Secure Processor

Zhao Zhang, Fall 2005

Security

Authentication, Encryption, Integrity

An adversary should not be able to
- Pretend to be someone in communication
- Understand the content
- Change the content without being detected

Why Security by Hardware

System Security: Secure OS, application software, etc
Why offer security by Hardware?
- Protect intellectual property
- To counter physical attacks
- Support secure computing in a non-secure environment

Scenarios of Secure Computing

- A copyrighted program runs only on a validated hardware platform
- A workstation running in a public lab
- A server runs programs submitted by a remote use
- A client submit programs to a remote server
- A centralized server collects information from distributed sensor nodes

Software-Only Approaches?

A server may handle spurious programs without hardware support
- Binary instrumentation
- OS call interception
- Virtual machine
- But hardware support can be more efficient

How about the other scenarios?

Secure Co-Processor

"Applications cannot be more secure than the kernel functions they call, and the operating system cannot be more secure than the hardware that executes its commands."
Hardware Security Support

Tamper-evident environment (TE)
Ensures that any physical or software tampering to alter the behavior, state or data of a program will be detected.

Private tamper-resistant environment (PTE)
Protect the confidentiality of a program and detect tampering.

Hardware Security Support

Must counter many types of physical attacks from an adversary:
- A logic analyzer may read memory transactions between processor and memory.
- Freezing or X-raying a chip may keep its memory contents.
- EM signals or power variation may disclose the activities inside a processor.

Secure Co-Processor

It is feasible to protect a small box against physical tampering:
- Use an enclosure to protect the whole unit (processor, memory, bus).
- Use tamper sensors to detect opening or drilling.
- Take use of temperature and radiation sensors.
- Use aluminum enclosure to reduce radiation.
- Use long-lived batteries to power the co-processor.


Secure Co-Processor

Uses of secure co-processor:
- Check the integrity of host at booting.
- Audit trails to ensure data integrity.
- Provide copyright protection.
- Keep core secrets of large applications.

Use cryptographic checksums, public key encryption.

IBM 4758 co-processor

Application software is built upon secured hardware and firmware.
Private and Tamper-Resistant Running Environment
- XOM: eXecute Only Memory
- AEGIS
  - Suh et al. “Architecture for Tamper-Evident and Tamper-resistant Processing”

XOM Architecture
Objective: Programs run without being copied or tampered
- Prevent hackers
- Protect copyright
- No trust on OS

Approaches: Provide a compartment for each program

Software Distribution Method
Source: Lie’s talk in ASPLOS’00

Loading Secure Code
Source: Lie’s talk in ASPLOS’00

Security Boundary
Anything outside the processor is not trustable
- Anything stored in off-chip cache and main memory must be encrypted

Memory Encryption
XOM Key Table (Store symmetric Session keys)
Protection Against OS

OS needs to save and restore program states on context switches

- On interrupt, register values are encrypted
- OS can save and restore them but cannot read or change them

AEGIS

Share merits of XOM and provide several enhancements over XOM

- A memory integrity verification scheme based on hash tree

Other Work

- Improvements on memory encryption and integrity verification
- Hardware support for intrusion detection
  - Buffer overflow detection
  - Track of information flow
- Trusted Computing Platform Alliance
  - Led by Intel

AEGIS

- Fixed a vulnerability of XOM on memory replay attack
- Can support a larger number of processes with the memory integrity verification
- Other improvements on processor efficiency