Advanced Systems Security: Attacks on SGX

Trent Jaeger
Systems and Internet Infrastructure Security (SIIS) Lab
Computer Science and Engineering Department
Pennsylvania State University
Intel SGX

- Hardware support that eliminates need to trust the operating system
  - Aim to prevent “cold boot” attacks
  - Does it prevent all OS attacks?
- Some types of attacks become more significant when you do not trust the operating system
  - Iago attacks
  - Side channels
  - Runtime attacks (ROP)
Cold Boot Attacks

- An attacker with physical access to a computer is able to retrieve encryption keys from a running operating system after using a cold reboot to restart the machine.

- **Problem:** Transient memory may retain values across reboots - for hours by cooling them with a refrigerator.

- Assume you have a system that has been booted securely, so it runs only secure software.
  - And you want to extract secret keys used by such a machine.

- **Attack**
  - Memory modules are removed from victim system.
  - Place in a compatible machine under the attacker's control, which is then booted to access the memory.
SGX Blocks Cold Boot Attacks

- How does SGX prevent the Cold Boot attack?
Threats to SGX Processes

• However, threats remain for SGX processes
  ‣ What do you think are the sources of threats?
Threats to SGX Processes

• However, threats remain for SGX processes
  ‣ What do you think are the sources of threats?
  ‣ All the untrusted software – especially the operating system
Operating System Is Threat

• Since the operating system was built to be trusted, it performs actions that may be exploited against SGX
  ‣ That have not typically been exploited
    • At least not to this extent

• Types of attacks
  ‣ Iago attacks
    • Attacks through system call responses
  ‣ Side channel attacks
    • Attacks through shared storage and/or operation timing
Iago Attack

- What is one major thing we depend on from the OS?
  - System call responses

- While it is hard to prove that an operating system should be trusted (e.g., verification in the reference monitor concept), we typically assume the OS is benign

- But, what if it is not
  - Iago attacks paper – Checkoway and Shacham [ASPLOS 2013]
  - **Definition:** Attacks in which a malicious kernel induces a protected process to act against its interests by manipulating system call return values
Iago Attack

- Example
  - Kernel becomes an *active network adversary* for a trusted application that needs to communicate remotely
  - Why is this an issue?
Iago Attack

• Example
  ‣ Kernel becomes an **active network adversary** for a trusted application that needs to communicate remotely
  ‣ Why is this an issue?

• Trusted inputs obtained from kernel to perform crypto operations
  ‣ Kernel can manipulate /dev/random
  ‣ VMM could prevent such an action
    • But attack is more subtle
Iago Attack

- Example
  - Kernel becomes an active network adversary for a trusted application that needs to communicate remotely
- Application depends on kernel for inputs to crypto
  - Kernel could replay the client connection’s messages from one client for a fake client
  - Kernel could return same values for getpid and time as prior connection to reduce entropy
  - Even getpid is an issue – used as a non-repeating nonce for Apache child process, but malicious OS can repeat PIDs
Iago Attack

• Example
  ‣ Kernel becomes an **active network adversary** for a trusted application that needs to communicate remotely

• Application depends on kernel for inputs to crypto
  ‣ Kernel could replay the client connection’s messages from one client for a fake client
  ‣ Kernel could return same values for `getpid` and `time` as prior connection to reduce entropy
  ‣ **Even if trusted entity (VMM or SGX) is used for time source, the kernel can replay with limit (same second)**
Side Channels

• Another challenge is created by side channels available in computing systems
  ‣ Side channels are channels created as side effects of an implementation
  ‣ Rather than channels designed into a system

• An adversary may learn unauthorized information via side channels, as they are not monitored
  ‣ Typically, a victim – with access to secret data – produces a signal on one or more side channels
  ‣ An adversary can also take actions to increase the bandwidth and reliability of the side channel
Side Channels

• Classic side channel attacks measure the time for the victim to perform an operation using secret data

• Timing channels
  ‣ Can attack a cryptosystem if an operation takes a different amount of time based on the inputs provided, such as the key value
  
  • Does your program have an algorithm whose execution time is dependent on the value of secret inputs?
  
  • Square-and-multiply and modular exponentation algorithms used in cryptography have different execution times depending on the number of ‘1’ bits in the input
SGX Side Channels

• The SGX approach results in a variety of side channels because we do not trust any other software
  ‣ Page faults
    • Noise-free, but coarse-grained (page granularity)
  ‣ Measure cache hit/miss timing
    • Fine-grained (cache line granularity), but can be noisy
  ‣ Branch prediction
    • Other paper
    • Can manage execution in a fine-grained way using small time slices
Cache Channels

- The SGX approach results in a variety of side channels because we do not trust any other software
  - One popular kind of side channel is a cache side channel
  - In a cache side channel, the adversary primes (fills) or flushes (invalidates) cache entries shared with the victim to detect victim accesses

- One attack PRIME and PROBE
  - Fill a cache line shared with a victim – subsequent access by adversary will show a slowdown if victim accessed entry

- If cache line use depends on input value – detect value
Cache Channels

- The SGX approach results in a variety of side channels because we do not trust any other software
  - One popular kind of side channel is a cache side channel
  - In a cache side channel, the adversary primes (fills) or flushes (invalidates) cache entries shared with the victim to detect victim accesses
- One attack **FLUSH and RELOAD**
  - Flush cache line with `clflush` and reload after victim runs to detect performance
- **Advantage**: Flushes LLC which applies to all cores
Runtime Attacks

- SGX may have side channels, but at least it runs programs in a manner that is encrypted to adversary
- Should make some runtime attacks harder
  - Such as return-oriented attacks
- But does it?
Take Away

• Problem: Do not want to trust systems software
  ‣ However, we have not considered the OS as an adversary deeply yet

• Attacks
  ‣ Iago attacks – OS as an active man-in-middle
  ‣ Side channel attacks – even more side channels and more effective attacks when controlled by the OS
  ‣ Runtime attacks – still possible against encrypted processes

• Lots of future work to close these holes