Reusable Enclaves
for Confidential Serverless Computing

Shixuan Zhao
PhD Student @ SecLab
CSE, The Ohio State University
zhao.3289@osu.edu

A joint work with Pinshen Xu, Guoxing Chen, Mengya Zhang, Yinqian Zhang and Zhiqiang Lin
(Confidential) Serverless - What is

The old way

- Your own server/VM
- Libs, OS, updates...
- Too heavy for a small app

Hello world!
Serverless

- Platform does them!
- In JS, Python...
- Commercialised

Hello world!

AWS Lambda
IBM Cloud Functions
Azure Functions
Google Cloud Functions
(Confidential) Serverless - What is Serverless
(Confidential) Serverless - What is

Confidential serverless

Motivations

Design

Implementation

Evaluation

Conclusion

1/17

Internet

Gateway

Web Server

Controller

Dispatcher

Backend Environment

Executor

Enclave-Protected Runtime

Remote User

Gateway Server

Backend
The cold start problem

Root cause
The cold start problem

Executor life cycle

Backend Executor

1 Executor Deployment

Frontend Gateway

1 User Func Uploading

Payload
The cold start problem

Executor life cycle
The cold start problem

Executor life cycle

**Backend Executor**
- 1. Executor Deployment
- 2. Payload Execution
- 3. Return Result

**Frontend Gateway**
- 1. User Func Uploading
- 2. Execution Request
- 3. Return Result

**Payload**
- Event/Request
- Result
The cold start problem

Executor life cycle

1. **Executor Deployment**
2. **Payload Execution**
3. **Return Result**
The cold start problem

Executor life cycle
The cold start problem

Penalty of Cold Start

Facts [1]

- Industry median memory
  - 170 MiB
- 50% workloads ends within
  - 1 s

The solution?

Executor life cycle
The solution?

Keep warm?

- Keep user environment + runtime?
- Reset runtime?
The solution?

Keep warm?

- Keep user environment + runtime? Precious enclave memory!
- Reset runtime? Buggy Runtime!

⚠️ Not good enough for confidential serverless!
Our solution

Question
Can we reset the enclave?

Backend Environment
Executor
Enclave-Protected Runtime

Runtime
User Code

Keep Warm
4 Environment Reset
Reuse

Only Runtime

Can we reset the enclave?
Our solution

Challenges

How to reset?

How to prove the reset?

How to secure the reset?
# Our solution

## Challenges

<table>
<thead>
<tr>
<th>How to reset?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enclave snapshot &amp; rewinding</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>How to prove the reset?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nested attestation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>How to secure the reset?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-Layer Intra-Enclave Compartmentalisation (MLIEC)</td>
</tr>
</tbody>
</table>
Our solution

Challenges

How to reset?
Enclave snapshot & rewinding

How to prove the reset?
Nested attestation

How to secure the reset?
Multi-Layer Intra-Enclave Compartmentalisation (MLIEC)

Generic architecture-independent method!
Enclave snapshot & rewinding

- Reset: Bring the enclave back to a known good state
- Take a snapshot and rewind

What a snapshot needs?
- Small memory footprint
- Fast to rewind
Enclave snapshot & rewinding

Initially...

- Stack, heap: Empty (zeros)
- .text: Read only (for now)
- .data, .bss
Enclave snapshot & rewinding

Initially...
- Stack, heap: Empty (zeros)
- .text: Read only (for now)
- .data, .bss

New reset module!
- Snapshot = copy
- Rewinding = copy back + zeroing
Nested attestation

Typical enclave attestation:
- Boot time only

How to...
- Prove the reset indeed took place?
- Prove the reset is correct?
- User workload attestation?
Nested attestation

New attestation module!

- Public-private key pair
- Reports with reset info
- User payload info

Diagram:

- Stack
- Heap
- ... (omitted)
- .bss
- ...
- .data
- ...
- .text
- .snapshot
- .reset
- .cdata
- .attest
- Metadata

Stack
Heap...
.data...
.text
.snapshot
.reset
.cdata
.attest
Metadata
Multi-Layer Intra-Enclave Compartmentalisation

**Observations**

- Runtime (.text) can be buggy
- Must not touch snapshots and attestation data
- Layers of security
Multi-Layer Intra-Enclave Compartmentalisation

- A higher-security layer can access lower one’s data
- Not vice versa!

Software-Fault Isolation (SFI)
- Inspired by SGX-Shield [1]
- Compiler techniques

MLIEC: SMA & aligned branching

Shepherded memory access

- R/W boundary for each layer
- Only allow R/W above the boundary
MLIEC: SMA & aligned branching

Shepherded memory access
- R/W boundary for each layer
- Only allow R/W above the boundary

Steps
- Step 1: Get offset from boundary
- Step 2: Make offset positive
- Step 3: Add back to boundary
- Step 4: Access

Before
```plaintext
mov %rax, (%rdx)
```

After
```plaintext
mov %rdx, %r14
sub %r15, %r14
shl $1, %r14
shr $1, %r14
add %r15, %r14
mov %rax, (%r14)
```
MLIEC: SMA & aligned branching

Problem...

Branching to arbitrary address can bypass the SMA

```
mov %rdx, %r14
sub %r15, %r14
shl $1, %r14
shr $1, %r14
add %r15, %r14
mov %rax, (%r14)
```
MLIEC: SMA & aligned branching

Problem...
Branching to arbitrary address can bypass the SMA

Aligned branching
- Branching to arbitrary address can bypass the SMA
- Emit code into fixed-size blocks (e.g., 32 bytes)
- Force all branching aligned to the block size

32-byte block
- mov %rdx, %r14
- sub %r15, %r14
- shl $1, %r14
- shr $1, %r14
- add %r15, %r14
- mov %rax, (%r14)
MLIEC: SMA & aligned branching

Problem...
Branching to arbitrary address can bypass the SMA

Aligned branching
- Branching to arbitrary address can bypass the SMA
- Emit code into fixed-size blocks (e.g., 32 bytes)
- Force all branching aligned to the block size

Before
```
jmp *%rax
```

After
```
and ~$32, %rax
jmp *%rax
```
MLIEC: Unaligned critical functions

Problem...

Not all functions can be instrumented...

- Security:
  - Boundary setup
  - RWX-granting
- Performance
  - memcpy
MLIEC: Unaligned critical functions

Problem...
Not all functions can be instrumented...
- Security:
  - Boundary setup
  - RWX-granting
- Performance
  - memcpy
- Checks + main logic executed as a whole without instrumentation
⇒ Control Flow Integrity (CFI)
MLIEC: Unaligned critical functions

Traditional CFI
Trap & check
• Slow!

⚠️
• Branches are aligned
⇒ Can’t branch to unaligned target
MLIEC: Unaligned critical functions

Traditional CFI

Trap & check
  • Slow!

⚠️
  • Branches are aligned
    ⇒ Can’t branch to unaligned target

Solution

• Emit a bomb before each block
• Chain blocks with jumps
MLIEC: Multi-layer compartmentalisation

Why

• Least privilege principle
• Attestation > reset > runtime

How

• Flexible boundary
• Linker script

```
1  ...
2  SECTION
3  {
4    ...
5    .attest
6    __boundary_1 = ;
7    .cdta
8    .reset
9    __boundary_2 = ;
10   .snapst
11   ...
12   .text
13   ...
14   __boundary_3 = ;
15   .data
16   ...
17   .bss
18   ...
```
MLIEC: Dynamically-loaded code

Read-only code: Not enough for serverless

**Ahead-of-Time (AoT)**
- Bytecode to native binaries
- Good performance
- May contain any code/instructions...

[Diagram showing backend environment with executor, enclave-protected runtime, and user function]
**Motivations**

Read-only code: Not enough for serverless

**Ahead-of-Time (AoT)**

- Bytecode to native binaries
- Good performance
- May contain any code/instructions...

**Solution**

Use MLIEC techniques when compiling AoT binaries
MLIEC: Dynamically-loaded code

RWX granting function

- AoT requires RWX area
- Protect RWX granting with unaligned critical functions
- Disable it before user code execution
Implementation

- MLIEC: LLVM-based toolchain
- Enclave: Intel SGX
- Frontend: OpenWhisk
  - Open source platform
  - Widely adopted
- Backend: WAMR
  - Open source
  - AoT mode

Code based on WebAssembly on OpenWhisk (WOW) [1]

Implementation

Motivations

Design

Implementation

Evaluation

Conclusion

Implementation

WAMR Enclave

Invoker (WOW Executor)

Regular Heap/Stack/Data

WebAssembly Micro Runtime

Snapshot

Reset Module

Communication

Attestation

OCall Handlers

Untrusted Communication

OpenWhisk

Nginx

OpenWhisk Controller

Kafka

Gateway Enclave

SGX Attestation Communication

Out of Enclave

Internet

In Enclave

Layer 3

Layer 2

Layer 1

Remote User

Gateway Server

Backend
Implementation

- LLVM: 1070 LoC
- OpenWhisk:
  - Action: 107 LoC
  - Gateway Untrusted: 1478 LoC
  - Gateway Enclave: 1978 LoC
- Executor:
  - WOW: 1457 LoC
  - Enclave: 4098 LoC
- Total: 10188 LoC
Relaunch vs. reuse

![Graph comparing relaunch and reset times with heap size (MiB) on the x-axis and time (ms) on the y-axis. The graph shows that relaunch/reset time decreases as heap size increases.](image-url)
Instrumentation overheads

Motivations

Design

Implementation

Evaluation

Conclusion

Instrumentation overheads

- Baseline (Our System)
- AoT
- Classic
- Fast

NUMERIC SORT
STRING SORT
BITFIELD
FP EMULATION
FOURIER

ASSIGNMENT
IDEA
HUFFMAN
NEURAL NET
LU DECOMPOSITION
Geo Mean
Real-world end-to-end

Conclusion

Reset benefits are significant

Solved cold start problem in confidential serverless with reusable encalves

- Enclave snapshot & rewinding
- Nested attestation
- MLIEC
Q&A

Source Code
https://github.com/OSUSecLab/Reusable-Enclaves

SecLab @ OSU
https://go.osu.edu/seclab

Teecert Labs @ SUSTech
https://teecertlabs.com

NSEC @ SJTU
https://nsec.sjtu.edu.cn