RSAConference2020

San Francisco | February 24 – 28 | Moscone Center

HUMAN ELEMENT

SESSION ID:XXX-XXXX

Secure Sandboxing in a Post-Spectre World

Tyler McMullen

CTO
Fastly
@tbmcmullen

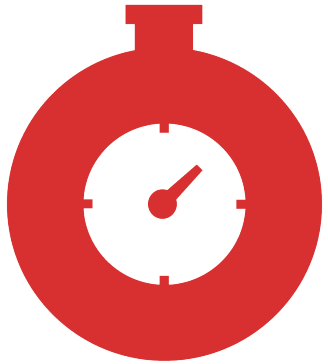
Jonathan Foote

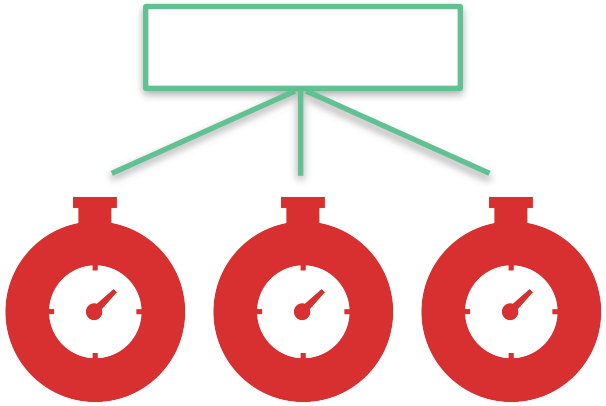
Principal Security Architect
Fastly
@footePGH

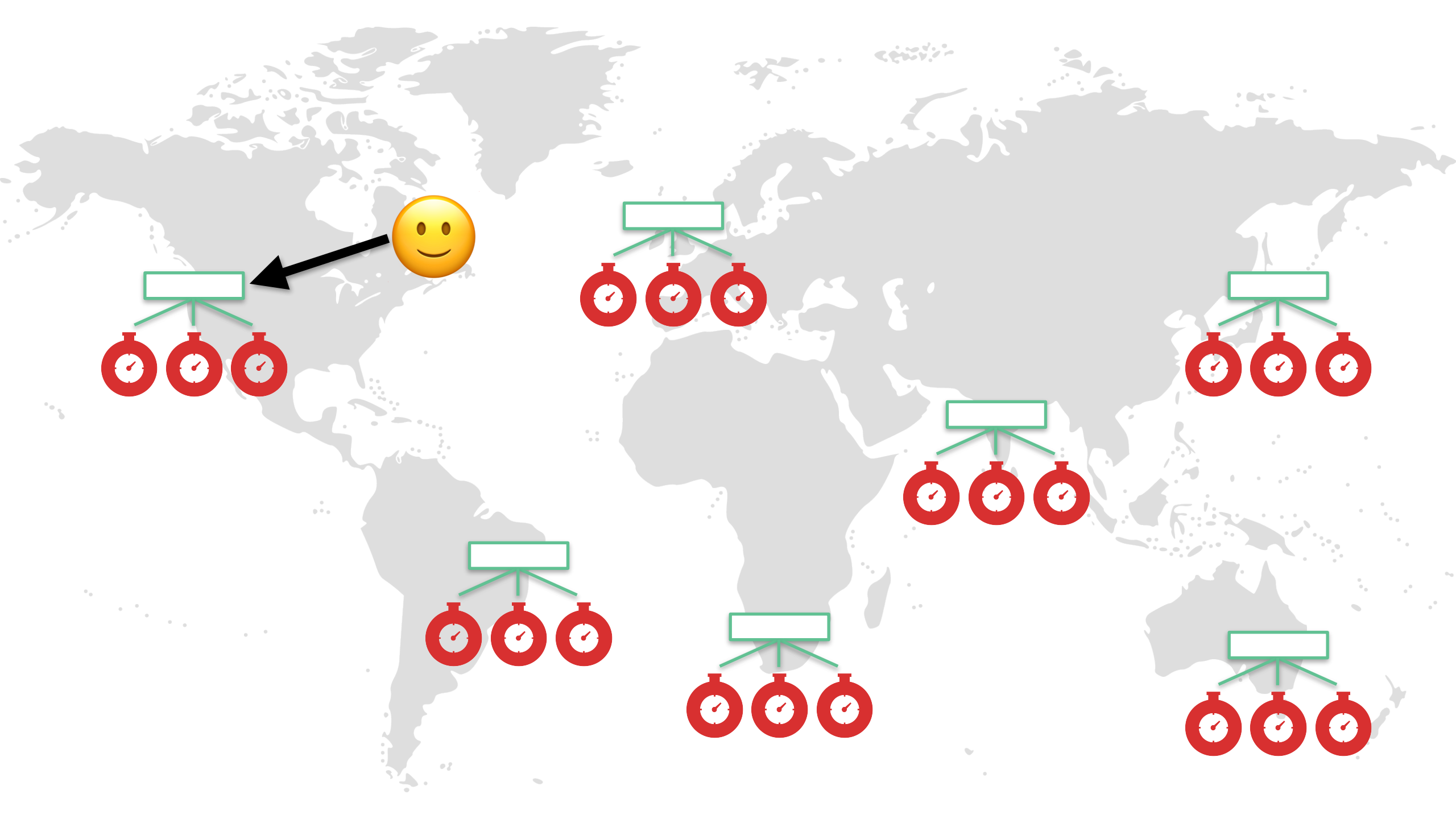
https://meltdownattack.com/

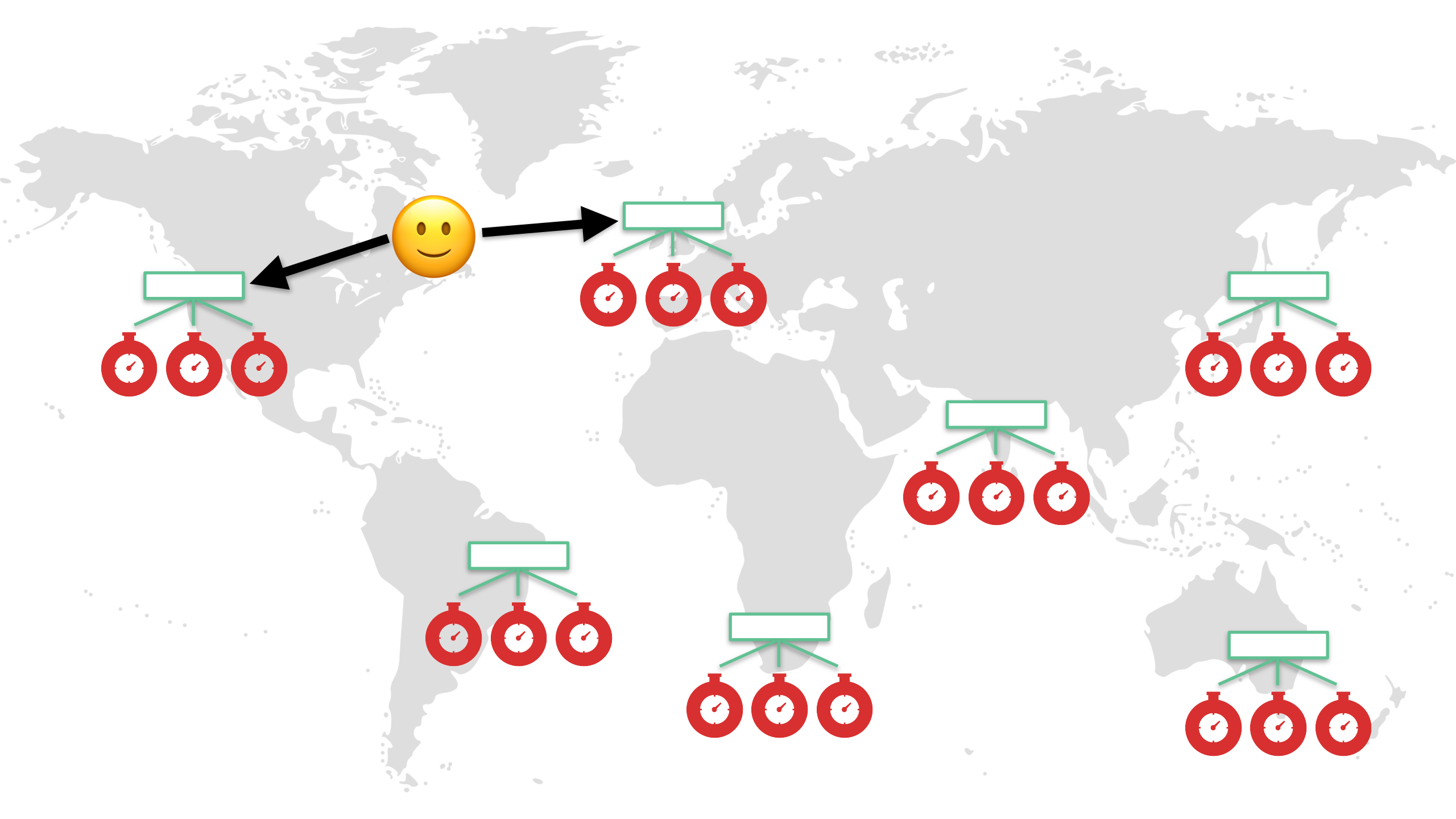
RSAConference2020

Edge compute

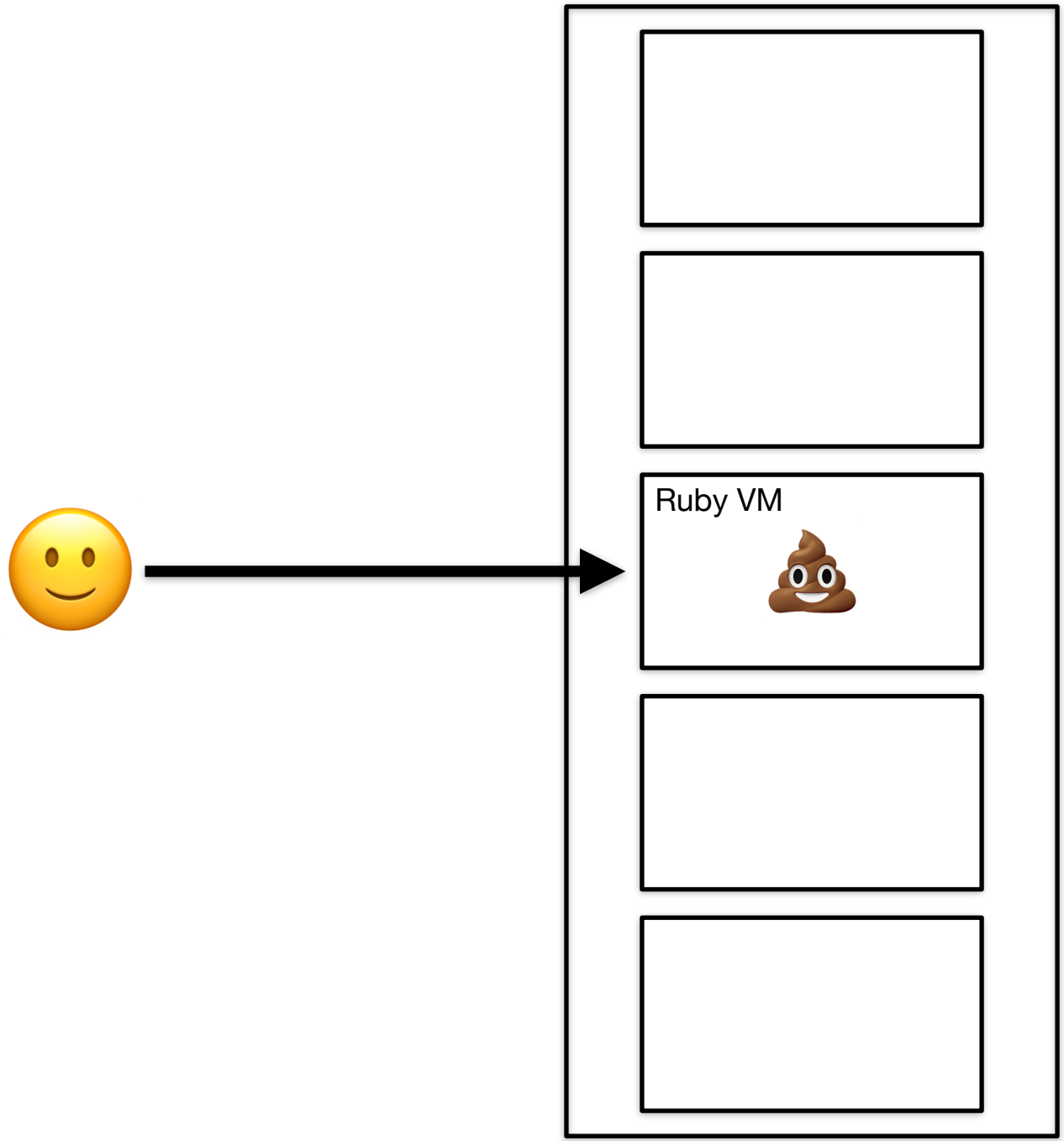


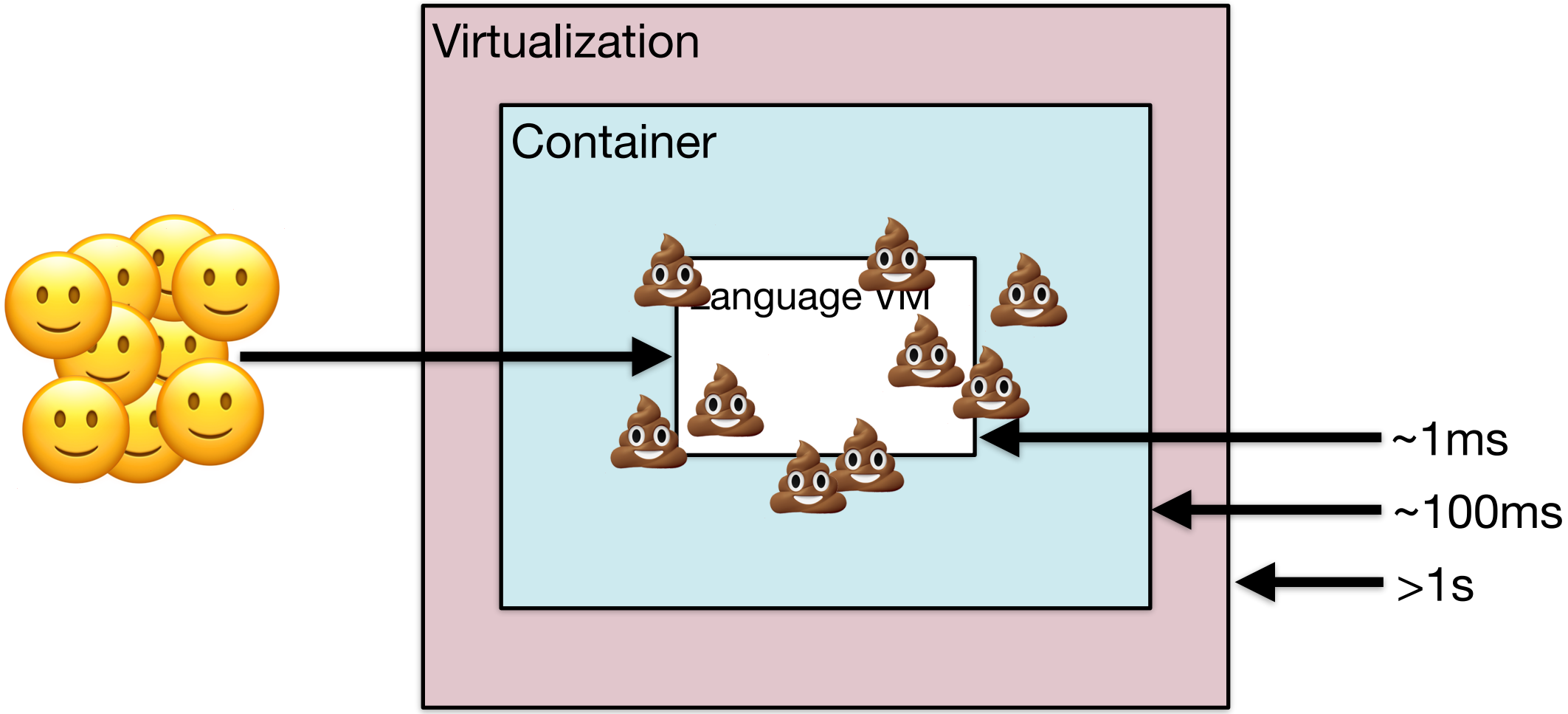


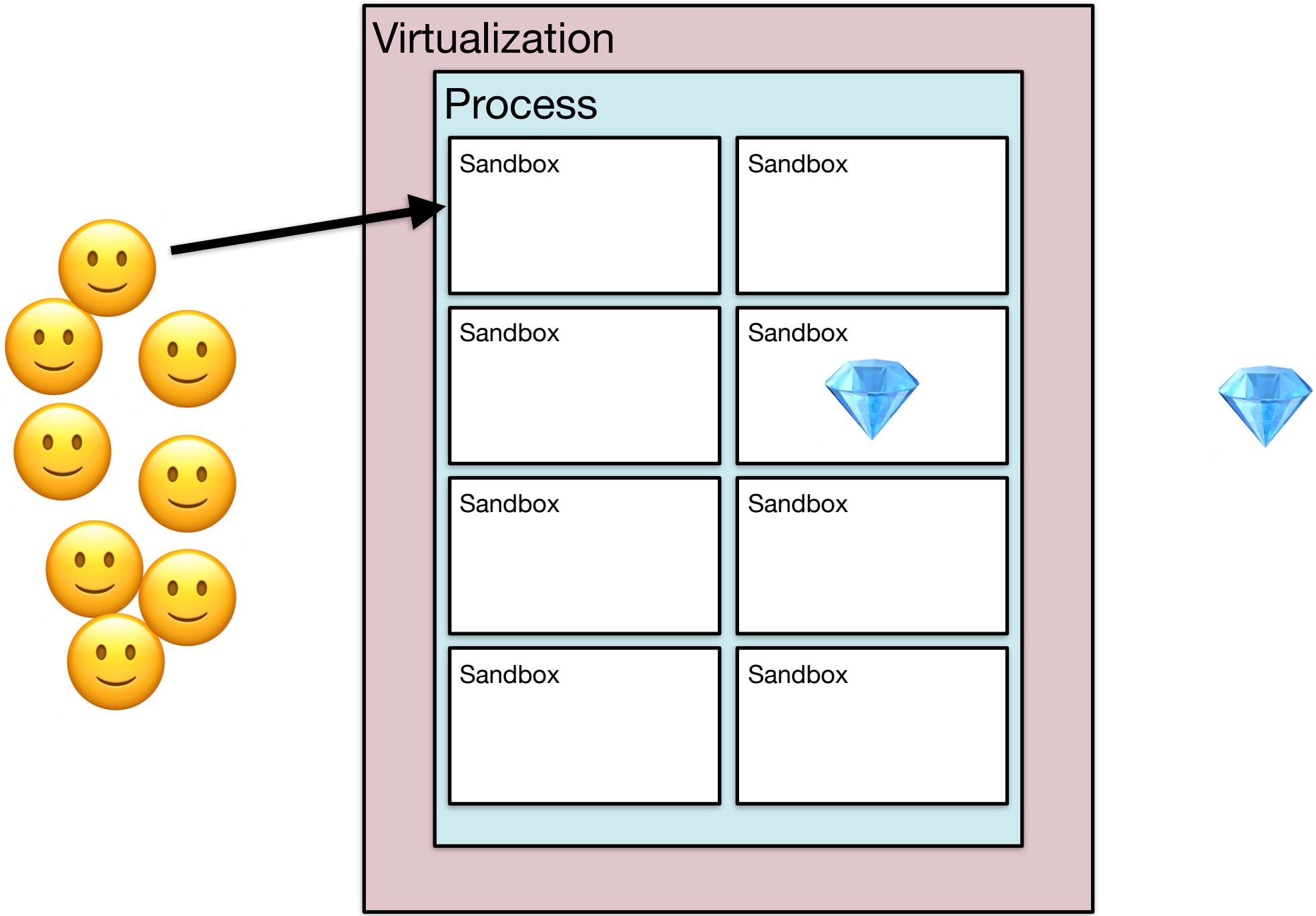




FaaS

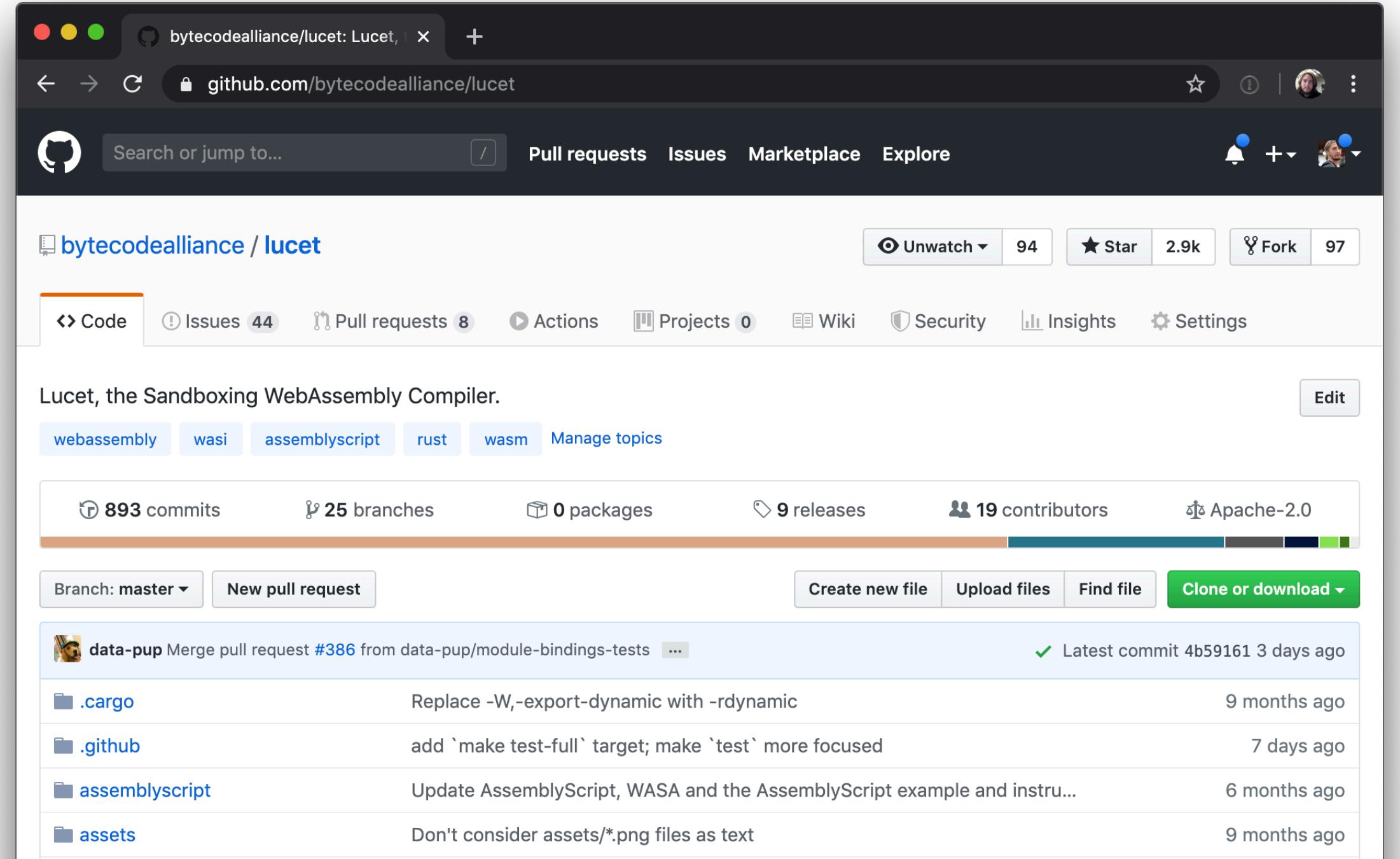

Ruby VM

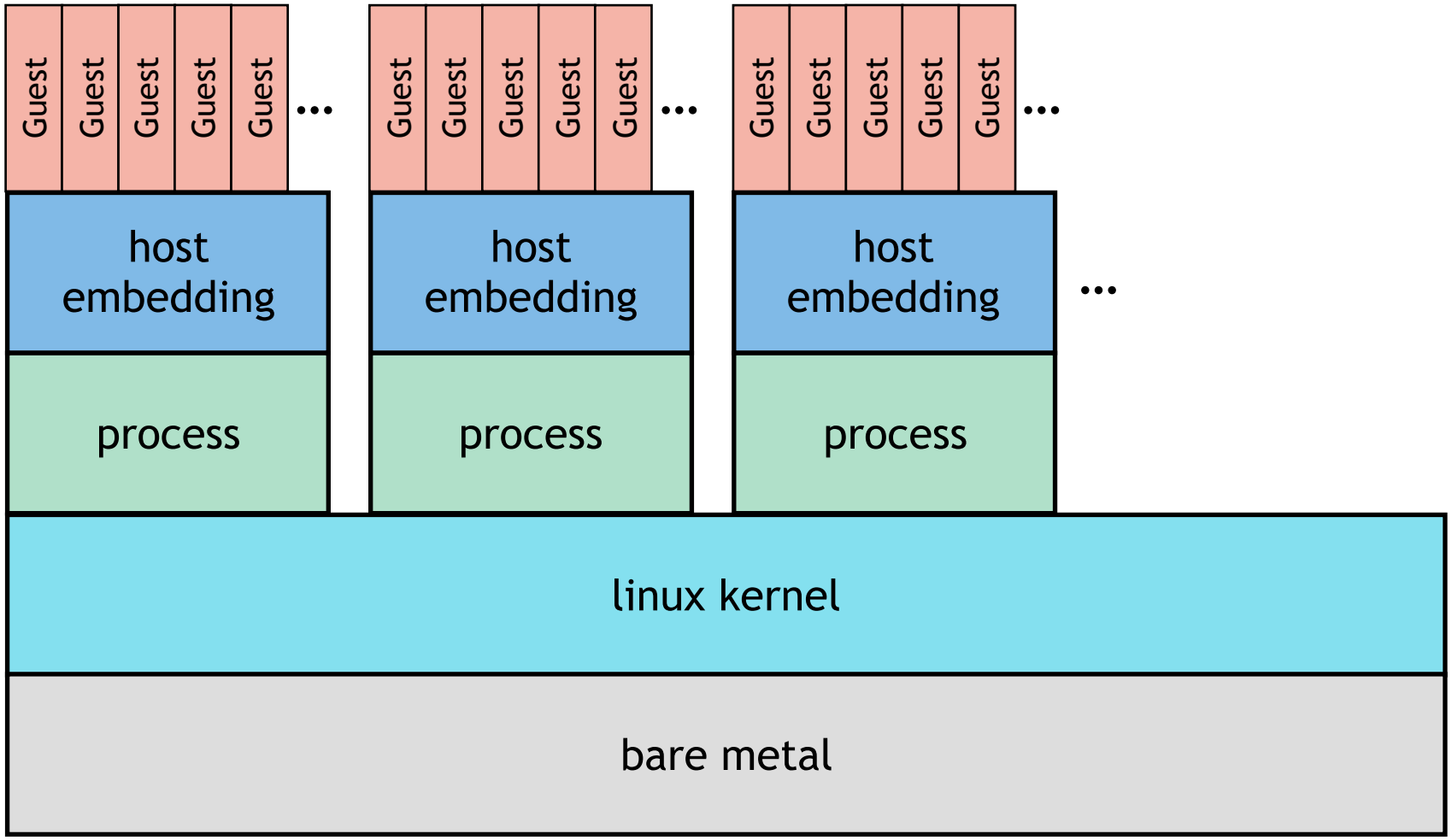

Virtualization
Container
Language VM
~1ms
~100ms
>1s

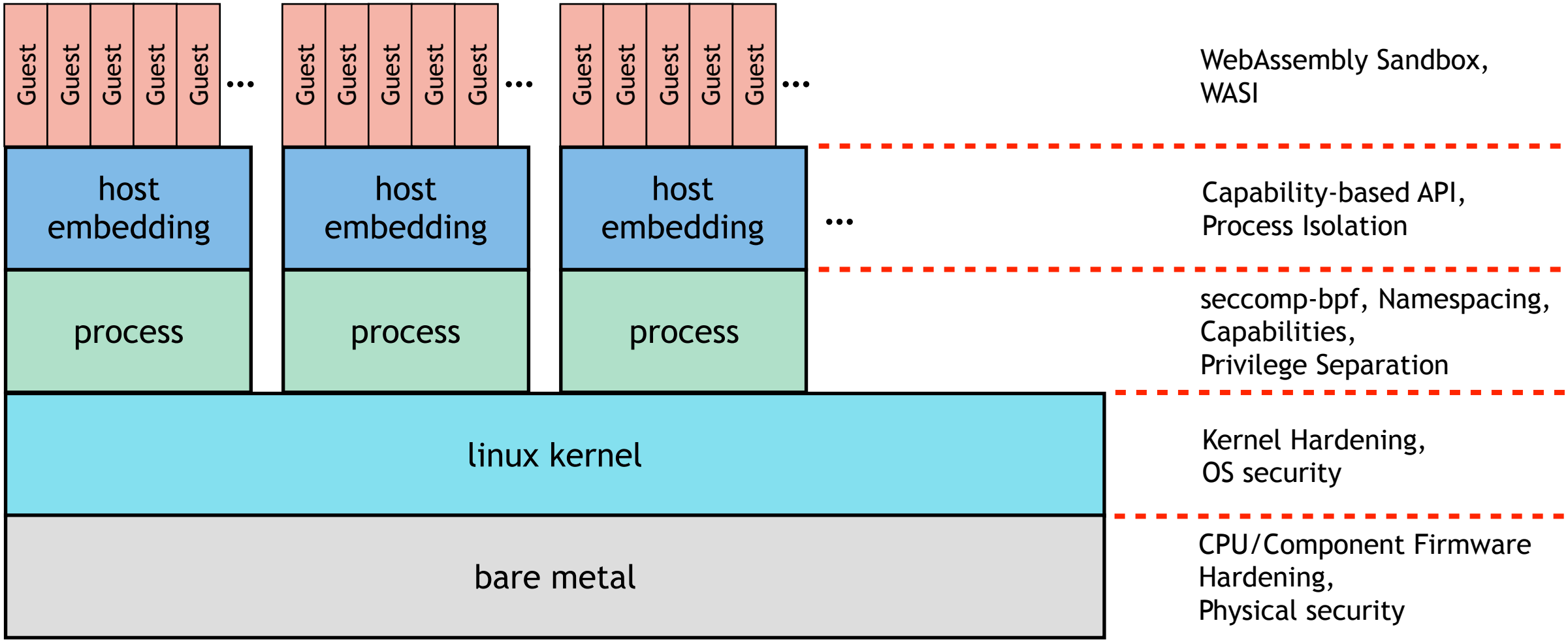

Virtualization
Process
Sandbox
Sandbox
Sandbox
Sandbox
Sandbox
Sandbox
Sandbox
Sandbox

Granular Sandboxing


WA


bytecodealliance/lucet: Lucet,
github.com/bytecodealliance/lucet
Search or jump to...
Pull requests
Issues
Marketplace
Explore
bytecodealliance / lucet
Unwatch 94
Star 2.9k
Fork 97
Code
Issues 44
Pull requests 8
Actions
Projects 0
Wiki
Security
Insights
Settings
Lucet, the Sandboxing WebAssembly Compiler.
Edit
webassembly
wasi
assemblyscript
rust
wasm
Manage topics
893 commits
25 branches
0 packages
9 releases
19 contributors
Apache-2.0
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
data-pup Merge pull request #386 from data-pup/module-bindings-tests
Latest commit 4b59161 3 days ago
.cargo
Replace -W,-export-dynamic with -rdynamic
9 months ago
.github
add `make test-full` target; make `test` more focused
7 days ago
assemblyscript
Update AssemblyScript, WASA and the AssemblyScript example and instru...
6 months ago
assets
Don't consider assets/*.png files as text
9 months ago


Guest
Guest
Guest
Guest
Guest
...
Guest
Guest
Guest
Guest
Guest
...
Guest
Guest
Guest
Guest
Guest
...
host embedding
host embedding
host embedding
...
process
process
process
linux kernel
bare metal


Guest
Guest
Guest
Guest
Guest
...
Guest
Guest
Guest
Guest
Guest
...
Guest
Guest
Guest
Guest
Guest
...
host embedding
host embedding
host embedding
...
process
process
process
linux kernel
bare metal
WebAssembly Sandbox, WASI
Capability-based API, Process Isolation
seccomp-bpf, Namespacing, Capabilities, Privilege Separation
Kernel Hardening, OS security
CPU/Component Firmware Hardening, Physical security

RSAConference2020

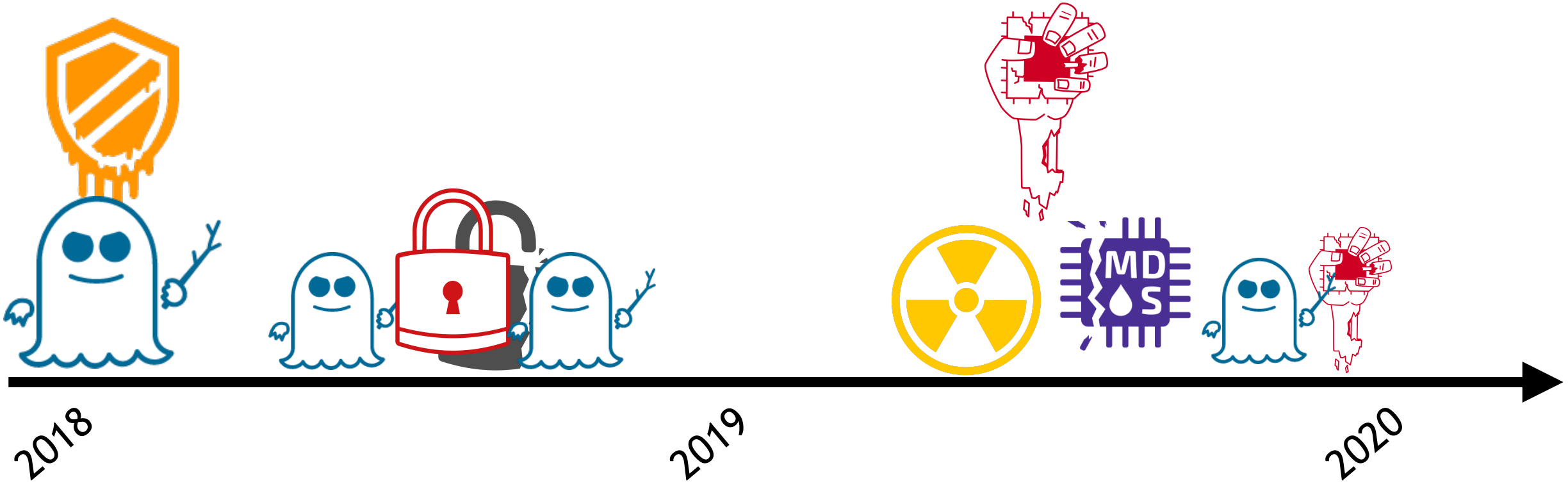
Transient execution security

RSAConference2020

Research and development evolution

Attacks and defenses

https://meltdownattack.com/



https://cpu.fail/

https://meltdownattack.com/

https://foreshadowattack.eu/

Expectations

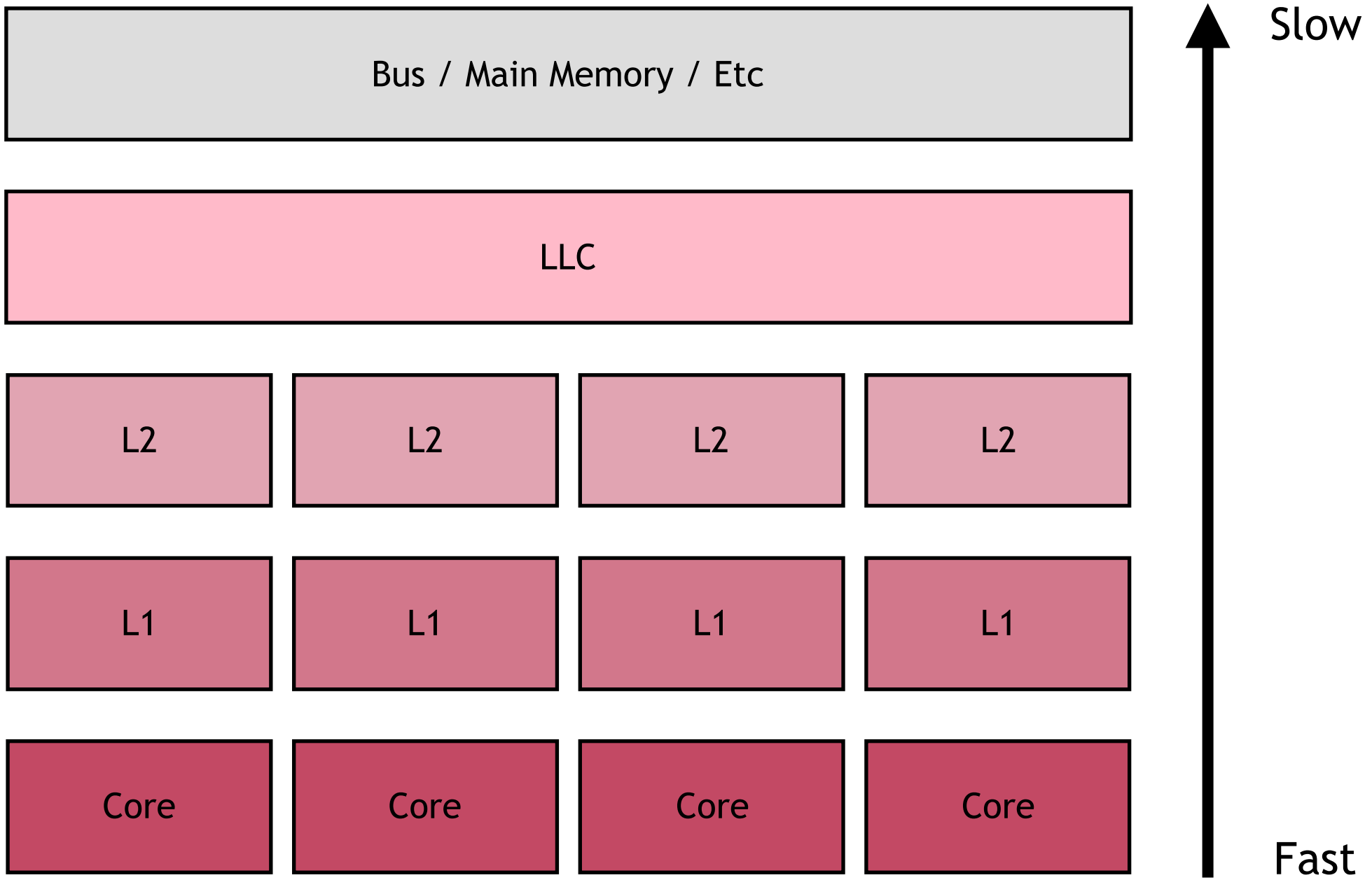
- Advances in offensive security research
  - Unexplored CPU components, new techniques
- Significant improvements in HW-supplied defenses
  - Intel, AMD
- Continued exposure of Fastly network
  - New hardware
  - New execution models
- Malicious exploitation

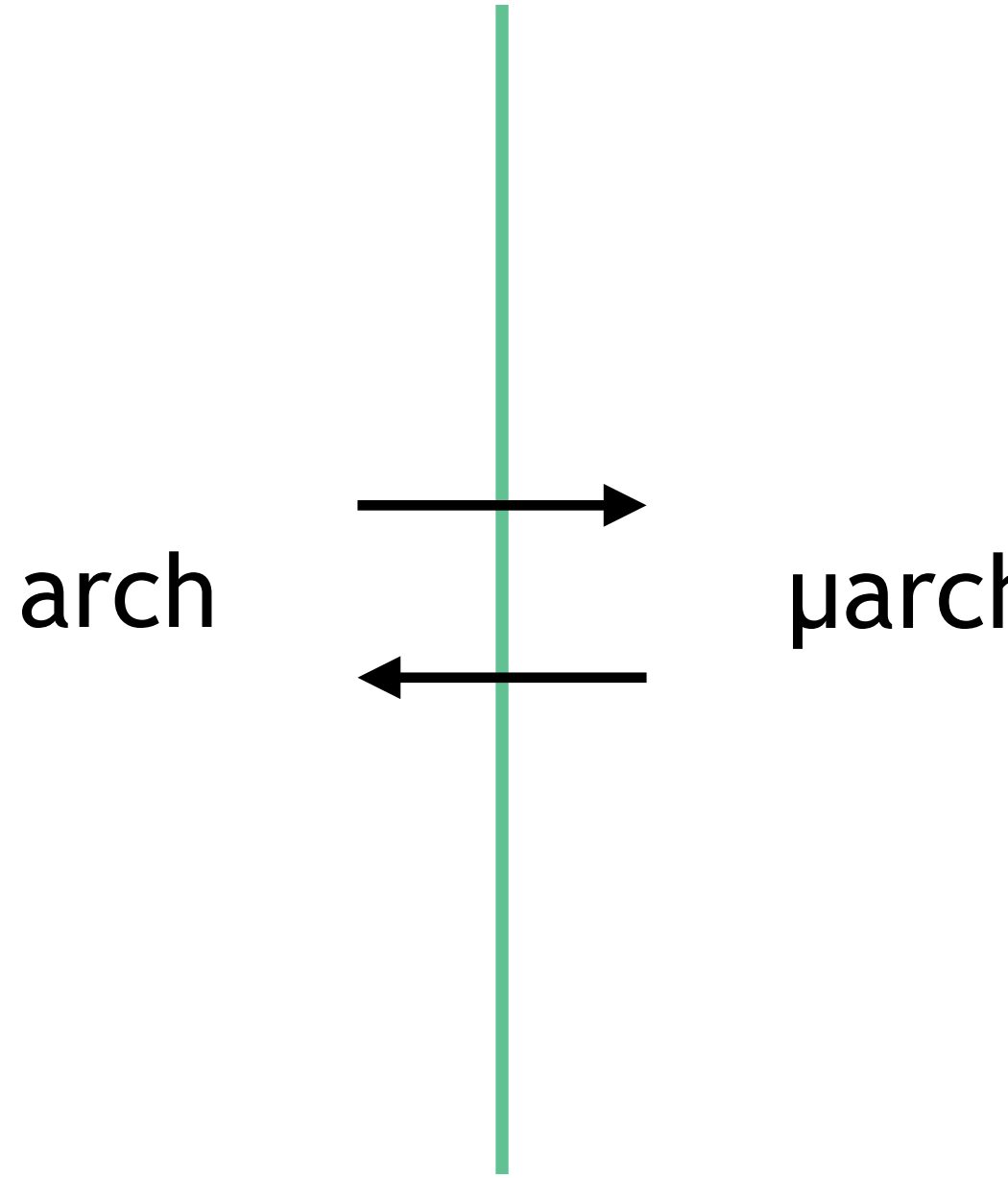
https://www.fastly.com/network-map

RSAConference2020

Transient execution attacks

Concepts


Bus / Main Memory / Etc
LLC
L2
L2
L2
L2
L1
L1
L1
L1
Core
Core
Core
Core
Slow
Fast


arch
μarch

RSAConference2020

Transient execution attacks and defenses

Analysis by example

arch

μarch

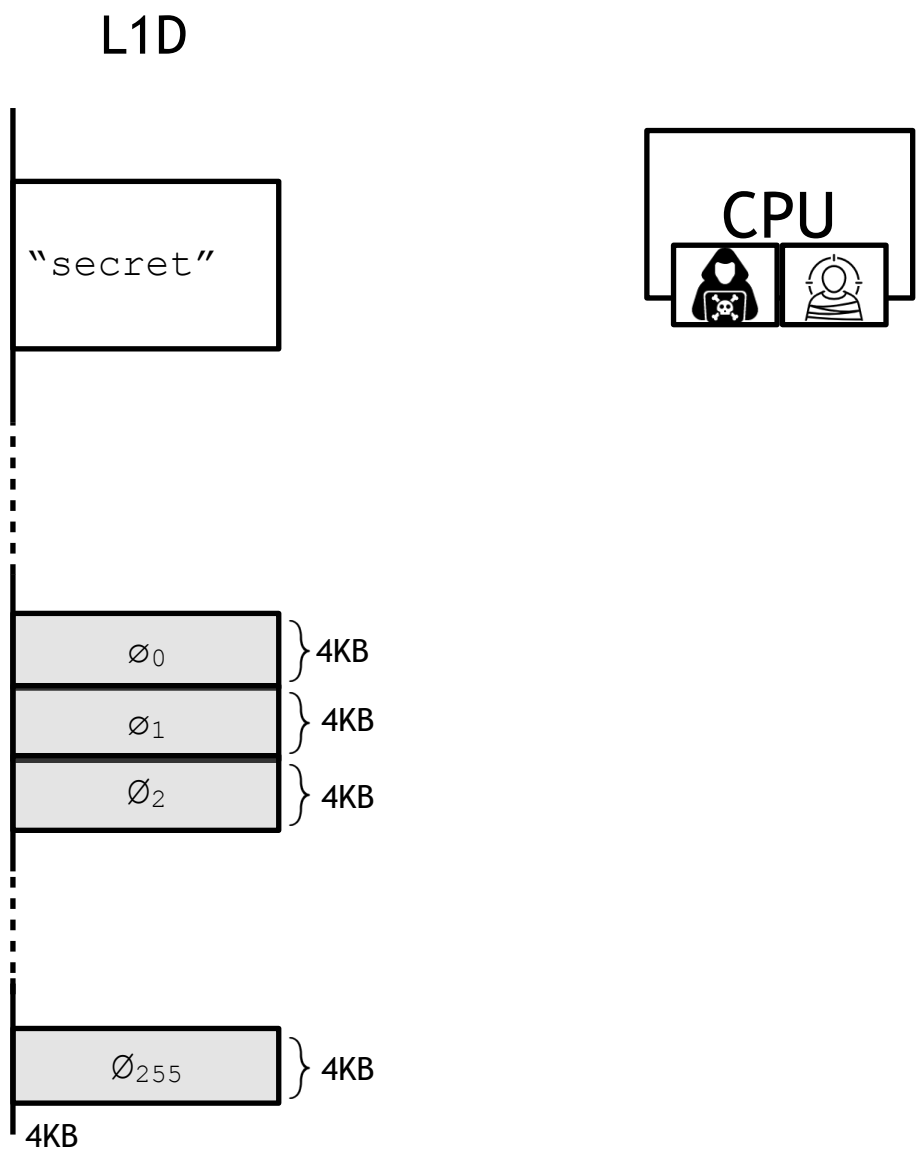
arch

μarch

arch

```
→ secret = "secret"; // or set affinity, etc.
→ oracle = malloc(4096 * 256);



→       for cache_line in oracle {
            clflush(cache_line);
        }
```

µarch

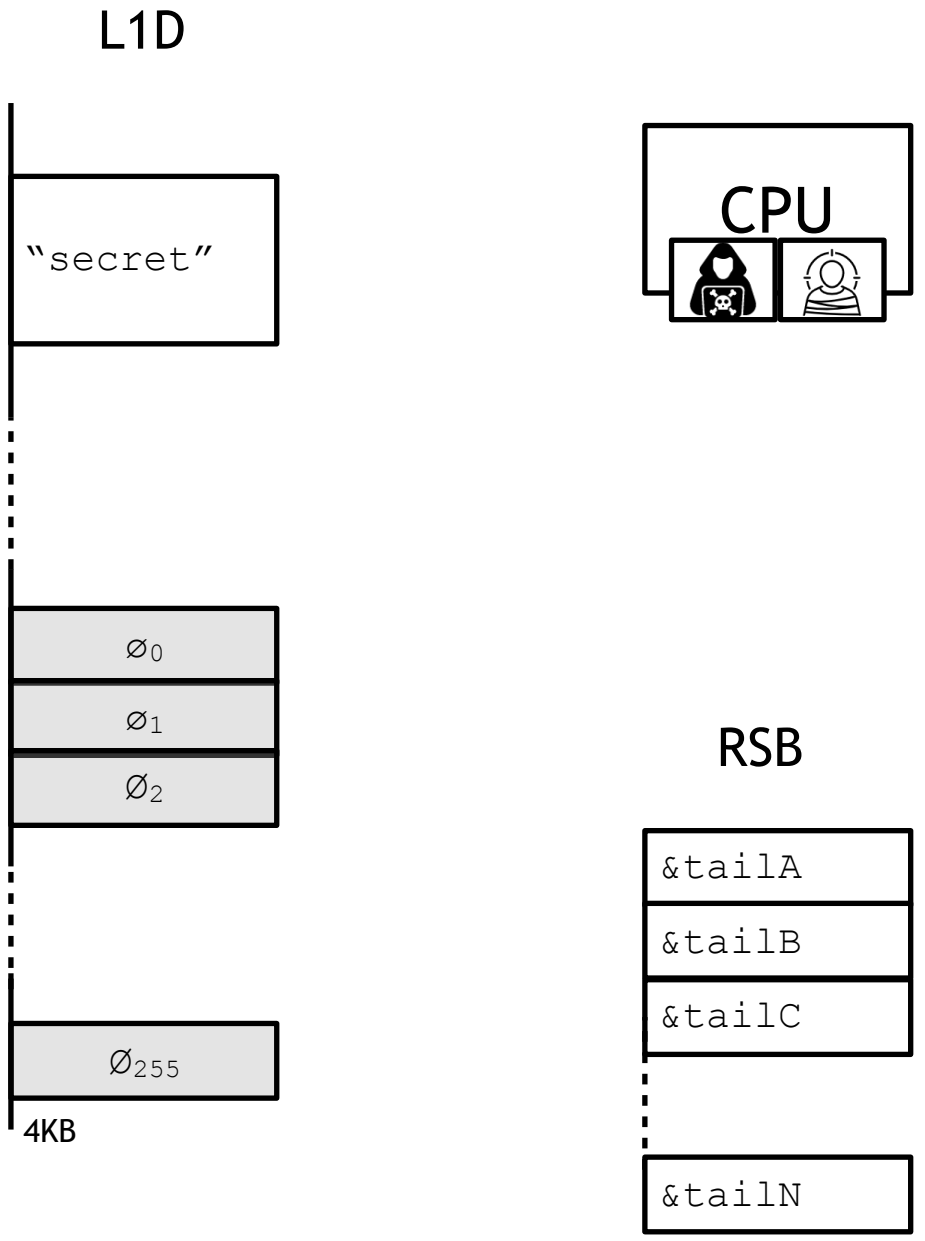


arch

```
secret = "secret"; // or set affinity, etc.
oracle = malloc(4096 * 256);


    for cache_line in oracle {
        clflush(cache_line);
    }
```

```
→ gadget:
    // ...
    ret2spec_gadget(i) {
        // ...

        *(oracle + secret[j]);
    }
```

µarch



Priming the system: defenses

- Disallow clflush and proxies
- Disallow large array allocations
- Process isolation
- Secrets management
- Detect suspicious code patterns
- Detect suspicious L*D and allocation activity
- …

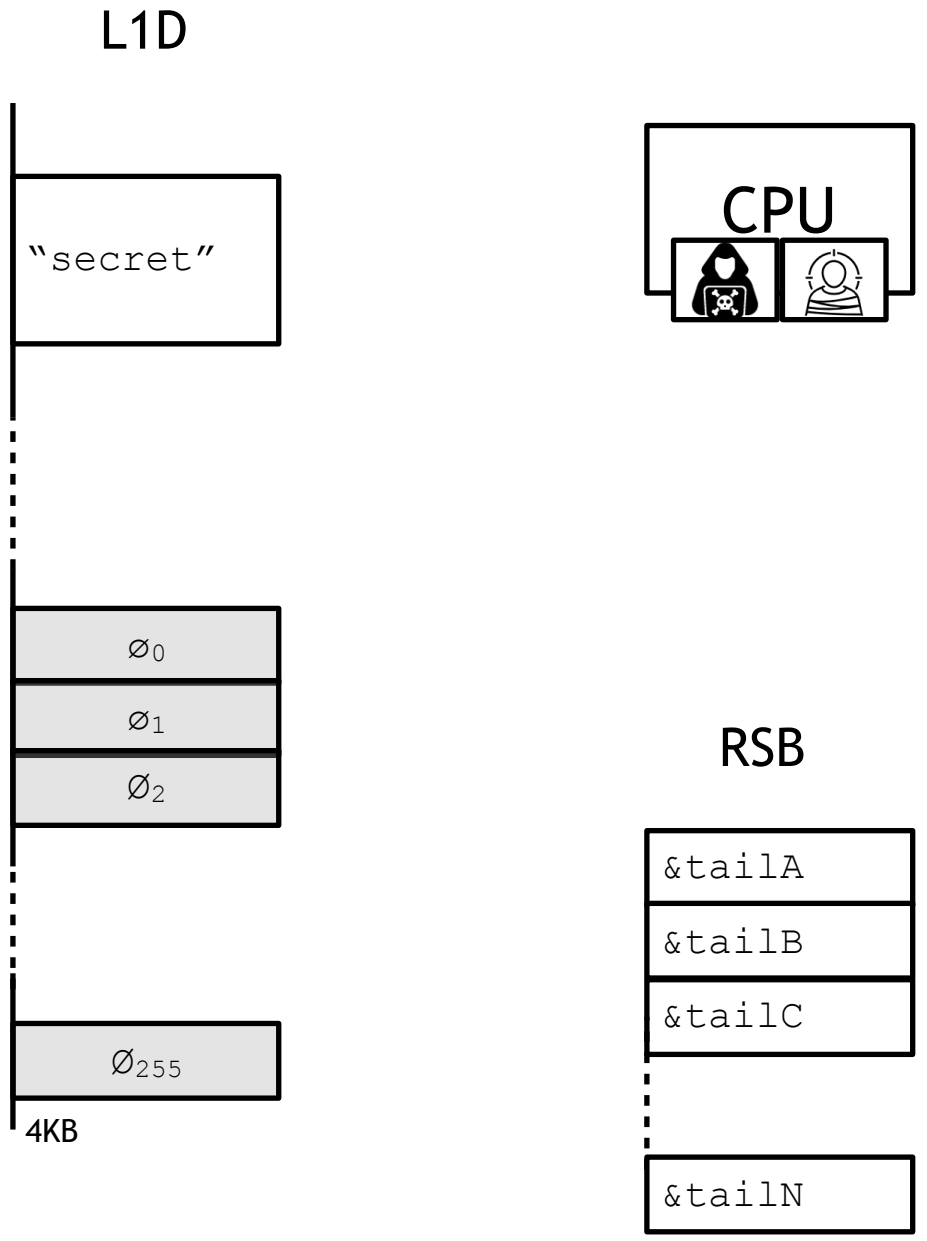
https://www.fastly.com/network-map

arch

```
secret = "secret"; // or set affinity, etc.
oracle = malloc(4096 * 256);


    for cache_line in oracle {
        clflush(cache_line);
    }
```

```
gadget:
    // ...
    ret2spec_gadget(i) {
        // ...

        *(oracle + secret[j]);
    }
```

µarch

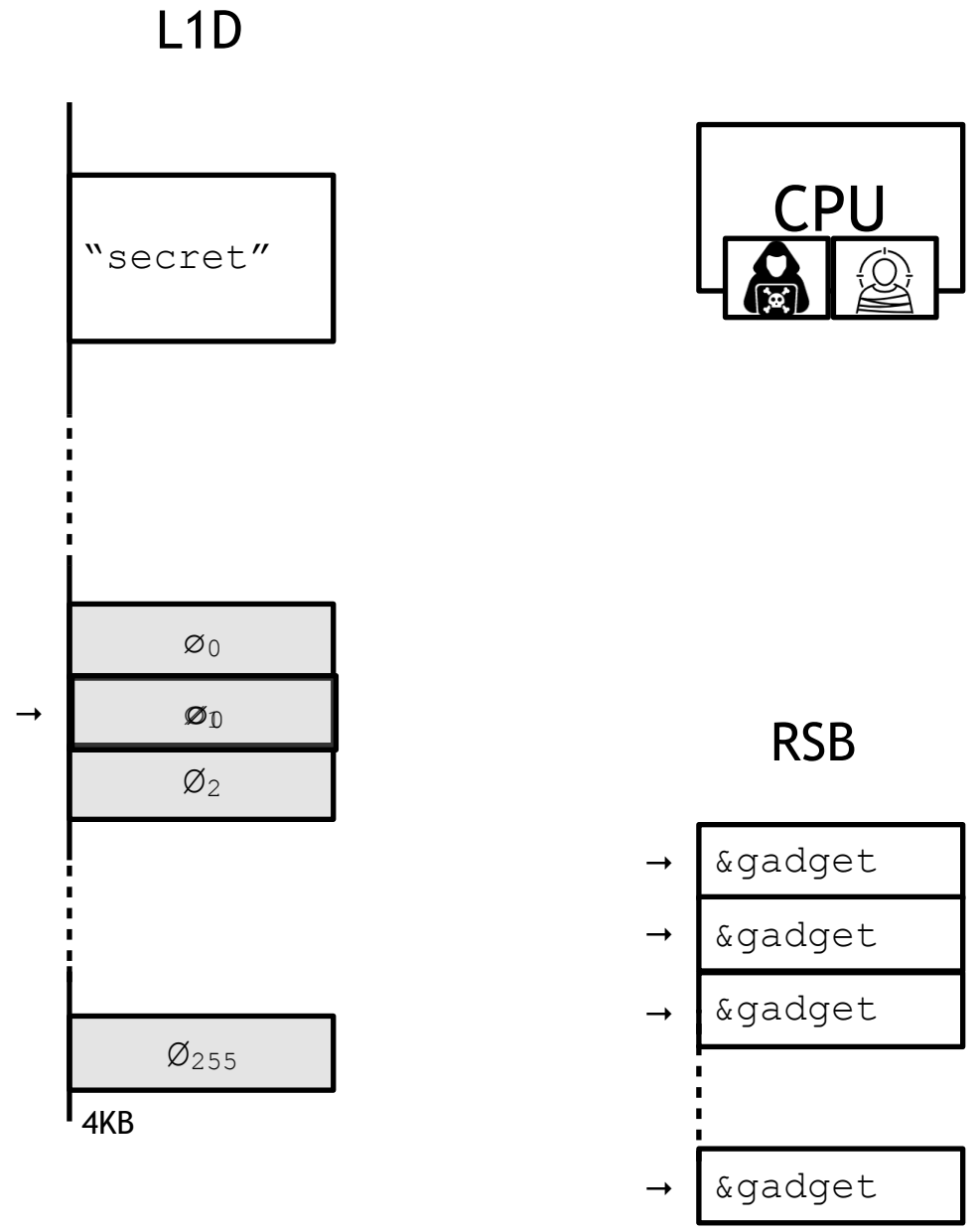


arch

```
secret = "secret"; // or set affinity, etc.
oracle = malloc(4096 * 256);



    for cache_line in oracle {
        clflush(cache_line);
    }

→   ret2spec_recurse();
```

```
gadget:
    // ...
    ret2spec_gadget(i) {
        // ...

        *(oracle + secret[j]);
    }
```

µarch



Problematic transient execution: defenses

- Apply microcode patches
- Limit reach of user-supplied code
- Flush shared resources by security domain
- Co-schedule by security domain
- Detect suspicious code patterns, runtime activity
- …

https://www.fastly.com/network-map

CPU component	Trans exec technique	ECP prevention controls
BPU::BTB	Bounds check bypass / Spectre v1	32-bit addressing for guest programs Array index masking or lfence/serialization for hostcalls *Process isolation by security domain*
	Branch target injection / Spectre v 2	Retpoline for indirect calls within Wasm guests Retpoline for host or IBPB on hostcall invocation *IBPB on hostcall return* *IBRS on guest context switch* Enable STIBP globally *Process isolation by security domain*
…	…	…
Memory disambiguator	Speculative Store Bypass (SSB) / Spectre v 4	32-bit addressing for guest programs Array index masking or lfence/serialization for hostcalls *SSBD*
Line Fill Buffer (LFB), Load port, Store buffer	Microarchitectual data sampling (MDS) / Zombieload (variants 1-5) / RIDL	Co-schedule physical hardware by security domain *Process isolation by security domain* VERW (or equivalent) on host+guest context switches *Disable TSX* *Enable KPTI* *Enable SMAP* *Disable SGX*
…	…	…

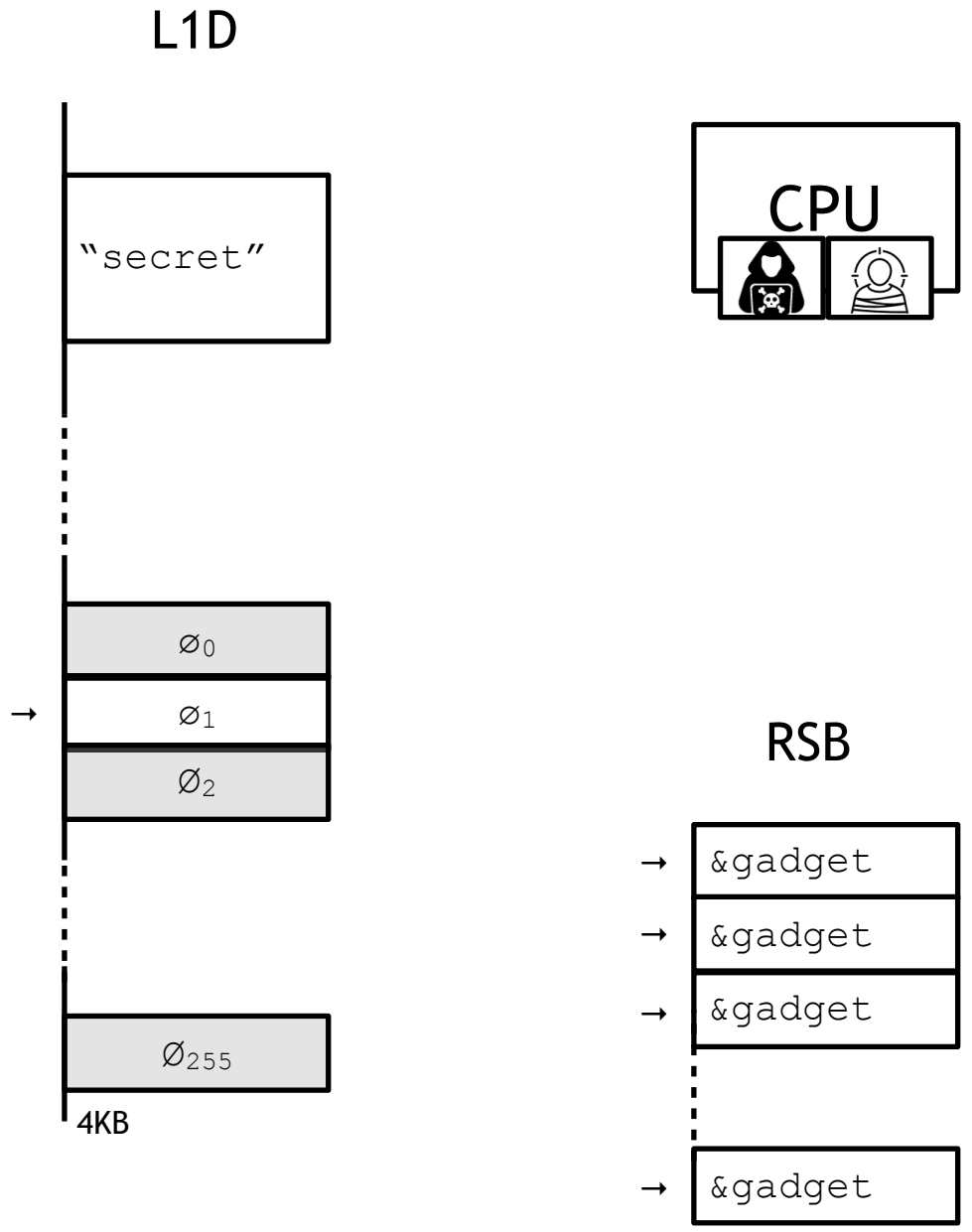
arch

```
secret = "secret"; // or set affinity, etc.
oracle = malloc(4096 * 256);
```

```
    for cache_line in oracle {
        clflush(cache_line);
    }

    ret2spec_recurse();
```

```
gadget:
    // ...
    ret2spec_gadget(i) {
        // ...

        *(oracle + secret[j]);
    }
```

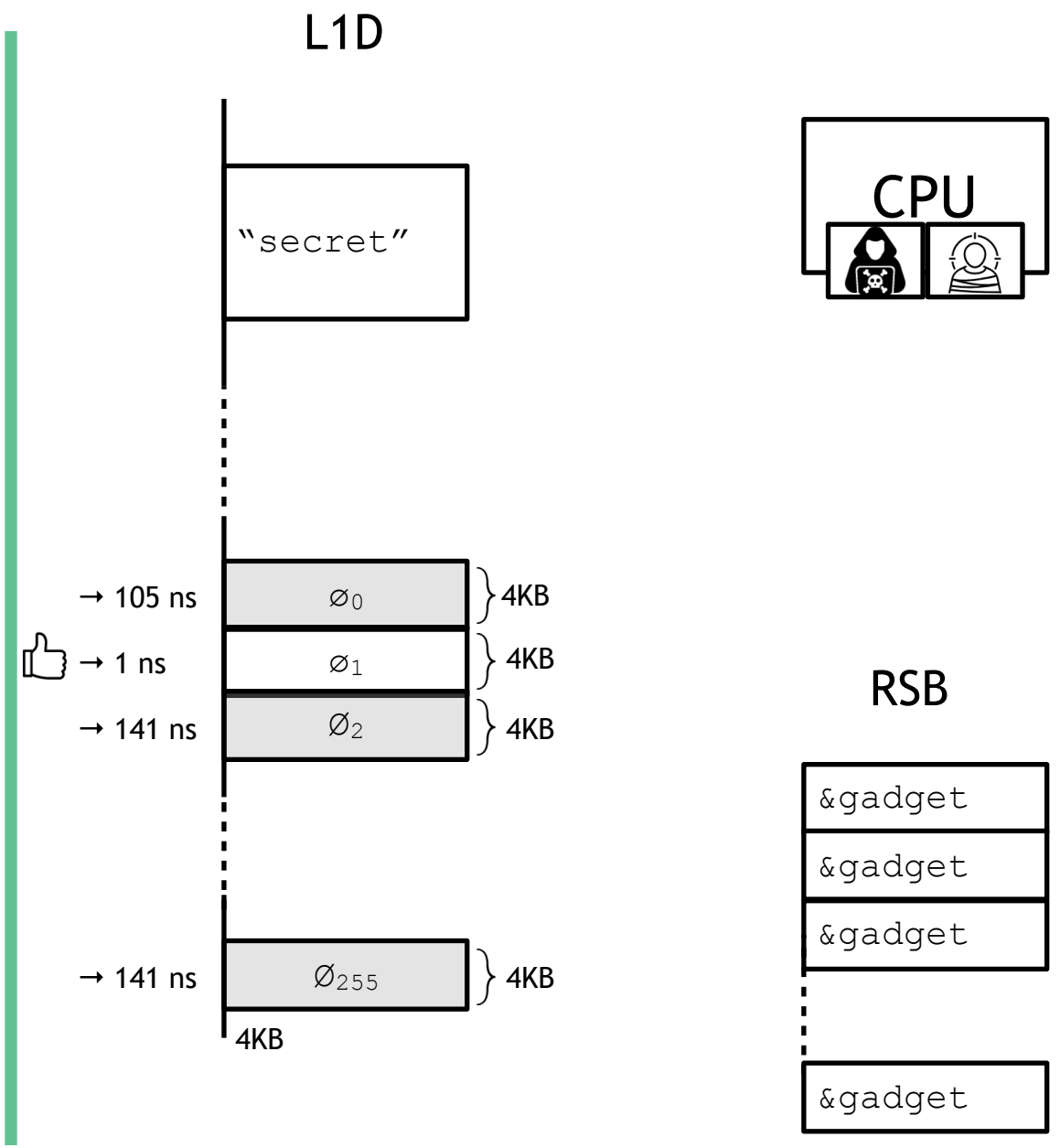
µarch



arch

```
secret = "secret"; // or set affinity, etc.
oracle = malloc(4096 * 256);



    for cache_line in oracle {
        clflush(cache_line);
    }

    ret2spec_recurse();

→   for block in oracle {
        mfence; lfence; rdtsc;
        // ...
        update_scores();
    }



gadget:
    // ...
    ret2spec_gadget(i) {
        // ...

        *(oracle + secret[j]);
    }
```

µarch



Measuring side effects: defenses

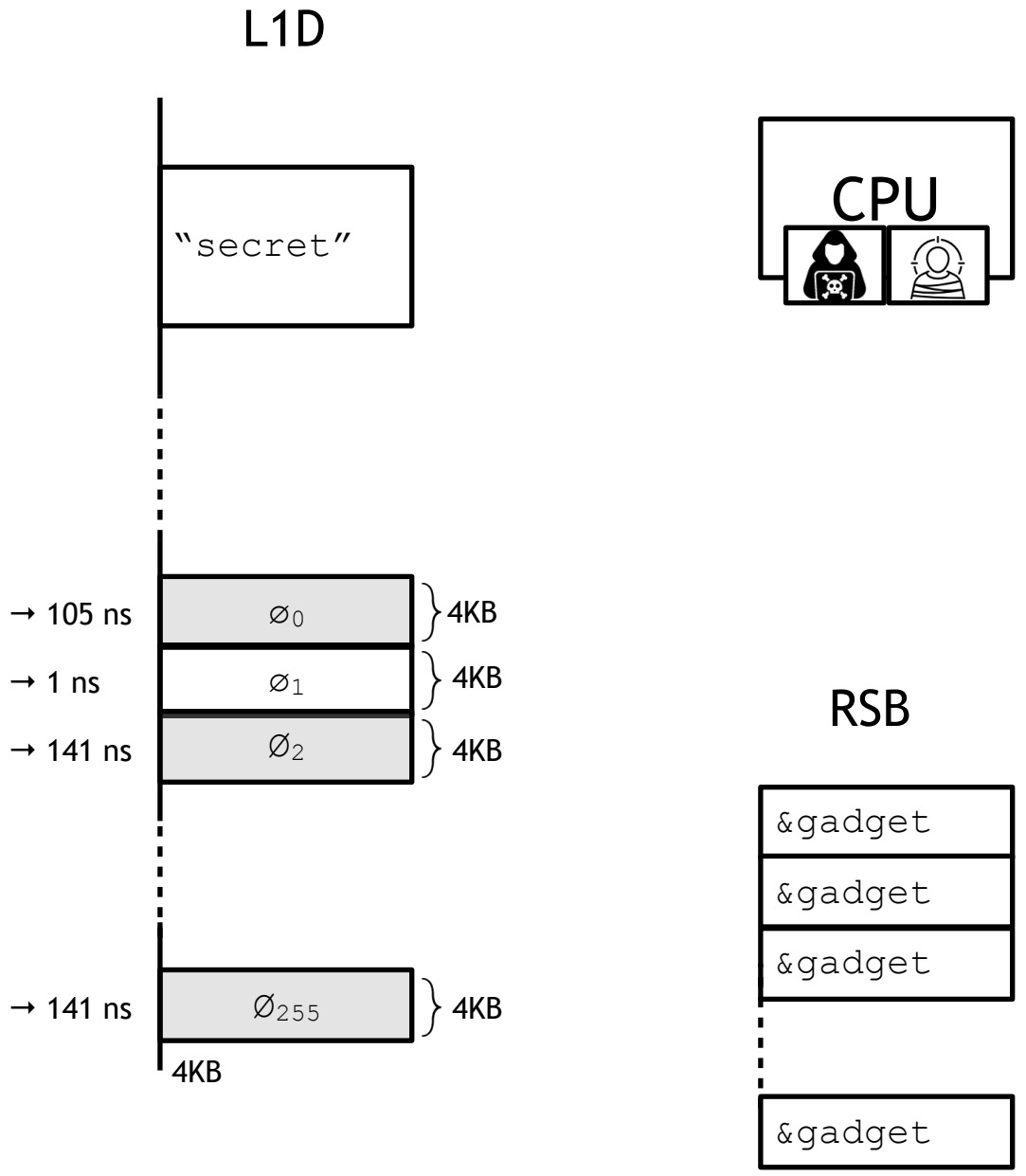
- Disallow asm
- Disallow fine-grained timers/primitives; add jitter
- Deterministic ordering of all events
- Array preloading, non-deterministic array access, buffer ASLR, etc.
- …

https://www.fastly.com/network-map

arch

```
secret = "secret"; // or set affinity, etc.
oracle = malloc(4096 * 256);



    for cache_line in oracle {
        clflush(cache_line);
    }

    ret2spec_recurse();

    for block in oracle {
        mfence; lfence; rdtsc;
        // ...
        update_scores();
    }



gadget:
    // ...
    ret2spec_gadget(i) {
        // ...

        *(oracle + secret[j]);
    }
```

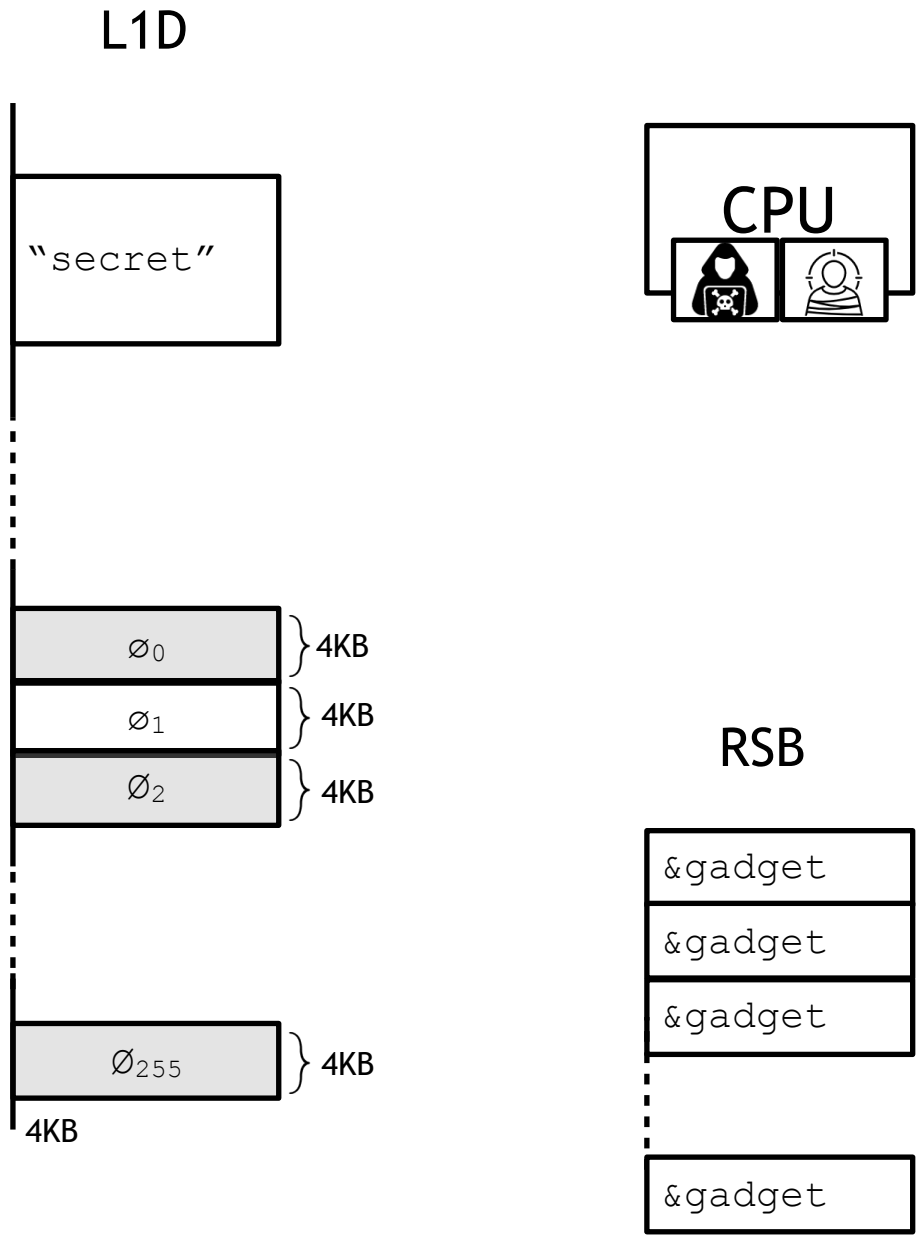
µarch



arch

```
secret = "secret"; // or set affinity, etc.
oracle = malloc(4096 * 256);

→ for k in (len(secret) * LRG_NUMBER) {

    for cache_line in oracle {
        clflush(cache_line);
    }

    ret2spec_recurse();

    for block in oracle {
        mfence; lfence; rdtsc;
        // ...
        update_scores();
    }
}


gadget:
    // ...
    ret2spec_gadget(i) {
        // ...

        *(oracle + secret[j]);
    }
```

µarch



Orchestrating the attack: defenses

- Disallow longer, continuous runtimes
- Isolate workloads by security domain
  - “Quarantine” low-trust workloads
- Detect suspicious code patterns, runtime activity
- …

https://www.fastly.com/network-map

RSAConference2020

Transient execution attacks and defenses

Approach for Fastly Compute@Edge

flush+reload

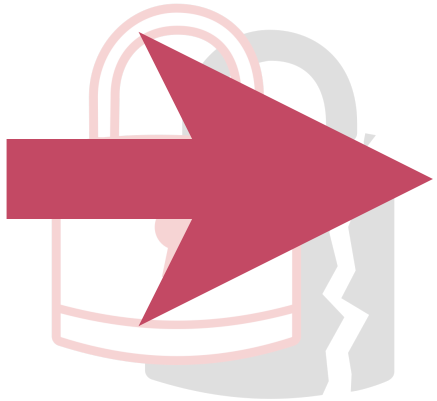
evict+time

prime+probe

flush+flush

prime+abort

...

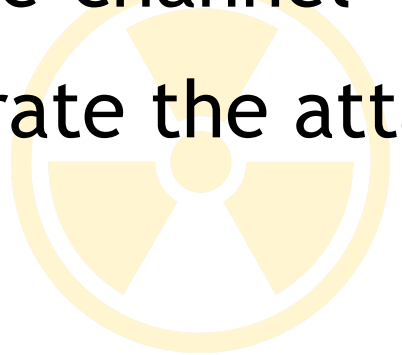


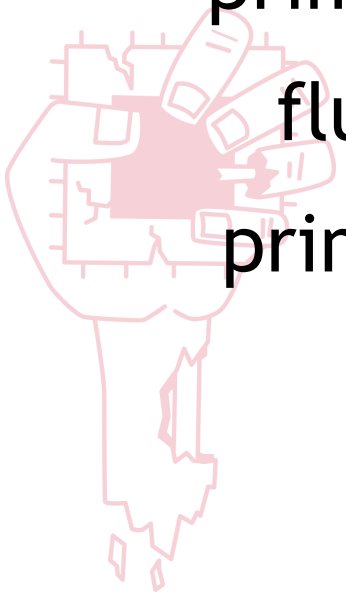
Prime the system

Invoke problematic transient execution

Read side-channel

Orchestrate the attack





https://cpu.fail/

https://meltdownattack.com/

https://foreshadowattack.eu/

Prime the system

Disallow direct, granular flushing of cache lines
Process isolation by security domain
…

Invoke problematic transient execution

Apply microcode patches
Limit reach of user-supplied code
…

Read side-channel

Apply coarse-grained, jittered timers
Non-deterministic array access
…

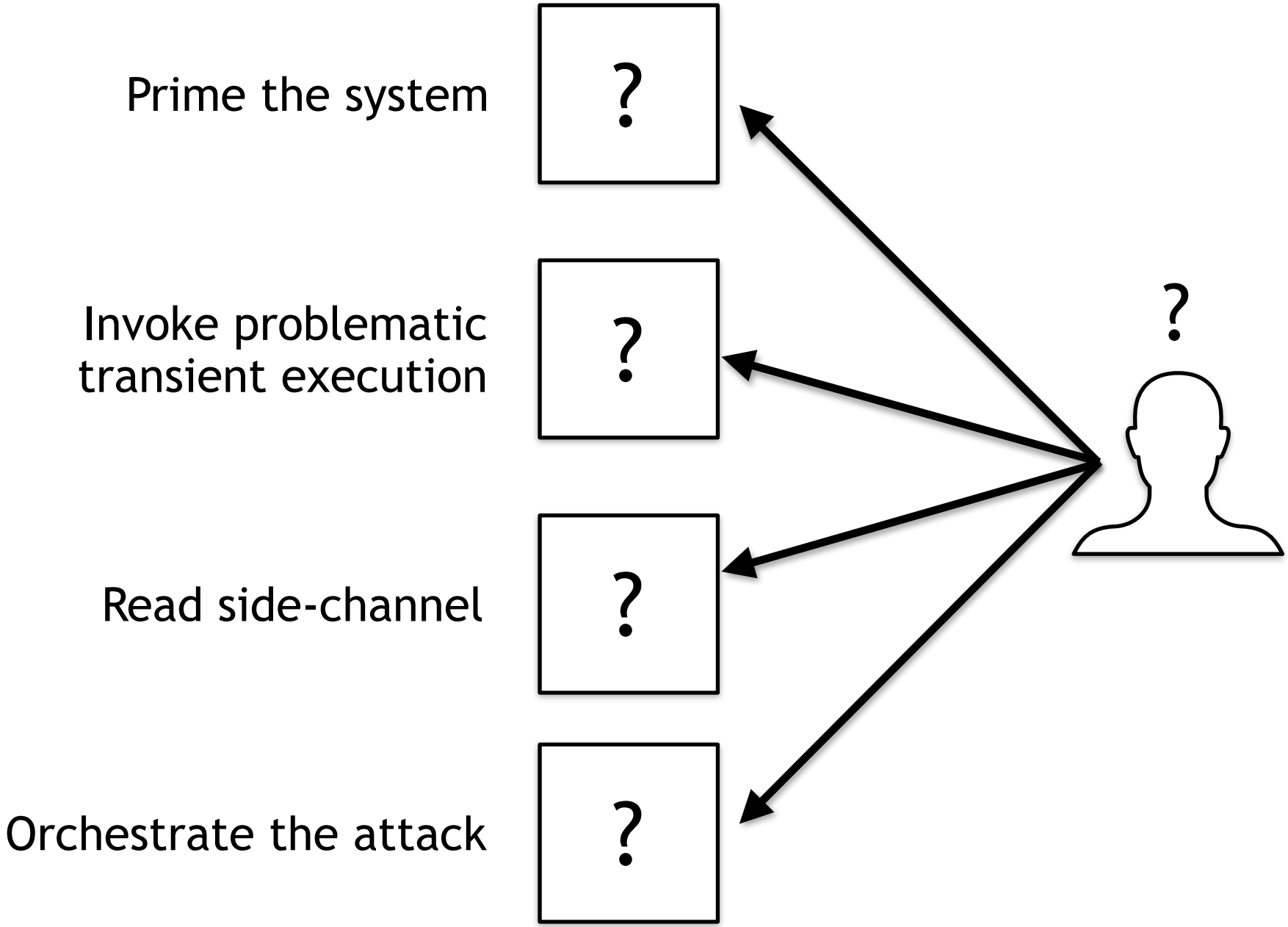
Orchestrate the attack

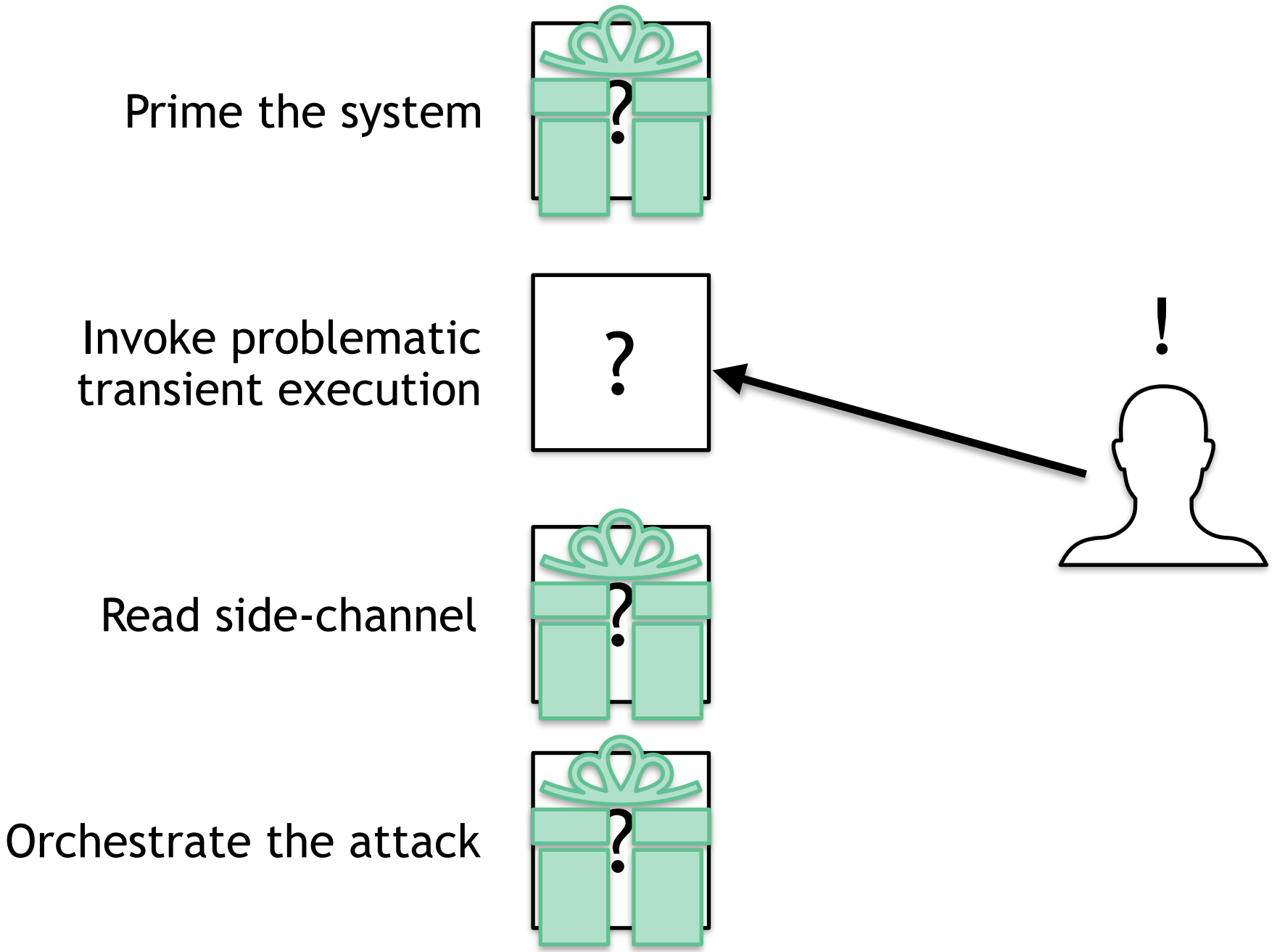
Time-limit, safely schedule workloads
Detect and respond to suspicious workloads
…

https://cpu.fail/

https://meltdownattack.com/

https://foreshadowattack.eu/


Prime the system
?
Invoke problematic transient execution
?
Read side-channel
?
Orchestrate the attack
?
?


Prime the system
?
Invoke problematic
transient execution
?
!
Read side-channel
?
Orchestrate the attack
?

Expectations (review)

- Advances in offensive security research
- Continued exposure of Fastly network
- Significant improvements in HW-supplied defenses
- Malicious exploitation

https://www.fastly.com/network-map

Coverage of approach

- Malicious exploitation
  - → Known attacks are evaluated and prevented
- Significant improvements in HW-supplied defenses
  - → Fit into defined categories of defense
- Continued exposure of Fastly network
  - → Framework can be applied across CPU architectures and execution models
- Advances in offensive security research
  - → Mitigate risk of practical attacks while defenses are rolled out

https://www.fastly.com/network-map

RSAConference2020

Conclusion

Defense is best approached holistically

https://www.fastly.com/network-map

Validate with adversarial analysis

https://www.fastly.com/network-map

Transient execution attacks are here to stay

https://www.fastly.com/network-map

RSAConference2020

Thanks