Intel SGX

Mengjia Yan
Fall 2020

Based on slides of Intel SGX Tutorial
Recap: Address Translation

Virtual Address Space (Programmer's View)

Page Table per process

Physical Address Space (limited by DRAM size)
Recap: Address Translation

Virtual Address Space (Programmer's View)

Virtual Address (VA) -> Page Table

Page Table per process

Physical Address Space (limited by DRAM size)

Physical Address (PA) -> 4KB Pages

System software handles "page fault"

6.888 L3 - Intel SGX
Recap: Process Isolation

Virtual Address Space (Programmer's View)

Physical Address Space (limited by DRAM size)

Process 1
- Page Table per process
  - VA
  - PA

Process 2

4KB

4KB

4KB

4KB

6.888 L3 - Intel SGX
Page De-duplication and Copy-on-write

If two pages have the same content, use page de-duplication to reduce memory footprint.
Page De-duplication and Copy-on-write

Upon write access, copy the page and change the mapping.

Virtual Address Space (Programmer's View)

Physical Address Space (limited by DRAM size)

4KB (copied)

Page Table per process

VA

PA

Process 1

Process 2

4KB

4KB

4KB

4KB

6.888 L3 - Intel SGX
Recap: Secure Boot

- Static root of trust for measurement (SRTM)

TPM + firmware

1. Measure (hash)

Boot Loader

2. Report (extend)

OS kernel

3. Load

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

PCR: platform configuration register
Software Attestation

• Defend against replay attack: Freshness

Processor Chip
(w/ BIOS, OS, Apps)

TPM
$K_{pri}$

measurement
List + nonce

Verifier

How to know this key belongs to a specific TPM?

nonce

6.888 L2 - Secure Processors in Industry
Software Attestation

• Need public key infrastructure

Processor Chip (w/ BIOS, OS, Apps)

TPM

AIK_{pri}

Chip Manufactory

Root Key

\[ RK_{pri} \]

\[ RK_{pub} \]

Attestation Identity Key

\[ AIK_{pub} \]

Verifier

Send AIK_{pri} to TPM using secure communication protocol

Measurement List + nonce

Sign

Works as Certificate Agent
Intel TXT, AMD PSP, Google Titan

Intel TXT Dynamic trust of measurement


6.888 L2 - Secure Processors in Industry
Security Vulnerabilities of Using TPM

- Vulnerable to bus tapping 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
Why Shrink TCB?

• Software bugs
  • SMM-based rootkits
  • Xen 150K LOC, 40+ vulnerabilities per year
  • Monolithic kernel, e.g., Linux, 17M LOC, 100+ vulnerabilities per year

• Remote Computing
  • Remote computer and software stack owned by an untrusted party
  • Examples
Secure Remote Computing

• Example: Video processing
Shrink Trusted Computing Base (TCB)

Arm TrustZone

Intel SGX

AMD SEV

Trusted
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
Privileged Software Attacks

• Manipulate everything

• Directly see and modify application code and data
  → Need to encrypt secret data
  → Need to verify integrity (software attestation)

• Mess up with
  • Address translation
  • Process initialization and context switch
  • Interrupts, I/Os
  • etc.

→ 6.888 L3 - Intel SGX 17
Enclave High-level View

• Goal: A protected environment that contains the code and data of a security-sensitive computation.
SGX HW TCB

Low impact on chip’s HW design

Processor Chip (socket)

- core
- L1/L2
- LLC

Memory Management Unit (MMU)

- Integrated Memory Controller

System Bus (logically)

Memory (DRAM)

Non-volatile storage device

other I/O Device

6.888 L3 - Intel SGX
Intel SGX Security Mechanisms

**Isolation**

- Ring 3
- Ring 0
- Ring -1
- Ring -2

**Enclave**

- App
- Guest OS
- Hypervisor
- SMM
- Hardware

**Attestation**

- Client Application
- Enclave
- Remote Platform

**DRAM Protection**

- Processor Chip (socket)
- Core L1/L2
- Core L1/L2
- ... LLC
- Integrated Memory Controller
- Memory (DRAM)

---

6.888 L3 - Intel SGX
SGX Access Control

• Assume software attestation is done
• Can have multiple enclaves

Performance issues.

PRM size is 128MB in SGX V1.0
*All enclaves loaded at the same time cannot exceed said ~90MB*
Enclave Address Translation

Virtual Address Space (Programmer's View)

Enclave Linear Range (ELRANGE)

4KB

Page Table per process

4KB

PA

This is what we want. But OS is not trustable.

Physical Address Space (limited by DRAM size)

4KB

Processor Reserved Memory (PRM)

4KB
Malicious Address Translation

Virtual Address Space (Programmer's View)

- Enclave Linear Range (ELRANGE)
- Page Table per process

Physical Address Space (limited by DRAM size)

if (PA belongs to PRM) {
    check whether in enclave mode
    if (NOT enclave access) {
        return a value 0xffffffff
    }
}

Easy to implement in MMU
Malicious Address Translation

Virtual Address Space (Programmer's View)

Enclave Linear Range (ELRANGE)

Physical Address Space (limited by DRAM size)

Page Table per process

if (in enclave mode) {
    compare PA with RPM range
    if (NOT in RPM) {
        #Signal Fault
    }
}

Also Easy to implement in MMU
Malicious Address Translation

Virtual Address Space (Programmer's View)

Physical Address Space (limited by DRAM size)

Page Table per process

4KB

Enclave Linear Range (ELRANGE)

4KB

Processor Reserved Memory (PRM)

4KB (belong to a different enclave)

4KB

How to block such attacks?

6.888 L3 - Intel SGX
SGX Memory Organization

- Keep page mapping metadata in PRM
- MMU performs extra checks

- Enclave pages (code, data)
- Meta data per enclave
  - enclave page mapping information, enclave thread context information, etc.
Enclave Page Mapping Information

Virtual Address Space (Programmer's View)

- 4KB
- Enclave Linear Range (ELRANGE)
- 4KB

Page Table per process

- VA
- PA

Physical Address Space (limited by DRAM size)

- 4KB
- Processor Reserved Memory (PRM)
- 4KB

Enclave Page Cache Mapping (EPCM)

- Stored in PRM
- {PA, VA, Enclave ID}

if (PA belongs to PRM) {
  compare VA in EPCM
  if (NOT match) {
    #Signal Fault
  }
}
Enclave Page Mapping Information

Virtual Address Space (Programmer's View)

- Enclave Linear Range (ELRANGE)
- Enclave Page Linear Range (EPCM)

Page Table per process

- VA
- PA

Physical Address Space (limited by DRAM size)

- Processor Reserved Memory (PRM)

Enclave Page Cache Mapping (EPCM)

Stored in PRM

{PA, VA, Enclave ID}

Problem: pages are allocated and selected by system software.
So far ......

• Once the enclave is initialized correctly, it can be isolated from system software using
  • Hardware access control (supported by MMU)
  • Hardware support for secure context switch

• How to ensure the initialization is correct?
  • Software Attestation (similar to secure boot)
Enclave Initialization

• BIOS setup PRM region
Enclave Initialization

• Enclave creation (ECREATE)
Enclave Initialization

- Add page (EADD)
- Measure (EEXTEND)
Enclave Measurement

• Hardware generates a cryptographic log of the build process
  • Code, data, stack, and heap contents
  • Location of each page within the enclave
  • Security attributes (e.g., page permissions) and enclave capabilities

• Enclave identity (MRENCLAVE) is a 256-bit digest of the log that represents the enclave
Enclave Initialization

- Add page (EADD)
- Measure (EEXTEND)
- Init (EINIT)
  - Finalize measurement
- Active (EENTER)
  - Switch to enclave mode

Problem:
No measurement after EINIT
Enclave Attestation and Sealing

- HW based attestation provides evidence that “this is the right application executing on an authentic platform” (approach similar to secure boot attestation)

HW-signed blob that includes enclave identity information

trusted communication channel

EREPORT
Protect Memory

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

- Integrated Memory Controller

- System Bus (logically)

- Memory (DRAM)
- Non-volatile storage device
- Other I/O Device
Confidentiality Protection with Encryption

- Secret key is stored inside chip
- For freshness, encrypt with nonce (counter)
- \{nonce, ciphertext\} per cache block are stored externally in DRAM
Integrity Protection with Hash

• For each cache line: \{\text{ciphertext} + \text{nonce} + \text{hash}\}

• Problem:
  • Need to store hashes or nonces on-chip \(\rightarrow\) high on-chip storage requirement
  • Too much storage requirement (~64bits / block) \(\rightarrow\) high off-chip storage requirement

• General solution:
  • Integrity Tree (Merkle tree)
Operations on Merkle Tree

- Only need to store the root node on chip
- How to verify block B1?
- Write to block B3?

root = Hash(f_{2i} || f_{2i+1})

f_{i} = Hash(g_{2i} || g_{2i+1})

g_{i} = Hash(h_{2i} || h_{2i+1})

h_{i} = Hash(B_{i})

Secure processor (trusted)

root

f_{0}  f_{1}

h_{0}  h_{1}  h_{2}  h_{3}  h_{4}  h_{5}  h_{6}  h_{7}
B_{0}  B_{1}  B_{2}  B_{3}  B_{4}  B_{5}  B_{6}  B_{7}
Next Lecture:
Side Channel Introduction