**VoltPillager**: Hardware-based fault injection attacks against Intel SGX Enclaves using the SVID voltage scaling interface

*Zitai Chen, Georgios Vasilakis, Kit Murdock, Edward Dean, David Oswald, Flavio D. Garcia*

University of Birmingham, UK
Threat Model of SGX
What are some of the use cases for Intel® SGX?

Intel® SGX allows you to run applications on untrusted infrastructure (for example public cloud) without having to trust the infrastructure provider with access to your applications.

Source: Fortanix Intel SGX

Enarx threat model

Enarx is built with these principles in mind:

- Don’t trust the host
- Don’t trust the host owner
- Don’t trust the host operator
- All hardware cryptographically verified
- All software audited and cryptographically verified

Source: Enarx Threat Model
https://github.com/enarx/enarx/wiki/Threat-Model

- Untrusted OS
- Untrusted owner
- Untrusted Infrastructure

Source: Intel® SGX for Dummies (Intel® SGX Design Objectives)

8. Enable applications to define secure regions of code and data that maintain confidentiality even when an attacker has physical control of the platform and can conduct direct attacks on memory.
Previous Works

- Faulting Multiplication
- Faulting RSA in SGX
- Faulting AES-NI in SGX
- Memory Corruption

Kit Murdock et al. Plundervolt: Software-based Fault Injection Attacks against Intel SGX
In: 41st IEEE Symposium on Security and Privacy (S&P'20)

**VoltPillager:** Hardware-based fault injection attacks against Intel SGX Enclaves using the SVID voltage scaling interface
SVID Bus

- 3 Wire interface
  - CLK, DATA and ALERT (Not required)
- Clock @ 25MHz
- Logical High >0.64V, Low <0.45V

Ref:
1. L6751C Digitally controlled dual PWM for Intel VR12 and AMD SVI
2. 8th Generation Intel® Core™ Processor Families Datasheet, Volume 1 of 2
**SVID Bus – Which wire?**

**ISL95824 Data Short**

4+2 Multiphase PWM Regulator for Intel IMVP8 Desktop CPUs

- 1*A4 page long
- Does not show pin definition
- No information about the signal

**VoltPillager:** Hardware-based fault injection attacks against Intel SGX Enclaves using the SVID voltage scaling interface
SVID Bus – Which wire?

VoltPillager: Hardware-based fault injection attacks against Intel SGX Enclaves using the SVID voltage scaling interface
SVID Protocol

SVID signals and data frame

VID: 1 byte, computed as:

\[ \text{VID} = \left[ \frac{U - 0.245}{0.005} \right] \]

(voltage U in volt)

VID Commands: 5 bits

<table>
<thead>
<tr>
<th>Command name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended</td>
<td>0x00</td>
</tr>
<tr>
<td>SetVID-Fast</td>
<td>0x01</td>
</tr>
<tr>
<td>SetVID-Slow</td>
<td>0x02</td>
</tr>
<tr>
<td>SetVID-Decay</td>
<td>0x03</td>
</tr>
<tr>
<td>SetPS</td>
<td>0x04</td>
</tr>
<tr>
<td>SetRegADR</td>
<td>0x05</td>
</tr>
<tr>
<td>SetRegDAT</td>
<td>0x06</td>
</tr>
</tbody>
</table>
VoltPillager: Hardware undervolting

$30
Teensy 4.0 with modified SPI driver and VoltPillager firmware

Direct trigger input using GPIO pin
Trigger over USB (serial)

Voltage Control Disabled by INTEL-SA-00289

Untrusted Code
SGX
CPU

Voltage Regulator
SVID Bus

VoltPillager: Hardware-based fault injection attacks against Intel SGX Enclaves using the SVID voltage scaling interface
Let’s Inject Some Fault
// configure the glitch
// Z170 2GHz
configure_glitch_with_delay(1, 0.83, 35, 0.63, -30, 0.83, 100);

// Target ecall
flag1++;
asm volatile("": "memory");
// TRIGGER
TRIGGER_SET
sgx_ret = rsa_dec_ecall(eid, &res_var, buffer, iterations);
if (SGX_SUCCESS != sgx_ret) {
    printf("[ERROR]: sgx error 0x%llx\n", sgx_ret);
}
asm volatile("": "memory");
flag1++;

// RESET TRIGGER
TRIGGER_RST
Fault Injection with VoltPillager

- Multiplication Fault
- RSA Fault (in SGX)
- AES-NI Fault (in SGX)
  - mbedtls_aesni
  - Open Enclave file-encryptor
- Delayed-Write Fault

VoltPillager: Hardware-based fault injection attacks against Intel SGX Enclaves using the SVID voltage scaling interface
Fault OE-FILE-ENCRIPTOR
Voltpillager: Hardware-based fault injection attacks against Intel SGX Enclaves using the SVID voltage scaling interface
Delayed-write fault
Delayed-Write Fault – Initial PoC

VoltPillager: Hardware-based fault injection attacks against Intel SGX Enclaves using the SVID voltage scaling interface
Delayed-Write Fault – Practical Exploitation

```c
uint32_t array[8] = { 0 }; // Attacker-supplied out-of-bounds size
int copy_size = 7;

// Ensure we stay within bounds
if(copy_size >= 5) copy_size = 4;

// overwrite elements 4, 3, 2, 1
while(copy_size >= 1) {
    array[copy_size] = 0xabababab;
    copy_size--;
}
```

**Normal execution:**

```
00... AB... AB... AB... AB... 00... 00... 00...
```

**Fault 1 causing out-of-bounds underflow:**

```
AB... AB... AB... AB... AB... 00... 00... 00...
```

**Fault 2 causing out-of-bounds overflow:**

```
00... AB... AB... AB... AB... AB... AB... AB...
```
VoltPillager: Hardware-based fault injection attacks against Intel SGX Enclaves using the SVID voltage scaling interface.

Voltage 0.788574. Undervolting: 0mV mV.
Intel’s response
“... opening the case and tampering of internal hardware to compromise SGX is out of scope for SGX threat model. Patches for CVE-2019-11157 (Plundervolt) were not designed to protect against hardware-based attacks as per the threat model” - Intel

But……A lot of developers still think SGX can protect against hardware tampering.

**VoltPillager**: Hardware-based fault injection attacks against Intel SGX Enclaves using the SVID voltage scaling interface
Summary

- 1st hardware based undervolting against Intel CPUs
- Physical access -> CVE- 2019-11157(Plundervolt)
- Discovered a delayed memory write fault
- Build for $30
- Rethink of Intel SGX Threat Model
Thank you.

https://zt-chen.github.io/voltpillager/