UVA ENGINEERING
LINK LAB

UNIVERSITY of VIRGINIA

Tock: A Safe and Secure Operating System for Root-of-Trust Hardware

Zero Trust Hardware Architectures Workshop (ZTHA)

September 4, 2024

Brad Campbell – bradjc@virginia.edu

http://www.cs.virginia.edu/~bjc8c/

Chromebook

AMD Ryzen AI 300

Tock: embedded operating system

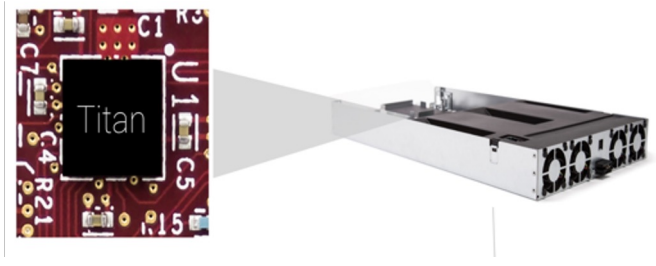
- Key design goals
  1. Safety
  2. Security
  3. Multiprogrammability
- Targets microcontrollers
  - Ex: Cortex-M, RISC-V 32 bit
  - 16-256kB RAM, 256kB-1MB code
  - No virtual memory

- Industry buy-in
  - Google OpenTitan, OpenSK
  - Microsoft
  - HPE
  - Infineon
  - OxidOS Automotive

Overview

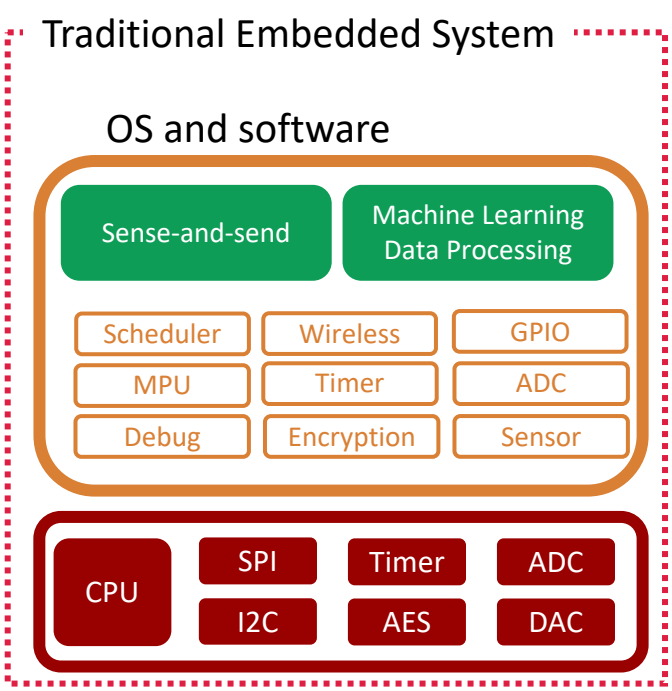
- Tock Operating System
  - What is it?
  - Tock Threat Model
  - Dynamic Memory Allocation
  - Processes and Updates
- Features in-the-works
  - Emerging use cases
- Community

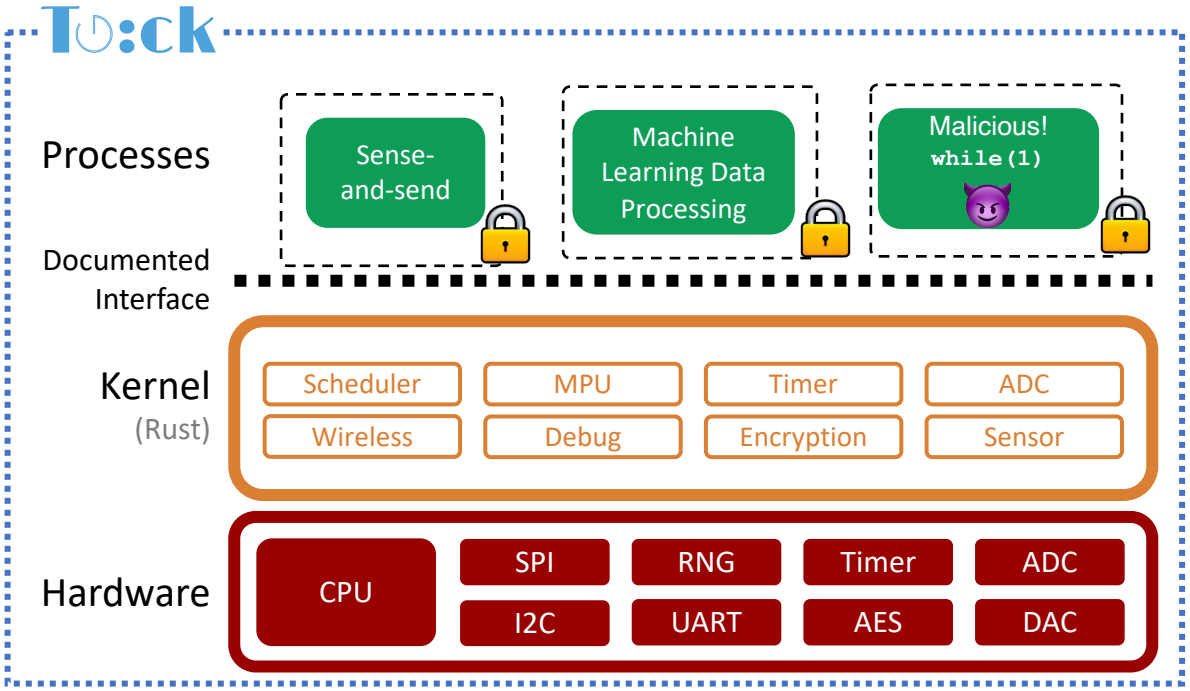
OpenSK 3D printed case



Tock OS architecture

- Tock: new OS for IoT emphasizing safety and reliability
  - Written in the Rust programming language (emphasizes safety and robustness)
- Individual processes are “sandboxed”
  - Cannot access or affect any other process
  - If a process is buggy or malicious it does not compromise the entire system



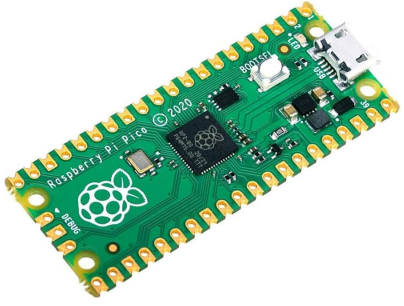


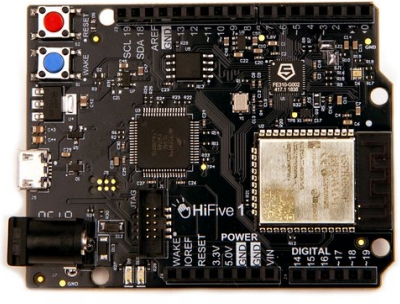
Kernel written entirely in Rust

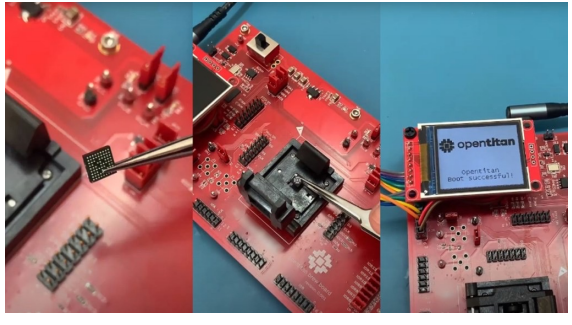
UNIVERSITY of VIRGINIA

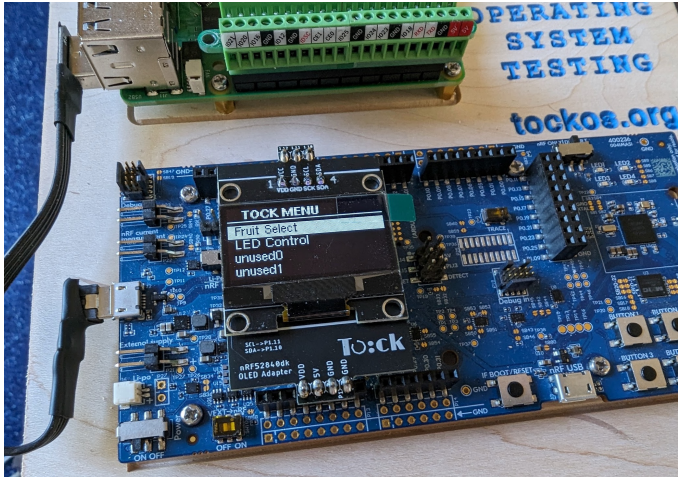
- Rust: type- and memory-safe systems language
  - Types enforced by compiler: no buffer overflows, null dereferences, or arbitrary memory accesses
  - Fast: statically compiled, within 30% performance with C
  - No garbage collection, all memory lifetimes tracked
- …but low-level OS code is fundamentally memory-unsafe
  - Memory-mapped I/O
  - Interrupts
  - Context switches
  - System calls
- Rust provides the `unsafe` keyword as an escape hatch
  - Disables certain (not all) compiler checks
  - Additional language features allowed (e.g. dereferencing pointers)
  - Tock very explicit about where `unsafe` is used

Support for 30+ boards



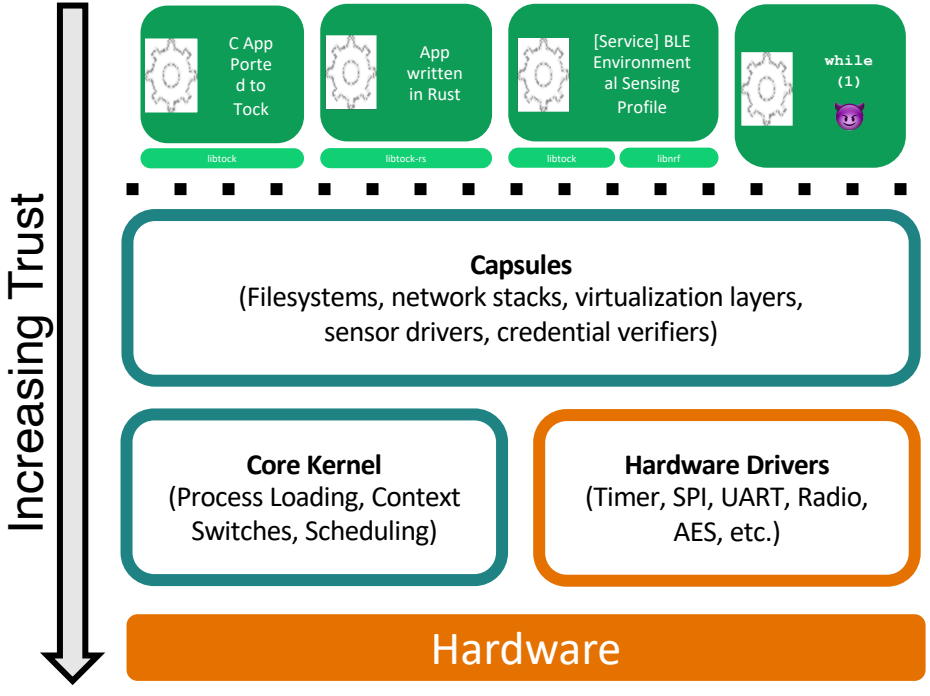






Architectural trust layers

- Applications
  - Completely untrusted
  - Isolated using MPU
- Capsules
  - Untrusted
  - In Rust, no `unsafe`
- Core Kernel & Drivers
  - Trusted
  - Limited `unsafe`

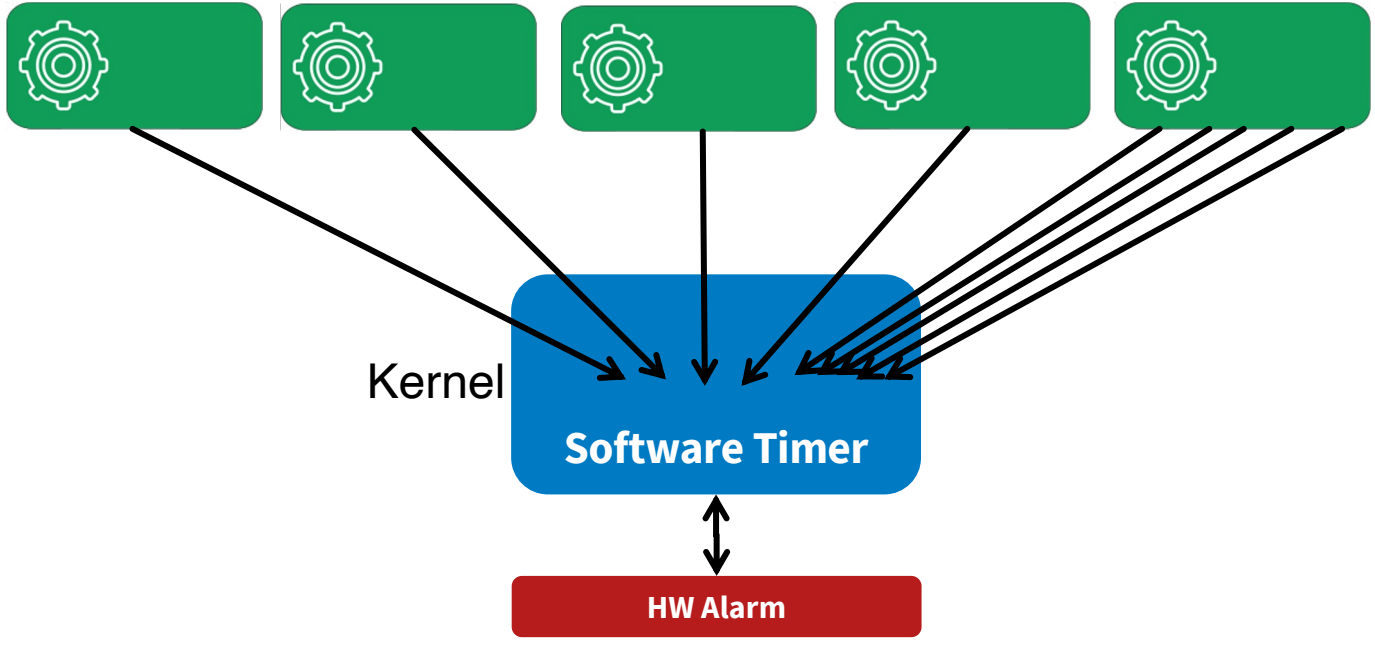


Tock Threat Model

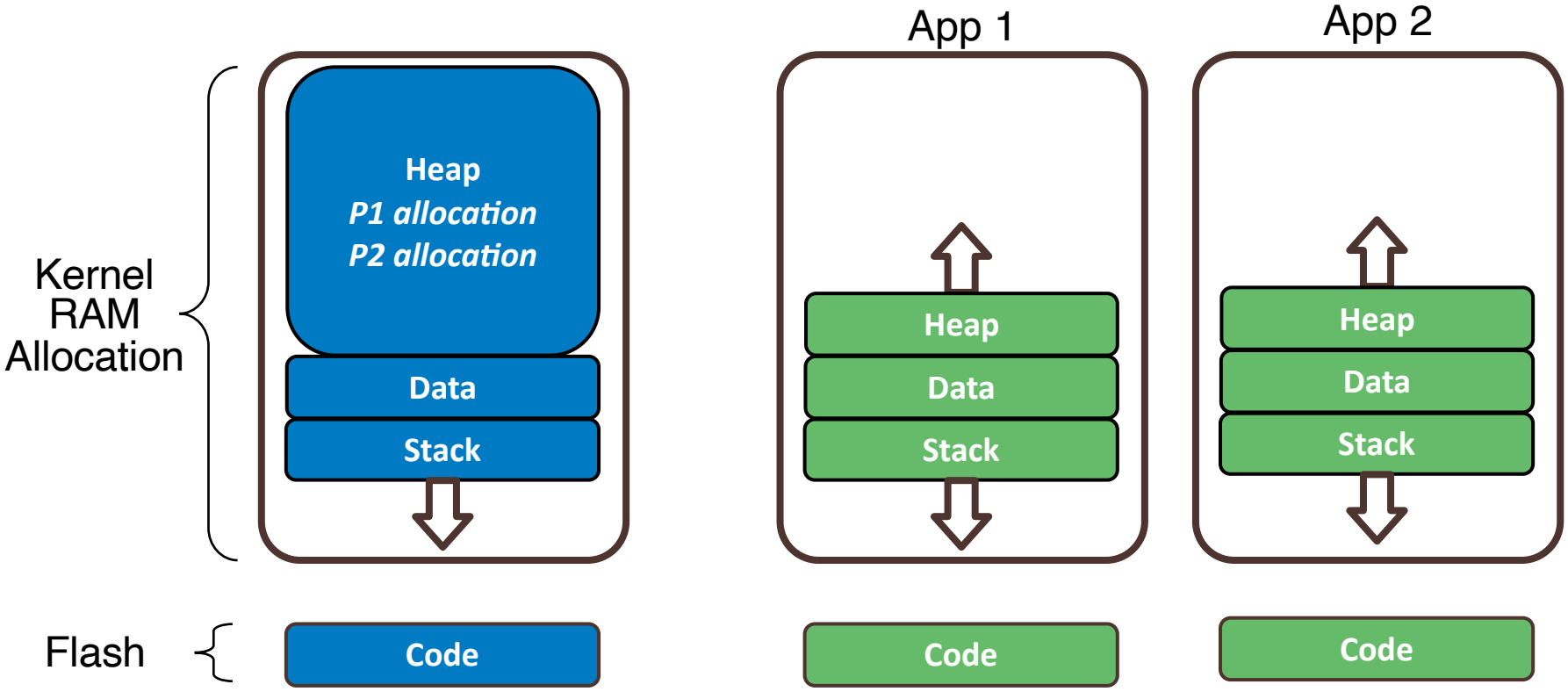
- Formal Definition of what the kernel guarantees
- Isolation Provided to Processes
  - **Confidentiality**: A process' data may not be accessed by other processes or by capsules, unless explicitly permitted by the process.
  - **Integrity**: Process data may not be modified by other processes or by capsules, except when allowed by the process.
  - **Availability**: Processes may not deny service to each other at runtime.
- Isolation Provided to Kernel Code
  - **Confidentiality**: Kernel data may not be accessed by processes or capsules, except where explicitly permitted by the owning component.
  - **Integrity**: Processes and capsules may not modify kernel data except through APIs intentionally exposed by the owning code.
  - **Availability**: Processes cannot starve the kernel of resources or otherwise perform denial-of-service attacks against the kernel.
- Implementing these guarantees
  - Rust compiler
  - Hardware memory protection
  - Application format
  - Software capabilities
  - Code review and software architecture
- More detail: https://book.tockos.org/doc/threat_model/threat_model

Challenge: ensuring reliability with limited resources

- Dynamic applications can lead to resource exhaustion in the kernel
- What happens when `malloc()` fails inside the kernel?!?
  - Crash??

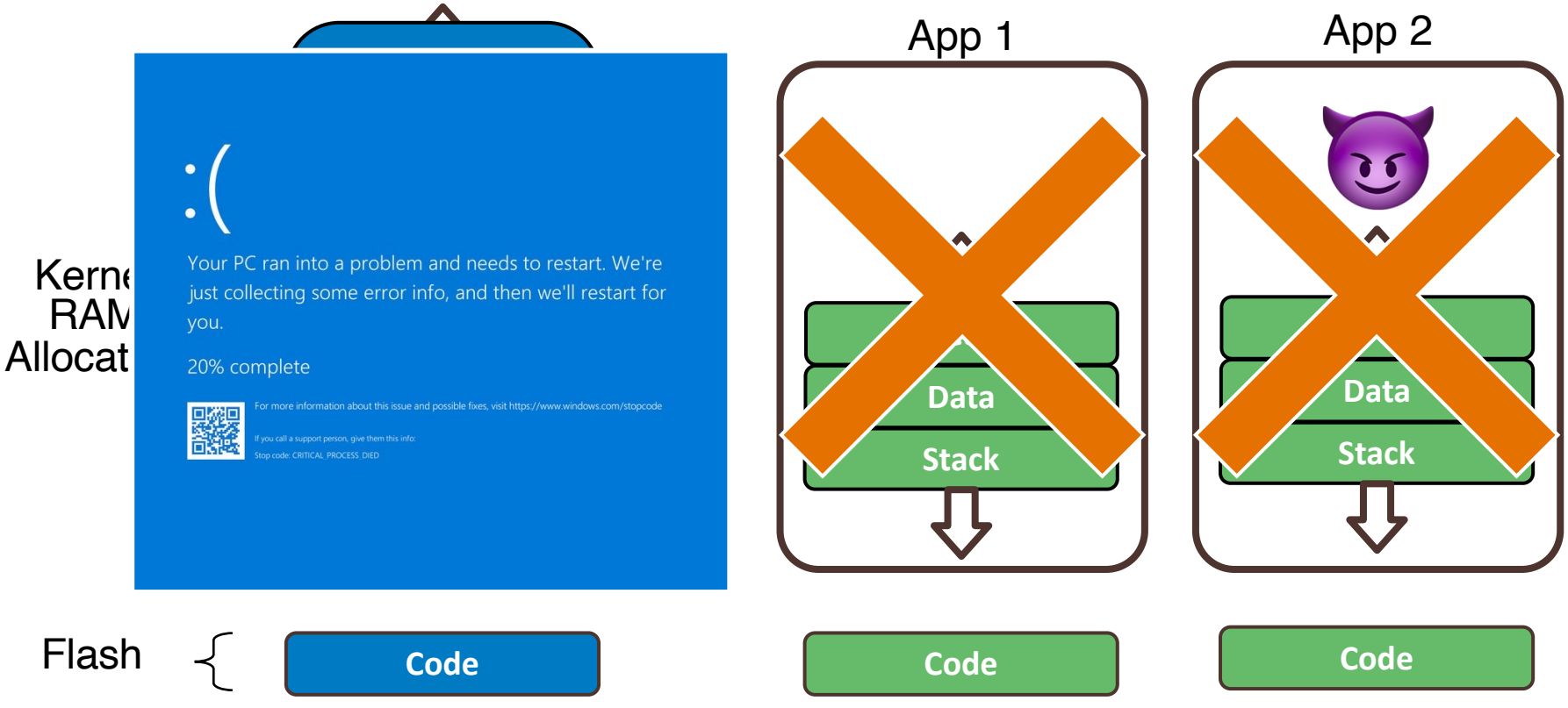


Dynamic allocation in the kernel is OK for a while…

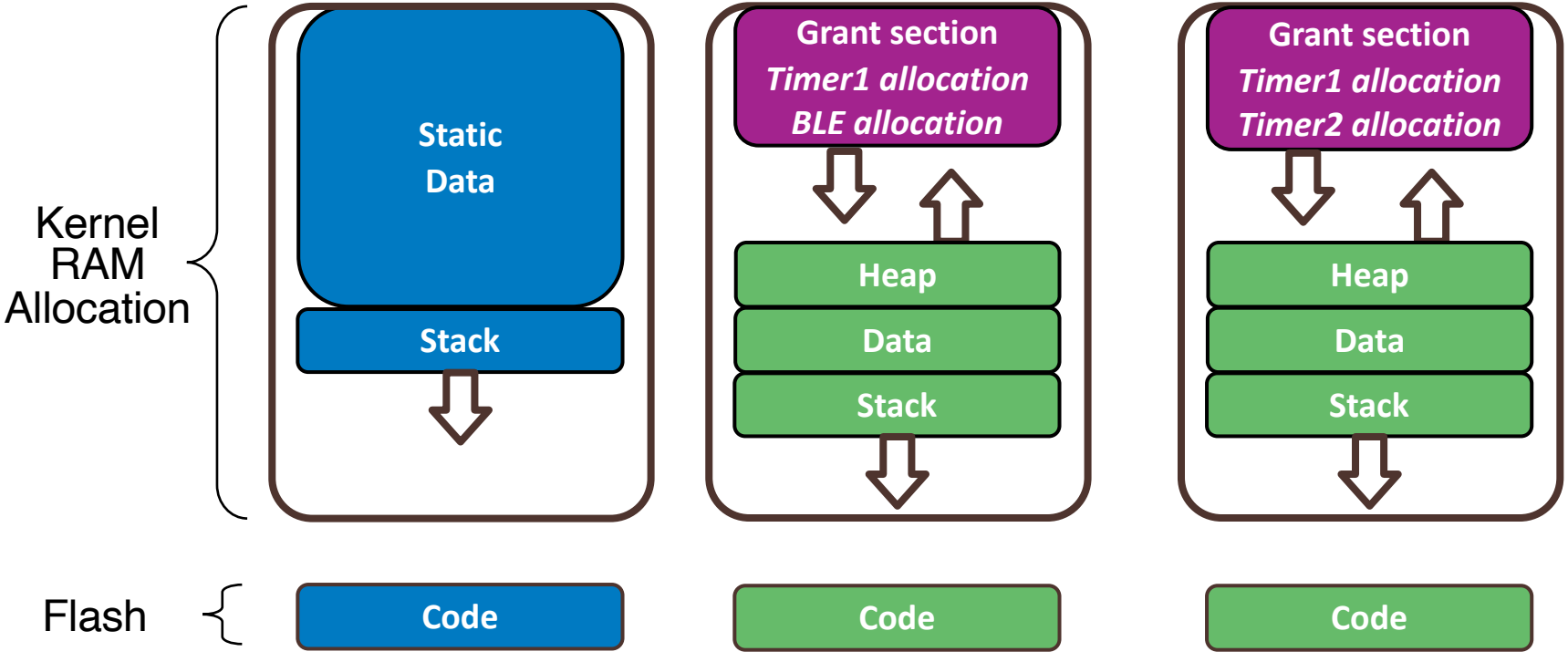



One month
later

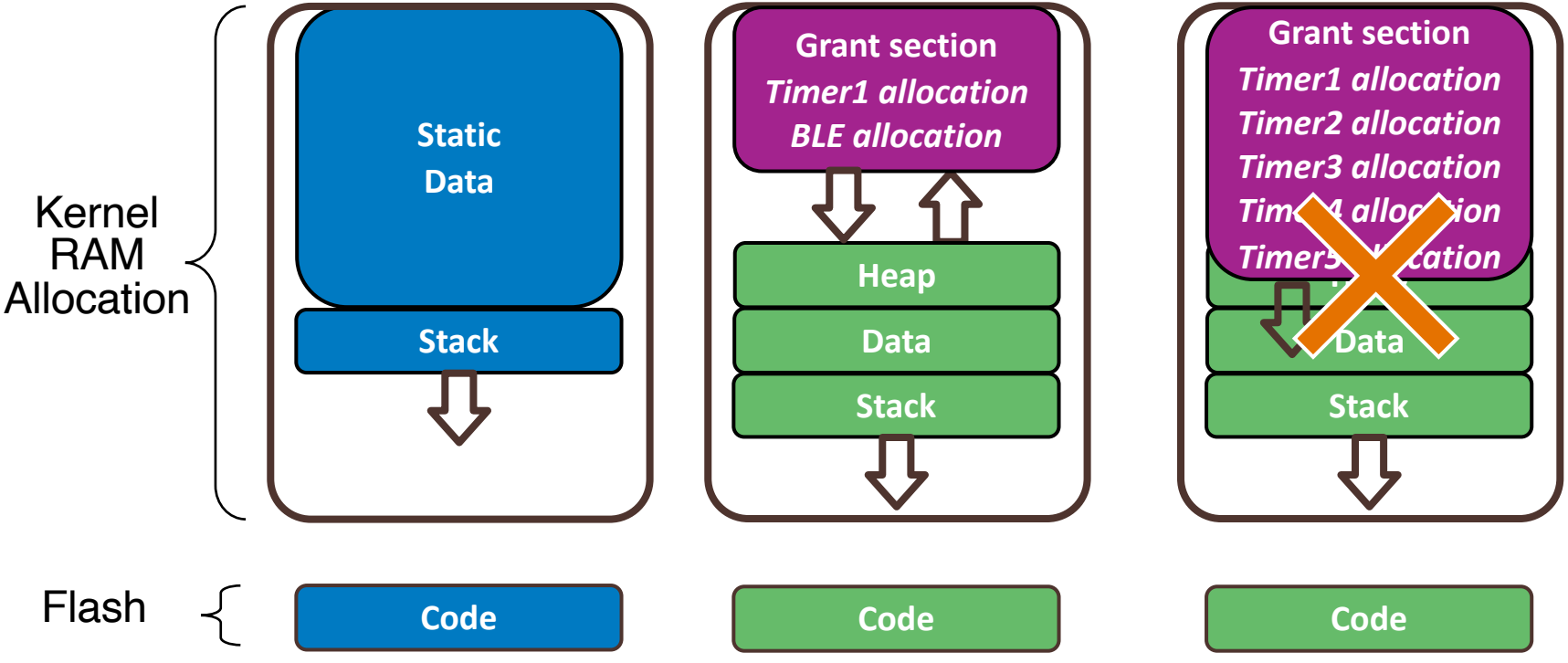
A broken/buggy/malicious app exhausts the kernel's heap!



Fix: all allocation is done in “Grant” regions inside of process memory space

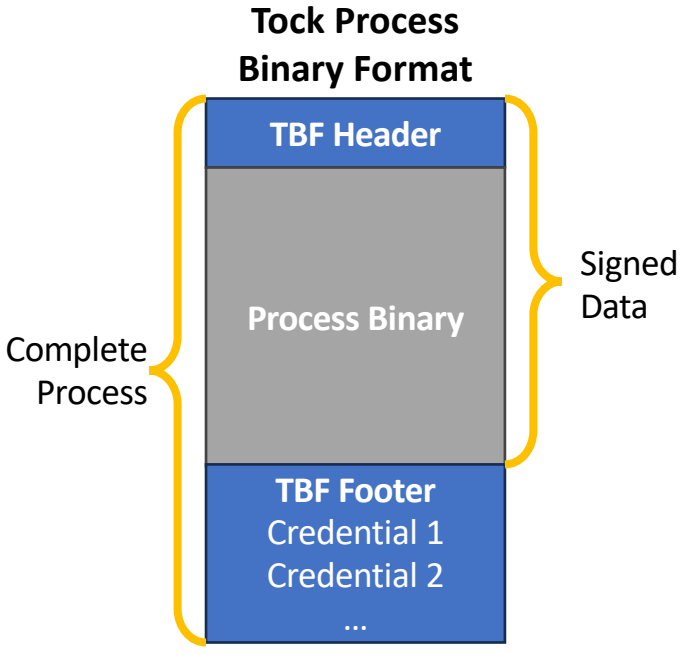


If a process exhausts its Grant region, only that process will fail/crash



Process format and credentials in TockOS

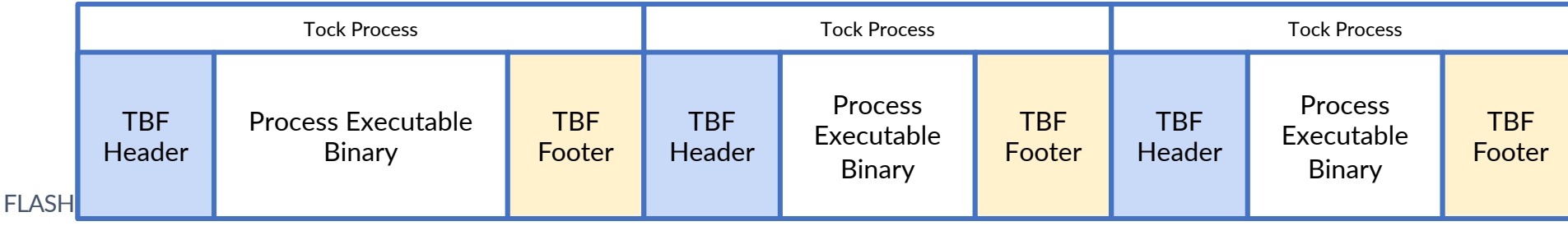
- Each Tock process is in Tock Binary Format (TBF)
- TBF Header
- Process Binary
  - Actual instructions and data for the process (compiled from any language)
- TBF Footer
  - List of credentials for the process
  - Ex: hash, HMAC, signature



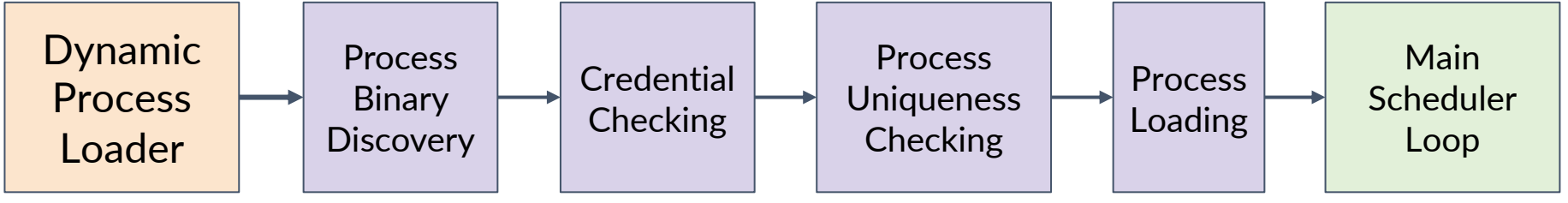

```
TBF Footers
  Footer
    footer_size          :       9520       0x2530
  Footer TLV: Credentials (128)
    Type: SHA256 (3) ✓ verified
    Length: 32
  Footer TLV: Credentials (128)
    Type: Reserved (0)
    Length: 9472
```

Processes can be updated individually

Processes stored sequentially in flash:



Core kernel workflow:



Overview

- Tock Operating System
  - What is it?
  - Tock Threat Model
  - Dynamic Memory Allocation
  - Processes and Updates
- Features in-the-works
  - Emerging use cases
- Community

Panic-free code: Converting runtime checks to compile-time checks

University of Virginia

- Simple example: what happens if:

```
uint8_t buffer[10];
x = buffer[15];
```

- C: memory bug
- Rust: system panic
- Crowdstrike failure shows the downsides of kernel panics!

Let the compiler reason about where crashes can happen

- Unrecoverable errors become `abort()`

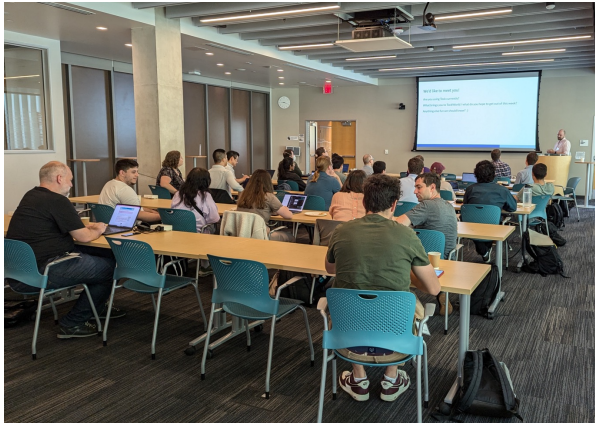
- At compile time, verify the compiler did not insert any `panic()` calls
  - Parse the generated ELF or LLVM IR
  - Fail if `panic()` is present
  - Panic is how rust signals a runtime error
    - Better than a security vulnerability
    - Still results in a security crash

- Use alternatives to avoid `panic()` calls
  - Eg: replace `buffer[15]` with `buffer.get(15).unwrap_or(0)`

Growing Community around Tock

UNIVERSITY of VIRGINIA

- Tock World 7 Meeting – June 26-28, 2024
  - Meeting of users from academia and industry
    - Held at UCSD
  - Three-day workshop
    - Developers day
    - Community day
    - Tutorial day
  - Shared progress on secure app updates
- Establishing a foundation to steward Tock
- Tutorials & Documentation
  - book.tockos.org
- Open-source project
  - github.com/tock/tock



TockOS

http://tockos.org

Thank you! Questions?

University of Virginia

Brad Campbell – bradjc@virginia.edu

http://www.cs.virginia.edu/~bjc8c/

- Tutorials & Documentation
  - book.tockos.org
- Open-source project
  - github.com/tock/tock

TockOS

http://tockos.org