Swivel: Hardening WebAssembly against Spectre

Shravan Narayan, Craig Disselkoen, Daniel Moghimi, Sunjay Cauligi,

Evan Johnson, Zhao Gang, Anjo Vahldiek-Oberwagner, Ravi Sahita,

Hovav Shacham, Dean Tullsen, Deian Stefan

What is WebAssembly (Wasm)?

Platform-independent bytecode

Runs C/C++/Rust in the browser

Designed for isolation

Stack	`local.get localidx` `local.set localidx`
Linear Memory	`load offset1 offset2` `store offset1 offset2`
Globals	`globals.get globalidx` `globals.set globalidx`
Control Flow	`load offset1 offset2` `store offset1 offset2`

Wasm is used outside the browser

Securing Firefox with WebAssembly

By Nathan Froyd

Posted on February 25, 2020 in Featured Article, Firefox, Rust, Security, and WebAssembly

Protecting the security and privacy of individuals is a central tenet of Mozilla's mission, and so we constantly endeavor to make our users safer online. With a

...

So today, we're adding a third approach to our arsenal. RLBox, a new sandboxing technology developed by researchers at the University of California, San Diego, the University of Texas, Austin, and Stanford University, allows us to quickly and efficiently convert existing Firefox components to run inside a

WebAssembly on Cloudflare Workers

10/01/2018

Kenton Varda

Announcing Lucet: Fastly's native WebAssembly compiler and runtime

Published March 28, 2019 EDGE

Pat Hickey
Principal Software Engineer

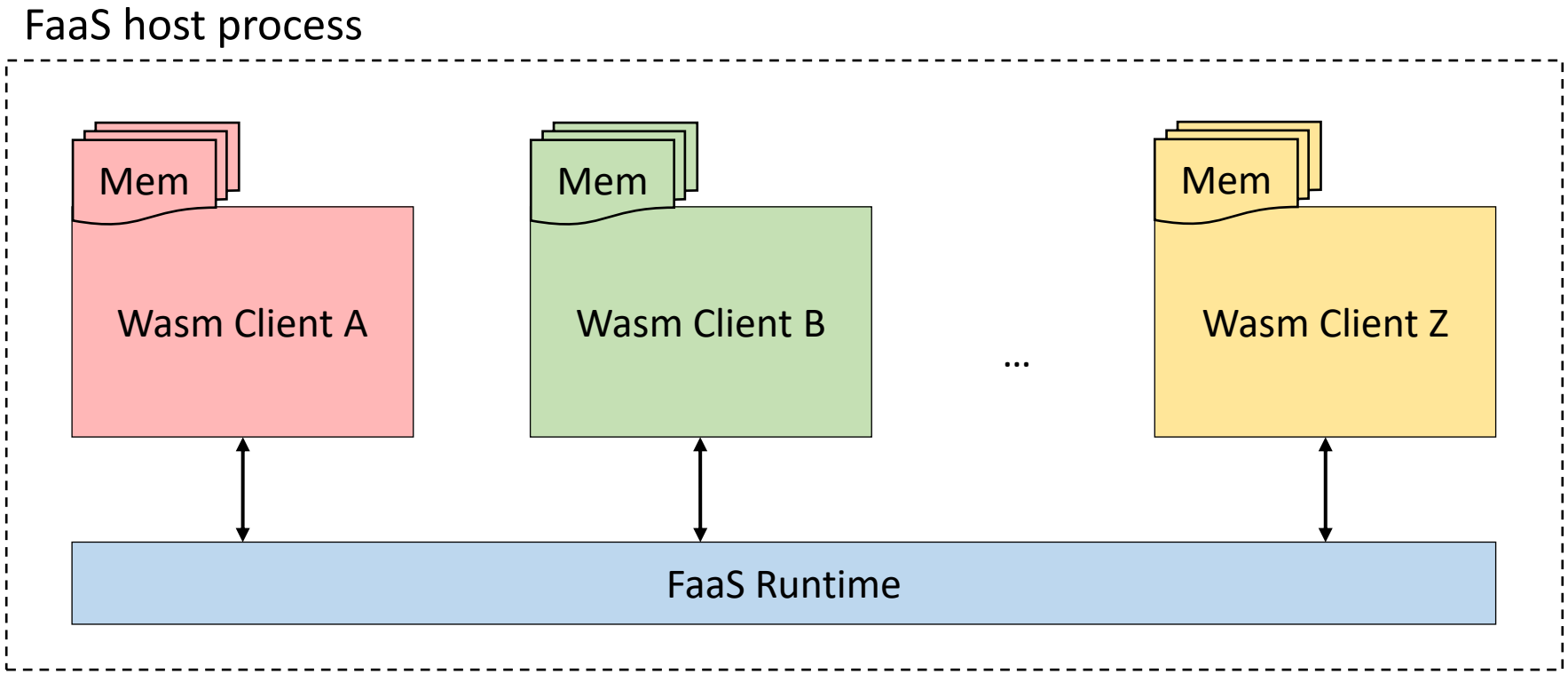
Today, we are thrilled to announce the open sourcing of Lucet, Fastly's native WebAssembly compiler and runtime. WebAssembly is a technology created to enable web browsers to safely execute programs at near-native speeds. It has been shipping in the four major browsers since early 2017.

Veracruz: privacy-preserving collaborative compute

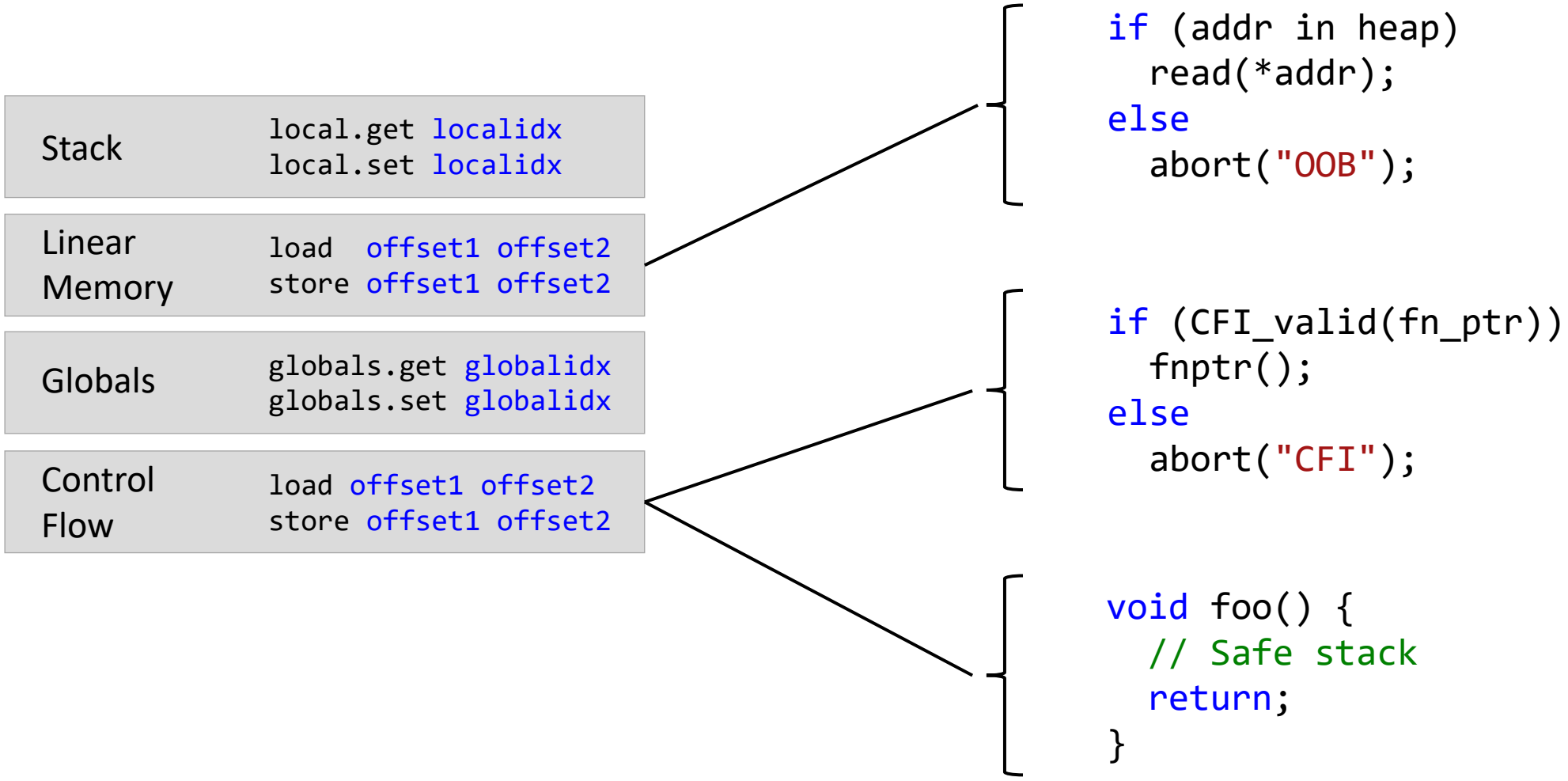
Project Oak

The goal of Project Oak is to create a specification and a reference implementation for the secure transfer, storage and processing of data.

Wasm on FaaS platforms

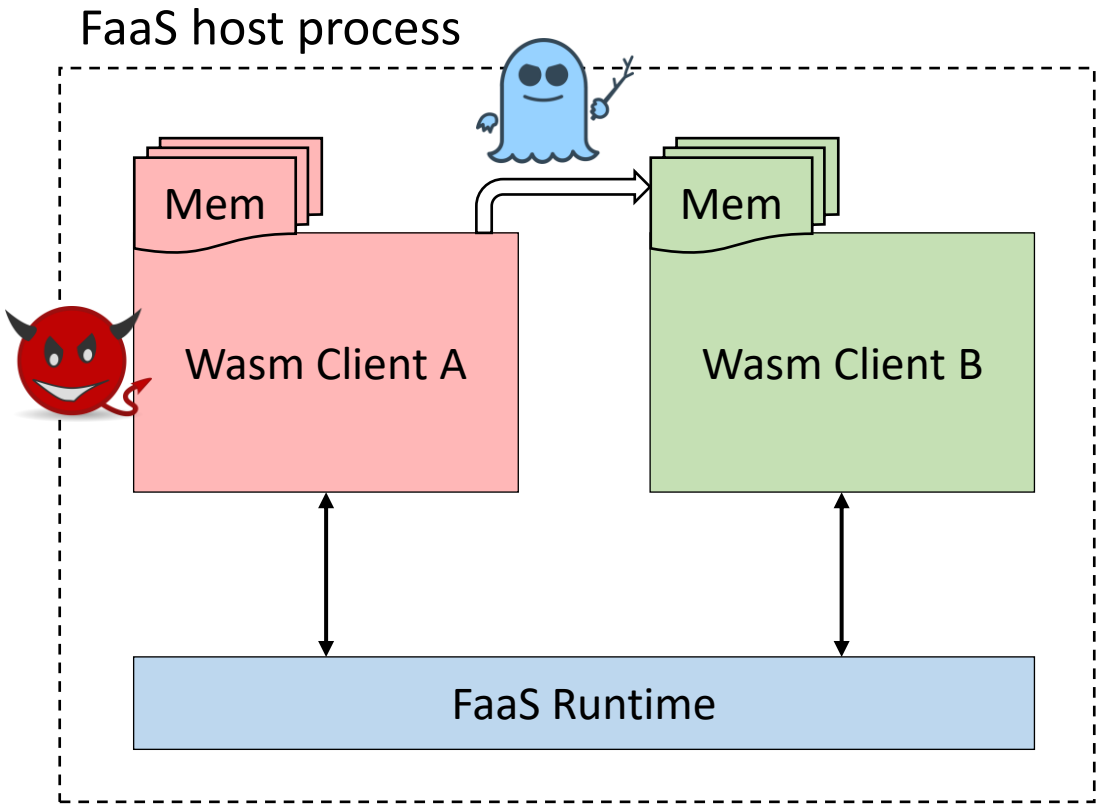


How does Wasm enforce isolation?



Problem: Spectre breaks Wasm isolation

Eg: Using Spectre-PHT to break isolation



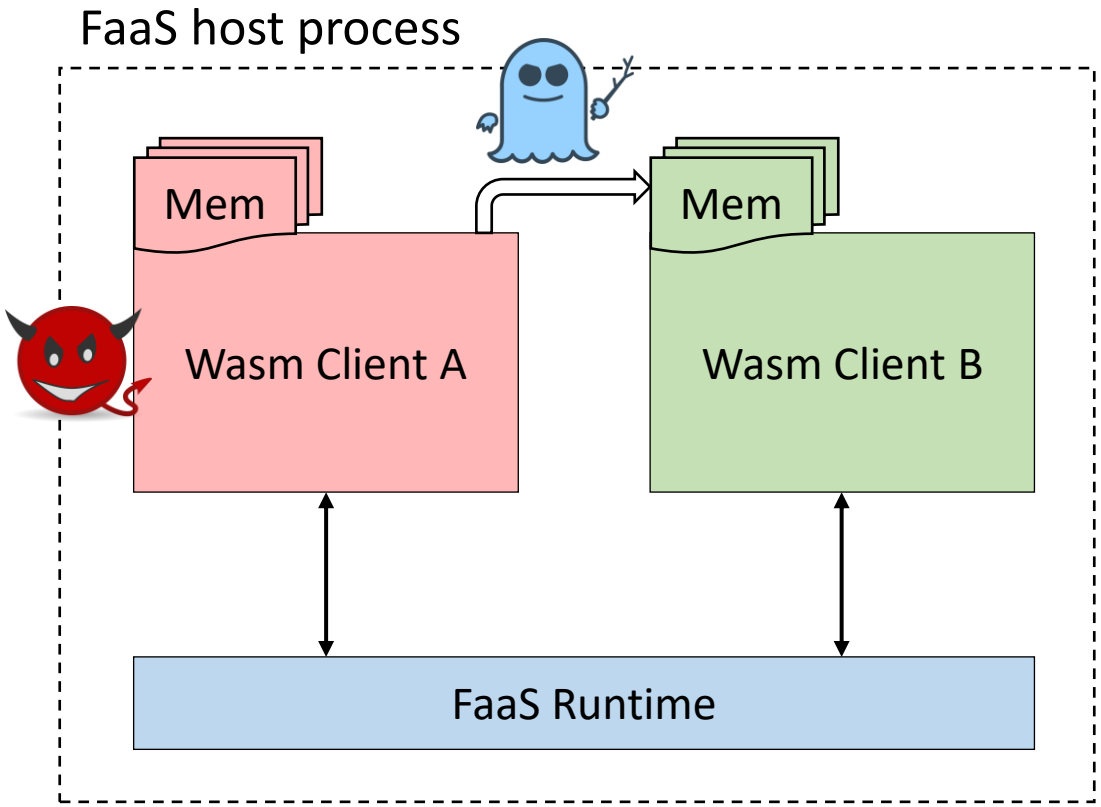

```
if (addr in heap) {
  x = read(*addr);
  y = read(x);
}
```

OOB read

Leaks data via cache

Expected: false
Predicted: true

Eg: Using Spectre-{BTB, RSB}

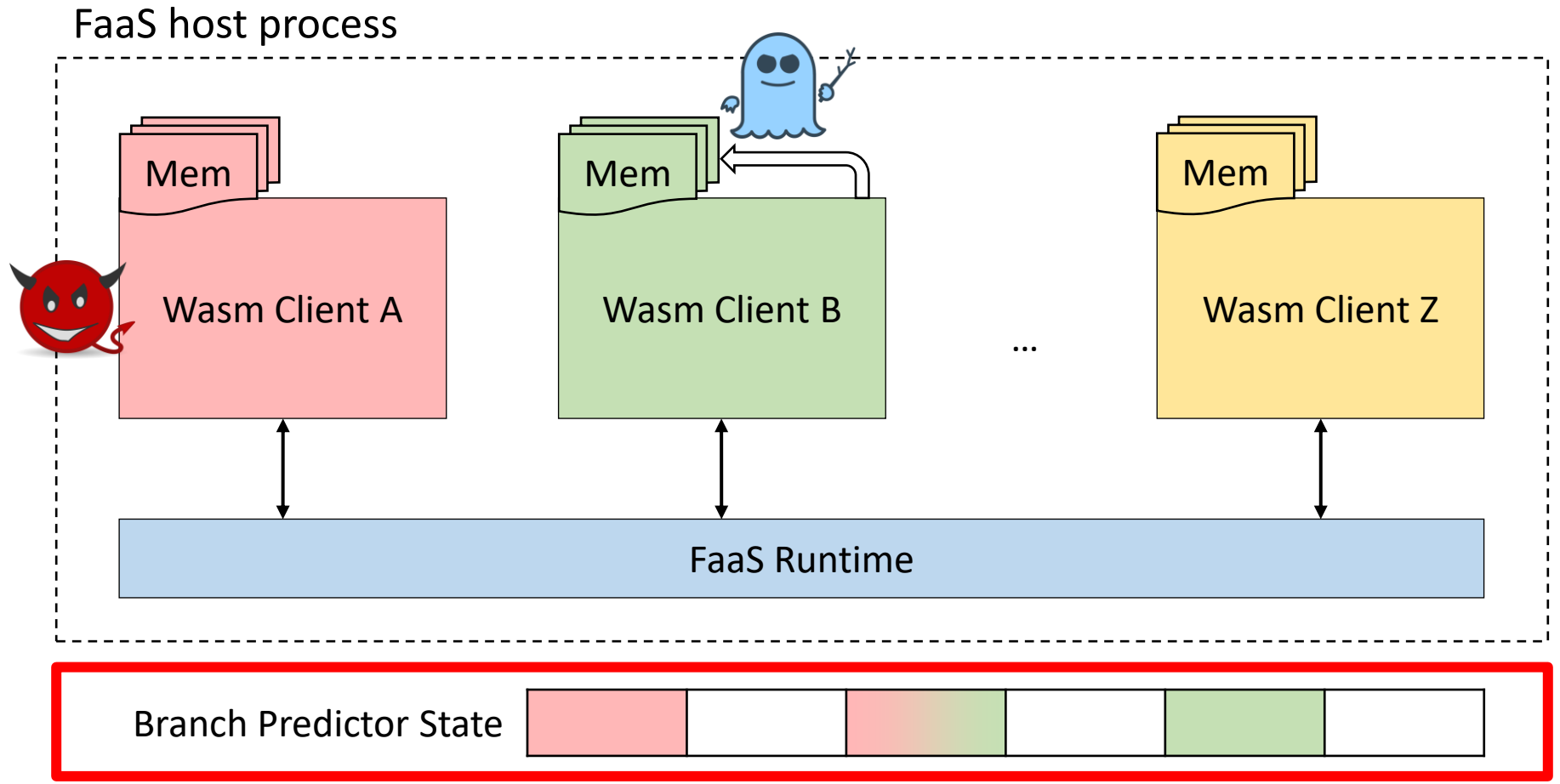



```
(*fnptr)();
return;
```

Expected: 0x1111111
Predicted: 0xbadc0de

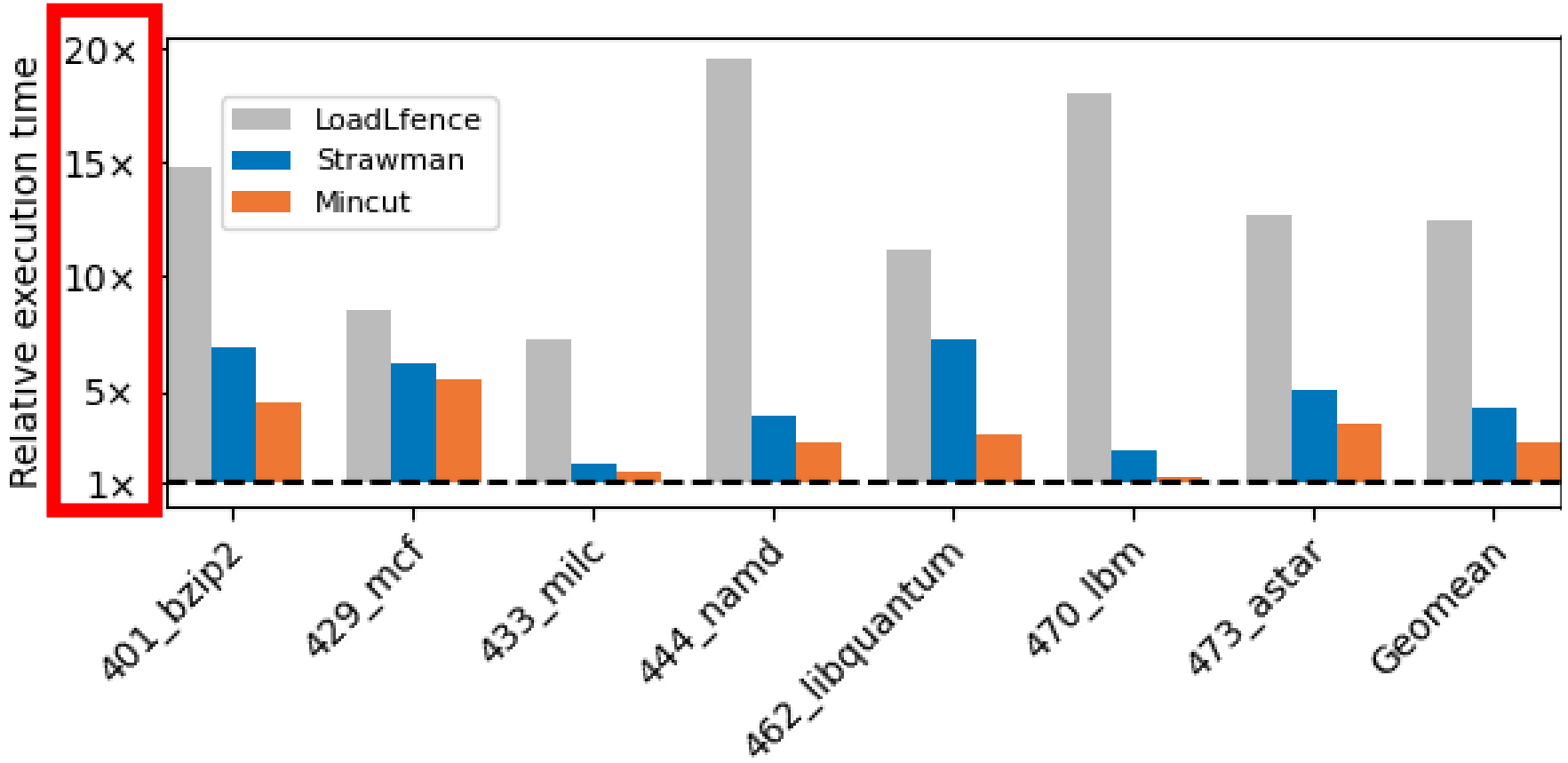
Speculative JOP/ROP

Alternately: Poisoning → Self exfiltration



Solution: Add fences!

We tried this: it’s too slow!



Our solution: Swivel

Swivel is a Wasm compiler that prevents:

- Breakout and poisoning attacks via Spectre-{PHT, BTB, RSB}

Swivel has two backends:

- Swivel-SFI: safety using only software checks
- Swivel-CET: safety with existing hardware extensions, allows hyperthreading

Fundamental problem

Wasm safety checks: function granularity

Speculative control flow can start anywhere

- Can bypass security checks

We need a new abstraction when compiling Wasm

Wasm code

```
func_foo:
  ...
  mem_bounds_check <reg_mem>
  call bar
  ...
  load <reg_mem>
  jmp  ...
```

Key abstraction: Linear blocks (LB)

Like basic blocks, except …

- Instruction sequences that end in a jump or call instruction
- Must include safety checks within the block
- Checks are speculatively safe

Wasm code

```
func_foo:
  ...
  mem_bounds_check <reg_mem>
  call bar
  ...
  load <reg_mem>
  jmp  ...
```

Wasm code with LB

```
linear_block_1:
  ...
  call bar
```

```
linear_block_2:
  ...
  safe_mem_bounds_mask <reg_mem>
  load <reg_mem>
  jmp  ...
```

1. Terminator is control flow / call inst

2. Checks are in same LB as instruction

3. Checks are speculatively safe masks

Swivel-SFI: Builds on Linear blocks (LB)

Spectre-PHT: LBs handles Spectre-PHT breakout attacks

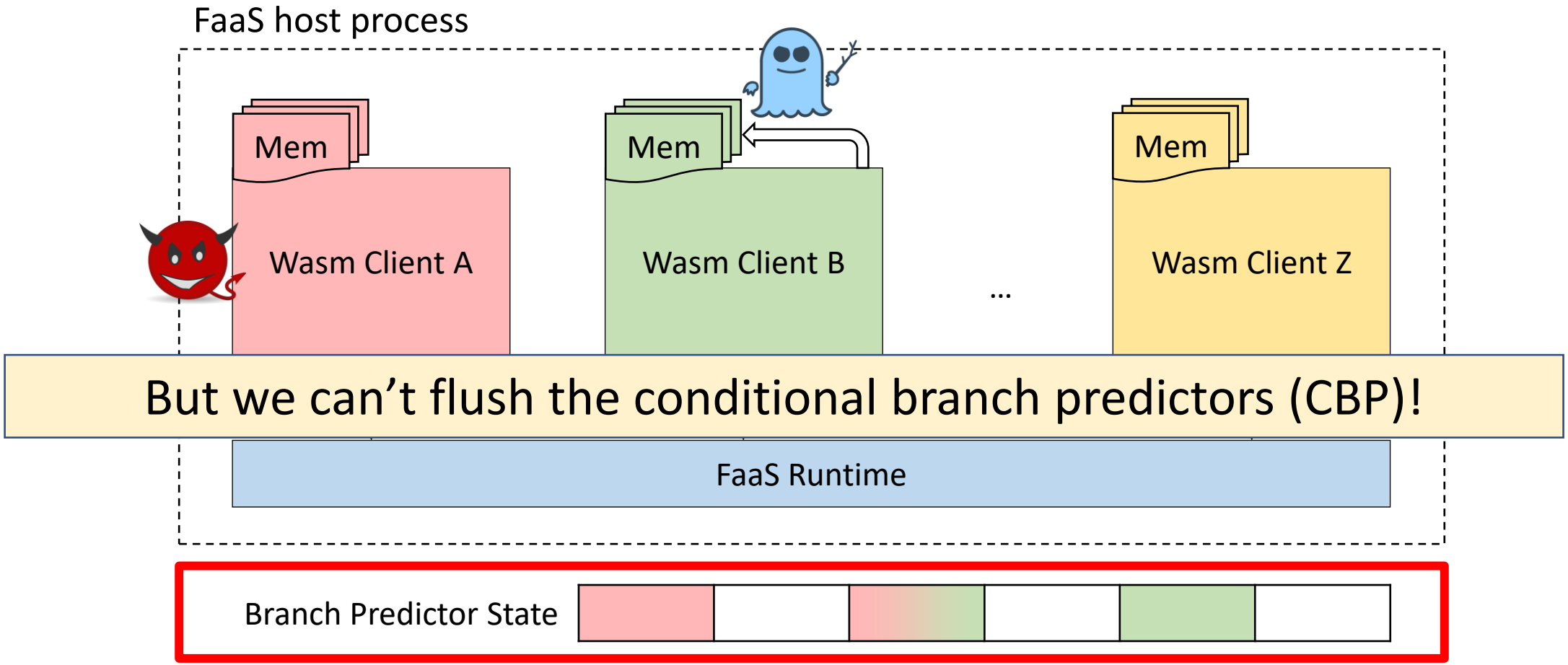
Spectre-BTB: LBs ensure that BTB targets only predict LBs

- Problem: BTB may not be empty when we enter the sandbox
- Solution: Flush the BTB before entering the sandbox

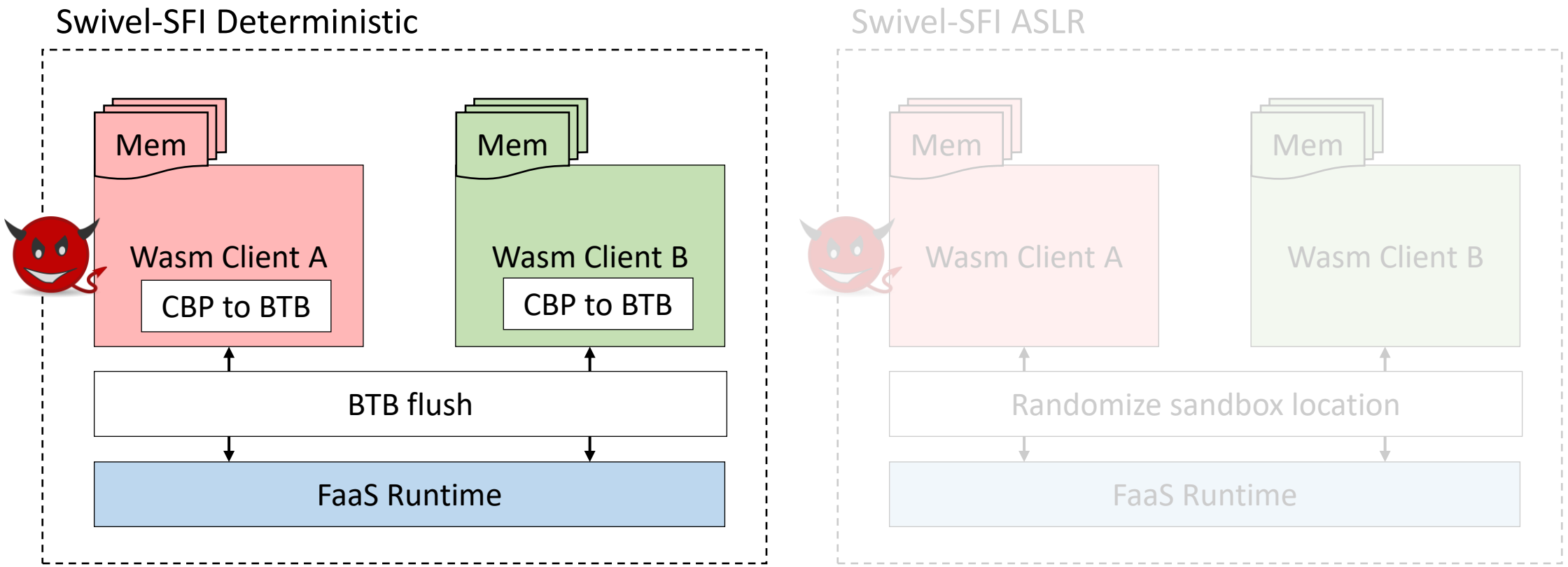
Spectre-RSB: LBs ensure that RSB only predict LBs

- Problem: RSB underflow → predict arbitrary target
- Solution: Separate control stack + use jumps instead of returns

What about sandbox poisoning attacks?



What about sandbox poisoning attacks?



Swivel's security guarantees

Attack variant		Swivel-SFI		Swivel-CET	
		ASLR	Det	ASLR	Det
Spectre-PHT	in-place	●	●	●	●
	out-of-place	◐	●	◐	●
Spectre-BTB	in-place	●	●	●	●
	out-of-place	●	●	●	●
Spectre-RSB	in-place	●	●	●	●
	out-of-place	●	●	●	●

Evaluation

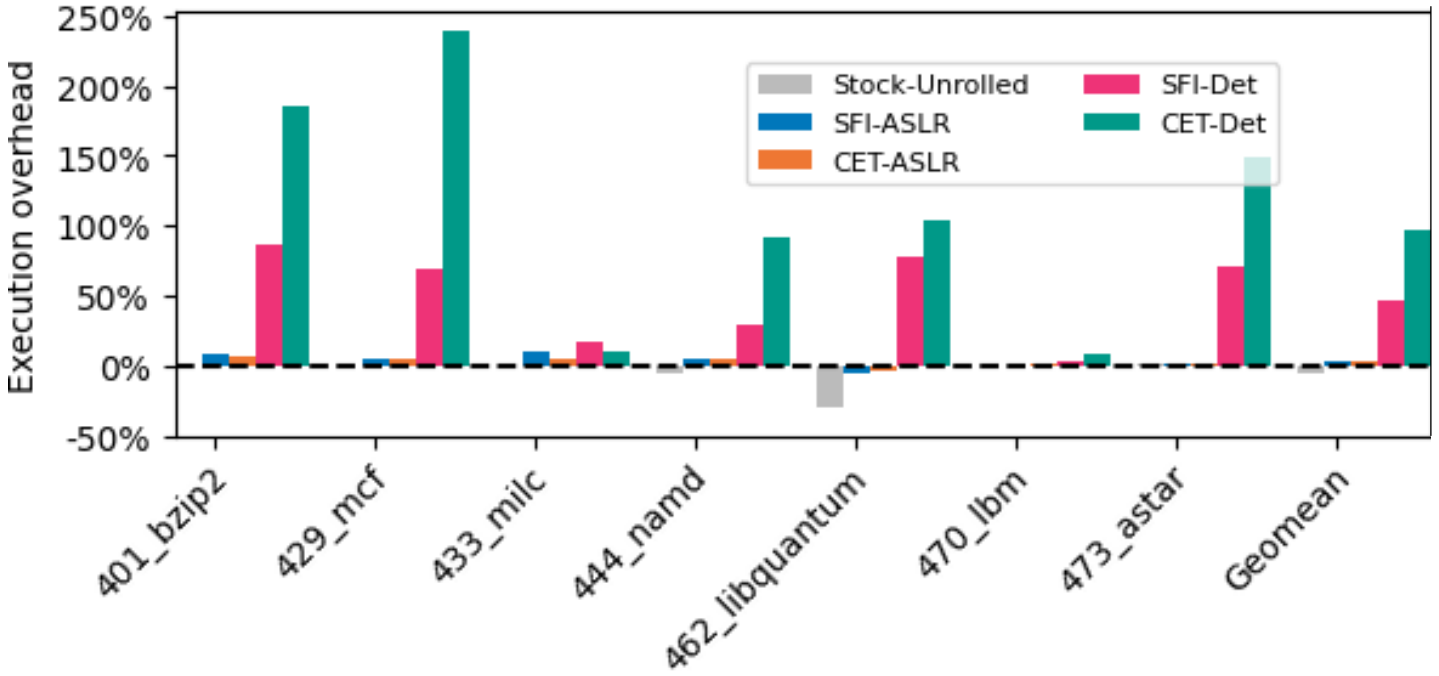
Performance

- Standard benchmark suites – SPEC 2006
- Macro benchmark – mock FaaS platform with Swivel services
- Baseline: Stock (insecure) Wasm

Security

- Implemented POC’s for Spectre-{PHT, BTB, RSB}

SPEC 2006 benchmark



Swivel ASLR: < 10%
Swivel Det: 3% to 240%

Fences are too slow!
Std fence solutions: 6x to 19x
Min fence solutions: 2x to 5x

FaaS platform benchmark

Swivel Protection	XML to JSON		Templated HTML		Image classification	
	Throughput	Perf Loss	Throughput	Perf Loss	Throughput	Perf Loss
Stock (unsafe)	531	-	4.81k	-	2.05	-
Swivel-SFI ASLR	459	13.6%	803	83.3%	2.03	1%
Swivel-SFI Det	350	34.1%	2.90k	39.7%	1.11	45.9%
Swivel-CET ASLR	498	6.2%	898	81.3%	2.02	1.5%
Swivel-CET Det	338	36.3%	3.50k	23.2%	1.26	38.5%

Summary

Swivel secures Wasm from Spectre attacks

Swivel-SFI: backward compatible approach
Swivel-CET: leverages hardware extensions, supports hyperthreading

Key abstraction: linear blocks

https://swivel.programming.systems

 @ShrNarayan