Secure Processors in Industry

Mengjia Yan
Fall 2020

Based on slides from Christopher W. Fletcher and Jakub Szefer
Reminder

• Fill the google form
  • [https://forms.gle/G6gh6sEYJ4UY24ePA](https://forms.gle/G6gh6sEYJ4UY24ePA)

• First review will be due @ 09/27 (2.5 weeks from now)
Recommended Reading

• *Intel SGX Explained; Victor Costan, Srini Devadas*
  • Great refresh on computer architecture
  • Background on cryptographic
  • Basic SGX programming model and architecture support (next lecture)
Outline

• IBM secure coprocessor 3848 and follow-ons
• Trusted Platform Module (TPM)
• Intel TXT, AMD

• Arm TrustZone
• Intel SGX
• AMD SEV
Physical Attacks
Computing Model

Processor Chip (socket)
- core L1/L2
- ... core L1/L2
- LLC

Processor Chip (socket)
- core L1/L2
- ... core L1/L2
- LLC

Memory (DRAM)

Non-volatile storage device

ASIC/Co-processor

System Bus (logically)

other I/O Devices
Hardware Adversary

• Pre-fab adversary (HW trojans)

• Physical attacks
  • Generally require physical access
  • Classified according to cost
  • A cold boot attack example

Advanced Hardware Hacking Techniques; Joe Grand; DEFCON’12
A Cold Boot Attack Example

https://www.youtube.com/watch?v=vWHDqBV9yGc

Gutmann et al. “Data Remanence in Semiconductor Devices”
More Physical Attack Examples

Tap board used to intercept data transfer over Xbox's HyperTransport bus from [http://www.xenatera.com/bunnie/proj/anatak/xboxmod.html](http://www.xenatera.com/bunnie/proj/anatak/xboxmod.html)

Physical Tamper Resistance

- Standalone security modules to protect cryptographic keys and personal identification numbers (PINs)
- A history lesson of physical security by IBM 4758

**Tampering Detection**
- Robust metal enclosures.
  - Open the lid → disconnect power supply
- Drill through the lid
- Photocells and tilt devices

**Tampering Evident**
- “Potting” the device in a block of epoxy resin
- Patience: Scrape away the epoxy
- Tamper-sensing barriers: nichrome wire wound around the device
IBM 4758 Secure Co-Processor

- Memory remanence
  - constant movement of values from place to place
- Cold boot
  - detects changes of temperature
- X-ray
  - a radiation sensor
- Power side channels
  - Solid aluminium shielding and a low-pass filter (a Faraday cage)

Photo of IBM 4758 Cryptographic Coprocessor (courtesy of Steve Weingart) from https://www.cl.cam.ac.uk/~rnc1/descrack/ibm4758.html

Expensive. Other secure processors only focus on a limited set of physical attacks.
IBM 4758 and Follow-ons

- The first FIPS 140-1 Level 4 validation, arguably the only general-purpose computational platform validated at this level by 2001
- A multipurpose programmable device
- Secure Boot and SW attacks (discussed later)

From Dyer et al. “Building the IBM 4758 Secure Coprocessor”

Bond et al. “API-Level Attacks on Embedded Systems.”
Trusted Platform Module (TPM)

- “Commoditized IBM 4758”
- Standard LPC interface – attaches to commodity motherboards
- Weaker computation capability

Uses:
- Verify platform integrity (firmware+OS)
- Disk encryption and password protection
Software Attacks
Software Stack

Intel’s Privilege Level

<table>
<thead>
<tr>
<th>Less Privilege</th>
<th>More Privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring 3</td>
<td>Ring -2 SMM (firmware)</td>
</tr>
<tr>
<td>Application Enclave application</td>
<td>Hypervisor (VMM)</td>
</tr>
<tr>
<td>Ring 2</td>
<td>Ring -1 SMM: system management mode</td>
</tr>
<tr>
<td>Ring 1</td>
<td>Ring 0 Guest OS</td>
</tr>
<tr>
<td>Ring 0</td>
<td>Ring -2 SMM (firmware)</td>
</tr>
<tr>
<td>OS kernel</td>
<td>Host operating system/Hypervisor</td>
</tr>
<tr>
<td>SMM</td>
<td>Hardware</td>
</tr>
</tbody>
</table>

User application

Host operating system/Hypervisor

Hardware
Process Isolation When Sharing Hardware

• Share HW resources in SMT contexts, same processor chips, across sockets.
Virtual Address Abstraction

Benefits of virtual memory abstraction:
- Over-commit memory: the illusion that they own all resources
- Security: process isolation
- Programmability: software independent of DRAM size
Page Table

- Page table:
  - A data structure to store address translation entries
  - Multi-level trees

- Page table entry attributes:
  - Writable (W), Executable (X), Supervisor (S), etc.
  - E.g., data execution prevention (DEP)

- MMU (memory management unit)
  - A hardware unit performs address translation

- TLB:
  - Caches for page tables
Trusted computing base (TCB)

- Trusted computing base (TCB)
  - TCB is trusted to be correctly implemented
  - Vulnerabilities or attacks on TCB nullify TEE protections
  - TCB may not be trustworthy

- Attacks, e.g., Rootkit, may change the **integrity** of TCB

- How to verify platform (HW + low-level SW) integrity
Platform Initialization (Booting)

For remote system management. Also manage booting.

6.888 L2 - Secure Processors in Industry
Boot Process (UEFI)

How to perform the measurement?

Root of trust
Cryptographic Hashing (e.g., SHA 1-3)

\[ \text{Hash}(m) = h \]

- **One-way hash**
  - Practically infeasible to invert, Difficult to find collision

- **Avalanche effect**
  - “Bob Smith got an A+ in ELE386 in Spring 2005”→01eace851b72386c46
  - “Bob Smith got an B+ in ELE386 in Spring 2005”→936f8991c111f2cefaw

Use as fingerprints
Secure Boot using TPM

- Static root of trust for measurement (SRTM)

1. Measure (hash)
2. Report (extend)
3. Load

PCR: platform configuration register

Compared to expected values locally or submitted to a remote attester.
Software Attestation

• Report a measurement list to a remote verifier
• Problem: How can the verifier know the list is not faked?
Public Key Cryptography (e.g., RSA, EC)

• A pair of keys:
  • Private key ($K_{pri}$ – kept as secret); Public key ($K_{pub}$ – safe to release publicly)

• Encryption:
  • $\text{Encrypt(plaintext, } K_{pub}\text{)} = \text{ciphertext}$
  • $\text{Decrypt(ciphertext, } K_{pri}\text{)} = \text{plaintext}$

• Digital signatures:
  • Proof that msg comes from *whoever owns private key corresponding to* $K_{pub}$
  • $\text{Sign(msg):}$
    • $h = \text{Hash(msg)}$; signature = $\text{Encrypt(h, } K_{pri}\text{)}$
    • Return $\{\text{signature, msg}\}$
  • Verify:
    • $\text{Decrypt(signature, } K_{pub}\text{)} ?= \text{Hash(msg)}$
Software Attestation

Processor Chip (w/ BIOS, OS, Apps)

TPM

$K_{pri}$

sign

Measurement List

Verifier

Replay attack: How can the verifier know the list is the latest?

$K_{pub}$
Software Attestation

- Defend against replay attack: Freshness

Processor Chip (w/ BIOS, OS, Apps)

TPM

$K_{pri}$

sign

Measurement List + nonce

nonce

Verifier

$K_{pub}$

How to know this key belong to a specific TPM?
Software Attestation

- Need public key infrastructure

Processor Chip (w/ BIOS, OS, Apps)

\[ \text{TRM} \]

\[ \text{AIK}_{\text{pri}} \]

\[ \text{Measurement List + nonce} \]

Verifier

\[ \text{AIK}_{\text{pub}} \]

Chip Manufactory

\[ \text{Root Key} \]

\[ \text{RK}_{\text{pri}}, \text{RK}_{\text{pub}} \]

Works as Certificate Agent

Attestation

Identity Key

6.888 L2 - Secure Processors in Industry
Security Objectives Summary

• Privacy
  • Alice sends msg $m$ to Bob. Only Bob should be able to read $m$. (asymmetric or symmetric encryption)

• Integrity
  • Alice sends msgs $m_1 \ldots m_n$ to Bob.
  • Authenticity: Bob receives msg $p$. Bob can verify $p \in m_1 \ldots m_n$. (Hash)
  • Freshness: Bob has received msgs $p_1 \ldots p_n$. Bob can verify $p_i = m_i$. (Hash+nonce)

• Identity
  • Bob wants to know if Alice is really Alice.

• Availability
  • Does Bob ever see the n messages?

Protocols can be constructed using crypto primitives and infrastructures
Intel TXT

• Uses TPM for software attestation
• Dynamic root of trust for measurement (DRTM)
  • PCRs 17-22 are reset by the SINIT ACM, every time a TXT VM is launched
• Marketed as more secure, but there are various attacks targeting TXT
Open-source Choice: Google Titan

Security Vulnerabilities of Using TPM

• Vulnerable to bus sniffing attacks

• TPM Reset attacks
  • SW reports hash values

• Bugs in the trusted software

---

Han et al. A Bad Dream: Subverting Trusted Platform Module While You Are Sleeping. Usenix Security’18
Wojtczuk et al. Attacking Intel TXT® via SINIT code execution hijacking. 2011
So Far ……

The trend: shrink TCB. Why?

6.888 L2 - Secure Processors in Industry
Shrink Trusted Computing Base (TCB)

Arm TrustZone

Intel SGX

AMD SEV

6.888 L2 - Secure Processors in Industry
Arm TrustZone

Recommend not to have secret data in DRAM.

But just recommendation.

Cache is dynamically partitioned by tagging each cache line with a security bit.

from Hua et al. vTZ: Virtualizing ARM TrustZone. Usenix’17
Shrink Trusted Computing Base (TCB)

6.888 L2 - Secure Processors in Industry
Next Lecture:
Intel SGX