SGXLock: Towards Efficiently Establishing Mutual Distrust Between Host Application and Enclave for SGX

Yuan Chen¹, Jiaqi Li¹, Guorui Xu¹, Yajin Zhou¹, Zhi Wang², Cong Wang³, Kui Ren¹

¹Zhejiang University, ²Florida State University, ³City University of Hong Kong

USENIX Security 2022
Motivation

➢ SGX: a popular TEE solution

➢ Application Components with SGX
  ➢ Enclave: Sensitive code/data
  ➢ Host application: Main application logic

➢ Problematic Assumption of SGX
  ➢ Enclave is considered as trusted, while the host app is untrusted
  ➢ In reality, enclave and host app are mutual untrusted
    ➢ e.g., third-party enclaves, enclaves with flaw
Enclave-Host Asymmetry

- Introduced by the problematic assumption of SGX
  - Blind trust of the host app to the enclave

- Control Flow Asymmetry
  - Enclave can jump to arbitrary locations of the host app
  - The host app enters enclave via pre-defined entry

- Data Access Asymmetry
  - Enclave can access host memory
  - Not vice verse
Our solution: SGXLock

➢ **Goal**: Eliminate enclave-host asymmetry and establish mutual distrust

➢ Control flow asymmetry elimination
  ➢ Leverage single-step mode

➢ Data access asymmetry elimination
  ➢ Leverage Intel MPK
Control Flow Asymmetry Elimination

- Based on single-step mode
  - The execution inside enclave is treated as a single instruction for single-step mode
- Workflow
  - Enter enclave with single-step mode enabled
  - Execute inside enclave
  - When EEXIT, single-step exception triggered
Data Access Asymmetry Elimination

- Based on Intel MPK
- Initialize
  - Allocate different MPK keys (i.e., E, H) for the enclave and the host app
  - Allocate the parameter buffer for data interaction, assigned with MPK key E
- Before entering enclave
  - Copy input into parameter buffer
  - CPU access permission: {E,H} \(\rightarrow\) {E}
- After exiting enclave
  - CPU access permission: {E} \(\rightarrow\) {E,H}
  - Copy output from parameter buffer
Challenges

C1. PKRU register update inside the enclave
   - PKRU represents the CPU’s access permission to MPK keys
   - Two ways to update PKRU inside the enclave
     - XSTORE instruction
     - WRPKRU instruction

C2. Host stack pointer manipulation
   - Enclave can manipulate host stack pointer
Solution of Challenges

➢ For C1.
  ➢ XRSTORE: PKRU is restored from mem as processor’s extended state
  ➢ Solution: disable the bit 9 of enclave’s XFRM field
  ➢ WRPKRU: Update PKRU directly
  ➢ Solution: binary inspection to avoid the occurrence of WRPKRU inside the enclave

<table>
<thead>
<tr>
<th>What to inspect</th>
<th>When to inspect</th>
<th>Who to inspect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static inspection</td>
<td>Plain enclave code</td>
<td>At enclave creation</td>
</tr>
<tr>
<td>Dynamic inspection</td>
<td>Dynamic enclave code</td>
<td>At enclave runtime, triggered by $W \oplus X$ violation</td>
</tr>
</tbody>
</table>

➢ For C2.
  ➢ Host stack integrity check based on a secret key
Experimental Setup

- Implementation based on Intel SGX SDK v2.9.1 for Linux
- Ubuntu 18.04.4 (Kernel v5.4.28) with SGX driver v2.6 installed
- Intel i7-10700F CPU (2.90GHz), which supports SGX and MPK
Micro-Benchmarks

- Raw ECALL/OCALL latency
  - Raw means no workload for ECALL/OCALL

<table>
<thead>
<tr>
<th></th>
<th>Original (cycle)</th>
<th>SGXLock (cycle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECALL</td>
<td>7636</td>
<td>11662 (52.7%)</td>
</tr>
<tr>
<td>OCALL</td>
<td>5908</td>
<td>9588 (62.3%)</td>
</tr>
</tbody>
</table>

SGXLock introduces relatively high latency overhead for host-enclave interaction.
Macro-Benchmarks

- Three representative scenarios
  - ML inference service, Database operation, HTTP web server

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Overhead</th>
<th>OCALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML inference service</td>
<td>0.84%</td>
<td></td>
</tr>
<tr>
<td>Database operation</td>
<td>1.26%</td>
<td>~ 13k OCALLs/s</td>
</tr>
<tr>
<td>HTTP web server</td>
<td>3.98%</td>
<td>~ 30k OCALLs/s</td>
</tr>
</tbody>
</table>

SGXLock is efficient in the above real-world scenarios, even with high-frequency OCALLs.
Conclusion & Takeaway

- Blind trust of the host app to the enclave introduces enclave-host asymmetry
- SGXLock: a defense solution to confine an untrusted enclave’s behavior
- Evaluation from real-world scenarios shows the efficiency of SGXLock
Thank you for listening!
Questions?

Yuan Chen
yuanchen96@zju.edu.cn