Binary Code Retrofitting and Hardening Using SGX

Shuai Wang, Wenhao Wang, Qinkun Bao, Pei Wang, XiaoFeng Wang, and Dinghao Wu

The Pennsylvania State University, Indiana University Bloomington, Institute of Information Engineering
Motivation

- Available in Intel Commercial CPUs
- Hardware isolated memory regions
- Protection under a strong adversary model
- A bit performance penalty (~10%)
Motivation

- Available in Intel Commercial CPUs
- Hardware isolated memory regions
- Protection under a strong adversary model
- A bit performance penalty

Can binary code hardening benefit from SGX?
Motivation

- Graphene-SGX, Haven
  - Large TCB (53 kloc for Graphene-SGX)
Motivation

- Graphene-SGX, Haven
  - Large TCB (53 kloc for Graphene-SGX)
- Our solution
  - Techniques to dissect binary code into multiple components
  - Put into separated enclaves
Background on SGX

- Two capabilities
  - change in enclave memory access semantics
  - protection of the address mappings of the application
Background on SGX

- Life cycle

Enclave Initialization (ECREATE/EINIT)

- EENTER
- ERESUME
- AEX
- EEXIT

Enclave Destroy (EREMOVE)
Background on SGX

- Life cycle

Enclave Initialization (ECREATE/EINIT)

- Enclave Destroy (EREMOVE)

Life cycle:
- Enclave Initialization (ECREATE/EINIT)
  - EENTER
  - ERESUME
- Enclave mode
- non-enclave mode
- AEX
- EEXIT

Enclave Destroy (EREMOVE)
Background on SGX

- Controlled enclave entry
- Separated stack
- CPU state and registers are cleared if exceptions occur inside the enclaves.

```c
/*
 * Function: enclave_entry
 * The entry point of the enclave.
 */

#define DECLARE_GLOBAL_FUNC enclave_entry

/*
 * Dispatch code according to CSSA and the reason of EENTER
 * eax > 0 - exception handler
 * edi >= 0 - ecall
 * edi == -1 - do_init_enclave
 * edi == -2 - oret
 */

/* Registers
 * No need to use any register during the dispatch
 */
```
Methodology
Methodology

Interface library: maintain routine code for ecall and ocall
Methodology

In-place binary editing: Trampoline code
Challenges

- Binary code reassembly disassembling
  - Uroboros

- How to generate enclave libraries
  - Intel SGX SDK

- Binary instrumentation to jump to the enclave entry
  - Trampoline code

- Exceptions
  - Customized exception handling inside the enclaves
Challenges

- Binary code reassembly disassembling
  - Uroboros
- How to generate enclave libraries
  - Intel SGX SDK
- Binary instrumentation to jump to the enclave entry
  - Trampoline code
- Exceptions
  - Customized exception handling inside the enclaves
Some technique details

- In-place binary editing
  - Trampoline code

```assembly
1  trampoline_foo:
2    push  %rbp
3    mov  %rsp,%rbp
4    push  $return_addr
5    push  %rax
6    mov  $sgx_interface_foo,%rax
7    xchg  %rax,(%rsp)
8    ret
9    pop  %rbp
10   ret
```
Some technique details

- Exceptions
  - Customized exception handling inside the enclaves

```c
1  exception_exit:
2    mov %gs:0x0,%rax
3    mov %rax,%rbx
4    call update_ocall_lastsp
5    mov 0x20(%rbx),%rdx
6    mov 0x98(%rdx),%rbp
7    mov 0x90(%rdx),%rsp
8    mov $target_addr,%rbx
9    mov $EXIT, %rax
10   enclu
```
Proof-of-concept implementation

- Extend Uroboros with SGX instrumentation functionalities.
  - Employ the core functionality of Uroboros to identify program relocation symbols (e.g., code pointers).
  - Use industrial standard reverse engineering tool (IDA-Pro) to recover the function type information.

- Implement the instrumentation functionality in Scala, with over 1,700 LOC.

- The proof-of-concept implementation of the exception handling mechanism adds 56 lines of C code.
Evaluation

- Evaluations mainly focus on understanding the **feasibility** and **cost** of the instrumentation products.
- Two major factors would contribute to the **performance penalty** of the SGX protected code:
  - Execution slowdown of **code components inside enclaves**.
  - **Cross-enclave control flow transfers**, e.g., enclave ECALL.
Evaluation Setup

- Our preliminary evaluation instruments sensitive procedures provided by cryptographic libraries.

- AES implementation in OpenSSL (version 0.9.7)
  - Write sample code to trigger the encryption and decryption functions in the library.
  - key length is set as 256.
  - AES electronic codebook (ECB) mode.
Evaluation Setup

<table>
<thead>
<tr>
<th>Evaluation</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td>AES_decrypt, AES_encrypt, AES_ecb_encrypt, enc, dec</td>
</tr>
<tr>
<td>Two</td>
<td>AES_decrypt, AES_encrypt</td>
</tr>
</tbody>
</table>

To measure the performance cost of code within enclave (first factor):

- All encryption/decryption computations are performed within one enclave.
- Pointers on key and data blocks are passed in through the interface.
Evaluation Setup

<table>
<thead>
<tr>
<th></th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation One</td>
<td>AES_decrypt, AES_encrypt, AES_ecb_encrypt, enc, dec</td>
</tr>
<tr>
<td>Evaluation Two</td>
<td>AES_decrypt, AES_encrypt</td>
</tr>
</tbody>
</table>

To measure the impact of inter-enclave control flow transfers *(second factor)*:
- Put the block-level encryption/decryption functions into the enclave.
- Control the number of inter-enclave control transfers by changing the length of the input data.
Evaluation Results

4× overhead over computation without SGX when processing over 100k data blocks, overhead is 6.91%.
Evaluation Results

<table>
<thead>
<tr>
<th>Case</th>
<th>Input Bin (KB)</th>
<th>Output Bin (KB)</th>
<th>Interface Libs (KB)</th>
<th>Enclaves (KB)</th>
<th>Output Total (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation One</td>
<td>48</td>
<td>48</td>
<td>16</td>
<td>116</td>
<td>180</td>
</tr>
<tr>
<td>Evaluation Two</td>
<td>48</td>
<td>48</td>
<td>12</td>
<td>108</td>
<td>168</td>
</tr>
</tbody>
</table>

We measure the size increase in terms of multiple components:
- Size of output binary is identical with the input, since we perform in-place binary instrumentation.
- Both SDK routines and our routine code introduce size increase.
- The overall size increase is within a reasonable extent.
  - **Evaluation One** has three more functions than **Evaluation Two**.
Future works

- Limitations
  - How to reliably recover the function prototype?
  - How to deal with the shared variables among several isolated enclaves?
  - Some instructions/operations may not be supported inside the enclaves.
  - ...

Thanks!

Contact: ww31@indiana.edu