Hardware-assisted Trusted Execution Environments

Look Back, Look Ahead

N. Asokan

https://asokan.org/asokan/

@nasokan
Hardware-assisted TEEs are pervasive

Hardware support for
- Isolated execution: Isolated Execution Environment
- Protected storage: Sealing
- Ability to convince remote verifiers: (Remote) Attestation

Trusteed Execution Environments (TEEs)
Operating in parallel with “rich execution environments” (REEs)

Cryptocards, Trusted Platform Modules, ARM TrustZone, Intel Software Guard Extensions


Concerns with TEEs: flaws

TPM Reset Attack
50,012 views

Evan Sparks
Published on Jun 18, 2007

A demonstration of a vulnerability in the TCG architecture running TPM without restarting the platform.


CLKSCREW: Exposing the Perils of Security-Oblivious Energy Management

Authors:
Adrian Tang, Simha Sethumadhavan, and Salvatore Stolfo, Columbia University
Distinguished Paper Award Winner!


Foreshadow (security vulnerability)

From Wikipedia, the free encyclopedia

Foreshadow is a vulnerability that affects modern microprocessors that was first discovered by two independent teams of researchers in January 2018, but was first disclosed to the public on 14 August 2018 (CVE-2018-0120). The vulnerability is a speculative execution attack on Intel processors that may result in the loss of sensitive information stored in personal computers or third-party clouds. There are two versions: the first version (original Foreshadow) (CVE-2018-0120) targets data from SGX-protected enclave, and the second version (modified Foreshadow) (CVE-2018-3694 and CVE-2018-3695) targets virtual machines (VMs), hypervisors (VMM), operating system (OS) kernel memory, and System Management Mode (SMM) memory. Intel considers the entire class of speculative execution side channel vulnerabilities as "1. Terminal Fault" (1TF). A listing of affected Intel hardware has been posted.

Foreshadow is similar to the Spectre security vulnerabilities discovered earlier to affect Intel and AMD chips, and the Meltdown vulnerability that also affected Intel; however, AMD products, according to AMD, are not affected by the Foreshadow security flaws. According to one expert, "[Foreshadow] lets malicious software hijack secure areas that even the Spectre and Meltdown flaws couldn’t touch." Nonetheless, one of the variants of Foreshadow goes beyond Intel chips with SGX technology and affects all Intel Core processors built over the last seven years. Foreshadow may be very difficult to exploit and there seems to be no evidence to date (19 August 2018) of any serious hacking involving the Foreshadow vulnerability. Nevertheless, applying software patches may help alleviate some concerns, although the balance between security and performance may be a worthy consideration. Companies performing cloud computing may see a significant decrease in their overall computing power; individuals, however, may not notice any significant performance impact. The fix, according to Intel, is by replacing today’s processors instead of further updates. These changes begin with our next-generation Intel Xeon Scalable processors (code-name "Cascade Lake").

Concerns with TEEs: suspicions of motives

Software

MS Palladium protects IT vendors, not you – paper

Anderson gives us the FAQs

By John Lettice 28 Jun 2002 at 10:27


Problem: Third-party uncertainty about your software environment is normally a feature, not a bug


Outline

A Look Back: How did TEEs start?

What are some (useful) applications for TEEs?

What are the downsides of relying on hardware-assisted TEEs?

(How) can we deal with these downsides?

What are other examples of hardware-assisted security?
Look Back
Platform security for mobile devices

Mobile network operators:
1. Subsidy locks → immutable ID
2. Copy protection → device authentication, app. separation
3. ...

End users:
1. Reliability → app. separation
2. Theft deterrence → immutable ID
3. Privacy → app. separation
4. ...

Regulators:
1. RF type approval → secure storage
2. Theft deterrence → immutable ID
3. ...

Closed → Open
Different Expectations than for PCs!
Early adoption of software platform security

~2001

J2ME

~2004

Symbian OS Platform Security

Software Development Using the Symbian OS Security Architecture

~2008

iOS

Mobile software platform security is now widely deployed

Example: regulatory compliance

The IMEI shall not be changed after the ME’s final production process. It shall resist tampering, i.e. manipulation and change, by any means (e.g. physical, electrical and software).

NOTE: This requirement is valid for new GSM Phase 2 and Release 96, 97, 98 and 99 MEs type approved after 1st June 2002.

Secure storage of RF configuration parameters

Early TEEs for mobile phones
Nokia Radio Application Processor (RAP), ca. 2001

Saara Matala & Thomas Nyman, “Historical insight into the development of Mobile TEEs”, Aalto SSG research group blog (2019)
Mobile TEEs: Motivation

Business requirements:
- mobile payment
- subsidy lock

Regulatory requirements:
- tamper-resistant IMEIs
- secure storage for RF

Engineering constraints:
Cost of discrete security chip too high on bill of materials!

New approach: “processor secure environments”

Generic low-cost enabler emerged as skunkworks project within Nokia
(rather than point solutions for particular use cases)
Mobile TEEs: Development

1982 Texas Instruments, Gutttag US4521853A
“Secure microprocessor/microcomputer with secured memory”

1982 Texas Instruments, Gutttag and Nussarallah US4521853A
“Security bit for designating the security status of information stored in a nonvolatile memory”

1996 Intertrust, Ginter et al US 5892900A
“Systems And Methods For Secure Transaction Management And Electronic Rights Protection”

2002 Nokia, Kiiveri and Paatero US9111097B2
“Secure execution architecture”

2003 Texas Instruments OMAP 161x and 73x processors

2004 ARM TrustZone
Mobile TEEs: Deployment

First deployment: Nokia 6630 (“Charlie”)
- first 3G phone with TI OMAP 1710 processor (June 2004)

ARM TrustZone currently widely deployed
- [TrustZone-M for Cortex-M class microcontrollers](2016)

Ca. 2008, TEE unheard of in academic circles
- first papers in FC 2008, ASIACCS 2009
  - [A Platform for OnBoard Credentials](AE08), Financial Cryptography and Data Security (2008)
  - [On-board credentials with open provisioning](KEAR09), ACM ASIACCS (2009)

Intel SGX
- SkyLake (2015); wide availability of SDK “democratized” TEE research
Mobile TEEs: Standardization

REE (Rich Execution Environment)

- App
- App
- TPM API
- TEE Client API
- Mobile OS

TEE

- Trusted app
- MTM / TPM 2.0 Trusted app
- TEE Internal API
  - trusted user interface
  - secure element
  - TEE management
- biometrics
- sockets
- debug

Trusted OS

Roots of Trust
- Device Hardware
- Device Firmware

TEE entry
Using TEEs
Original motivations (for mobile TEEs)

Tamper-resistant device identifiers (IMEIs)
for various use cases including theft protection, subsidy lock, and DRM

Sealed storage
for secure storage of RFID configuration data

Mobile payments

Boot integrity
TEE applications: academic literature (1/2)

Private membership test for malware scanning, private contact discovery,


Protection of password-based web authentication


Secure accounting for function-as-a-service (FaaS) settings


Scalable consensus for blockchains and cryptocurrencies


Examples only, not a complete list
TEE applications: academic literature (2/2)

Private neural network evaluation

High-performance remote ORAM

Verifiable computation

Authenticated data feeds

Examples only, not a complete list
**TEE applications: commercial deployments**

**Digital rights management (e.g. Widevine L1 & L2 content decryption)**
- Widevine DRM Architecture Overview
  - [Link](https://www.androidauthority.com/widevine-explained-821935/)

**Runtime integrity (e.g. OS kernel integrity monitoring)**
  - [Link](https://doi.org/10.1145/2660267.2660350)

**Local user authentication (e.g. password authentication, biometrics)**
- Android Gatekeeper
  - [Link](https://source.android.com/security/authentication/gatekeeper)
- Android Fingerprint HAL
  - [Link](https://source.android.com/security/authentication/fingerprint-hal)
- Windows Hello
  - [Link](https://docs.microsoft.com/en-us/windows/security/information-protection/tpm/how-windows-uses-the-tpm)

**Property Attestation (e.g. proof that cryptographic credential is protected by TEE)**
- Android Key and ID Attestation
  - [Link](https://source.android.com/security/keystore/attestation)
- MirrorLink Content Attestation
  - [Link](https://www.etsi.org/deliver/etsi_ts/103500_103599/10354404/01.03.00_60/ts_10354404v010300p.pdf)

  Also see [KAE11] Practical Property-Based Attestation on Mobile Devices, TRUST (2011)

  *Examples only, not a complete list*
Downsides of TEEs
Downsides of TEE-based solutions

Difficulty of developer access

Risk of TEE compromise
Difficulty of developer access

TEEs were closed systems

Tools for TEE software development were cumbersome and/or expensive

Device or TEE vendor controls what applications allowed to executed in the TEE

- Ordinary developers cannot deploy TEE apps without vendor approval
Risk of TEE compromise

**Software attacks**
- Many trusted applications are written in *unsafe languages*
- Correct trusted code can be vulnerable to *confused-deputy* attacks
- Difficult even for *hardware security module* vendors [CC19]

**Side-channel attacks**
- Timing
- Memory access
- Electromagnetic emanations

[CC19] “Everybody be Cool, This is a Robbery!”, Black Hat (2019)
Dealing with downsides
Challenge: easy development

GlobalPlatform standards for TEE interfaces
http://www.globalplatform.org/specificationsdevice.asp

Open-source tools for TEE app development available:
• OP-TEE https://www.op-tee.org/
• Open-TEE https://open-tee.github.io/
• OpenEnclave https://openenclave.io

Developing TEE applications is no longer cumbersome or expensive

Challenge: open deployment

On-board Credentials (ObC) (2006-2009)

- **open** credential platform leveraging TEE functionality
- allows any developer to write/use TEE apps
- deployed in Nokia and Windows smartphones (2009-2012)

- **applications:**
  - RSA SecurID, mobile ticketing (trialed at NY MTA LIRR in 2011), even "soft SIM"

Other efforts to address the deployment hurdle:

- “User centric provisioning” work from Royal Holloway
- GlobalPlatform white paper
  - “A New Model: The Consumer-Centric Model and How It Applies to the Mobile Ecosystem” Whitepaper (2012)

Challenge: Dealing with TEE compromise

Hardware attacks pose a serious threat
   No longer reasonable to assume hardware security to be inviolable

Abandon hardware-assisted TEEs altogether?
   Instead rely only on cryptographic techniques like MPC?

TEEs still hold the promise of efficient solutions
   Hardware-assistance and cryptography are not mutually exclusive!
   Defense-in-depth is desirable
   Novel approaches for dealing with TEE compromise may be feasible
Challenge: Dealing with TEE compromise

Prevent
Formal verification
Minimal coupling between TEE and REE

Tolerate
Replication / redundancy
Application-specific mitigation
Formal Verification

Formal verification can prevent software vulnerabilities

Good track record with protocol specs and implementations

- TLS 1.3 [CMSv16]
- miTLS, a formally-verified TLS implementation [BFKPS13]

Applicable to platform security

- seL4, a formally-verified microkernel [K+09]
- ProvenCore, ProvenCore-M, commercial formally-verified kernels [L15]

Caveat: Formal analysis is only as good as the underlying model

- seL4 needed to be patched for Meltdown like everything else

Recall: TEE system architecture

Figure adapted from: Global Platform. TEE system architecture. 2011.
TEE hardware realization alternatives

**Legend:**
- **SoC**: system-on-chip
- **OTP**: one-time programmable

**Figure adapted from: Global Platform. [TEE system architecture](https://example.com). 2011.**
Minimal coupling

- TEE closely coupled to REE
  - Intel SGX
  - ARM Trustzone
  - TPM

- TEE more isolated from REE

Larger attack surface

Greater sharing of resources (OS services, cache, memory, processor)

Smaller attack surface
Minimal coupling in the real world

Discrete security processors in modern smartphones
• Apple Secure Enclave Processor (SEP)
  Apple, iOS Security Whitepaper, May 2019
• Google Titan M
  Google Device Security Group, Building a Titan, Android Developers Blog, October 2018

Physical isolation mitigates against entire classes of hardware-level exploits
• Processor, caches, memory, and persistent storage are not shared with main OS
Layered defense using multiple TEEs

Enables division of tasks (and secrets) between two (or more) elements

**Improved security for stored secrets**

**Sensitive peripheral management (e.g. camera LED indicator, microphone disconnect)**

**Trusted path (e.g. isolated circuit to side buttons)**

**Insider attack resistance (e.g. firmware updates require device owner’s cooperation)**
Application-specific mitigation

Premise: Exploiting a hardware compromise may leave tell-tale signs

Approach: Use application-specific domain knowledge for detection or mitigation of the effects of hardware compromise
Example: Proof of Elapsed Time (PoET)

PoET is a replacement for Proof of Work in Bitcoin-like blockchains

**Proof of Work:**
First miner to **solve puzzle** wins (gets to proposes next block)

\[
\text{Work} \sim \text{Exp (difficulty)}
\]

*Proposals can be made at a rate proportional to computational power*

**Proof of Elapsed Time:**
TEE issues **attestation** after waiting (idly) for a while; First miner to get the attestation wins

\[
\text{Idle wait time} \sim \text{Exp (difficulty)}
\]

*Proposals can be made at a rate proportional to the number of idle CPUs*

*Intel, Hyperledger Sawtooth Documentation, 2015*
Example: Dealing with TEE compromise

Problem: A compromised TEE can win every block

Statistical solution: refuse blocks from machines that have won too many times
• Before: compromised TEEs give attacker unlimited power
• After: attacker power proportional to # of compromised TEEs

“Design for Failure”

Open question: How can TEE-using applications detect/mitigate effects of TEE-compromise?

Cross-layer design for security

- Frank Piessens (2019)

Hennessy and Patterson on cross-layer design (for performance):
• “Achieving significant gains through such approaches will require a vertically integrated design team that understands applications, domain-specific languages and related compiler technology, computer architecture and organization, and the underlying implementation technology”
• “In this new era, vertical integration has become more important, and teams that can examine and make complex trade-offs and optimizations will be advantaged”

Such cross-layer design can have similar benefits for security
• Not surprising, as there are often significant trade-offs between security and performance
Carrying security information across layers

- Frank Piessens (2019)

Applications may have precise information about what data in the program is confidential

In state-of-practice compilation, this information is lost

By preserving this information during compilation, we can use it to selectively close micro-architectural channels used in transient execution attacks

Other types of hardware-assisted security
Software Attacks against TEEs


Protection against software attacks

Novel hardware architectures
   CHERI, fat pointers, …

Hardware extensions rolled out by processor vendors
   x86:
      • Memory Protection Keys (MPK)
      • Memory Protection eXtensions (MPX)
   ARM:
      • Pointer Authentication (PA)
      • Memory Tagging Extensions (MTE)
      • Branch Target Indication (BTI)
How to utilize hardware-security primitives?

New hardware primitives are being rolled out
• Can we efficiently combine them to achieve new security properties?

How do different techniques compare?
• e.g., ARM PA and ShadowStack achieve similar security for return-address protection?

Understanding can help hardware vendors to choose which mechanisms to deploy
Forthcoming IEEE SP Special Issue

IEEE Security and Privacy Magazine
Special Issue on Hardware-assisted Security
mid-to-late 2020

Submissions: Dec 22, 2019

https://computer.org/digital-library/magazines/sp/call-for-papers-special-issue-on-hardware-assisted-security
Takeaways

TEEs have been around for more than two decades
   Dominant design choice informed by cost and usability considerations

Unconditional trust in hardware-TEEs is no longer acceptable

TEEs are still useful: defense-in-depth, novel mitigations for TEE failure possible?

Other hardware-assisted security mechanisms to harden software are emerging

https://asokan.org/asokan/
@nasokan

Thanks to my students:
Lachlan J. Gunn, Hans Liljestrand, Thomas Nyman
Acknowledgments

Icons on Platform security for mobile devices and Dealing with TEE compromise made by those-icons, freepik, and Good Ware from www.flaticon.com licensed by CC 3.0 BY
Icons on Mobile TEEs: Motivation made by Good Ware from www.flaticon.com licensed by CC 3.0 BY
Nokia 6630 image by JotWu from en.wikipedia.org licensed by CC 3.0 BY
Web icon on title slide from material.io licensed by Apache 2.0