Autarky: Closing controlled channels with self-paging enclaves

Meni Orenbach, Technion
Andrew Baumann, Microsoft Research
Mark Silberstein, Technion
Public cloud computing

Sensitive data

Enclave

Enclave

Enclave

Microsoft Azure

Alibaba Cloud

IBM Cloud
Intel SGX

- Isolated user-mode environment
- Commodity CPUs
- Small trusted computing base
  - CPU
  - Enclave’s code and data
    - Confidentiality
    - Integrity
Page fault side-channel attack

• OS-level attacker
  • **Induces** page faults
  • **Tracks** faulted address
  • **Infer secrets content** that depends on page access patterns
    • Control-dependent accesses
    • Data-dependent accesses

---

**Xu, Y., Cui, W. and Peinado, M., 2015.**

**Controlled-Channel Attacks:**
Deterministic Side Channels for Untrusted Operating Systems.
Controlled-channel attack

- Precursor to other attacks
  - Foreshadow [Usenix Security’18]
  - Sgxspectre [arXiv’18]
  - LVI [IEEE S&P’20]
  - Microscope [ISCA’19]
  - Zombieload [CCS’19]

Why?

- Attacker controls the channel
- Precise
- No noise
Agenda

- Background
- Controlled-Channel Attack
- Self-Paging Enclaves
- Evaluation
SGX virtual memory protection

• **SGX validates** the OS does not insert *spurious mappings*

  ![Diagram of page table and reverse page table]

  **Page table (maintained by OS)**

<table>
<thead>
<tr>
<th>VA</th>
<th>PA</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>f0000</td>
</tr>
</tbody>
</table>

  **(Inaccessible by OS)**

<table>
<thead>
<tr>
<th>PA</th>
<th>VA</th>
</tr>
</thead>
<tbody>
<tr>
<td>f0000</td>
<td>10000</td>
</tr>
</tbody>
</table>

• **SGX does not** validate the *presence* of expected mappings
The missing component

Validate presence of expected mappings

Side-channel attacks defense

Active mapping attacks defense

SGX Reverse page table

Validate mapping
Implication: Controlled channel attack

Enclave

Application code

for (i=0;i<key_len;i++)
    if (key[i] == 1)
        mul(msg);

PF addr: 0x2000

Page fault on 0x2000

Resume

Operating System

I know that key[i]=1

resolve fault

SGX Reverse page table

Branch in page 0x5000

Function in page 0x2000

<table>
<thead>
<tr>
<th>VA</th>
<th>PA</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td>f0000</td>
<td>1</td>
</tr>
<tr>
<td>2000</td>
<td>e0000</td>
<td>0</td>
</tr>
</tbody>
</table>
Existing Software Mitigations

- Detect attack due to high frequency of exceptions
  - **Restrict** demand-paging
  - False positive occurrence
- Provably obfuscate all memory accesses
  - Orders of magnitude **performance impact**

---

Existing Hardware Mitigations

• Private enclave page tables

Requires major changes to SGX internals since SGX is entangled with the x86 architecture
Our solution: Autarky

- Minimal extension to SGX OS-hardware interface
  - **Backward-compatible** with SGX
  - **Validate presence** of expected mappings
Design principles

- Force the OS to call the enclave on every page fault
- Give enclave power to control all page faults
- Enclave-OS cooperative paging
- Hide fault information from the OS
- Enclave can enforce its own paging policy
Design overview

- Enclave
  - Legacy application
- Autarky runtime
  - Paging mechanism
  - Paging policy
  - Attack detection
- Operating System
  - Autarky paging module
  - Cooperative paging

Part of Library OS, SDK, etc.
Self-Paging Enclaves

Enclave
Application code
mov %rax, 0(10000)
PF addr: 0x10000

Page fault
0x0
Resume

Operating System

SGX Reverse page table

<table>
<thead>
<tr>
<th>VA</th>
<th>PA</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>f0000</td>
<td>0</td>
</tr>
</tbody>
</table>
Self-Paging Enclaves

Enclave
- Application code: `mov %rax, 0(10000)`
- PF addr: 0x10000

Self-paging fault handler
- Secure tracking:
  - VA 10000: Present: 1

Operating System
- Page fault 0x0
- Resume
- Enter page fault handler
- Exit
- VA 10000: PA f0000: P 0
- Attack detected!

SGX Reverse page table
Enclave can protect against spurious page faults.

Original attack required millions of page faults. Removing control is a huge improvement.
Support for legitimate page faults

Enclave

Application code

mov %rax, 0(10000)

PF addr: 0x10000

Self-paging fault handler

Secure tracking

<table>
<thead>
<tr>
<th>VA</th>
<th>Present</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>0</td>
</tr>
</tbody>
</table>

Operating System

Page fault 0x0

Fetch(10000)

Naïve paging policy leaks

SGX Reverse page table
Paging policy: part of the enclave’s runtime

Control the leakage
Rate-limiting policy

• Used by state-of-the-art software mitigations
  • Put a **limit** on the rate of exceptions
  • **Low** security guarantees

ORAM policy

• **Provably obfuscates** distribution of memory accesses
• Prior solutions show substantial **performance cost**
• Autarky is order-of-magnitude **faster** and makes it practical
  • Invoke ORAM only for paging

See paper for more details

Novel page clusters policy

Some applications do not need oblivious paging across all pages
Page clusters: cooperative paging for all pages in the cluster
Actual faulted address is hidden from the OS
Actual page access is not leaked

Upon page fault: Fetch all pages belonging to cluster C
Page clusters policy use cases

Similarly for libraries:
Attacker learns library access, not which function executed.

Attacker learns victim access to a dictionary. Not which word queried.
More details

- SGX1 and SGX2 cooperative paging mechanisms
- Eliminate accessed, dirty bit leakage
- Practical optimizations
  - Remove extra enclave crossing on page faults
  - Remove all enclave crossings on page faults
Agenda

- Background
- Controlled-Channel Attack
- Self-Paging Enclaves
- Evaluation
Memcached stores > 2x available memory
Issuing random 1KB GET requests

Throughput increases due to less paging
28% slower due to enclave crossing overhead
Just 7% slower
Throughput decreases as each page fault fetches 10 pages
ORAM only 60% slower compared to insecure baseline
ORAM has better cache utilization than page clusters
Conclusion

- Autarky **mitigates** the controlled-channel attack
  - **Practical modifications** to the architecture
  - Runtime with a secure paging policy
- Maintains **backward compatibility**
  - Operating system
  - Demand-paging
- Attack is not unique to SGX enclaves
  - Retrofit Autarky for other enclave environments!

Thank you!