Trusted Hardware

Sai Krishna Deepak Maram, CS 6410
Move to a Cloud-based model

User apps
Software
Hypervisor
OS

PaaS
Cloud Provider manages the stack

User apps
Software
Hypervisor
OS
Move to a Cloud-based model

User apps
Software
Hypervisor
OS

Privileged code
Software
Hypervisor
OS

Malicious cloud provider?
Can you trust the cloud?

- Huge Trusted Computing Base (TCB)
- Cloud Provider's software
- Management stack
- Sysadmins
What do we want?
Shielded Execution using SGX

Confidentiality: The execution state is unobservable to the rest of the system.

Integrity: If the program completes, its output is the same as a correct execution on a reference platform.
Is shielded execution sufficient?
Remote attestation

- **Goal:** Allow cryptographic verification that specific software has been loaded within an enclave
  - While an enclave is initialized, its contents is cryptographically hashed by the CPU forming the enclave’s measurement
- Generated using a key burnt on the SGX chip
  - Root of trust: Intel
  - Intel attestation service (IAS) for verification
How does SGX achieve this?
Memory protection

- **EPC** (Enclave Page Cache)
  - A separate region in physical memory
  - All enclave pages reside here
  - Hardware tracks meta info corresponding to each page
  - Virtualized
Memory protection

- EPC (Enclave Page Cache): A separate region in physical memory
  - Encrypted and integrity-protected before writing to the main memory

- Same page table as the underlying OS
  - Access checks are performed to ensure any other application (not even other enclaves) can access an enclave's data
Execution lifecycle (high-level)

- Loading stage: Performed by untrusted code
  - Enclave is initialized by copying code/data into EPC Pages
  - At the end of which, contents are hashed to compute enclave’s measurement hash
- Enclave mode:
  - Special instructions to create an enclave, add pages to enclave and exit an enclave
  - Similar to switching from user to kernel mode
  - Secure mechanisms to handle interrupts (or) page faults to protect from OS exception handlers
Before SGX?
Trusted Platform Module (TPM)

- Attestation-based
- Can be used with commodity systems
- **Weak security**
  - Much bigger TCB than SGX: Measurement hash covers all the OS modules and device drivers
  - Very hard to keep an up-to-date list of the hashes
  - Many more attacks....
How to port legacy applications into SGX?
Developing applications in SGX

- Untrusting OS: Makes it harder
- Any function call (or) syscall made outside the enclave are not guaranteed to return
- Even if data returns, enclave cannot trust the data returned
Haven

- Haven design goals:
  - Mutual distrust b/w guest and host
  - Run legacy apps inside SGX without any modifications
- Application interacts only with LibOS
  - Assumes libOS is carefully implemented
- Shield module interacts with the untrusted host OS

Figure 2: Haven components and interfaces
How Haven handles Iago attacks

Iago attacks: “Malicious kernel attempts to subvert an isolated application by exploiting its assumption of correct OS behaviour, for example when using the results of system calls”

LibOS: Implement entire OS as part of the *Trusted Computing Base*. Limits the interaction of enclave app with the actual OS, thus reducing the attack surface.
Haven

- Unmodified binaries
- Subset of Windows, enlightened to run in-process
- Shields LibOS from lago attacks
  - Includes typical kernel functionality
    - Scheduling, VM, file system
  - Untrusted interface with host
- Picoprocess (protects host from guest)
- Enclave (protects guest from host)
  - Application
  - Library OS (Drawbridge)
  - Shield module
  - Mutual distrust
  - Untrusted runtime
  - Drawbridge host
  - Windows kernel
  - SGX driver
LibOS and Exokernels

Both bring OS level functionalities to the user space, but for what reasons?

- Efficiency in Exokernel: “Move OS functionality to the user space to grant more flexibility”
- Security in Haven’s LibOS: “Move OS functionality into the enclave to reduce attack space”
Haven Performance

- 35% - 65% slowdown
- Depends on the exact use case
Haven influencing SGX
Haven influencing SGX design

- Dynamic memory allocation
  - SGX does not allow addition of enclave pages after the creation of enclave
- Exception Handling
  - SGX does not allow handling of all exceptions
- Some other limitations

Fixed in v2.0
SGX: What's new?

- Latest v2.3
  - Trusted randomness, other crypto operations
  - File abstractions inside an enclave
- Baidu’s Rust [SGX SDK](https://github.com/baidu/sgx-sdk)
  - Dockerized
  - Runs a simulated version on machines without SGX chip as well
Is SGX secure?

- Sophisticated side channel attacks
- [Foreshadow](#) - Usenix’18
  - Speculative execution
Trusted hardware makes the attacker’s job costly
Discussion

Haven
- Exokernel connection to Haven
- Impact of Haven and why it’s not more widely used?

SGX
- Does trusted hardware solve the problem of security in the cloud?
- Can SGX still be useful in face of side channel attacks?
Thank you!

References:
1. Haven, Slides
2. Intel SGX explained