caught storing fingerprint data in unencrypted plain text

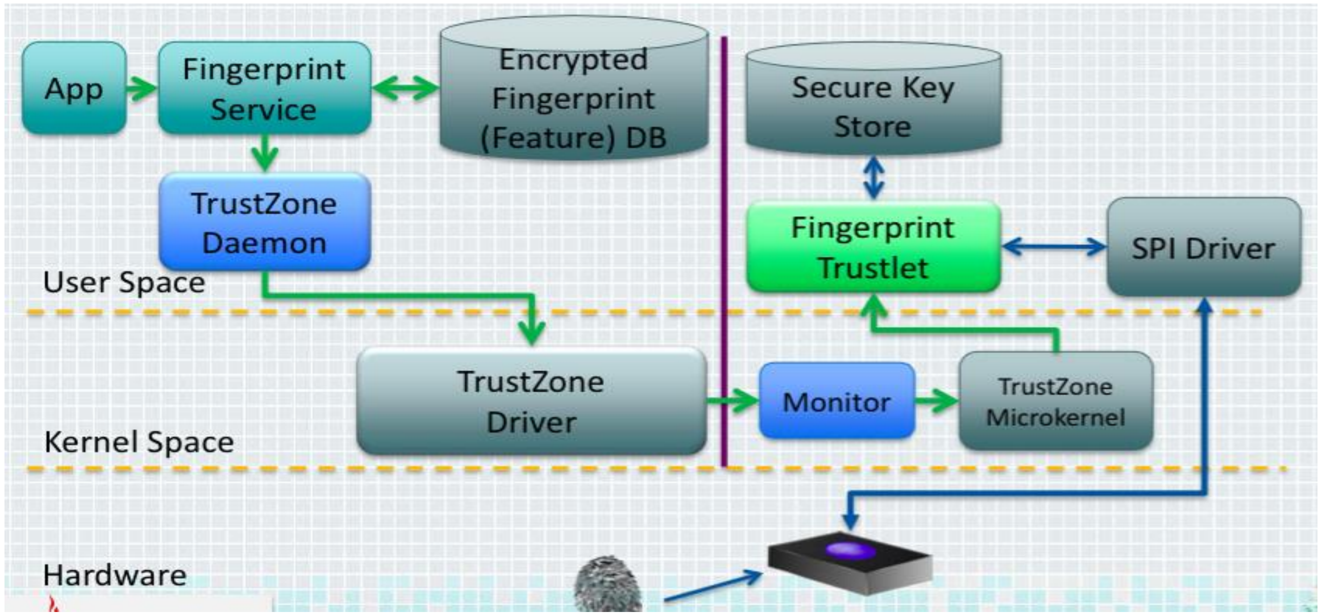
By Joel Hruska on August 10, 2015 at 4:24 pm | 40 Comments

Share This article

For the past few years, both Apple and the various Android manufacturers have been pushing the idea of fingerprint readers, typically on the dubious grounds that biometric security is a better choice compared to a good passcode. New research from the security firm FireEye seems to blow that claim wide open, however. According to FireEye, multiple Android manufacturers protect your fingerprint so poorly, it can be read by plugging the phone into a computer and knowing which folder to access.

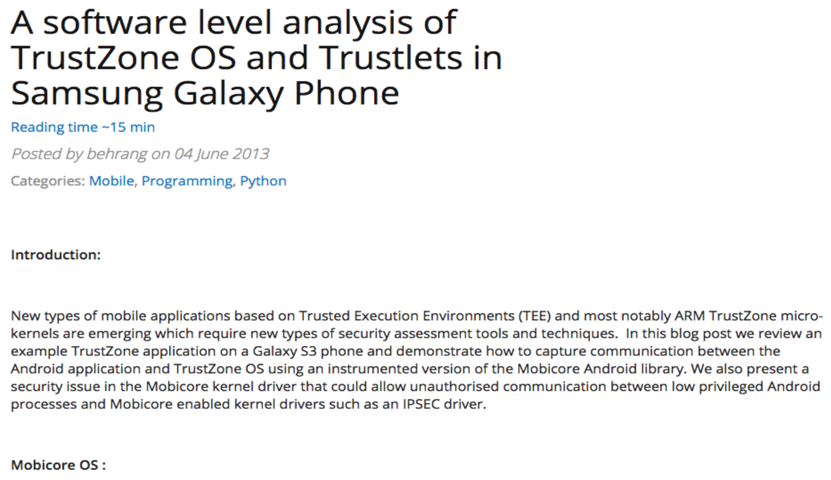
Attacks on Peripherals

- Blackhat'15: Samsung S5 has bug on fingerprint reader



Attack on Interaction between REE & TEE

- TEE Driver Privilege Escalation: CVE-2015-4421
  - Incorrect boundary check
- Samsung TEE Kernel MITM



A software level analysis of TrustZone OS and Trustlets in Samsung Galaxy Phone

Reading time ~15 min

Posted by behrang on 04 June 2013

Categories: Mobile, Programming, Python

Introduction:

New types of mobile applications based on Trusted Execution Environments (TEE) and most notably ARM TrustZone micro-kernels are emerging which require new types of security assessment tools and techniques. In this blog post we review an example TrustZone application on a Galaxy S3 phone and demonstrate how to capture communication between the Android application and TrustZone OS using an instrumented version of the Mobicore Android library. We also present a security issue in the Mobicore kernel driver that could allow unauthorised communication between low privileged Android processes and Mobicore enabled kernel drivers such as an IPSEC driver.

Mobicore OS :