OpenSGX: An Open Platform for SGX Research

Prerit Jain, Soham Desai, **Seongmin Kim***, Ming-Wei Shih, JaeHyuk Lee, Changho Choi, Youjung Shin, Taesoo Kim, Brent Byunghoon Kang, Dongsu Han
Trusted Execution Environment (TEE)

- Hardware technologies for trusted computing
  - Isolated execution: integrity of code, confidentiality
  - To protect application from untrusted platform
Trusted Execution Environment (TEE)

AMD, ARM Partner on Future TrustZone Security Platform

BY DAMON POETER  JUNE 13, 2012 05:15PM EST  1 COMMENT
Trusted Execution Environment (TEE)

AMD, ARM Partner on Future TrustZone Security

German Federal Government Certifies Infineon TPM
Trusted Execution Environment (TEE)

AMD, ARM Partner on Future TrustZone Security

German Federal Government Certifies Infineon TPM

Intel alters design of ‘Skylake’ processors to enhance security

October 3rd, 2015 at 12:04 pm - Author Anton Shilov
Trusted Execution Environment (TEE)

AMI, ARM Partner on Future TrustZone Security

Intel alters design of ‘Skylake’ processors to enhance security

October 3rd, 2015 at 12:04 pm - Author Anton Shilov

• Practical limitations of TEEs
  – Trusted Platform Module (TPM) : Poor performance
  – ARM TrustZone : Compatibility (only for embedded devices)
Intel SGX

• An extension of x86 Instruction Set Architecture (ISA)
  – Offers native performance, Compatibility with x86
  – Application keeps its data/code inside the “enclave”
Intel SGX

- An extension of x86 Instruction Set Architecture (ISA)
  - Offers native performance, Compatibility with x86
  - Application keeps its data/code inside the “enclave”
Intel SGX 101: Isolated Execution

- Smallest attack surface by reducing TCB (App + processor)
- Protect app’s secret from untrusted privilege software
Intel SGX 101: Isolated Execution

- Smallest attack surface by reducing TCB (App + processor)
- Protect app’s secret from untrusted privilege software
Intel SGX 101: Isolated Execution

- Smallest attack surface by reducing TCB (App + processor)
- Protect app’s secret from untrusted privilege software
Intel SGX 101: Isolated Execution

- Smallest attack surface by reducing TCB (App + processor)
- Protect app’s secret from untrusted privilege software
Intel SGX 101: Isolated Execution

- Smallest attack surface by reducing TCB (App + processor)
- Protect app’s secret from untrusted privilege software
Intel SGX 101: Isolated Execution

- Smallest attack surface by reducing TCB (App + processor)
- Protect app’s secret from untrusted privilege software
Intel SGX 101: Remote attestation

• Attest an application on remote platform
  – Check the integrity of enclave (hash of code/data pages)
  – Verify whether enclave is running on real SGX CPU
  – Can establish a “secure channel” between enclaves

1. Request
2. Create REPORT
3. Sign with EPID group key (Create QUOTE)
4. Send QUOTE
5. Verify
Intel SGX 101: Remote attestation

- Attest an application on remote platform
  - Check the **integrity of enclave** (hash of code/data pages)
  - Verify whether **enclave is running on real SGX CPU**
  - Can establish a “**secure channel**” between enclaves

**Intel SGX brings new opportunities for enhancing security of applications**

1. Request
2. Create REPORT
3. Sign with EPID group key (Create QUOTE)
4. Send QUOTE
5. Verify

**EPID key**

**Enclave Quoting**

**Remote platform**

**User platform**

16
SGX Research: Current Status

• Pioneering research: Adopting SGX on cloud computing (Haven [OSDI14], VC3 [S&P15])

• Confidentiality verification of SGX program (Moat [CCS15])

• Adopts SGX on networking [HotNets15]
SGX Research: Current Status

• However, software technologies for SGX lag behind their hardware counterpart

SGX CPU and SDK is now available! But..

• Specification for SGX [revision 1 & 2] is not fully available on the SGX hardware (only functionalities in revision 1)
• SGX technology has a complex license model
OpenSGX: Design Goal

• Offers a complete platform for SGX research
  – To explore software and hardware design space of SGX
  – To develop and evaluate SGX-enabled applications
OpenSGX: Design Goal

• Offers a complete platform for SGX research
  – To explore software and hardware design space of SGX
  – To develop and evaluate SGX-enabled applications

• Fills non-trivial issues on SGX software components
  – Support for system software and user-level APIs
  – Familiar programming model and interface
  – Secure design to defend against potential attack vectors (e.g., lago attacks)
OpenSGX: Design Goal

- Offers a complete platform for SGX research
  - To explore software and hardware design space of SGX
  - To develop and evaluate SGX-enabled applications

- Fills non-trivial issues on SGX software components
  - Support for system software and user-level APIs
  - Familiar programming model and interface
  - Secure design to defend against potential attack vectors (e.g., Iago attacks)

- Non goal: security guarantee
OpenSGX: Approach

- Using userspace emulation of QEMU
  - Binary translation to support SGX instructions
  - QEMU helper routine to implement complex instructions

Host (single address space)

QEMU

```
entry()
    // Binary Translation
    if(opcode == 0x0f01d7) {
        helper_enclu();
    }
    ...
```

Helper routine
- Set registers
- Operates SGX instructions
OpenSGX: Approach

- Using userspace emulation of QEMU
  - Binary translation to support SGX instructions
  - QEMU helper routine to implement complex instructions
OpenSGX: Component Overview

• Emulated SGX hardware
OpenSGX: Component Overview

- Emulated SGX hardware
- OS emulation layer
OpenSGX: Component Overview

- Emulated SGX hardware
- OS emulation layer
- OpenSGX user library
OpenSGX: Component Overview

- Emulated SGX hardware
- OS emulation layer
- OpenSGX user library
- OpenSGX toolchain
OpenSGX: Component Overview

- Emulated SGX hardware
- OS emulation layer
- OpenSGX user library
- OpenSGX toolchain

- Enclave loader

OpenSGX toolchain

SGX OS Emulation

SGX QEMU (HW emulation)
OpenSGX: Component Overview

- Emulated SGX hardware
- OS emulation layer
- OpenSGX user library
- OpenSGX toolchain

- Enclave loader
- Performance monitor
- Enclave debugger
OpenSGX: Component Overview

- Emulated SGX hardware
- OS emulation layer
- OpenSGX user library
- OpenSGX toolchain

- Enclave loader
- Performance monitor
- Enclave debugger

Enclave Program

Enclave loader
Runtime library
SGX Libraries
- Trampoline
- Stub

OpenSGX toolchain

SGX OS Emulation

SGX QEMU (HW emulation)
OpenSGX: Component Overview

- Emulated SGX hardware
- OS emulation layer
- OpenSGX user library
- OpenSGX toolchain
- Enclave loader
- Performance monitor
- Enclave debugger

### Enclave Program

```c
void enclave_main()
{
    char *hello = "hello sgx!\n";
    sgx_enclave_wriate(hello, strlen(hello));
    sgx_exit(NULL);
}
```

$ opensgx hello.sgx hello.conf
hello sgx!
OpenSGX: Component Overview

- Emulated SGX hardware
- OS emulation layer
- OpenSGX user library
- OpenSGX toolchain
- Enclave loader
- Performance monitor
- Enclave debugger

Enclave Program

```c
void enclave_main(){
    char *hello = "hello sgx!\n";
    sgx_enclave_write(hello, strlen(hello));
    sgx_exit(NULL);
}
```

$ opensgx hello.sgx hello.conf
hello sgx!
Hardware Emulation

- Emulates all data structures (e.g., EPCM) and processor key
- EPC Memory management
  - Direct mapping on virtual memory
  - Access protection: Instrument memory access
Hardware Emulation

• Emulates all data structures (e.g., EPCM) and processor key
• EPC Memory management
  – Direct mapping on virtual memory
  – Access protection: Instrument memory access

Virtual address space
Hardware Emulation

- Emulates all data structures (e.g., EPCM) and processor key
- EPC Memory management
  - Direct mapping on virtual memory
  - Access protection: Instrument memory access

![Diagram of Virtual Address Space]

EPC_begin

EPC_end

Virtual address space
Hardware Emulation

- Emulates all data structures (e.g., EPCM) and processor key
- EPC Memory management
  - Direct mapping on virtual memory
  - Access protection: Instrument memory access

Virtual address space

1. Prohibit access from host to EPC
2. Prohibit others enclaves’ EPC to current enclave’s EPC
Hardware Emulation

- Emulates all data structures (e.g., EPCM) and processor key
- EPC Memory management
  - Direct mapping on virtual memory
  - Access protection: Instrument memory access

```
QEMU's translation routine

...  
Case (Load | Store) { 
  1. Prohibit access from host to EPC
  2. Prohibit others enclaves’ EPC to current enclave’s EPC
}
...
```
Instruction Support

• OpenSGX supports most instructions specified
  – 21 out of 24 instructions
  – Except for debugging related instructions (e.g., EDBGRD)
  – Instead, it offers rich environment for debugging since it is a “software emulator” (e.g., GDB stub)
Instruction Support

• OpenSGX supports most instructions specified
  – 21 out of 24 instructions
  – Except for debugging related instructions (e.g., EDBGRD)
  – Instead, it offers rich environment for debugging since it is a “software emulator” (e.g., GDB stub)

• Provides simple C APIs which wraps assembly code
  – User-level instructions (ENCLU) : accessible to user-level APIs
  – Super-level instructions (ENCLS) : Requires system support
OS Emulation Layer

• Emulate OS to execute the privileged SGX instructions
  – Bootstrapping (EPC allocation)
  – Enclave initialization & page translation
  – Dynamic EPC page allocation

<table>
<thead>
<tr>
<th>System call</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_sgx_init()</td>
<td>Allocate EPC memory region</td>
</tr>
<tr>
<td>sys_init_enclave()</td>
<td>Create an enclave, Add and measure EPC pages</td>
</tr>
<tr>
<td>sys_add_epc()</td>
<td>Allocates a new EPC page to the running enclave</td>
</tr>
<tr>
<td>sys_stat_enclave()</td>
<td>Obtains the enclave statistics</td>
</tr>
</tbody>
</table>
OS Emulation Layer

• Emulate OS to execute the privileged SGX instructions
  – Bootstrapping (EPC allocation)
  – Enclave initialization & page translation
  – Dynamic EPC page allocation

Planning to extend the emulated OS for the system-level layer

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_init_enclave()</td>
<td>Create an enclave, Add and measure EPC pages</td>
</tr>
<tr>
<td>sys_add_epc()</td>
<td>Allocates a new EPC page to the running enclave</td>
</tr>
<tr>
<td>sys_stat_enclave()</td>
<td>Obtains the enclave statistics</td>
</tr>
</tbody>
</table>
Stub and Trampoline Interface

“A strict and narrow interface to handle enclave-host communication using shared data/code”
Stub and Trampoline Interface

“A strict and narrow interface to handle enclave-host communication using shared data/code”

- **Lib**
- **Code**
- **Heap**

Enclave

**Stub**: Shared data to specify the function code and arguments

**Trampoline**: Shared code to call user-level APIs in the wrapper

(Shared)

**Wrapper**

**Emulated OS**
“A strict and narrow interface to handle enclave-host communication using shared data/code”

Lib
malloc(){
  ...
  sgx_exit(tram);
  ...
}

Code
...
malloc(100);
...

Heap

Enclave

Stub : Shared data to specify the function code and arguments

Trampoline : Shared code to call user-level APIs in the wrapper

Wrapper

Emulated OS

(Shared)
Stub and Trampoline Interface

“A strict and narrow interface to handle enclave-host communication using shared data/code”

Lib
malloc()
{
    ...
    sgx_exit(tram);
    ...
}

Code
...
malloc(100);
...

Heap
...(Shared)

Enclave

Stub
heap_end
fcode
mcode
argument1
...

Trampoline
...
if (fcode == FUNC_MALLOC)
    alloc_tramp();
...

Trampoline : Shared code to call user-level APIs in the wrapper

Stub : Shared data to specify the function code and arguments

Wrapper

Emulated OS

FULL!
Stub and Trampoline Interface

“A strict and narrow interface to handle enclave-host communication using shared data/code”

Stub : Shared data to specify the function code and arguments

Trampoline : Shared code to call user-level APIs in the wrapper

Lib

<Specification>
  fcode : FUNC_MALLOC
  mcode : EAUG
  size: 100

Code
  ... malloc(100);
  ...

Heap

Enclave

(Shared)

Wrapper

Emulated OS

...
Trampoline and Stub Interface

“A strict and narrow interface to handle enclave-host communication using shared data/code”

Enclave

Lib
malloc(){
... sgx_exit(tram);
... }

Code
... malloc(100);
...

Heap

Trampoline

if (fcode == FUNC_MALLOC)
alloc_tramp();
...

(Shared)

Stub
heap_end
FUNC_MALLOC
EÅUG
100
...

User-level APIs to request system calls

alloc_tramp() {
... sys_add_epc();
... }

Wrapper

Emulated OS
Trampoline and Stub Interface

“A strict and narrow interface to handle enclave-host communication using shared data/code”

Lib
malloc()
... sgx_exit(tram);
...

Code
... malloc(100);
...

Heap

Enclave
(Shared)

Trampoline
... if (fcode == FUNC_MALLOC)
alloc_tramp();
...

User-level APIs to request system calls
alloc_tramp() {
    ...
    sys_add_epc();
    ...
}

Wrapper

Emulated OS
int sys_add_epc() {
    encls(EAUG, ...);
    ...
}

Stub
heap_end
FUNC_MALLOC EAUG 100 ...

Call EAUG
System Call
Trampoline and Stub Interface

“A strict and narrow interface to handle enclave-host communication using shared data/code”

Lib
malloc()
... 
sgx_exit(tram);
...

Enclave

Code
... 
malloc(100);
...

Heap

Lib
malloc()
... 
sgx_exit(tram);
...

Trampoline

if (fcode == FUNC_MALLOC)
alloc_tramp();

(Shared)

Stub
heap_end+4K
FUNC_MALLOC
EÅUG
100
...

User-level APIs to request system calls
alloc_tramp() {
... 
sys_add_epc();
... 
}

Wrapper

Emulated OS

Call EAUG
int sys_add_epc() {
encls(EÅUG, ...);
... 
System Call

System Call

Enclave

stub
... 
malloc(100);
...

Lib
malloc()
... 
sgx_exit(tram);
...

Trampoline

if (fcode == FUNC_MALLOC)
alloc_tramp();

(Shared)

Stub
heap_end+4K
FUNC_MALLOC
EÅUG
100
...

User-level APIs to request system calls
alloc_tramp() {
... 
sys_add_epc();
... 
}

Wrapper

Emulated OS

Call EAUG
int sys_add_epc() {
encls(EÅUG, ...);
... 
System Call

System Call
Evaluation: Tor Network

- Redesigns non-trivial application to use OpenSGX
- Tor: volunteer-based anonymity network
Evaluation: Tor Network

- Redesigns non-trivial application to use OpenSGX
- Tor: volunteer-based anonymity network

“Defend possible attacks on Tor components when they are compromised by adversaries”
Evaluation: Tor Network

- Redesigns non-trivial application to use OpenSGX
- Tor: volunteer-based anonymity network

“Defend possible attacks on Tor components when they are compromised by adversaries”

- Here, defense against network-level attacks on Tor is out of scope
SGX-enabled Tor Design

• Design goal
  – Protect data/code from adversary
  – Reducing Trusted Computing Base

Exit node (or directory server)  Separation  Enclave
SGX-enabled Tor Design

- Design goal
  - Protect data/code from adversary
  - Reducing Trusted Computing Base

Exit node (or directory server)

Enclave

Core operations (e.g., key creation, encryption, decryption, ...)

Key Relay table

Gen_key()
{
  ...
}

Encrypt()
{
  ...
}
SGX-enabled Tor Design

- Design goal
  - Protect data/code from adversary
  - Reducing Trusted Computing Base

Exit node (or directory server)
Rest of Tor operations
1. Send/receive packets
2. Initialize data structures

Enclave
Core operations (e.g., key creation, encryption, decryption, ...)

Gen_key()

Encrypt()

Key Relay table
SGX-enabled Tor Design

• Design goal
  – Protect data/code from adversary
  – Reducing Trusted Computing Base

Exit node (or directory server)
Rest of Tor operations
  1. Send/receive packets
  2. Initialize data structures
...

Other Tor nodes

Remote Attestation
Interaction

Enclave
Core operations (e.g., key creation, encryption, decryption, ...)

Key
Relay table
Interaction

SGX-enabled Tor Design

Design goal

- Protect data/code from adversary
- Reducing Trusted Computing Base

Exit node (or directory server)
Rest of Tor operations
  1. Send/receive packets
  2. Initialize data structures
...

Other Tor nodes

Remote Attestation
Interaction

Enclave
Core operations (e.g., key creation, encryption, decryption, ...)

Key
Relay table
Interaction
Performance Profiling

- Performance profiling of Tor exit node
  - Using OpenSGX performance monitor

<table>
<thead>
<tr>
<th></th>
<th>Code</th>
<th>Data</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenSSL</td>
<td>271</td>
<td>89</td>
<td>360</td>
</tr>
<tr>
<td>SgxLib</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Tor</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>278</strong></td>
<td><strong>91</strong></td>
<td><strong>369</strong></td>
</tr>
</tbody>
</table>

(Unit: Number of pages)

Required EPC: Less than 2MB
OpenSGX: Current Status

• Available at github, released in May 2015
  – Available in https://github.com/sslab-gatech/opensgx
  – 7 Contributors (Gatech, KAIST, Two sigma, MITRC, ...)
  – 31 unique cloners, 1,645 Views (Until January, 2016)

• What’s next?
  – Binary compatibility with Intel SGX hardware
  – Implement unsupported functionalities (e.g., multi-threading)

• Our current community
  🔵 TWO SIGMA
  🔴 Lua
Our Early Lessons on SGX

• Misconceptions on SGX
  – SGX for desktop-like environment: Needs secure I/O channel (integration with hardware technology such as Intel IPT)
  – Need EPID support for the remote attestation
Our Early Lessons on SGX

• **Misconceptions on SGX**
  – SGX for desktop-like environment: Needs secure I/O channel (integration with hardware technology such as Intel IPT)
  – Need EPID support for the remote attestation

• **Malicious use of Intel SGX**
  – Malware might be possible by abusing the isolation property
  – Fails on traditional signature-based AV programs
Conclusion

• We design and implement OpenSGX, fully functional and instruction-compatible SGX emulator
Conclusion

• We design and implement OpenSGX, fully functional and instruction-compatible SGX emulator

• As a showcasing application, we develop SGX-enabled Tor to enhance the security and privacy
Conclusion

• We design and implement OpenSGX, fully functional and instruction-compatible SGX emulator

• As a showcasing application, we develop SGX-enabled Tor to enhance the security and privacy

• OpenSGX offers opportunities to explore all components of SGX research
  – Hardware semantics (e.g., encryption scheme of MEE)
  – System software, enclave loader and user-level APIs
  – Redesigning unforeseen security applications (e.g., Tor)
Thanks!
Any Questions?
An adversary has control over all software components (including OS and hypervisor) and hardware except the CPU package

- Protection against denial-of-service is out of scope
## Comparison: Intel SGX vs OpenSGX

<table>
<thead>
<tr>
<th></th>
<th>Intel SGX</th>
<th>OpenSGX</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Hardware</td>
<td>Software Emulator</td>
</tr>
<tr>
<td><strong>Instructions</strong></td>
<td>16 ENCLS, 8 ENCLU</td>
<td>13 ENCLS, 8 ENCLU (Except debugging)</td>
</tr>
<tr>
<td><strong>Data structures</strong></td>
<td>Specified</td>
<td>○</td>
</tr>
<tr>
<td><strong>Paging</strong></td>
<td>Page table</td>
<td>Direct mapping</td>
</tr>
<tr>
<td><strong>System software</strong></td>
<td>Not specified</td>
<td>User level emulation</td>
</tr>
<tr>
<td><strong>User level APIs</strong></td>
<td>SDK is available (Only for Windows)</td>
<td>○</td>
</tr>
</tbody>
</table>
OpenSGX User Library

• Challenge 1: Facilitate the enclave programming
  – Custom in-enclave library: APIs for user-level SGX instructions
  – Porting standard C library (glibc)

• Challenge 2: Minimize attack surface between enclave and the potentially malicious host process
  – Function call relies on OS features will break an execution of enclave programs
  – Such functions open up new attack surfaces (e.g., lago attacks)
Defense against Iago attacks

- Iago attacks [ASPLOS’13]: Malicious OS tries to subvert trusted application by incorrect behavior
  - ex) adds incorrect EPC page for heap

```c
void *malloc(int size){
  if(cur_heap_ptr == heap_end) {
    stub->mcode = EAUG;
    exit(trampoline);
  }
  enclu(EACCEPT, ...);
}
```
Memory State of OpenSGX Program

User process (single address space)

Package Info
- Entry point
- Measurement
- Key

SGX Lib
- Trampoline
- Stub
- Wrapper

Code
- Lib
- EPC

Lib
- Data
- EPC

…

Data
- Stack
- EPC

…

Stack
- Heap
- EPC

Enclave Program

System call (e.g., sys_sgxinit())

SGX OS Emulation

QEMU SGX

ENCLU
(e.g., EENTER)

ENCLS
(e.g., EINIT)

System calls boundary

Privilege boundary
Attacks on Tor Components

- Tor network: uses 3-hop onion routing
  - Directory servers: Advertise available onion routers (ORs), vote for bad

When exit node is compromised, (unless end-to-end encryption is used)
1. Snooping or tampering of the plain-text
2. Break of anonymity: Bad apple attack

- Tor client
- Directory servers
- Tor network
- Entry
- Relay
- Exit
- Destination
Attacks on Tor Components

- Tor network: uses 3-hop onion routing
  - Directory servers: Advertise available onion routers (ORs), vote for bad exit nodes

When directory servers are compromised,
1. Tie-breaking attacks while voting
2. Admission of malicious ORs
Performance Profiling: CPU cycles

**<Directory Server>**

- Key generation
- Consensus creation
- Circuit establishment & Service
- Total

**<Tor Exit Node>**

- Key generation
- Consensus creation
- Circuit establishment & Service
- Total

- **2.8x**
- **2.7x**

- ENCLU(EEXIT, ERESUME) calls
- In-enclave library code to handle stub & trampoline interface
## Performance Profiling: TCB

<table>
<thead>
<tr>
<th></th>
<th>Code</th>
<th>Data</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenSSL</td>
<td>270</td>
<td>88</td>
<td>358</td>
</tr>
<tr>
<td>SgxLib</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Tor</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>276</td>
<td>90</td>
<td>366</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Code</th>
<th>Data</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenSSL</td>
<td>271</td>
<td>89</td>
<td>360</td>
</tr>
<tr>
<td>SgxLib</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Tor</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>278</td>
<td>91</td>
<td>369</td>
</tr>
</tbody>
</table>

(Unit: Number of pages)

- Required EPC size: **Less than 2MB** for each process
- TCB size : **54% smaller than** compared to Tor code base
OpenSGX implementation

- OpenSGX is an open source project!
  - Modified lines of code: 19K
  - First released in May, 2015
  - 7 Contributors (Gatech, KAIST)
  - 31 unique cloners, 1,645 Views (Until January, 2016)
  - Available at https://github.com/sslab-gatech/opensgx.git