OPEN SOURCE FIRMWARE CONFERENCE

Starting Soon:

Provable Security in Embedded Systems: Verification Work in Tock OS

By Samir Rashid

an integer overflow caused $370M to vaporize in 40 seconds

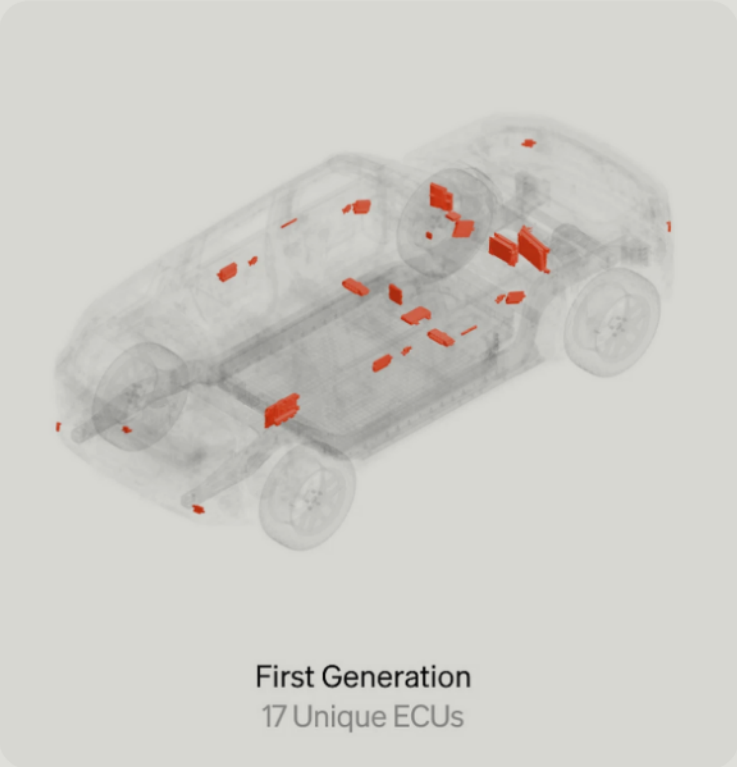
Thankfully Boeing has learned from this incident and provably avoids overflow

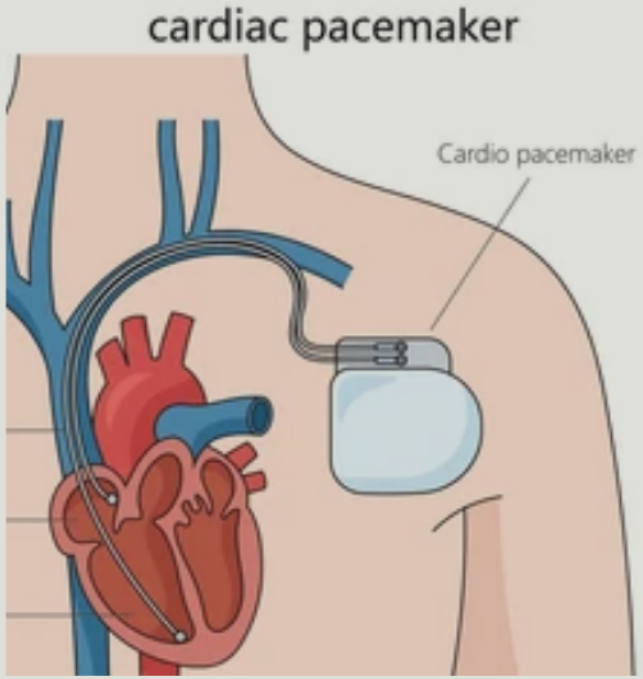
by restarting your plane...

The Register®

Boeing 787s must be turned off and on every 51 days to prevent 'misleading data' being shown to pilots

US air safety bods call it 'potentially catastrophic' if reboot directive not implemented


First Generation
17 Unique ECUs


cardiac pacemaker
Cardio pacemaker

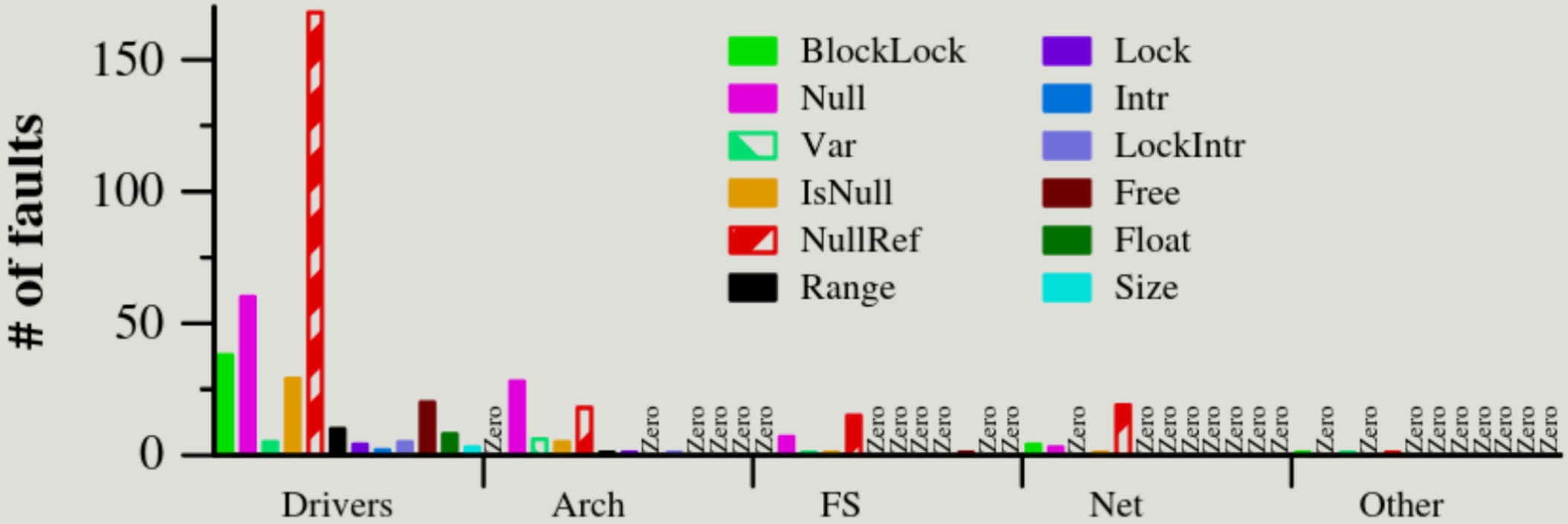
NEURALINK

Compile time techniques for safer firmware

Samir Rashid, Evan Johnson, Nico Lehmann, Ranjit Jhala

Open Source Firmware Conference, September 3, 2024

Presentation permalink:
godsped.com/safe-firmware



(a) Number of faults broken down per directory and category

Classifying bugs

- **Memory management**: buffer overflows, null dereference
- **Undefined behavior**

- **Faulty drivers**
- **Concurrency/Locking**

- **Logic bugs**: *semantic* errors
- **Malicious attacks**: `xz` poisoned contributions

1. Compartmentalizing firmware with Rust

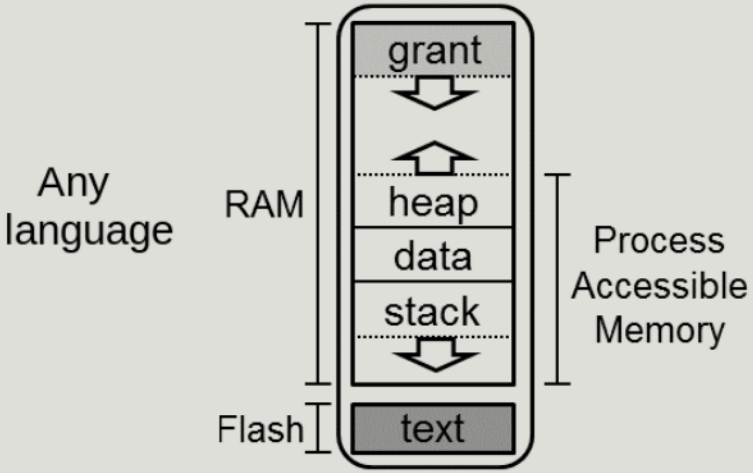
2. Lightweight verification for firmware

Tock OS

- embedded operating system
- designed for safe, robust multitenancy on microcontrollers
- used at UC San Diego, Princeton University, Stanford University
- used in industry by Google, Microsoft, HP, Western Digital
- OpenSK, opentitan

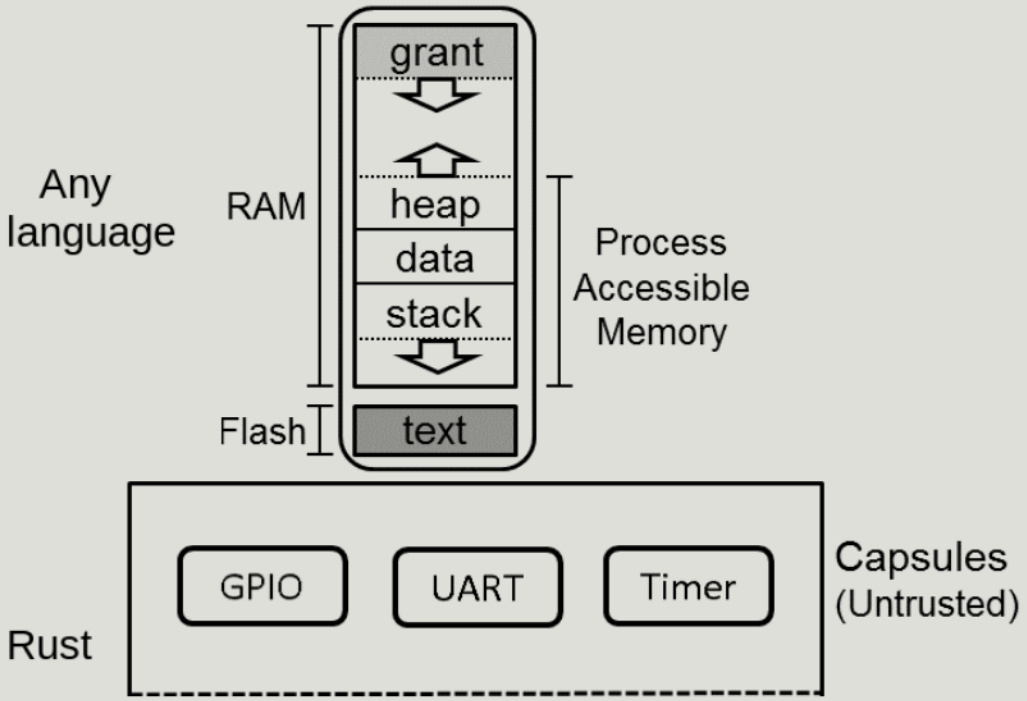
Processes: hardware protection

- isolated by Memory Protection Unit
- written in any language
- loadable and restartable



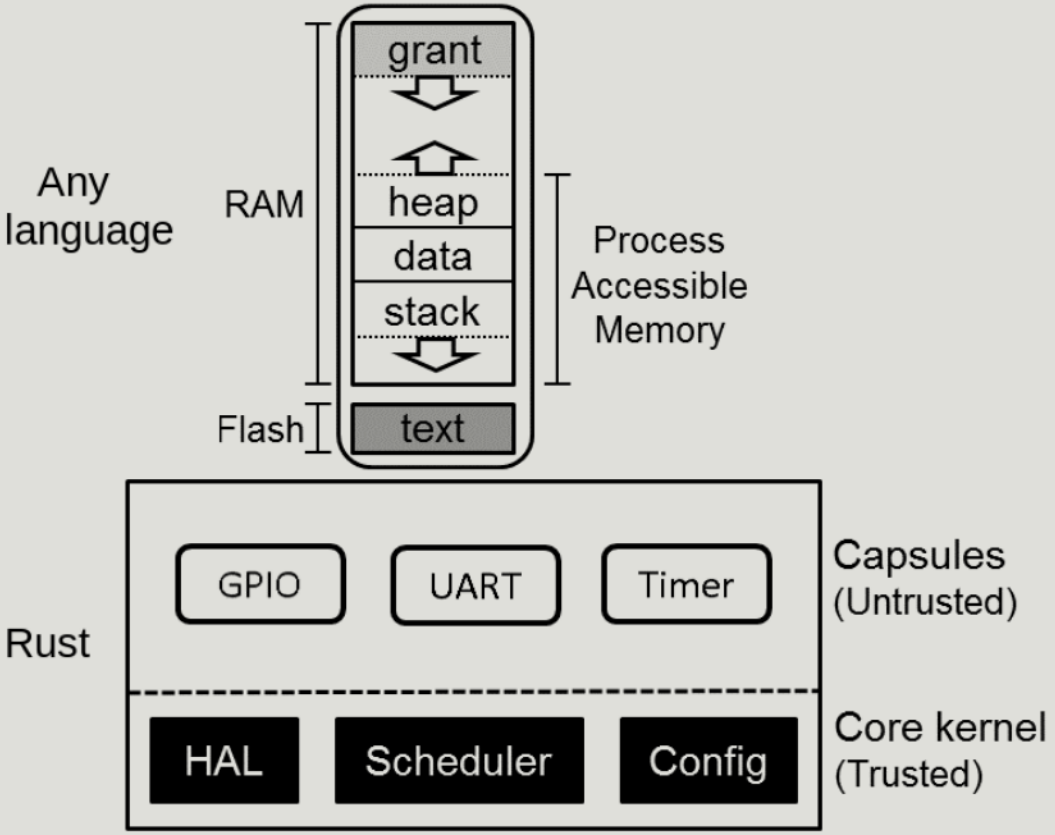
Capsules (drivers) are isolated by Rust

- written in **safe** Rust
- users choose which capsules to trust
- **zero-size capabilities** enable access to privileged functions
- **zero cost** overhead for this isolation via Rust language safety
- static memory usage
- **capsule threat model is buggy but not malicious**



Kernel

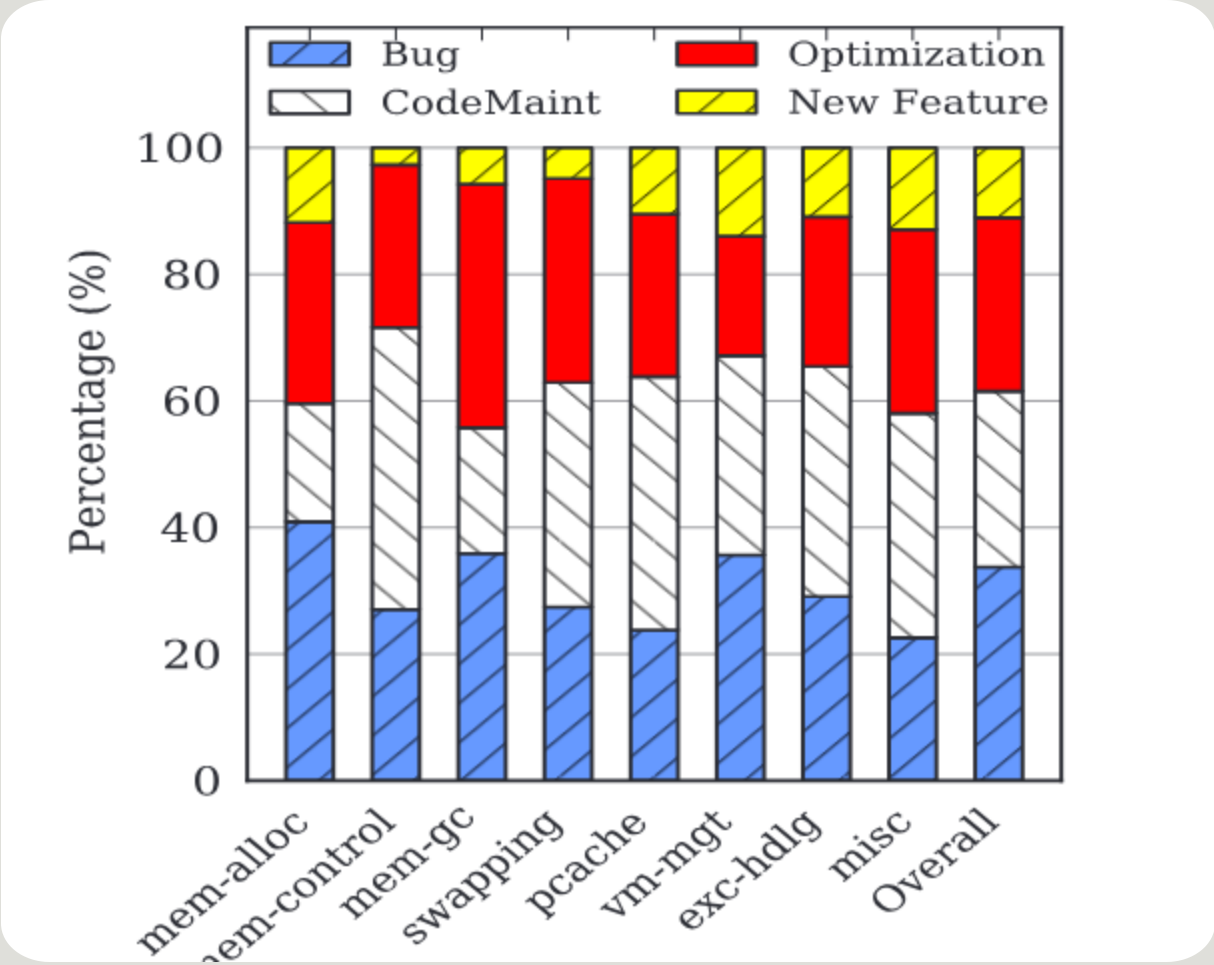
- small Trusted Compute Base
- safety via Rust's language sandbox



Secure by design

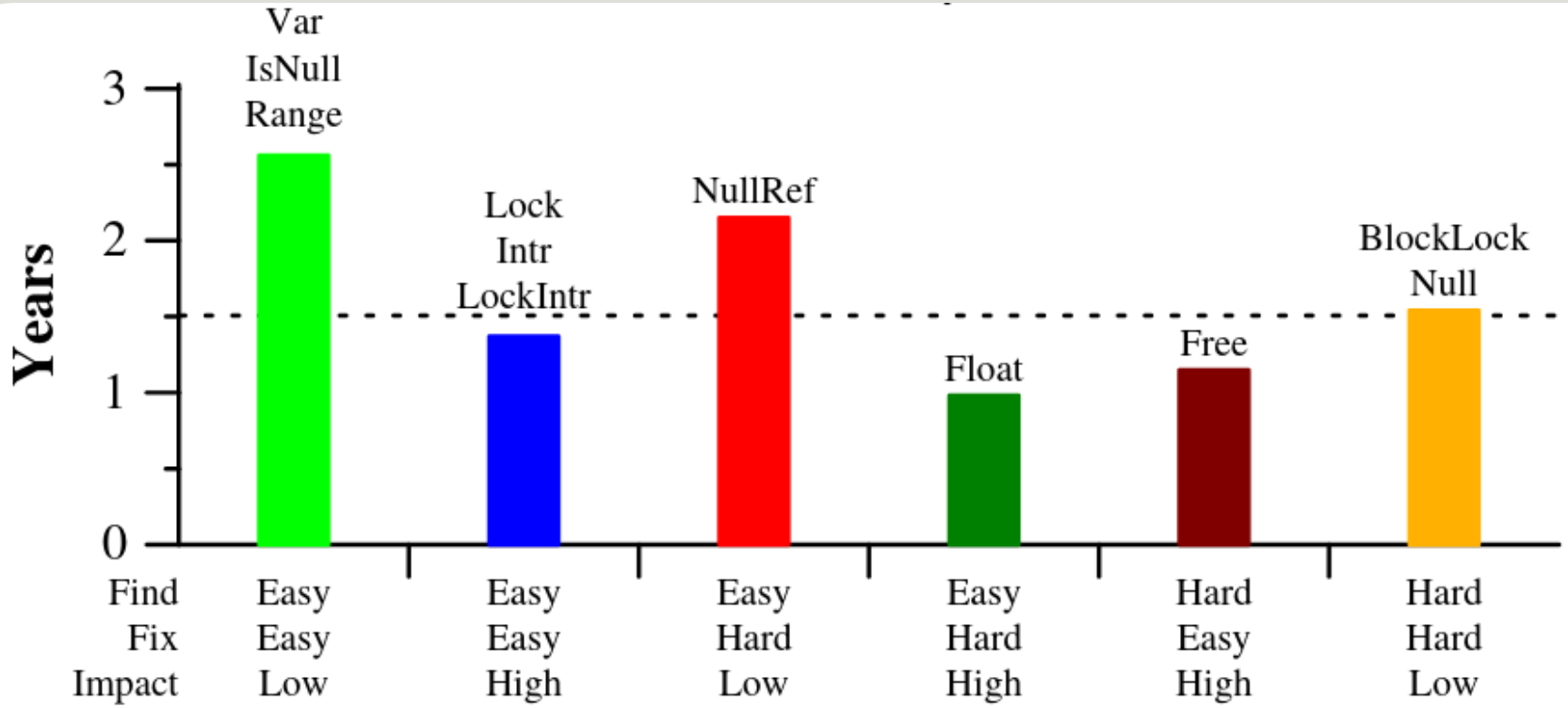
- **Rust**
  - **Memory management** ✅ Rust language safety
  - **Undefined behavior** ✅ undefined behavior is explicit with `unsafe`
- **Architecture**
  - **Faulty drivers** ✅ capsules are isolated
  - **Concurrency/Locking** ✅
- ❓ ❓ ❓
  - **Logic bugs** ⚠️ testing and code review is not enough
  - **Malicious attacks** ❌

you can still have A LOT OF bugs



Huang 2016

bugs stay undetected (and exploitable) for years



(b) Per finding and fixing difficulty, and impact likelihood

Palix 2011

1. Compartmentalizing firmware with Rust

2. Lightweight verification for firmware

what is formal verification?

> Formal verification is the process of using automatic proof procedures to establish that a computer program will do what it's supposed to.

- Amazon

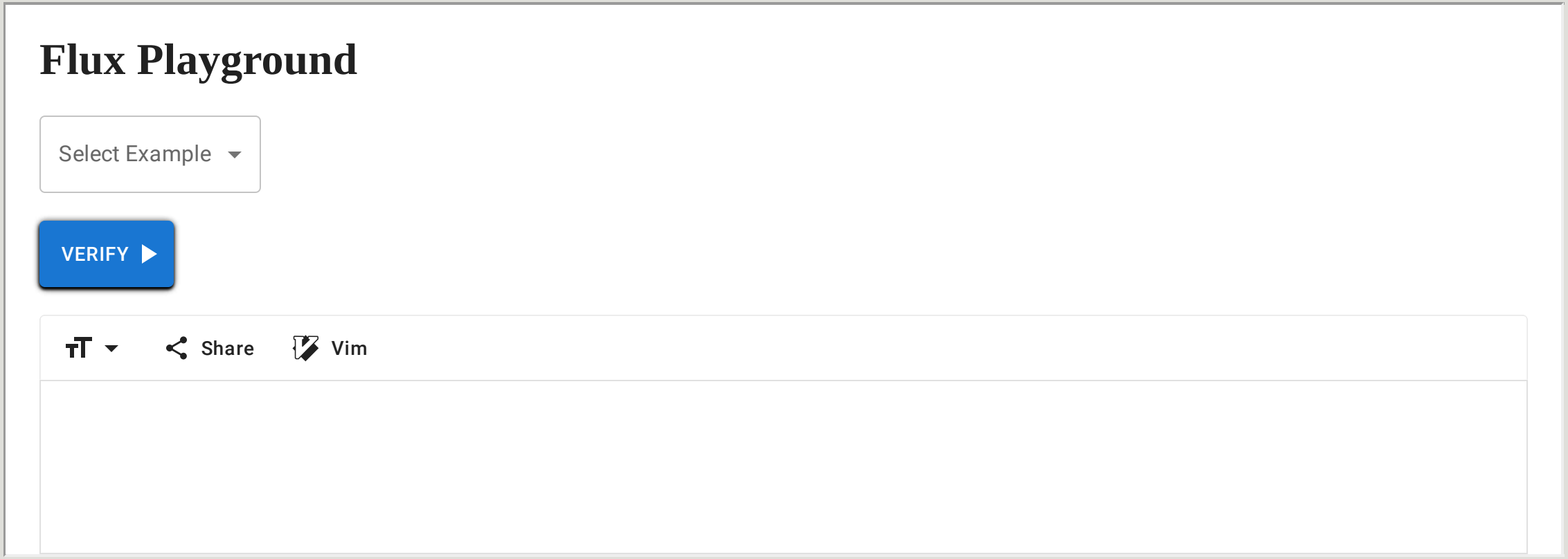
here are all the rumors you have heard

- "It is so slow"
- "I don't want to rewrite my code"
- "You need a PhD in formal methods first"

here are all the rumors you have heard

- "it is so slow" -> **takes seconds**
- "I don't want to rewrite my code" -> **incrementally retrofit existing code**
- "You need a PhD in formal methods first" -> **actually it's easy!**

Demo: `abs()`



https://flux.programming.systems/

Example of running the type system on a function

```
fn abs(x: i32) -> i32 {
  if x >= 0 {
    x
  } else {
    -x
  }
}
```

then we can add this

```
#[flux::sig(fn(x: i32) -> {v. i32[v] | v >= 0 && v >= x})]
```

fails because `abs(MIN_INT)` overflows

```
#[flux::sig(fn(x: i32{x != i32::MIN}) -> {v. i32[v] | v >= 0 && v >= x})]
```

liquid types

liquid types = types + *decidable* logical predicates

Demo: refine PMP from RISC-V spec

Goal: encode the spec into the code

we are still working on the formal model, but this is the high level for how you would apply liquid types to abstract properties

This means you can reduce trust on the kernel and rely on the specification of the flux annotations to maintain memory isolation.

OPEN SOURCE FIRMWARE CONFERENCE

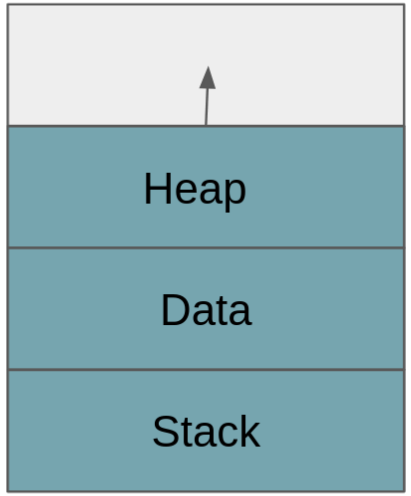
rv32i/{pmp,epmp}: invalidate unused regions on config switch #3541

tock#3541

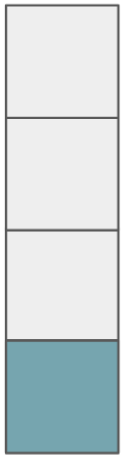
rv32i/{pmp,epmp}: invalidate unused regions on config switch #3541

Merged **bradjc** merged 1 commit into `tock:master` from `lschuermann:otdev/pmp-invalidate-fix` on Jul 18, 2023

Process 1

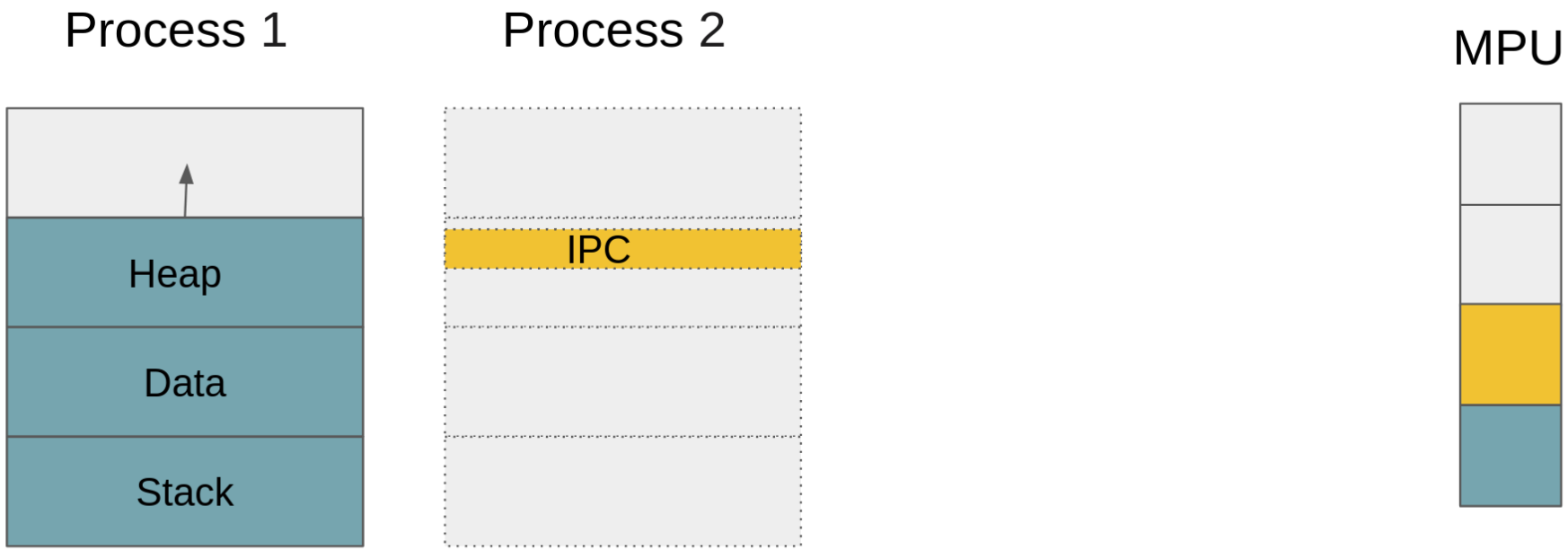


MPU



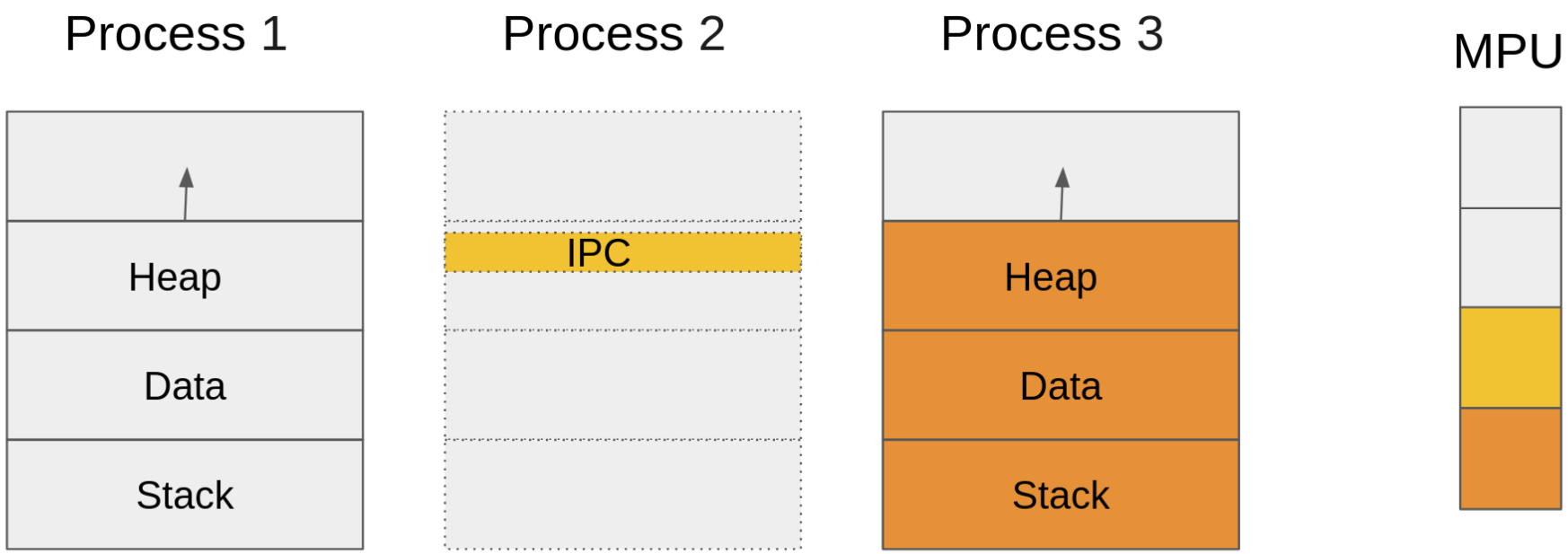
rv32i/{pmp,epmp}: invalidate unused regions on config switch #3541

Merged bradjc merged 1 commit into tock:master from lschuermann:otdev/pmp-invalidate-fix on Jul 18, 2023



rv32i/{pmp,epmp}: invalidate unused regions on config switch #3541

Merged bradjc merged 1 commit into tock:master from lschuermann:otdev/pmp-invalidate-fix on Jul 18, 2023



Summary

- **Rust**
  - **Memory management** Rust language safety
  - **Undefined behavior** is explicit and minimized
- **Architecture**
  - **Faulty drivers** capsules are isolated from kernel
  - **Concurrency/Locking** avoided
- **Lightweight verification**
  - **Logic bugs** via verification
  - **Malicious attacks** cannot break the easily auditable compile-time safety

Tock: safe embedded operating system

Tock is open source!

https://tockos.org/

Join Tock Slack, Matrix, verification mailing list

Flux: scalable Rust verification

> Proving should be as easy as programming

https://github.com/flux-rs/flux/

https://flux.programming.systems/

Presentation permalink: godsped.com/safe-firmware