A Formal Approach to Secure Computation

(Presented by Kevin Cheang)

ADEPT End-of-Project Party
Dec 9th 2021
Demand for Secure and Remote Computation

• Strong memory isolation through enclaves
  • An enclave is a program that provides memory isolation typically using hardware primitives and a software interface

• Why do we need them?
  • Remote computation on the cloud is very common (e.g. AWS)
  • Want our data to be kept secret
  • Recent attacks: Spectre, Spectre-NG, Meltdown, Foreshadow, Fallout, ...
Trusted Execution Environments

Formally verify enclave programs execute correctly and without revealing secrets!
A Formal Foundation for Secure Remote Execution of Enclaves

Pramod Subramanyan, Rohit Sinha, Illia Lebedev, Srinivas Devadas, Sanjit A. Seshia

CCS 2017
A Formal Foundation for the Secure Execution of Enclaves

• **Goal**: Introduce a novel formal verification methodology based on a trusted abstract platform to model enclaves and to prove secure remote execution on enclave platforms

• **Contributions**:
  • Formalization of enclave execution against a privileged adversary
  • Formalization of secure remote execution (SRE)
  • Decomposed SRE into secure measurement, integrity and confidentiality
  • Modeled the trusted abstract platform (TAP) and proved SRE on TAP
  • Refinement based methodology; showed SGX and Sanctum refines TAP
Secure Remote Execution

**SRE Definition:** A remote platform performs secure execution of enclaves if the execution follows the expected semantics and does not reveal anymore than what is allowed by the adversary’s observations.

**Theorem 3.2:** An enclave platform that satisfies secure measurement, integrity and confidentiality property for any enclave program also satisfies secure remote execution.
The Trusted Abstract Platform

platform
fetch | load | store | get_addr_map | set_addr_map | launch | destroy | enter | exit | pause | resume | measure

CPU state
pc
regs
mem

address translation state
addr_map
cache
os_metadata

enclave state
current_eid
owner
enc_metadata
A Formal Approach to Secure and Efficient Enclave Cloning

Dayeol Lee, Kevin Cheang, Alexander Thomas, Catherine Lu, Pranav Gaddamadugu, Anjo Vahldiek-Oberwagner, Mona Vij, Dawn Song, Krste Asanovic, Sanjit A. Seshia

[in submission at S&P 2022]
The Trusted Abstract Platform with Cloning

(recently submitted to S&P 2022!)
A Formal Approach to Secure Speculation

Kevin Cheang, Cameron Rasmussen, Pramod Subramanyan
Sanjit A. Seshia

CSF 2019
A Formal Approach to Secure Speculation

• **Goal:**
  • Verify that programs are not vulnerable to transient execution attacks running on a given microarchitectural design

• **Mitigations**
  • Software mitigations
    • compiler extensions (e.g. /QSpectre), retpolines, page table isolation
  • Hardware mitigations
    • constant time memory loads, cache partitioning, access time randomization, obfuscation of timers

• **Contributions**
  • Formulating a general property to capture transient execution attacks
  • Introduced an assembly intermediate representation for speculative platforms
  • Automated verification of absence of transient execution attacks
Problem Statement

• Given a platform model, an adversary model, and a program, determine if the program is vulnerable to a transient execution attack

```c
void victim_function_v01(size_t x) {
    if (x < array1_size) {
        temp &= array2[array1[x] * 512];
    }
}
```

```c
void victim_function_v01(size_t x) {
    if (x < array1_size) {
        std::atomic_thread_fence(std::memory_order_seq_cst);
        temp &= array2[array1[x] * 512];
    }
}
```
Secure Speculation Property

Public

$\pi_1$
Secret

$\pi_2$
Secret

$\pi_3$
Secret

$\pi_4$
Secret
Evaluation

- Paul Kocher’s list of 15 bounds check bypass examples
- Bounded model checking for exploit finding (5 steps)
- Inductive model checking for verification (1 step)

<table>
<thead>
<tr>
<th>Example</th>
<th>Ex1</th>
<th>Ex5</th>
<th>Ex7</th>
<th>Ex8</th>
<th>Ex10</th>
<th>Ex11</th>
<th>Ex15</th>
<th>Fig. 3c</th>
<th>NI</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC</td>
<td>6.6</td>
<td>9.0</td>
<td>10.2</td>
<td>5.7</td>
<td>9.6</td>
<td>6.4</td>
<td>5.8</td>
<td>6.6</td>
<td>12.9</td>
</tr>
<tr>
<td>Induction</td>
<td>5.0</td>
<td>5.0</td>
<td>5.7</td>
<td>4.6</td>
<td>5.8</td>
<td>5.9</td>
<td>4.8</td>
<td>4.8</td>
<td>5.4</td>
</tr>
</tbody>
</table>
Conclusion

• Recently advances in side-channel attacks warrants formal verification of microarchitectural models and trusted systems
• The Trusted Abstract Platform is an abstraction of enclave platforms and enables formal reasoning about secure remote execution
• A verification methodology was introduced to verify the absence of transient execution attacks on programs running on a microarchitecture
• Future work
  • Enabling automated and agile verification for enclave platform and microarchitectural processor designs using these existing methodologies
  • Showing refinement of TAP to the implementation of Keystone