Eleos: Exit-Less OS Services for SGX Enclaves

Meni Orenbach
Marina Minkin
Pavel Lifshits
Mark Silberstein

Accelerated Computing Systems Lab

Belgrade, Serbia
April 23-26, 2017
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

![Diagram of SGX enclaves and operating system](image-url)
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 memory
Unsecured access

Untrusted (Host & OS)

Trusted (Enclave)
Untrusted memory
Unsecured access

Dedicated SGX mem
Limited to: 128 MB
Secured access

Background:
Lifetime of a secured server

Untrusted (Host & OS)

Trusted (Enclave)
Background: Lifetime of a secured server

Untrusted (Host & OS) -> Wait for network requests -> Trusted (Enclave)
Background: Lifetime of a secured server

Untrusted (Host & OS) → Wait for network requests → Trusted (Enclave)

Host app

Lifetime of a secured server

Meni Orenbach, Technion
Wait for network requests

Enter enclave

Decrypt requests

Background: Lifetime of a secured server

Untrusted (Host & OS)

 Trusted (Enclave)

Host app

Host

dimensions: 794.0x595.0

[Image 346x156 to 520x348]

[Image -39x124 to 176x339]

[Image 682x451 to 754x509]
Background: Lifetime of a secured server

Untrusted (Host & OS)

Host app → Wait for network requests → Enter enclave → Decrypt requests → Process requests

Trusted (Enclave)
Background: Lifetime of a secured server

Untrusted (Host & OS)

Host app

Wait for network requests

Enter enclave

Trusted (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
  - Exit enclave
  - 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

Memory footprint

64 MB  11X

512 MB  34X
Executing a Key-Value Store in enclave is slower

Throughput: Slowdown factor

Memory footprint

- 64 MB: 11X
- 512 MB: 34X

Crashes in Windows
• 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

Enter enclave

~3,300 cycles

Decrypted requests

150 cycles/32B

Process requests

*100 cycles/32B

Encrypt responses

*150 cycles/32B

Trusted (Enclave)

Wait for network requests

Send responses

Enter enclave

~3,300 cycles

Decrypted requests

150 cycles/32B

Process requests

*100 cycles/32B

Encrypt responses

*150 cycles/32B
Overhead analysis

**Untrusted (Host & OS)**

- **Host app**
- Wait for network requests

**Trusted (Enclave)**

- Enter enclave
- Decrypt requests: 150 cycles/32B
- Process requests: *100 cycles/32B
- Encrypt responses: *150 cycles/32B
- Exit enclave
- Send responses

Approximate cycle counts:
- Decrypt requests: 3,300 cycles
- Process requests: ~3,800 cycles

Overhead analysis

Untrusted (Host & OS)

Wait for network requests

~3,300 cycles

Enter enclave

Decrypt requests
150 cycles/32B

Trusted (Enclave)

Process requests
*100 cycles/32B

Send responses

~3,800 cycles

Exit enclave

Encrypt responses
*150 cycles/32B

Exits causes indirect costs:
1.5X – 5X slower execution
FlexSC [OSDI'10] syscall analysis
Overhead analysis

Untrusted (Host & OS)

Enter enclave

Wait for network requests

Exit enclave

Host app

Trusted (Enclave)

Decrypt requests

150 cycles/32B

Process requests

*100 cycles/32B

Encrypt responses

*150 cycles/32B

~3,300 cycles

~3,800 cycles

Exits causes indirect costs:
1.5X – 5X slower execution
FlexSC [OSDI'10] syscall analysis

Send responses

FlexSC [OSDI'10] syscall analysis
Eleos does better!

Throughput: Slowdown factor

- Memory footprint
  - 64 MB
  - 512 MB

- SGX
- Eleos

Slowdown factor:
- 3.5x at 64 MB
- 5x at 512 MB
Eleos does better!

Throughput: Slowdown factor

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

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

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

Enclave
Trusted

System mem

Untrusted

SGX mem
Background: SGX paging

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

Enclave

Trusted

System mem

SGX mem

Hardware

Address translation

Untrusted
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
- System mem
- SGX mem
- Page table
- Hardware Address translation
- Swapped-out
- Untrusted
- Encrypted

24 April@Eurosys' 2017 Meni Orenbach, Technion
Background: SGX paging

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

- Enclave
  - Trusted
  - SGX-driver
    - Untrusted

- System mem
  - SGX mem
  - Encrypted
  - Swapped-out

- Hardware
  - Address translation

- Page table

- Fault handler
Background: SGX paging

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

Enclave
Trusted

System mem

SGX mem

Decrypted

Encrypted

Integrity validation

Swapped-out

SGX-driver
Untrusted

Hardware
Address translation

Page table

Fault handler

24 April@Eurosys' 2017
Meni Orenbach, Technion
Background: SGX paging

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

- Enclave
  - Trusted
  - SGX driver
    - Untrusted

- System mem
  - SGX mem
    - Encrypted
  - Decrypted
Background: SGX paging

secret.foo():
...
*p = 1;
*++p = 2;

Hardware
Address translation

Fast path

Page table

Fault handler

System mem

SGX mem

Decrypted

Encrypted

SGX driver
Untrusted

Enclave
Trusted

24 April@Eurosys' 2017
Meni Orenbach, Technion
Since SGX memory is small, paging is not as rare as in native applications. What are the overheads?
Background: SGX paging

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

Page table

Hardware
Address translation

Fault handler

SGX mem
Decrypted

System mem
Encrypted

Enclave
Trusted

SGX driver
Untrusted

24 April@Eurosys' 2017
Meni Orenbach, Technion
SGX paging overheads

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

Hardware
Address translation

System mem
SGX mem
Decrypted
Encrypted

Enclave Trusted

Enclave exit

SGX driver Untrusted

Page table
Fault handler

Enclave resume
SGX paging overheads

secret.foo():
...
*p = 1;
*(++p) = 2;

Enclave Trusted

Indirect costs

Enclave exit

SGX driver Untrusted

System mem

SGX mem

Decrypted

Encrypted

Hardware
Address translation

Page table

Fault handler

Enclave resume

Enclave
Trusted

24 April@Eurosys' 2017

Meni Orenbach, Technion
SGX paging overheads

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

Enclave exit
SGX driver Untrusted

Enclave Trusted
Indirect costs

Page table
Fault handler

Overheads: Untrusted software manages enclave memory

System mem
SGX mem
Decrypted
Encrypted
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

Fault handler

SGX driver
Untrusted

24 April@Eurosys' 2017

Meni Orenbach, Technion
Ideal in-enclave VM management

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

- Enclave
  - Trusted
  - Secret Foo
    - Hardware Address translation
    - Page table
    - Fault handler
  - System mem
    - SGX mem
  - System mem
Ideal in-enclave VM management

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

Hardware
Address translation
Page table
Fault handler

Enclave
Trusted

No available hardware

System mem
SGX mem
Ideal in-enclave VM management

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;
```

Enclave Trusted

System mem

- Software Address translation
- Page table
- Fault handler

SGX mem
SUVM: Secured user-space VM

secret_foo():

```cpp
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

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

System mem

SGX mem

Encrypted

Template class: SecuredPointer.

Enclave
Trusted

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.

System mem
  SGX mem
  Encrypted
  Swapped-out

Enclave
  Trusted

Software
  Address translation

Page table

Fault
  handler
SUVM: Secured user-space VM

secret_foo():

\[
\text{s_ptr<int> } p = \text{suvm_malloc(1024)};
\]

...  

\[
*p = 1;
\]

Template class: SecuredPointer.

Enclave Trusted

Software Address translation

Page table

Fault handler

System mem

SGX mem

Encrypted

Swapped-out
SUVM: Secured user-space VM

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

Template class: SecuredPointer.

System mem

Enclave
Trusted

SGX mem

Decrypted

Encrypted

Swapped-out

Integrity validation

Page table

Fault handler

Software Address translation
SUVM: Secured user-space VM

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

Template class: SecuredPointer.

Control path
in-enclave

SuVM: Secured user-space VM

Enclave
Trusted

System mem

SGX mem

Decrypted

Encrypted

Integrity validation

Swapped-out

Software
Address translation

Page table

Fault handler
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

Enclave
Trusted

System mem

SGX mem

Decrypted

Encrypted

24 April@Eurosys' 2017
Meni Orenbach, Technion
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
SUVM: Secured user-space VM

secret_foo():

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

Enclave
Trusted

System mem

Fast path
No page table
Lookup!

Page table

Fault handler

Software Address translation

SGX mem

Decrypted

Encrypted
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
- Face verification server

10Gb NIC

ID

? =

450MB DB (5X SGX mem)
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

Server threads

Eleos
Native
Memcached

Workload Generator (memaslap)

Memcached Graphene LibOS [Eurosys'2014]

~75 LOC modification for SUVM

10Gb NIC

GET(

500MB DB (5.5X SGX mem)
Memcached

Speedup compared to vanilla SGX (500 MB)

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

Server threads

1 Thread

4 Threads

No SGX Faults
Memcached

Speedup compared to vanilla SGX (500 MB)

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

No SGX Faults

Disclaimer: 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

Enclave

Get data

Operating System
Traditional SGX: Host-centric OS services

Enclave

Operating System

Get data

Data Unavailable
Traditional SGX: Host-centric OS services

- Enclave
  - Get data
- Operating System
  - Fetch data
- Data Unavailable
Traditional SGX: Host-centric OS services

Enclave

Operating System

Data Unavailable

Get data

Fetch data

Loading
Eleos Insight: Enclave-centric OS 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