Trusted Computing

Chengyu Song

Slides modified from
Dawn Song

OS security

- How to protect users and the system (e.g. other apps) from malicious apps
  - Access control

Trusted computing

- From another perspective: what if I want to run my code on a platform where I don't fully trust the owner?
  - Public cloud
  - PC: digital right management (DRM)
  - Mobile: bring-your-own-device (BYOD)
- Trusted computing: **how to establish certain degrees of trust**

Security concerns

- How to protect the confidentiality of my app and data?
- How to protect the integrity of my app and data?
- How to make sure it's really my app (identity)?
- How to make sure my app has really been run properly (verifiable results)?

Threat model

- What is your trusted computing base (TCB)
  - Operating system?
  - System administrators?
  - Hypervisor?
  - Hardware?

Confidentiality

- Means to protect the confidentiality?
    - System approach: isolation
    - Cryptographic approach: encryption
- *How to compute over encrypted programs and data?
    - Secure multi-party computation
    - Homomorphic encryption

Side-channel attacks

- System approaches can only block *direct* channels, but information can also leak through *indirect* channels (a.k.a. side-channels)
- Two necessary conditions
  - Difference in behaviors
    - Access pattern, timing, power consumption, electromagnetic, acoustic, etc.
  - A shared media to observe the behavior

Integrity

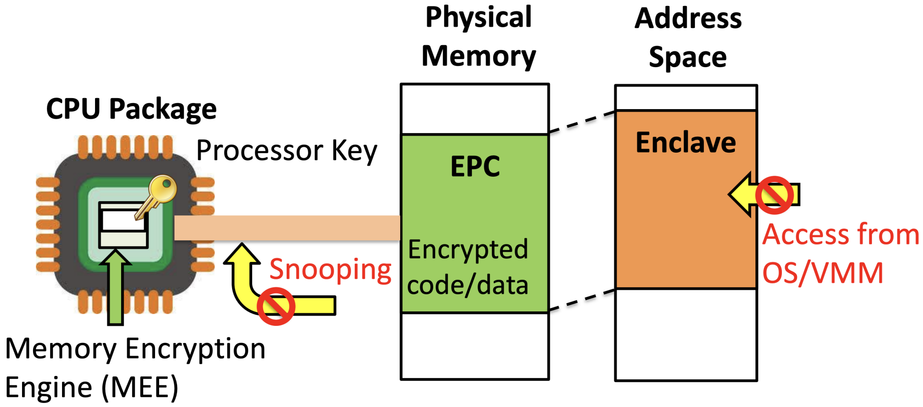
- Means to protect the integrity?
  - System approach: isolation
  - Cryptographic approach: MAC (message authentication code)
- How to scale?
  - Hash/Merkle tree

Replay attacks

- Using an old version to replace current version
- How to avoid?
    - Random nonce as challenge
    - Version

Operational mode

- In practice, we combine both system and cryptographic approaches
  - Isolation is more efficient, but has limited protection scope
  - Once the code/data leaves the protection scope, we rely on cryptographic



OpenSGX, NDSS 2016

How to provide trusted identity

- What is an identity?
  - A static/dynamic measurement
- How to establish trust
  - Root of trust

Root of trust

- A piece of hardware/software that is
  - Privileged enough for performing **measurement**
  - Capable of protecting itself
    - E.g., a standalone chip
  - Cryptographic provable identity
    - E.g., embedded private keys

Measurement

- A proof for the identity and integrity of system state
  - A chain of hashes
- Example: measured boot
  - Record the hash of the BIOS
  - Record the hash of the bootloader
  - Record the hash of the hypervisor/OS kernel
- How to record?
  - `new_hash = hash(old_hash || new measurement)`

Attestation

- A signed proof for the integrity measurement
  - Measurement results
  - A nonce to mitigate replay attack
  - Additional information from the software
  - Signed by a private key of the root of trust

Other operations

- Secure key generation and storage
- Seal: bind key to a measurement
  - E.g., only decrypt the disk image if the measurement of the OS is expected

Problems of integrity measurement

- Hidden assumption1: one must verify and trust the code
- Hidden assumption2: trust the binary
- Load-time integrity != run-time integrity
  - Why? Vulnerabilities!!

Commercial TEEs

- Intel SGX
- ARM TrustZone
- AMD SEV

SGX Applications (1)

- VC3: Trustworthy Data Analytics in the Cloud using SGX
- M2R: Enabling Stronger Privacy in MapReduce Computation
- SCONE: Secure Linux Containers with Intel SGX
- Oblix: An Efficient Oblivious Search Index
- Oblivious Multi-Party Machine Learning on Trusted Processors

SGX Applications (2)

- Ryoan: A Distributed Sandbox for Untrusted Computation on Secret Data
- Enhancing Security and Privacy of Tor's Ecosystem by using Trusted Execution Environments
- Secure Content-Based Routing Using Intel Software Guard Extensions
- SecureKeeper: Confidential ZooKeeper using Intel SGX

Attacks against TEE apps

- Side-channel attacks
- Exploit against vulnerabilities in TEE apps
- Iago Attack
    - Bad system calls