Eleos: Exit-Less OS Services for SGX Enclaves

Meni Orenbach
Marina Minkin
Pavel Lifshits
Mark Silberstein

Accelerated Computing Systems Lab

Haifa, Israel
What do we do?
Improve performance: I/O intensive & memory demanding SGX enclaves

Why?
Cost of SGX execution for these applications is high

How?
In-enclave System Calls & User Managed Virtual Memory

Results
Eleos vs vanilla SGX
2x ↑ Throughput: memcached & face verification servers
Even for 5x ↑ available enclave memory

Available for Linux, Windows* 

(*) Without Eleos, these applications crash in Windows enclaves
• Background
• Motivation
• Overhead analysis
• Eleos design
• Evaluation
SGX enclaves are already here!

- Secured execution environment
- Reversed sandbox
- Small TCB
- Private code & data
  - Confidentiality
  - Integrity
  - Freshness
- Only CPU is trusted
SGX enclaves are already here!

- Secured execution environment
- Reversed sandbox
- Small TCB
- Private code & data
  - Confidentiality
  - Integrity
  - Freshness
- Only CPU is trusted
SGX enclaves are already here!

- Secured execution environment
- Reversed sandbox
- Small TCB
- Private code & data
  - Confidentiality
  - Integrity
  - Freshness
- Only CPU is trusted
SGX enclaves are already here!

- Secured execution environment
- Reversed sandbox
- Small TCB
- Private code & data
  - Confidentiality
  - Integrity
  - Freshness
- Only CPU is trusted
SGX enclaves are already here!

- Secured execution environment
- Reversed sandbox
- Small TCB
- Private code & data
  - Confidentiality
  - Integrity
  - Freshness
- Only CPU is trusted
SGX enclaves are already here!

- Secured execution environment
- Reversed sandbox
- Small TCB
- Private code & data
  - Confidentiality
  - Integrity
  - Freshness

Let's look at how to secure server applications with enclaves.
Background:
Lifetime of a secured server

Untrusted (Host & OS)  Trusted (Enclave)
Background:
Lifetime of a secured server

Untrusted (Host & OS)

Trusted (Enclave)

Untrusted memory
Unsecured access
Background: Lifetime of a secured server

Untrusted (Host & OS)  Trusted (Enclave)

- Untrusted memory
- Unsecured access

- Dedicated SGX mem
- Limited to: 128 MB
- Secured access
Background:
Lifetime of a secured server

Host app

Wait for network requests

Untrusted (Host & OS)

Trusted (Enclave)
Background:
Lifetime of a secured server

Untrusted (Host & OS)  Trusted (Enclave)

Host app

Wait for network requests
Background: Lifetime of a secured server

Untrusted (Host & OS):
- Host app
- Wait for network requests

Trusted (Enclave):
- Enter enclave
- Decrypt requests
Background:
Lifetime of a secured server

Untrusted (Host & OS)

Wait for network requests

Trusted (Enclave)

Enter enclave

Decrypt requests

Process requests

Host app

App
Background:
Lifetime of a secured server

- Untrusted (Host & OS)
  - Host app
  - Wait for network requests

- Trusted (Enclave)
  - Enter enclave
  - Decrypt requests
  - Process requests
  - Encrypt responses
Background: Lifetime of a secured server

Untrusted (Host & OS)

Host app

Wait for network requests

Send responses

Trusted (Enclave)

Enter enclave

Decrypt requests

Process requests

Encrypt responses
SGX enclaves should be fast

- ISA extensions
- Implemented in HW & Firmware
- Same CPU HW
- In-cache execution suffers no overheads
SGX enclaves should be fast

- ISA extensions
- Implemented in HW & Firmware
- Same CPU HW
- In-cache execution suffers no overheads

However...
Executing a Key-Value Store in enclave is slower
Executing a Key-Value Store in enclave is slower

Throughput: Slowdown factor

<table>
<thead>
<tr>
<th>Memory footprint</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 MB</td>
<td>11X</td>
</tr>
<tr>
<td>512 MB</td>
<td>34X</td>
</tr>
</tbody>
</table>
Executing a Key-Value Store in enclave is **slower**

Throughput: Slowdown factor

- **Crashes in Windows**
  - 11X for 64 MB
  - 34X for 512 MB
• Background
• Motivation
• Overhead analysis
• Eleos design
• Evaluation
Overhead analysis

Untrusted (Host & OS)

Host app

Wait for network requests

Send responses

Trusted (Enclave)

Enter enclave

Decrypt requests
150 cycles/32B

Process requests
*100 cycles/32B

Exit enclave

Encrypt responses
*150 cycles/32B
Overhead analysis

Untrusted (Host & OS)

Host app

Wait for network requests

Send responses

Trusted (Enclave)

Enter enclave

Decrypted requests

150 cycles/32B

Process requests

*100 cycles/32B

Encrypt responses

*150 cycles/32B

Exit enclave

Wait for network requests

Send responses

3,300 cycles
Overhead analysis

Untrusted (Host & OS) →

Host
app
→
Wait for network requests

Enter enclave

→
3,300 cycles

Decrypt requests
150 cycles/32B

Process requests
*100 cycles/32B

Encrypt responses
*150 cycles/32B

Send responses

Exit enclave

Trusted (Enclave) →
Overhead analysis

Host app

Wait for network requests

Enter enclave

Decrypt requests 150 cycles/32B

Process requests *100 cycles/32B

Encrypt responses *150 cycles/32B

Send responses

Exits causes indirect costs: 1.5X – 5X slower execution

FlexSC [OSDI’10] syscall analysis

Untrusted (Host & OS)

Trusted (Enclave)

~3,300 cycles

~3,800 cycles
Overhead analysis

Untrusted (Host & OS)

Wait for network requests

Host app

Enter enclave

Decrypt requests

150 cycles/32B

~3,300 cycles

Exits causes indirect costs:
1.5X – 5X slower execution

FlexSC [OSDI'10] syscall analysis

Encrypt responses

*150 cycles/32B

~3,800 cycles

Process requests

*100 cycles/32B

Send responses

Trusted (Enclave)
Eleos does better!

Throughput: Slowdown factor

- **SGX**
- **Eleos**

Memory footprint:
- 64 MB: 3.5x slowdown
- 512 MB: 5x slowdown
Eleos does better!

Throughput: Slowdown factor

- SGX
- Eleos

How does Eleos achieve this?
Eleos: Exit-less services

Exit-less system calls with RPC infrastructure
Exit-less SGX paging
Eleos: **Exit-less** services

Exit-less system calls with RPC infrastructure

Exit-less SGX paging
Background: SGX paging

- System mem
- SGX mem

Dedicated memory
Enclave code & data
Limited to 128 MB
Background: SGX paging

secret_foo():
...
*p = 1;
Background: SGX paging

secret_foo():
...
*p = 1;

Hardware Address translation

Enclave
Trusted

System mem

SGX mem

Untrusted

22 May@Systor' 2017
Meni Orenbach, Technion
Background: SGX paging

secret_foo():
...
*p = 1;

Hardware
Address translation

Page table

System mem
SGX mem
Encrypted

Enclave
Trusted

Untrusted
Background: SGX paging

secret_foo():
...
*p = 1;

Enclave
Trusted

Hardware
Address translation

Page table

System mem

SGX mem

Encrypted

Swapped-out

Untrusted

22 May@Systor' 2017 Meni Orenbach, Technion
Background: SGX paging

secret_foo():
...
*p = 1;

Hardware Address translation

Page table

Fault handler

System mem

SGX mem

Encrypted

Swapped-out

Enclave
Trusted

SGX-driver
Untrusted

Meni Orenbach, Technion
Background: SGX paging

secret(foo):
...
*p = 1;

Enclave
Trusted

Page table

Fault handler

 système mem

SGX mem

Decrypted

Encrypted

Integrity validation

Swapped-out

SGX-driver
Untrusted
Background: SGX paging

secret_foo():
...
*p = 1;
*(++p) = 2;

Hardware
Address translation

Page table

System mem

SGX mem
Decrypted

SGX driver
Untrusted

Enclave
Trusted

Fault handler

Encrypted
Background: SGX paging

secret_foo():
...
*p = 1;
*(++p) = 2;

Hardware
Address translation

Page table

Fault handler

Fast path

Enclave
Trusted

System mem

SGX mem
Decrypted

Encrypted

SGX driver
Untrusted
Since SGX memory is small paging is not as rare as in native applications
What are the overheads?
Background: SGX paging

secret_boo():
...
*p = 1;
*(++p) = 2;

Hardware Address translation

Page table

Fault handler

SGX driver

Untrusted

System mem

SGX mem

Decrypted

Encrypted
**SGX paging overheads**

```c
secret_foo():
...
*p = 1;
*(++p) = 2;
```

- Enclave Trusted
- Enclave exit
- SGX driver Untrusted
- System mem
  - SGX mem
    - Encrypted
    - Decrypted
- Hardware Address translation
- Page table
- Fault handler
- Enclave resume

---

22 May@Systor' 2017

Meni Orenbach, Technion
SGX paging overheads

```
secret_foo():
...
*p = 1;
*(++p) = 2;
```

- Enclave
  - Trusted
- System mem
  - SGX mem
    - Encrypted
    - Decrypted
- Page table
- Hardware
  - Address translation
- Fault handler
- Enclave
  - Exit
  - Resume
- Indirect costs
- SGX driver
  - Untrusted
SGX paging overheads

```c
secret_foo():
...
*p = 1;
*(++p) = 2;
```

Indirect costs

Enclave exit

SGX driver
Untrusted

Enclave
Trusted

System mem

SGX mem

Decrypted

Encrypted

Page table

Hardware
Address translation

Fault handler

Enclave resume

Overheads: Untrusted software manages enclave memory
SGX paging overheads

secret_foo():
...
*p = 1;
*(++p) = 2;

Indirect costs
Enclave exit
SGX driver Untrusted

Overheads: Untrusted software manages enclave memory

Enclave Trusted

System mem

SGX mem
Encrypted
Decrypted

Page table
Fault handler
Hardware Address translation

22 May@Systor' 2017 Meni Orenbach, Technion

22 May@Systor' 2017 Meni Orenbach, Technion
Wanted: In-enclave virtual memory management

No more exits!
Ideal in-enclave VM management

Enclave
Trusted

secret_foo():
...
*p = 1;
*(++p) = 2;

Hardware
Address translation

Page table

System mem

SGX mem

SGX driver
Untrusted

Fault handler

22 May@Systor’ 2017
Meni Orenbach, Technion
Ideal in-enclave VM management

secret_foo():
...
*p = 1;
*(++p) = 2;

Hardware Address translation

Page table

Fault handler

System mem

SGX mem

Enclave Trusted

22 May@Systor' 2017
Meni Orenbach, Technion
Ideal in-enclave VM management

Enclave Trusted

secret_foo():
...
*p = 1;
*(++p) = 2;

Hardware Address translation

Page table

Fault handler

No available hardware

System mem

SGX mem
Ideal in-enclave VM management

Enclave

Trusted

secret_foo():
...
*p = 1;
*(++p) = 2;

Software
Address translation

Page table

Fault handler

System mem

SGX mem
SUVM: Secured user-space VM

secret_foo():
  s_ptr<int> p = suvm_malloc(1024);
...
  *p = 1;

Software
Address translation

Page table

Fault handler
SUVM: Secured user-space VM

secret_foo():
s_ptr<int> p = suvm_malloc(1024);
...
*p = 1;

Template class: SecuredPointer.

Enclave Trusted

System mem

SGX mem

Software Address translation

Page table

Fault handler
SUVM: Secured user-space VM

Enclave Trusted

secret_foo():
  s_ptr<int> p = suvm_malloc(1024);

  ...

  *p = 1;

Software Address translation

Page table

Fault handler

Template class: SecuredPointer

System mem

SGX mem

Encrypted

22 May@Systor' 2017

Meni Orenbach, Technion
SUVM: Secured user-space VM

Template class: SecuredPointer.

secret_foo():
  s_ptr<int> p = suvm_malloc(1024);
  ...
  *p = 1;

System mem
SGX mem

Swapped-out

Encrypted

Software
Address translation

Page table

Fault handler

Enclave
Trusted
SUVM: Secured user-space VM

```
secret_foo():
    s_ptr<int> p = suvm_malloc(1024);
    ...
    *p = 1;
```

Template class: SecuredPointer.
SUVM: Secured user-space VM

secret_foo():
  s_ptr<int> p = suvm_malloc(1024);
  ...
  *p = 1;

Template class: SecuredPointer.

System mem

SGX mem

Decrypted

Encrypted

Integrity validation

Swapped-out

Page table

Fault handler

Software Address translation

Enclave Trusted

22 May@Systor' 2017 Meni Orenbach, Technion
SUVM: Secured user-space VM

secret_foo():

\[ s\_ptr<int> p = suvm\_malloc(1024); \]

\[ \ldots \]

\[ *p = 1; \]

Template class: SecuredPointer.

Control path in-enclave

System mem

SGX mem

Decrypted

Page table

Fault handler

Address translation

Swapped-out

Integrity validation

Enclave

Trusted

Software

Meni Orenbach, Technion

22 May@Systor'17
SUVM: Secured user-space VM

secretFoo():

\begin{verbatim}
  s_ptr<int> p = suvm_malloc(1024);
  ...
  *p = 1;
  *(++p) = 2;
\end{verbatim}
SUVM: Secured user-space VM

secret_foo():
s_ptr<int> p = suvm_malloc(1024);
...
*p = 1;
*(++p) = 2;

Software
Address translation

Page table

Fault
handler

System mem

SGX mem

Decrypted

Encrypted

Enclave
Trusted
SUVM: Secured user-space VM

secret_foo():
  s_ptr<int> p = suvm_malloc(1024);
  ...
  *p = 1;
  *(++p) = 2;

Software
Address translation

Page table

Fault handler

System mem

SGX mem

Decrypted

Encrypted

Enclave
Trusted

Fast path
No page table
Lookup!
Wait...Software based VM management?

Based on software address translation on GPUs, ActivePointers [ISCA'2016]
SUVM key contributions

• Multi-threaded

**Compared to SGX:**
- Fast path: up to 20% overheads
- Slow path: Eliminates costs of exits

<table>
<thead>
<tr>
<th></th>
<th>1 Thread</th>
<th>4 Threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ</td>
<td>5.5x</td>
<td>7x</td>
</tr>
<tr>
<td>WRITE</td>
<td>3.5x</td>
<td>5.9x</td>
</tr>
</tbody>
</table>

Throughput speedup
Software address translation offers new optimizations

- Customized page size
- Customized eviction policy
- Multi-enclave memory coordination
- Write-back only dirty pages
- Sub-page direct access to backing store
Software address translation offers new optimizations

- Customized page size
- Customized eviction policy
- Multi-enclave memory coordination
- Write-back only dirty pages
- Sub-page direct access to backing store
Software address translation offers new optimizations

- Customized page size
- Customized eviction policy
- Multi-enclave memory coordination
- Write-back only dirty pages
- Sub-page direct access to backing store
• Background
• Motivation
• Overhead analysis
• Eleos design
• Evaluation
Biometric Identity checking server

Workload generator + ID = 10Gb NIC

Face verification server

450MB DB (5X SGX mem)

ID
Biometric Identity validating server

Speedup compared to vanilla SGX

- Eleos
- Native

Server threads:
- 1
- 2
- 4
Biometric Identity validating server

Speedup compared to vanilla SGX

Eleos Native

Server threads

1 2 4
Biometric Identity validating server

Speedup compared to vanilla SGX

Eleos scales better than vanilla-SGX: Saves inter-processor-interrupts
Biometric Identity validating server

Speedup compared to vanilla SGX

Eleos scales better than vanilla-SGX:
- Saves inter-processor-interrupts
- Saturate 10Gb network

Eleos Native

Server threads

1 2 4

0 0.5 1 1.5 2 2.5 3 3.5
Memcached

Workload Generator (memaslap)

Memcached Graphene LibOS [Eurosys'2014]

10Gb NIC

GET()

~75 LOC modification for SUVM

500MB DB (5.5X SGX mem)
Memcached

Speedup compared to vanilla SGX (500 MB)

- Eleos (500MB DB)
- vanilla SGX (20MB DB)

Server threads

<table>
<thead>
<tr>
<th></th>
<th>1 Thread</th>
<th>4 Threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>No SGX</td>
<td>2.0</td>
<td>1.8</td>
</tr>
<tr>
<td>Faults</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Eleos (500MB DB) shows a higher speedup compared to vanilla SGX (20MB DB) in both 1 and 4 threads scenarios.
Memcached

Speedup compared to vanilla SGX (500 MB)

- Eleos (500MB DB)
- vanilla SGX (20MB DB)

Discovery: Eleos+Graphene is 3x slower than native
Take aways

- Eleos eliminates enclave exits costs
- Eleos available for Windows and Linux
  - Makes memory demanding applications available on Windows today
- Eleos takes a modularize approach
  - Memory demanding app? Link to SUVM
  - I/O intensive app? Link to RPC
  - Maintaining small TCB
Traditional SGX: Host-centric OS services
Traditional SGX: Host-centric OS services
Traditional SGX: Host-centric OS services

Enclave

Get data

Data Unavailable

Operating System
Traditional SGX: Host-centric OS services

[Diagram showing enclave, operating system, and data flow]
Traditional SGX: Host-centric OS services
Eleos Insight: Enclave-centric OS services

Enclave

Get data

Fetch data

In-enclave Services
Take aways (2)

• Eleos adapts 'accelerator-centric management'
  – System calls: GPUfs [ASPLOS'13], GPUnet [OSDI'14]
  – Virtual memory: ActivePointers [ISCA'16]

• We can do more!
  – Asynchronous DMA host copies
  – Non-blocking enclave launches

More information at:
  “SGX Enclaves as Accelerators" [Systex'16]
Thank you

Code is available at:
https://github.com/acsl-technion/eleos

shmeni@tx.technion.ac.il
Backup slides