SGX-BOMB:
Locking Down the Processor via the Rowhammer Attack

Yeongjin Jang*, Jaehyuk Lee†, Sangho Lee‡, and Taesoo Kim‡

Oregon State University*
KAIST†
Georgia Institute of Technology‡
TL;DR

- SGX locks up the processor to protect enclaves on any integrity violation

- SGX-BOMB is a software attack that intentionally violates an enclave’s integrity to launch a DoS attack via the Rowhammer attack

- Processor does nothing if the attack is triggered and requires cold reboot

- Hard to detect this attack because it runs in an enclave

- Attackers could lockdown not only a user’s machine but also a cloud server by launching this attack
SGX Encrypts an Enclave’s Memory

• Memory Encryption Engine (MEE) handles the encryption

• Encrypts enclave’s data with processor’s key

• Attackers on the DRAM cannot see plaintext
  • Confidentiality

• Attackers could tamper ciphertext but...
  • Processor will authenticate data (integrity)

• Protect an enclave from hardware attackers
Integrity Tree Protects the Integrity of EPC

- Integrity Tree
  - A version tree that stores hash of data
  - Rooted at on-die SRAM
  - Parent node contains the hash of its children nodes
  - Updated on each write and checked on each read
  - Any integrity violation can be detected on read access
Intel Assumes Only Hardware Attackers Can Launch Attacks on EPC

• Processor isolates EPC from non-enclave accesses
  • Redirect all access to EPC to an abort page (if the origin is not a right enclave)
  • Return 0xffffffffffffffff for all memory read and ignore write
  • Rely on an extension to page table handler

• Threat model
  • Software attacker cannot access (read/write) to the EPC region
  • Only hardware attacker can tamper the integrity of ciphertext
On Integrity Violation

• Integrity violation infers an existence of a hardware attacker

• Intel took the *drop-and-lock* policy
  • Processor locks up the memory controller to stop running, to block any further damage on enclaves by the hardware attackers

• The processor must be rebooted
On Integrity Violation

• Integrity violation infers an existence of a **hardware attacker**

  No, that’s not true. Attackers can induce **bit-flips** in DRAM without directly accessing them by launching the **Rowhammer** attack in **software**

• Intel took the **drop-and-lock** policy
  • Processor locks up the memory controller to stop running, to block any further damage on enclaves by the hardware attackers

  • The processor must be rebooted
SGX-BOMB

• A processor Denial-of-Service attack against Intel SGX

• Intentionally trigger drop-and-lock policy by inducing integrity violation using the Rowhammer attack

• Fast, hideous, and could lockdown the entire server in the cloud

• Hard to detect; software fix is hard
The Rowhammer Attack [ISCA 2014]

• A disturbance attack on the DRAM
  • A hardware vulnerability

• Accessing different rows in a bank could induce disturbance in adjacent row

• Triggered by purely in software
The Rowhammer Attack [ISCA 2014]

• Access Row i-1 and i+1 for multiple times

• This will induce disturbance in $i^{th}$ row

A DRAM BANK

(i-1)th row
(i)th row
(i+1)th row

Row Buffer
The Rowhammer Attack [ISCA 2014]

• Access Row i-1 and i+1 for multiple times

• This will induce disturbance in i\textsuperscript{th} row
The Rowhammer Attack [ISCA 2014]

• The attack can flip multiple bits in a block
  • DRAM with ECC could not completely block this

• The attack is triggered by software
  • Breaks Intel’s threat assumption

• No memory access is required
  • The data will be mismatched with Integrity Tree
Launching Rowhammer in SGX

• Should know virtual addresses that map to interleaved rows
  • Enclave does not know the physical address (Ring 3)
  • Can be resolved with a timing side-channel (DRAMA [SEC 2016])
    • Accessing to a different row in the same bank will take more time
    • E.g., 500 cycles for buffered read, 550 cycles for read from a different bank, and 650 cycles for reading conflicting rows

• SGX does not have a timer (rdtsc is prohibited)
  • Get helped by ocall to call rdtsc after 1,000 times of access
  • Or, we can spawn a thread to count integers (to get # of cycles elapsed)
Step 1: Finding Rows in the Same Bank

- Fix an address (p1)
- For the addresses in enclave (p2),
  - Place a timer
  - Access p1 and p2 multiple times
  - Get the timer value and check
- Access time > THRESHOLD will be rows in the same bank
  - 600,000 in our test with i7-6700K
  - For 1,000 times of row access

```c
#define N_THRESHOLD (600000)
#define N_TIMES (1000)

// Runs outside of an enclave

bool check_addr_in_the_same_bank(uint64_t *p1, uint64_t *p2) {
    // returns -500000 if p1 and p2 are in the same row
    // returns -550000 if p1 and p2 are in different banks
    // returns > 600000 if p1 and p2 are in different rows
    size_t start_time = rdtscp();
    enclave_access_row(p1, p2, N_TIMES);
    size_t end_time = rdtscp();
    return (end_time - start_time > N_THRESHOLD );
}
```
Step 2: Finding 1-interleaved Rows (i-1, i, i+1)

- Current SGX driver for Linux uses a naïve scheduler for allocating memory in EPC

- Virtually adjacent rows are highly likely to be adjacent in the physical space, too

- Just picking two virtually adjacent rows in the middle (over 32MB space) would be sufficient for the attack
Step 3: Hammering Rows

```c
void dbl_sided_rowhammer(uint64_t *p1, uint64_t *p2, uint64_t n_reads) {
    while(n_reads-- > 0) {
        // read memory p1 and p2
        asm volatile("mov (%0, %%r10); : r"(p1) : "memory");
        asm volatile("mov (%0, %%r11); : r"(p2) : "memory");
        // flush p1 and p2 from the cache
        asm volatile("clflushopt (%0); : r"(p1) : "memory");
        asm volatile("clflushopt (%0); : r"(p2) : "memory");
    }
    chk_flip();
}
```
DEMO

• https://www.youtube.com/watch?v=X3R6pq1gyo
Result

• We observed that SGX-BOMB can happen in normal settings
  • Core i7-6700K (Skylake), 8GB DDR4-2133Mhz DRAM
  • Took 283 seconds

• Much faster attack time in higher refresh rate

<table>
<thead>
<tr>
<th>Refresh time (ms)</th>
<th>64 (default)</th>
<th>128</th>
<th>256</th>
<th>503</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack time</td>
<td>283</td>
<td>30</td>
<td>4s</td>
<td>1s</td>
</tr>
</tbody>
</table>
Implications of the SGX-BOMB attack

• SGX-BOMB on a cloud provider (e.g., Amazon EC2) could lock a processor in the cloud server

• This will lock the entire server instance because QPIs and NUMA would fail
  • All tenants suffer reboot

• Rebooting the cloud machine would affect on the SLA a lot
  • Amazon guarantees 99.95% SLA
  • Reloading working memory set in redis and memcached requires long time...

• The attack can also lock an end-user’s machine
The Rowhammer Attack in Enclaves

• SGX-BOMB attack is easier to launch than other attacks
  • Only require one flip in any block in the EPC region (~128MB)
  • Do not require a specific bit to flip; unlike flipping bits in private key (FFS), etc.

• Detection of SGX-BOMB is harder
  • Cannot inspect application; an enclave can load executables dynamically
  • Cannot use PMU to monitor in-enclave operations (ANVIL & Linux)
  • Anti side-channel inference (ASCI) in effect
Root-cause is in DRAM

• Not a design flaw of SGX
• Target Row Refresh (TRR)
  • Standardized in LPDDR3, but not in both DDR3 and DDR4

• Intel’s Pseudo-TRR (pTRR) is in the processor, but still non-compliant
  vulnerable DRAMs are in the market

• ECC could mitigate SGX-BOMB, but cannot completely block it
  • Multiple bit flips (2 or more) in one block are possible
Potential Software Mitigations?

- CATT/GATT [SEC 2017] could be a solution
  - Block any access to the adjacent rows of the rows of the EPC region
- Changing memory allocation scheduling also helps
  - Make finding adjacent row harder

- Use Uncore PMU for detection
  - ASCl does not hide information for Uncore PMU
  - e.g., [L3 miss from Uncore PMU] – aggregated([L3 access from core PMU])
    = [L3 access from enclaves]
Better Defense than Drop-and-Lock?

• It is the sole problem of a malicious enclave, but drop-and-lock stops all executions of a processor

• Better options?
  • Let regular operations go on while disabling further SGX execution
  
  • Just kill the target enclave that owns the violated block in EPC
    • EPCM contains the information

• Both approaches require hardware modification
Conclusion

• Intel SGX locks the processor if any of integrity violation detected on accessing EPC memory

• It assumes the violation can only happen if there is a hardware attack

• SGX-BOMB can tamper the data in EPC memory via the Rowhammer attack, which is in software manner, to trigger processor lock

• SGX-BOMB can lockdown cloud servers equipped with SGX and is hard to be detected by existing Rowhammer defenses