



SecLab

COMPUTER SECURITY LABORATORY



10th Aug., 2023

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



(Confidential) Serverless - What is

Serverless

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



AWS Lambda



IBM Cloud Functions



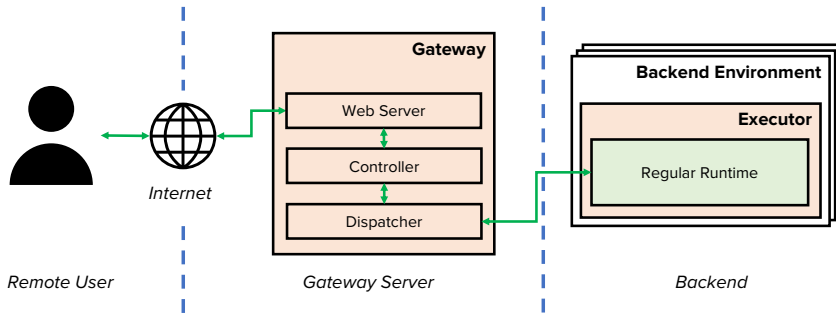
Azure Functions



Google Cloud Functions

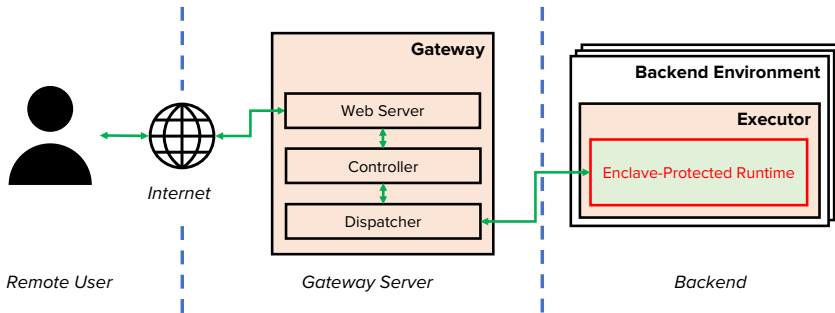
(Confidential) Serverless - What is

Serverless



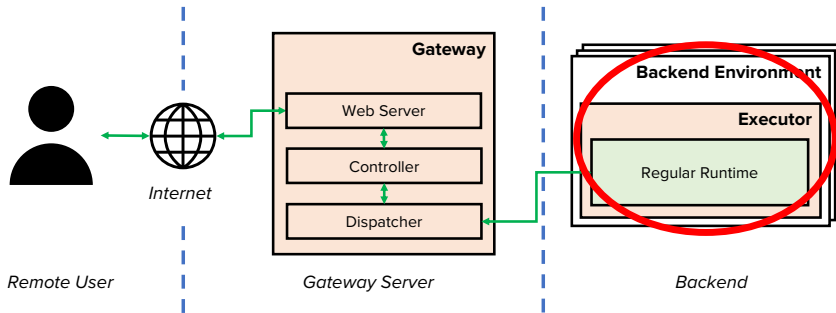
(Confidential) Serverless - What is

Confidential serverless



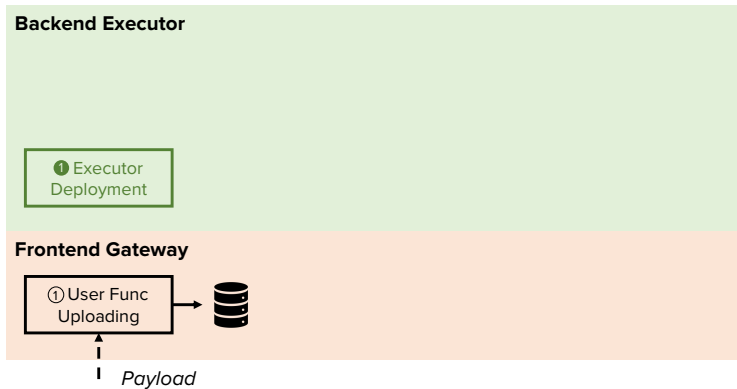
The cold start problem

Root cause



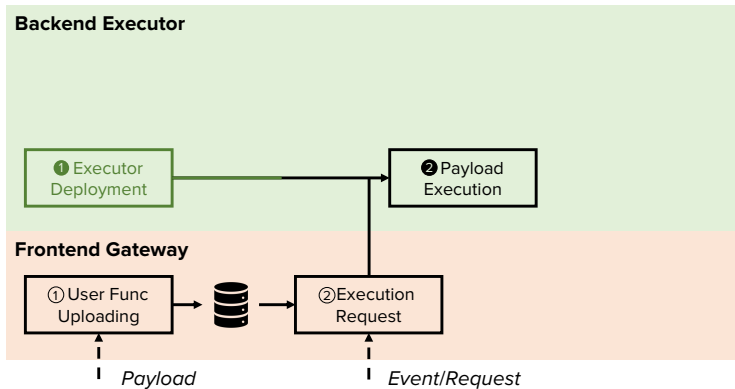
The cold start problem

Executor life cycle



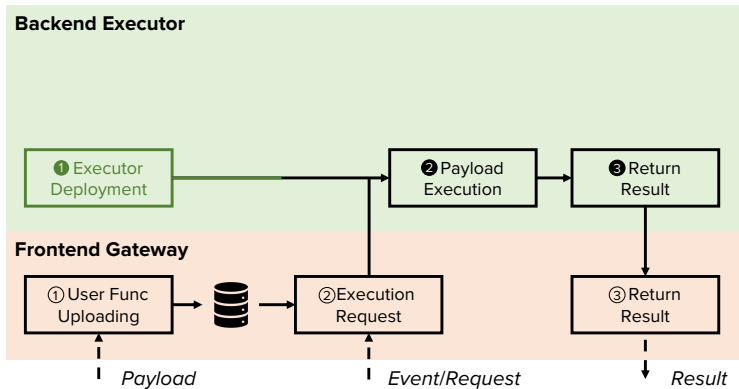
The cold start problem

Executor life cycle



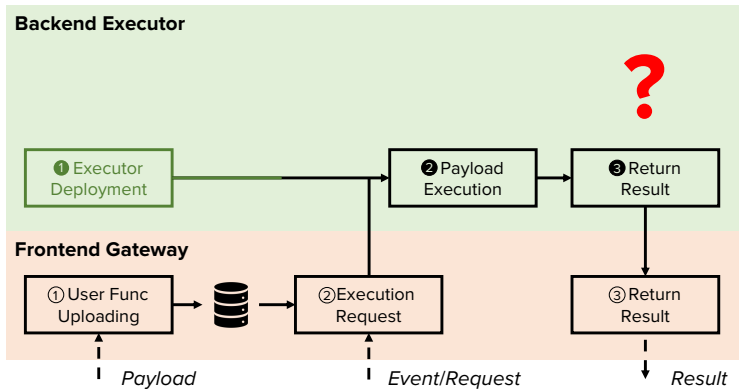
The cold start problem

Executor life cycle



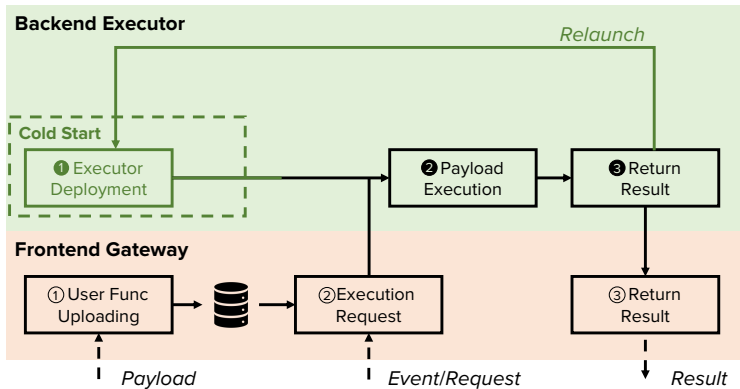
The cold start problem

Executor life cycle



The cold start problem

Executor life cycle

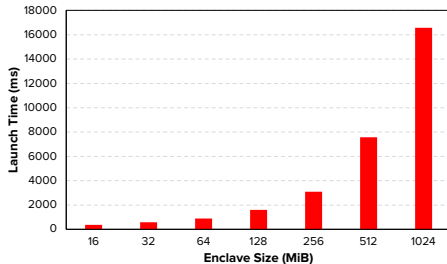


The cold start problem

Penalty of Cold Start

Facts [1]

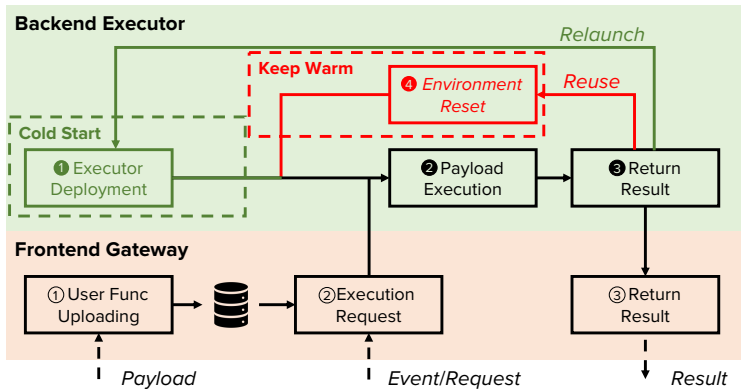
- Industry median memory
 - 170 MiB
- 50% workloads ends within
 - 1 s



[1] M. Shahrad, et al, 'Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider,' ATC 2020.

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!



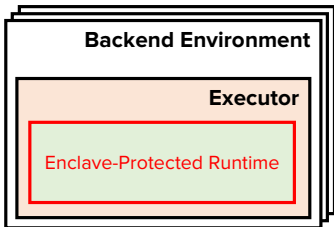
Only Runtime



Our solution

Question

Can we reset the enclave?



Our solution

Challenges

How to reset?

How to prove the reset?

How to secure the reset?

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)

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)

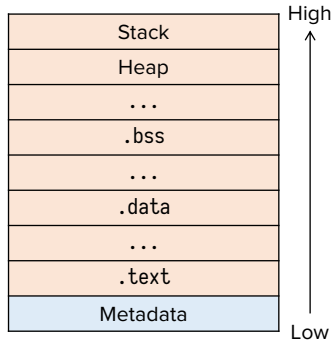
Generic architecture-independent method!

Enclave snapshot & rewinding

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

What a snapshot needs?

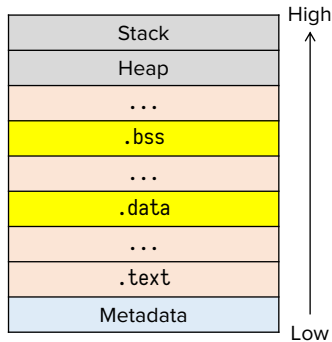
- Small memory footprint
- Fast to rewind



Enclave snapshot & rewinding

Initially...

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



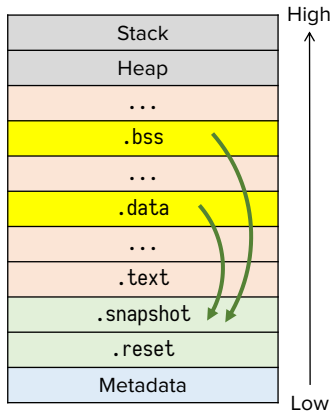
Enclave snapshot & rewinding

Initially...

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

New reset module!

- Snapshot = copy
- Rewinding = copy back + zeroing



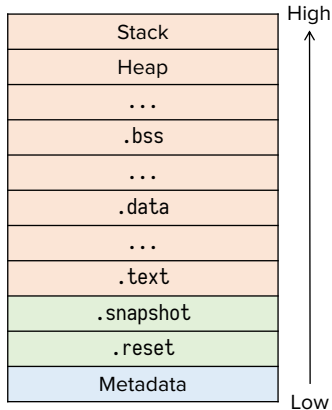
Nested attestation

Typical enclave attestation:

- Boot time only

How to...

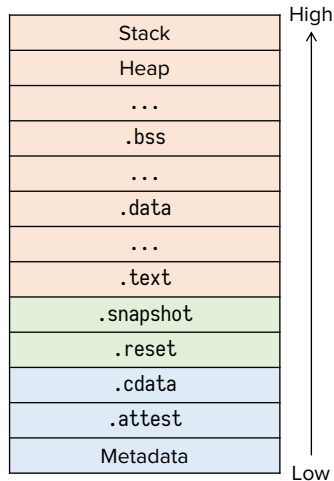
- Prove the reset indeed took place?
- Prove the reset is correct?
- User workload attestation?



Nested attestation

New attestation module!

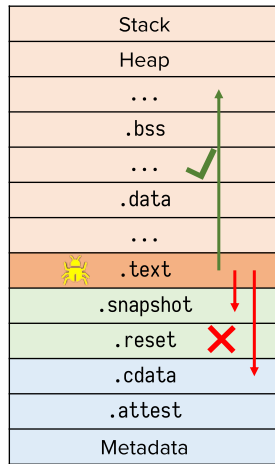
- Public-private key pair
- Reports with reset info
- User payload info



Multi-Layer Intra-Enclave Compartmentalisation

Observations

- Runtime (.text) can be buggy
- Must not touch snapshots and attestation data
- Layers of security



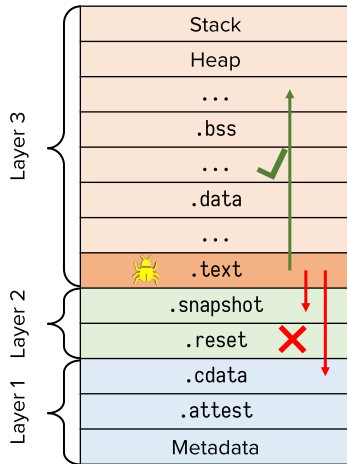
Multi-Layer Intra-Enclave Compartmentalisation

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

Software-Fault Isolation (SFI)

- Inspired by SGX-Shield [1]
- Compiler techniques

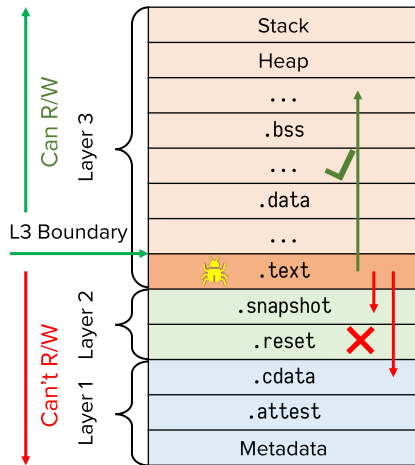
[1] J. Seo, et al, 'SGX-Shield: Enabling Address Space Layout Randomization for SGX Programs,' NDSS 2017.



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

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


After

	<code>mov</code>	<code>%rdx, %r14</code>
Step 1	<code>sub</code>	<code>%r15, %r14</code>
Step 2	<code>shl</code>	<code>\$1, %r14</code>
	<code>shr</code>	<code>\$1, %r14</code>
Step 3	<code>add</code>	<code>%r15, %r14</code>
Step 4	<code>mov</code>	<code>%rax, (%r14)</code>

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)
```

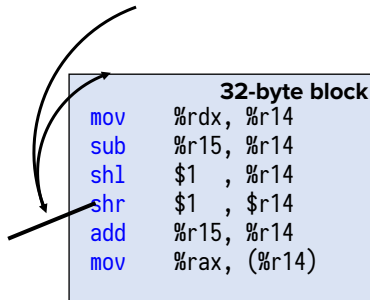
MLIEC: SMA & aligned branching

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



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



Security



Performance

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)



Security



Performance

MLIEC: Unaligned critical functions

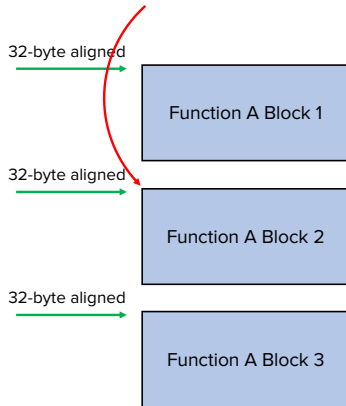
Traditional CFI

Trap & check

- Slow!



- Branches are aligned
- ⇒ Can't branch to unaligned target



MLIEC: Unaligned critical functions

Traditional CFI

Trap & check

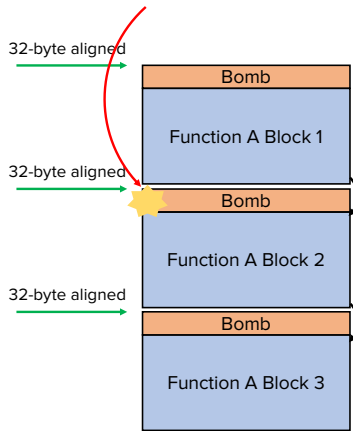
- Slow!



- 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

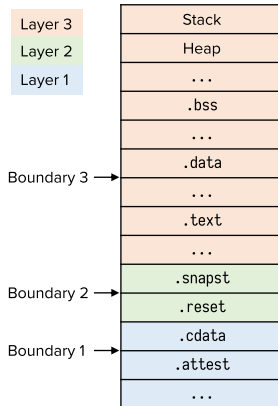
How

- Flexible boundary
- Linker script

```

1 ...
2 SECTION
3 {
4   ...
5   .attest
6   __boundary_1 = .;
7   .cdata
8   .reset
9   __boundary_2 = .;
10  .snapst
11  ...
12  .text
13  ...
14  __boundary_3 = .;
15  .data
16  ...
17  .bss
18  ...

```

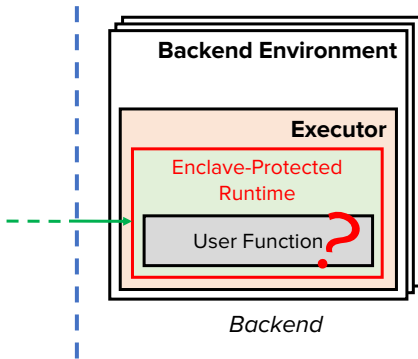


MLIEC: Dynamically-loaded code

Read-only code: Not enough for serverless

Ahead-of-Time (AoT)

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



MLIEC: Dynamically-loaded code

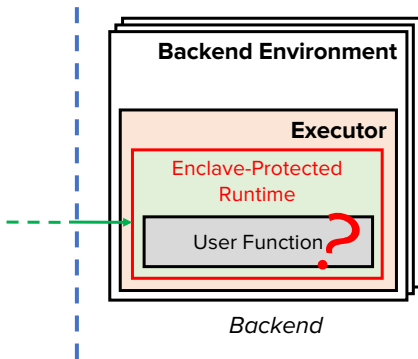
Read-only code: Not enough for serverless

Ahead-of-Time (AoT)

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

Solution

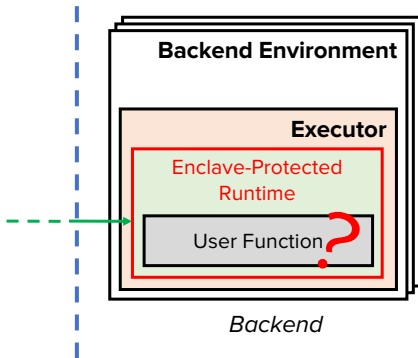
Use MLIEC techniques when compiling AoT binaries



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

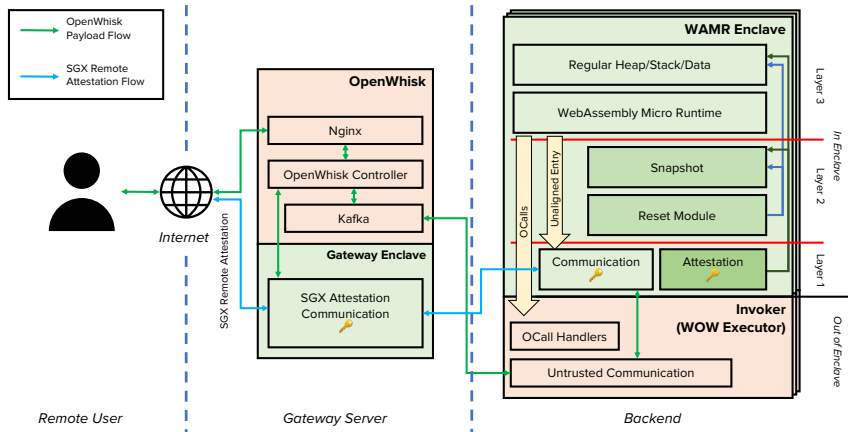
- MLIEC: LLVM-based toolchain
- Enclave: Intel SGX
- Frontend: OpenWhisk
 - Open source platform
 - Widely adopted
- Backend: WAMR
 - Open source
 - AoT mode



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

[1] P. Gackstatter, et al, 'Pushing Serverless to the Edge with WebAssembly Runtimes,' CCGrid 2022.

Implementation

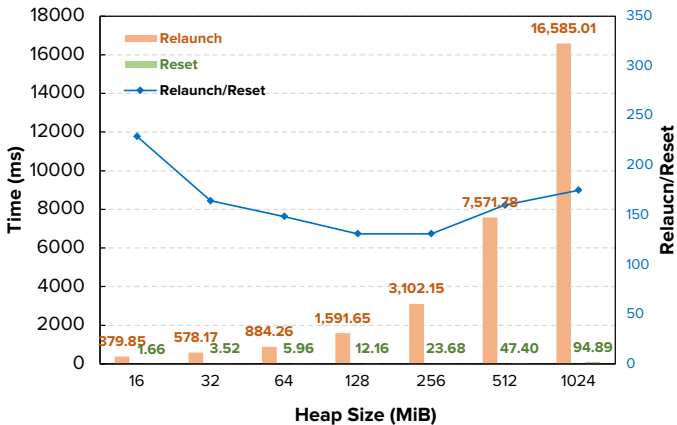


Implementation

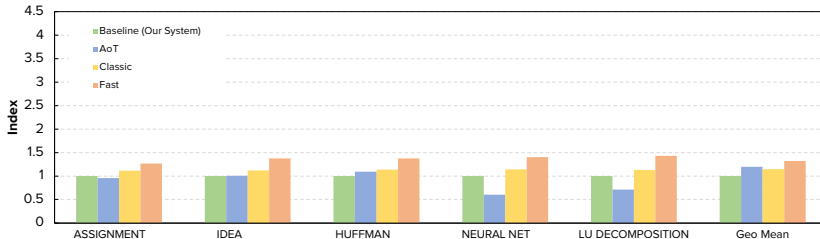
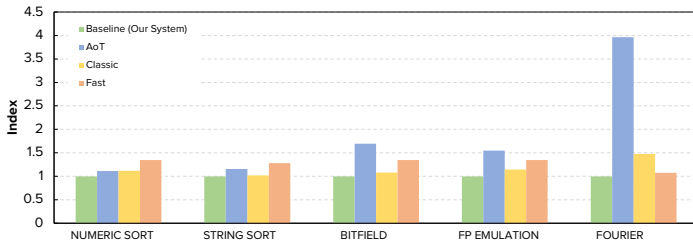
- LLVM: 1070 LoC
- OpenWhisk:
 - Action: 107 LoC
 - Gateway Untrusted: 1478 LoC
 - Gateway Enclave: 1978 LoC
- Executor:
 - WOW: 1457 LoC
 - Enclave: 4098 LoC
- Total: 10188 LoC



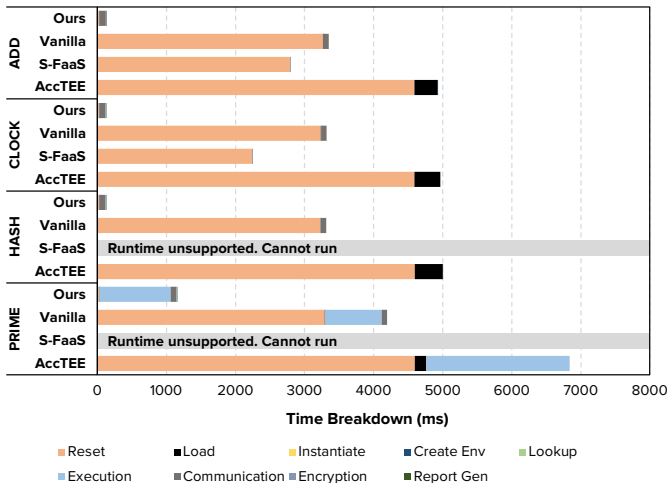
Relaunch vs. reuse



Instrumentation overheads



Real-world end-to-end



[1] F. Alder, et al, 'S-FaaS: Trustworthy and Accountable Function-as-a-Service Using Intel SGX,' CCS 2019.

[2] D. Goltzsche, et al, 'AccTEE: A WebAssembly-Based Two-Way Sandbox for Trusted Resource Accounting,' Middleware 2019.

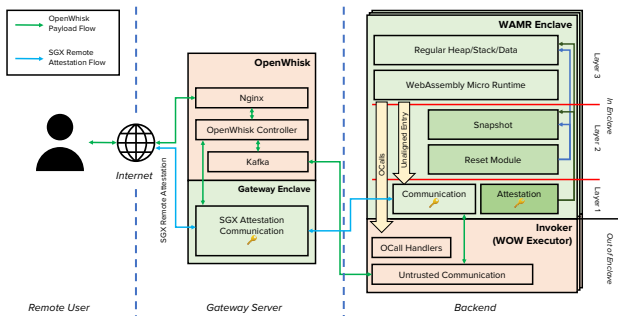
Conclusion



Reset benefits are significant

Solved cold start problem in confidential serverless with reusable encaves

- 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>