Common-sense applications of hardware-based TEEs

N. Asokan

http://asokan.org/asokan/
@nasokan

Acknowledgements: Thomas Nyman, Lachlan Gunn
Hardware-security mechanisms are pervasive

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

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

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
Distiguished 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. The vulnerability is a special cache timing 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-2018-3639) targets data from Sgx-encrypted, and the second version (modified Foreshadow-Adv) (CVE-2018-3639M and CVE-2018-3639Ml) targets Intel traces (VTM), hypervisors (VMW), operating system (OS) kernel memory, and System Management Mode (SMM) memory. Foreshadow consists of two classes of speculative execution side-channel vulnerabilities: 1. "Terminal Fault" (v1.1) and 2. "A rendering 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 break into secure areas even the Sgx and Meltdown flaws couldn’t crack." 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. However, 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, individually; however, may not likely see any performance impact, according to researchers. The real fix, according to Intel, is by replacing today’s processors with future Intel processors. These changes begin with our next-generation Intel Xeon Scalable processors (code-
Concerns with TEEs: suspicions of motives

**Software**

MS Palladium protects IT vendors, not you – paper

Anderson gives us the FAQs

By John Lettece 28 Jun 2002 at 10:27


---

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


---

Trust Intel – Next Generation of Backdooring?

We have seen that SGX offers a number of attractive functionality that could potentially make our digital systems more secure and third party servers more trusted. But does it really?

The obvious question, especially in the light of recent revelations about NSA backdooring everything and the kitchen sink, is whether Intel will have backdoors allowing “privileged entities” to bypass SGX protections?

Possible motivations for widespread deployment

Vendor lock-in

Restriction of digital rights

... 

Regulatory requirements

Protection of end-user data

...
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 (ca. 2001) → TrustZone®

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

Business requirements:
• mobile payment
• subsidy lock
• custom silicon consolidation

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

Supply-chain 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
  "Secure microprocessor/microcomputer with secured memory"

1996
- Intertrust
  "Systems And Methods For Secure Transaction Management And Electronic Rights Protection"

2002
- Nokia
  US9111097B2
  "Secure execution architecture"

2003
- Texas Instruments OMAP 161x and 73x processors

2004
- ARM TrustZone
  Tiago Alves and Don Felton

2008
- Nokia
  US9111097B2
  "Secure execution architecture"

1982 Texas Instruments, Guttag US4521853A
  "Secure microprocessor/microcomputer with secured memory"

1982 Texas Instruments, Guttag and Nussarallah US4521853A
  "Security bit for designating the security status of information stored in a nonvolatile memory"
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 academic circles**
  - first paper in FC 2008, ASIACCS 2009
- **Intel SGX**
  - SkyLake microarchitecture (2015)
  - wide availability of SDK “democratized” TEE research
Should we build systems that rely on TEEs?

Concerns with applicability of hardware-supported TEEs remain

But compelling common-sense applications exist

practical; protect end-users; address everyday needs

- Private membership test for malware scanning, private contact discovery,..
- Protection of password-based web authentication
- Secure accounting for function-as-a-service (FaaS) settings
- Blockchains and cryptocurrencies
- …
Can blockchains be made better using hardware-assisted security?

Lachlan J. Gunn, N. Asokan
Proof of Work + “longest chain” rule

Bitcoin, Ethereum, etc. all use Proof of Work to agree on the next block:

Miners decide which transactions include in their proposal for the next block
Proof of Work: use computation power to solve a puzzle; winner proposes next block
  • Chance of success proportional to amount of computation (work) performed
  • Fair: any miner expending the same amount of work has the same chance of winning

Miner 1 ➔ Miner 2 ➔ Miner 3

• Everyone follows the longest valid chain (chain with largest CPU power wins eventually)
What’s wrong with Bitcoin, anyway?

The luxury of not trusting anyone does not come for free:

- All transactions need to be **online**
- Slow: long confirmation time, low throughput
- Wasteful (energy expended on puzzle solving)
- Probabilistic finality
- Extremely **scalable**

![Annual Power Consumption](Data: Digiconomist, CIA World Factbook)

- Finland
- Belgium
- Philippines
- Venezuela
- Bitcoin
- Austria
- ETH
- Visa
Can hardware-assisted security improve blockchains?

Example approaches

• Changing the “business process”
• Replacing consensus (“longest chain” rule)
• ...

What challenges arise?
Changing the process
Proof of Elapsed Time

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
Replacing Consensus
Byzantine Consensus

Goals of classical Consensus schemes:
• Liveness: all (honest) nodes produce output
• Safety: all (honest) nodes output same value
• Finality: output values are definitive

Adversary model:
• Adversary can compromise some nodes
• Goals hold despite $f$ compromised nodes

Limits:
• No protocol can tolerate more than a third of nodes being compromised
The first practical protocol for Byzantine fault tolerance

Less scalable than Proof of Work.

$O(n^2)$ messages, $n = 3f + 1$

The landscape of consensus mechanisms

How can TEEs help design scalable consensus?

**Problem:** Compromised nodes can equivocate

**Solution:** Use attestation to prevent equivocation!
- Tolerate faults in \( \frac{1}{2} \) of the nodes

Applicability limited to permissioned settings

Chun et al., “Attested append-only memory: making adversaries stick to their word”, SOSP ‘07
MinBFT

Hardware-based monotonic counters → increase fault-tolerance

FastBFT

TEE-protected secret sharing, message aggregation → increase throughput

$O(n)$ messages, $n = 2f + 1$

Challenges
Challenges in relying on hardware-assistance

**TEE Availability:**
- TEEs will not be universally available:
  - Gradual rollout
  - Obsolescence
  - Revocation

**TEE Compromise:**
- Compromising some TEEs should not completely break the system
**Example: Dealing with TEE availability in consensus**

**Question:** Can we improve consensus protocols by adding only a few TEEs?

**Answer:**
- can increase throughput if $\#\text{TEEs} > 1$
- but fault tolerance cannot be increased if $(\#\text{TEEs} / \#\text{Nodes}) \leq 2/3$

**Open question:** (How) can we optimally increase fault tolerance when $2/3 < (\#\text{TEEs} / \#\text{Nodes}) < 1$

Example: Dealing with TEE compromise in PoET

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?

Concerns with applicability of hardware-supported TEEs remain

But compelling common-sense applications exist be practical; protect end-users; address everyday needs

Solutions must incorporate mitigations for:
TEE unavailability or compromise

Design for failure application- or system-level mitigations possible
On dealing with TEE compromise

Two types of settings where TEEs are useful:
1. Improving **functionality** without compromising security: e.g., PoET
2. Improving **security** (esp. where none exists today): e.g., SafeKeeper

TEE compromise is a major concern in Type 1 settings

In Type 2 settings, TEE compromise implies returning to current situation