Autarky: Closing controlled channels with self-paging enclaves

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

Sensitive data

Enclave

Meni Orenbach
Page fault side-channel attack

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

---

SGX virtual memory protection

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

  - Page table (maintained by OS)
  - Reverse page table (Inaccessible by OS)

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

- **SGX does not** validate the presence of **expected mappings**

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

Side-channel attacks defense

Active mapping attacks defense

Validate presence of expected mappings

Validate mapping

SGX Reverse page table
Implication: Controlled channel attack

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

Enclave

Application code

Page fault on 0x5000

Operating System

Branch in page 0x5000

Function in page 0x2000

SGX Reverse page table

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

Branch in page 0x5000

Function in page 0x2000

I know that key[i]=1

Resolve fault

SGX Reverse page table

<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 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

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 hardware/software co-design
  - **Backward-compatible** with SGX
  - **Validate presence** of expected mappings

---

**Diagram:**
- **Autarky**
  - Side-channel attacks defense
    - Active mapping attacks defense
  - SGX Reverse page table
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

Secure demand-paging
Design overview

Part of library OS/SDK. Current prototype based on Graphene

Enclave

Legacy application

Autarky runtime

Paging mechanism

Paging policy

Attack detection

Operating System

Autarky paging module

Cooperative paging

Part of library OS/SDK. Current prototype based on Graphene
Self-Paging Enclaves

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

Page fault 0x0
Resume

Operating System

Page fault addr: 0x10000

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
\texttt{mov} \%rax, 0(10000)
PF addr: 0x10000 pending exception
Self-page fault handler
Custom paging policy

Page fault 0x0
Resume

Operating System
Enter page fault handler
Exit

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)

Self-paging fault handler
Secure tracking

<table>
<thead>
<tr>
<th>VA</th>
<th>Present</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>1</td>
</tr>
</tbody>
</table>

Operating System

Fetch/Evict

SGX Reverse page table

VA | PA  | P  
---|-----|----
10000 | f0000 | 0

Attack detected!

Meni Orenbach
Enclave can protect against spurious page faults.

Original attack required millions of page faults. Removing control is a huge improvement.
Removing control raises the bar for other attacks

- Foreshadow [Usenix Security’18]
- SgxPectre [arXiv’18]
- LVI [IEEE S&P’20]
- Microscope [ISCA’19]
- Zombieload [CCS’19]

Inject page faults

- Speculative execution window is short
- Injecting faults offers a synchronization point with attacker’s code
- SGX page fetch instruction brings data to L1 cache
Optimizing page faults

Enclave transitions can impact performance

- Application code
  - `mov %rax, 0(10000)`

  - Page fault
    - Self-paging fault handler
      - Custom paging policy

  - Operating System

  Page fault
Demand-paging performance

![Chart showing demand-paging performance](chart.png)
Demand-paging performance

6% slowdown (2% with opt.)
No app changes

Meni Orenbach
Support for legitimate page faults

- **Application code**
  - `mov %rax, 0(10000)

- **Enclave**
  - PF addr: 0x10000

- **Self-paging fault handler**
  - Secure tracking
    - VA | Present
    - 10000 | 0

- **Operating System**
  - Page fault 0x0
  - Fetch(10000)

- **SGX Reverse page table**
  - Fetch(10000)

- Naïve paging policy leaks

Meni Orenbach
Paging policy: part of the enclave’s runtime

Control the leakage
Bounded-leakage 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
- Substantial **performance impact**
- Autarky achieves orders of magnitude **improvement**
  - Invoke ORAM only for paging

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

![Diagram showing a Spelling Server with options for English, Hebrew, and Greek languages, with an arrow from 'word, language' to the server.]
Page clusters policy use cases

Attacker learns victim access to a dictionary. Not which word queried.
Page clusters policy use cases

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

Similarly for libraries: Attacker learns library access, not which function executed.
Memcached stores > 2x available memory
Issuing random 1KB GET requests

Throughput (requests/s)
Memcached stores > 2x available memory
Issuing random 1KB GET requests

Throughput (requests/s)

Baseline | Rate Limit | 10-Page Cluster | ORAM

ORAM only 60% slower compared to insecure baseline
ORAM has better cache utilization than page clusters
Accessed, Dirty Bits Leakage

Accessed, Dirty bits changes should not be visible to the attacker

<table>
<thead>
<tr>
<th>VA</th>
<th>PA</th>
<th>P</th>
<th>A</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>f0000</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Accessed, Dirty Bits Leakage

Enclave Access?  
True

A/D bits are set?  
True

EPCM Validation

mov %rax, 0(10000)

TLB

Page table walk

Accessed, Dirty bits changes should not be visible to the attacker

<table>
<thead>
<tr>
<th>VA</th>
<th>PA</th>
<th>P</th>
<th>A</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>f0000</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Meni Orenbach
Summary

- Autarky mitigates the page fault side-channel attack
  - Practical hardware modifications
  - 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! Any questions?