LEDGER

Once upon a time in IoT

An industry-grade OS perspective for IoT security

SSTIC 2024 - Patrice HAMEAU, Victor SERVANT, Philippe THIERRY, Florent VALETTE
5-7 June 2024

What you will see (e.g. the agenda)

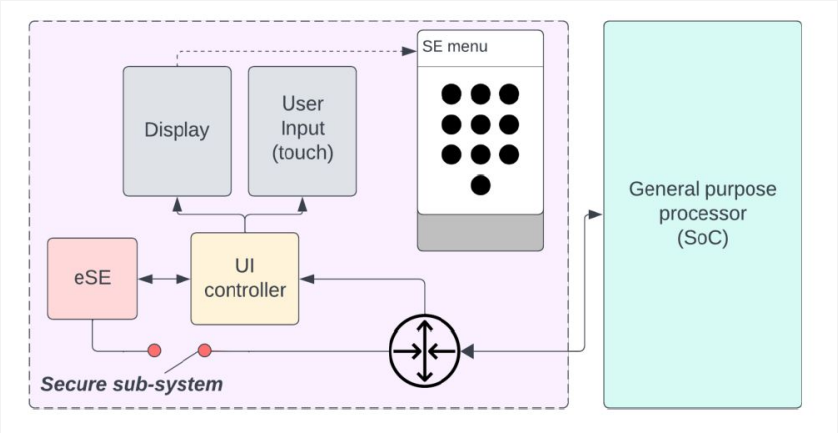
1. Once Upon A Time: last year!
2. You want a TUI?
3. A New OS is born:
   a. Welcome Outpost OS :)
   b. Key concepts & architecture
4. Security under the microscope
   a. Security mechanisms
   b. Resource isolation
   c. Predictable execution
   d. Runtime countermeasures & robustness
5. Let’s build it: the development environment
   a. SDK and Integrator
   b. Reproducible build And secure development chain
6. Comparison with other OS es
7. Back to the future: what’s next
8. Conclusion of our journey
9. This is the end

Once upon a time...

Once upon a time… last year!

- SSTIC '23 presentation of deported UI design:
  - Isolated secure enclave inside a SoC.
  - Driving a Trusted User Interface (input/output)
  - Acting as a virtual display for Android OS

- *Aiming at*
  - Supporting security-critical user interactions through a Trusted UI
  - Significantly reducing the attack surface compared to ARM TrustZone or virtualized implementations

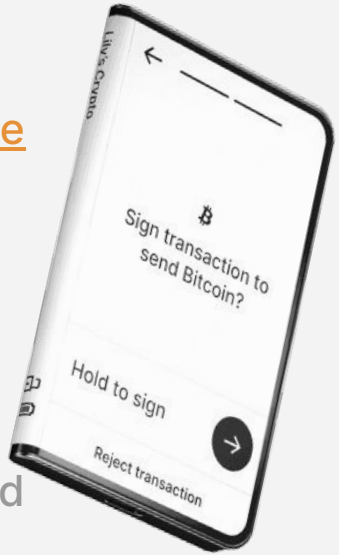


Forewords about Trusted User Interface (TUI)

- A secure, isolated, and trusted environment within a device, in charge of **user interactions**
- Unlock trust of UI based trusted functions (authentication, transactions confirmations, ...)

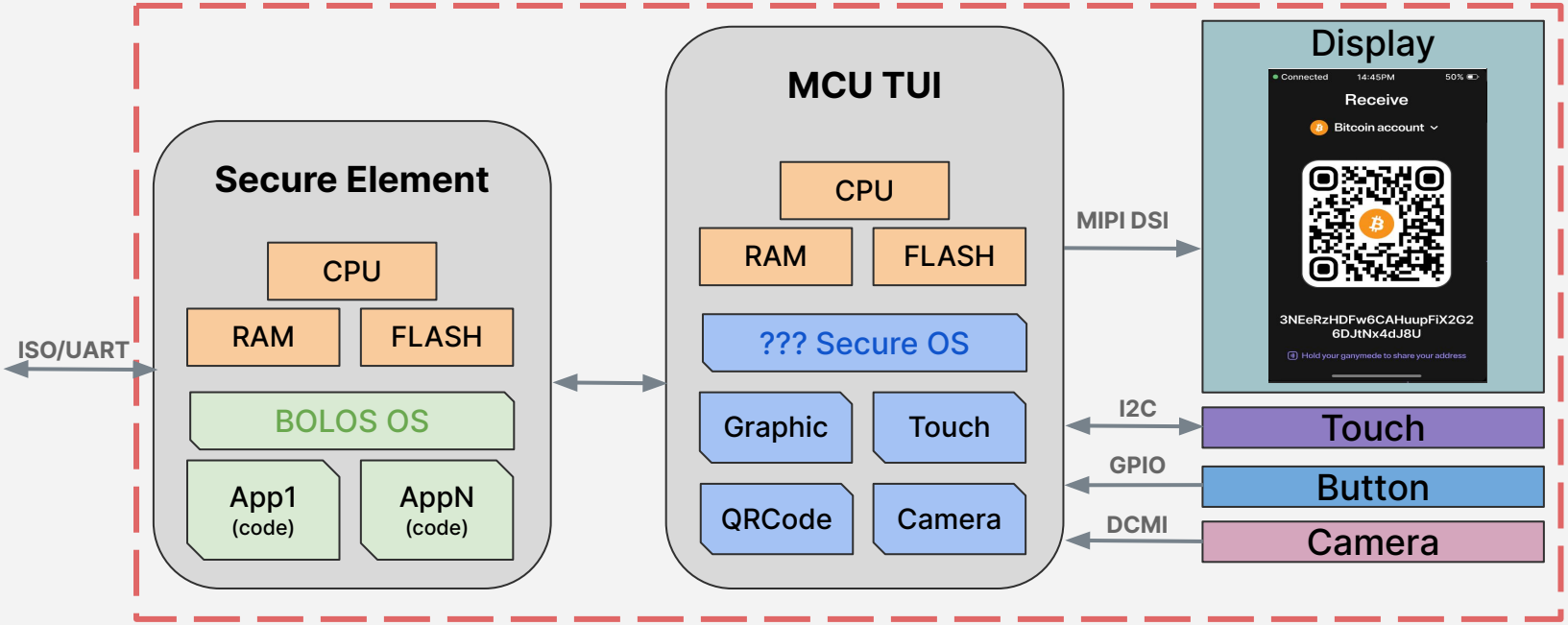
- **Key concepts in our product**
  - Our Web3 model relies on security-critical TUI interface (WYSIWYSign) driven by Secure Element

- New products... going to higher UI resolutions implies:
  - New hardware architecture with UI co-processor
  - With increased complexity of the firmware stack
  - While keeping highest security and robustness levels
  - And reusability for future products in regards of needed investments



Let’s design TUI for a high resolution display in embedded device!

- We already have the Secure Element Operating System (BOLOS)
- Let’s add a MCU as Secure Element graphical co-processor
  ... but we need a secure and robust OS on this MCU for the TUI!
  ... and we have not found an open one that check all our boxes 😞



Isolated TUI

All TUI elements are hardware-isolated by the Secure Element from device external interfaces (USB, BLE, ...)

A new OS is born...

A new OS is born

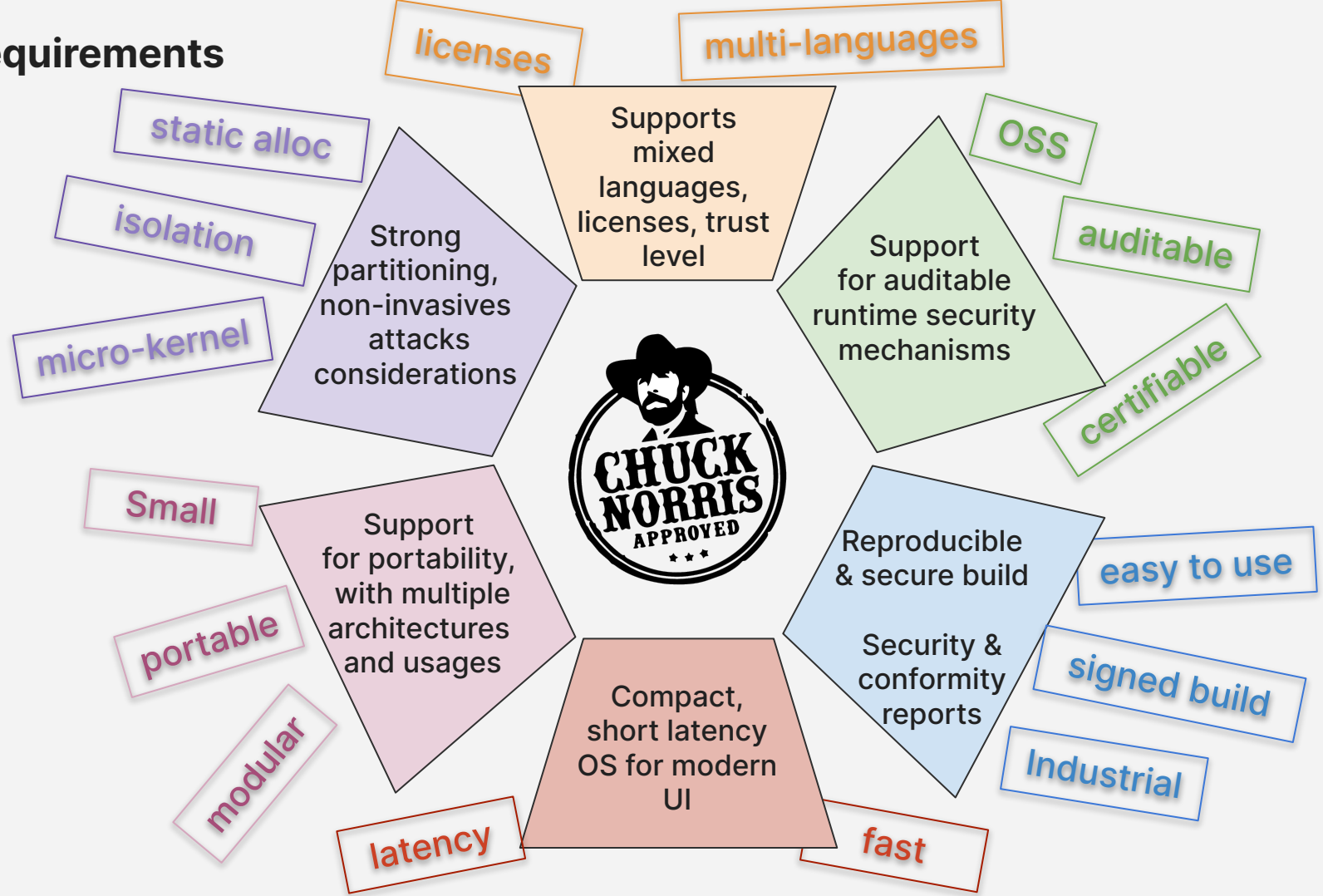
- 😔😔 So the we have not found the OS of our dreams from our wish list:

  - Industrial-grade Operating System
  - With high level of security and robustness
  - Full isolation of resources and applications
  - Complete Software Development Kit
  - Project-oriented and trustable development chain
  - Open-Source and auditable by our customers

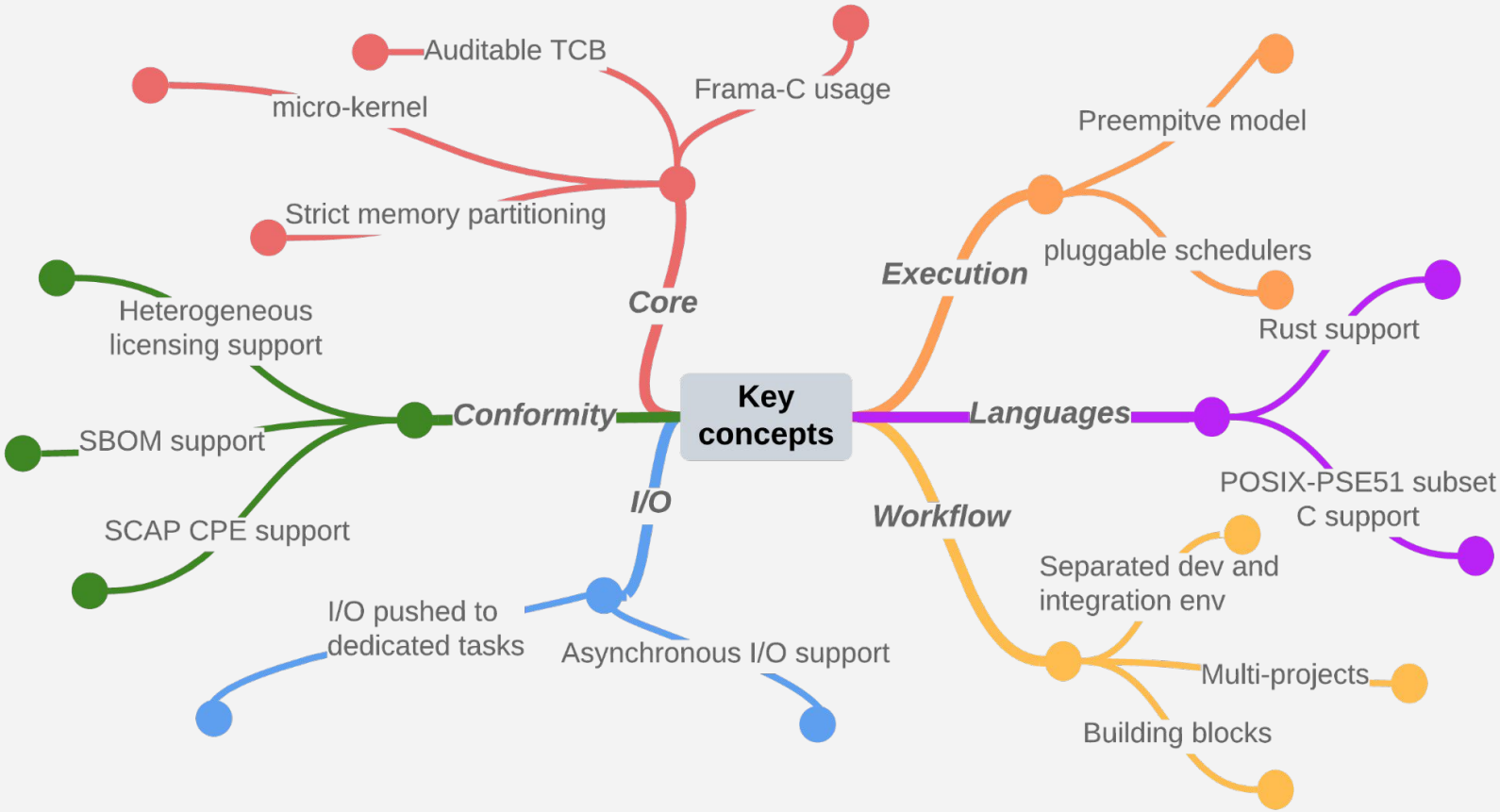
- Ok, not a problem, let's develop another one (yes we knew where we were going here from our past experiences 😅!)

- So let's start the journey of a new OS
  ... and its name is **Outpost OS**

OS requirements



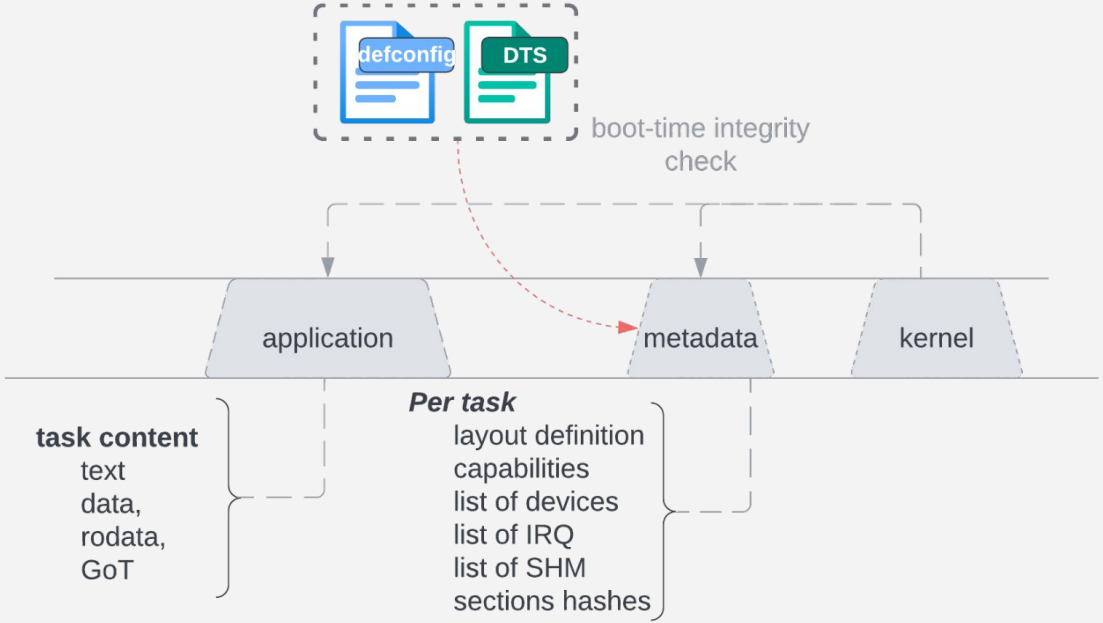
Experience-based Key concepts & Architecture



Security under the microscope

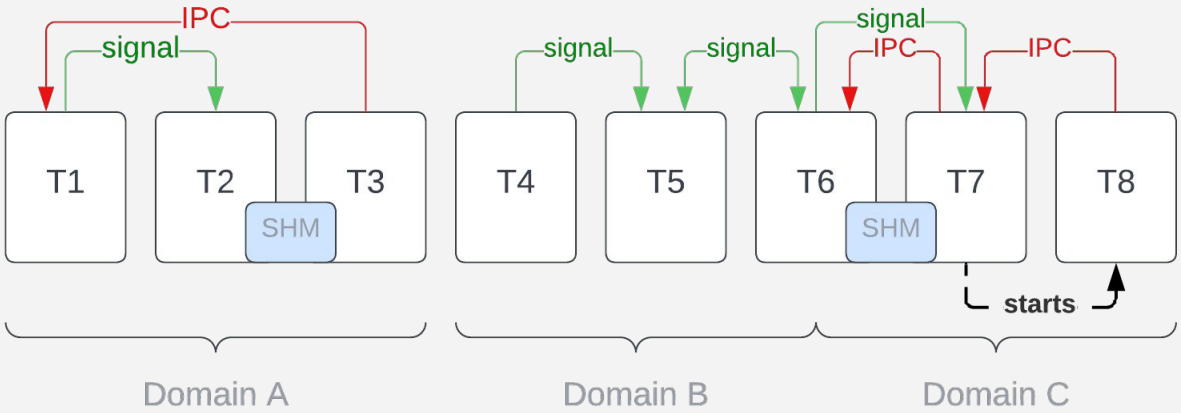
Resources Isolations

- dedicated and pre-allocated exclusive resources
- Using a three-third content repartition
- Init-time consistency check
- Kernel enforce run-time ressources isolation



Communication channels

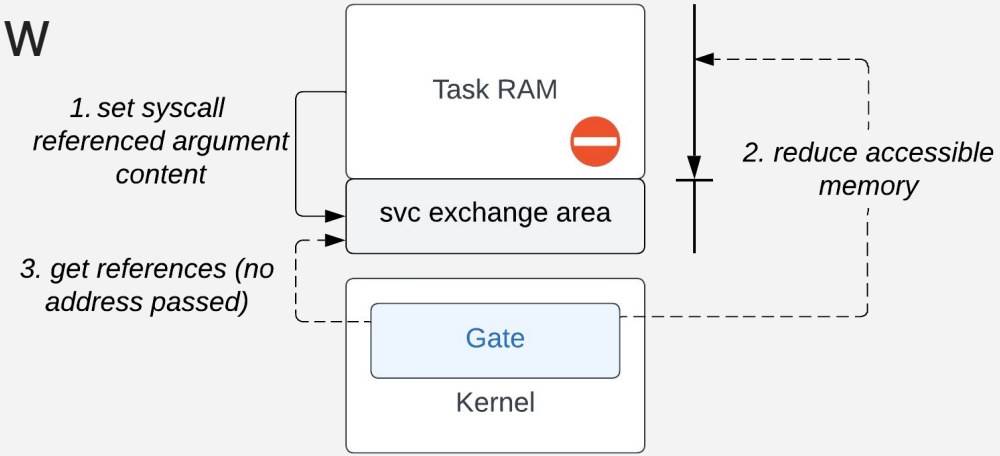
- IPC communications using single copy model
- Signals support
- Shared memories, defined as ressources with ownership and permissions
- Domains separating communicant task sets
- Lifecycle enhanced support (start & termination models, start capability)



Security Mechanisms

- No address dereference between kernel and userspace
- No task private data accessible from kernel
- Syscall-based ressource (un)mapping using MPU
- Bus-master ressources are not under direct control of user applications
- Capability-based SW & HW ressources access
- Whole kernel code checked for correctness

frama C
Software Analyzers



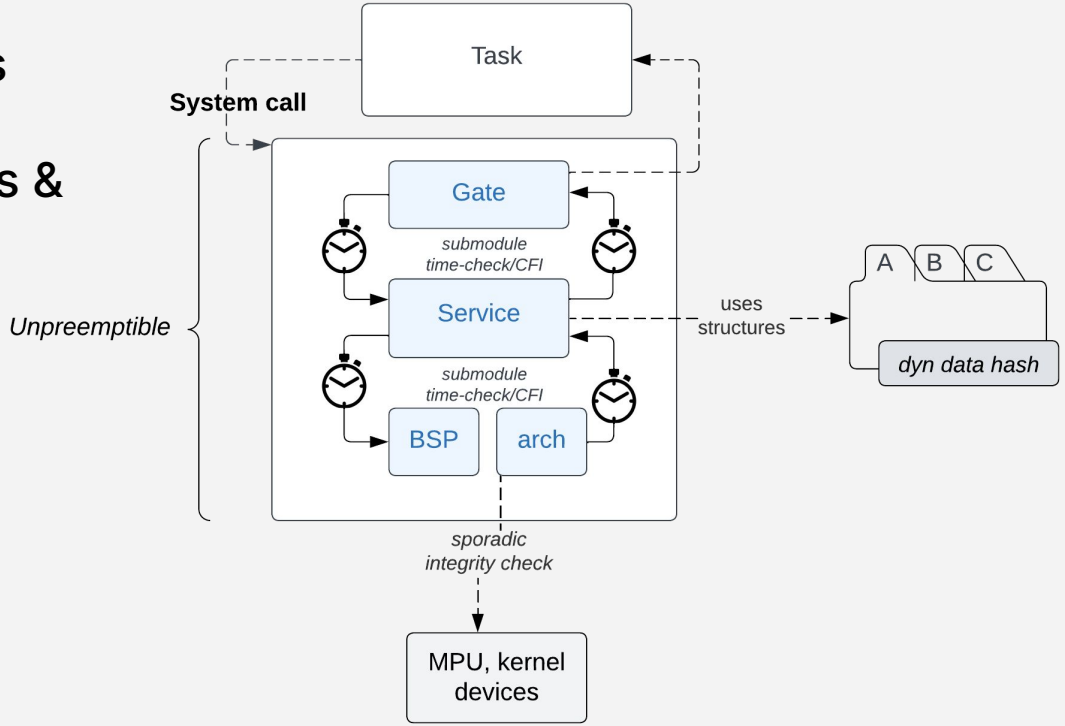
Measured Execution

Security

- No dynamic resources
- IRQ non-reentrant
- Measured kernel paths & CFI
- Dyn data integrity

Robustness

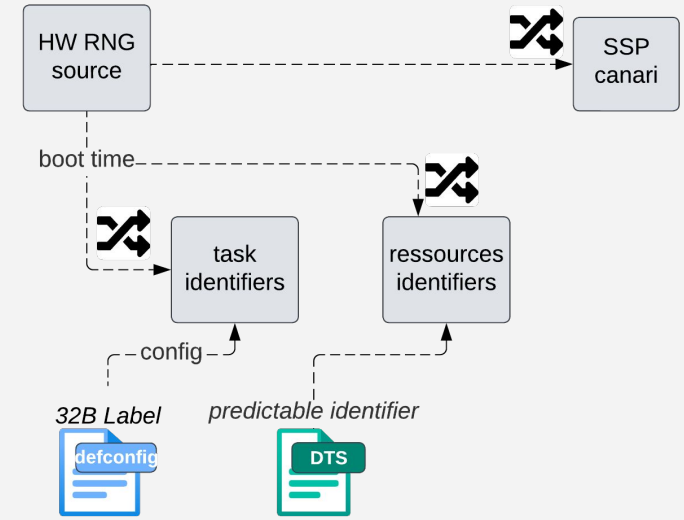
- Deadlock detection
- RRMQ* scheduler, no starvation

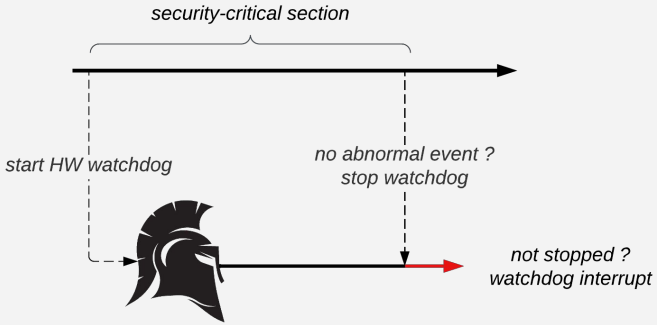


*: Round-Robin with Multi-queue and Quantum

Runtime counter-measures

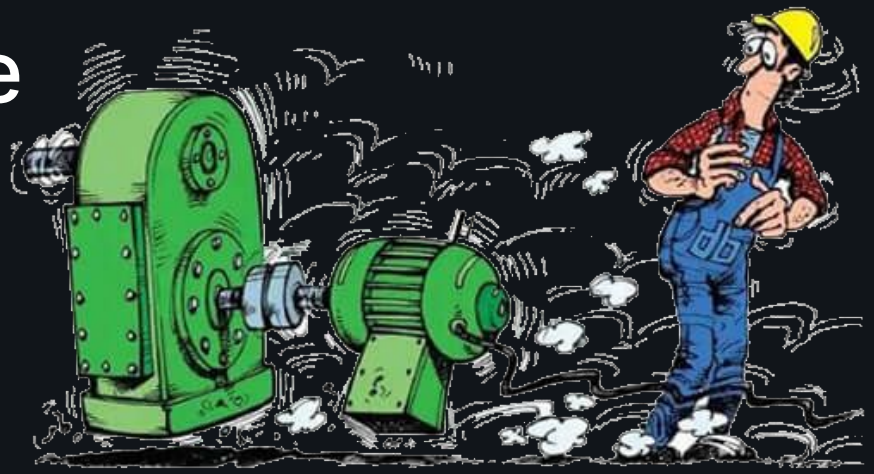
- Huge usage of boot-time forged random seeds
- Task identifier fully regenerated each time a task (re)start
- Loop double counters for critical code blocks
- Watchdog-based abnormal event detection (overconsumption, CFI check failure...)
- Storage of abnormal events on backuped memory
- Check of previous abnormal events at boot-time





Let’s build it:

A development environment with the greatest of care



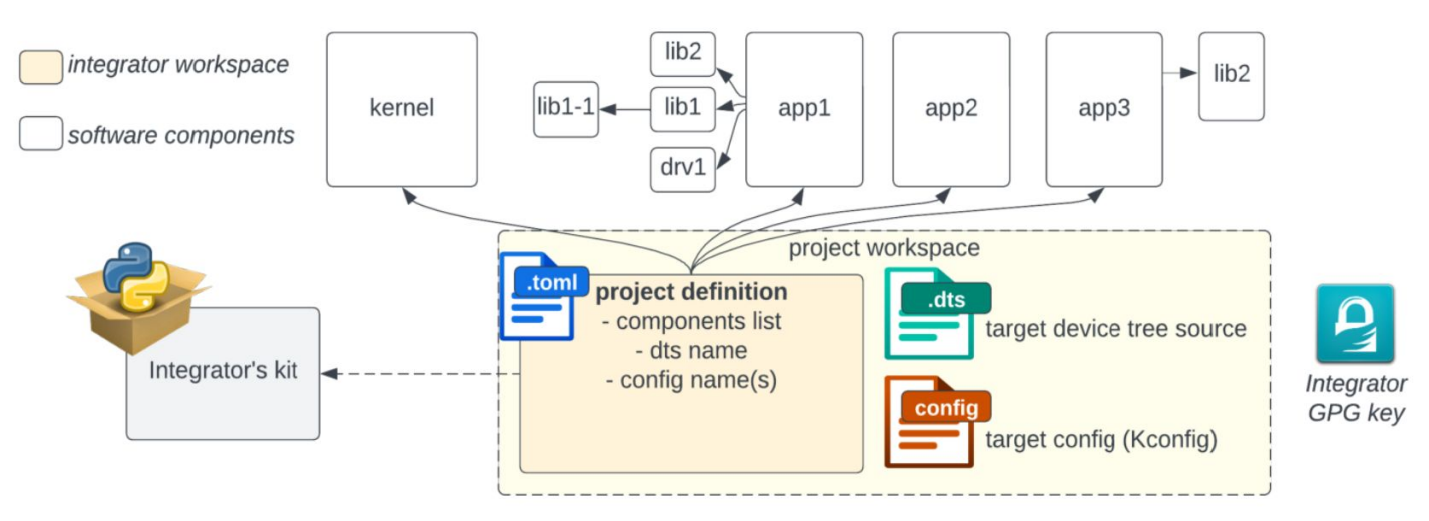
Software Development Kit

- Tools and resources needed to build user applications
  - Kernel UAPI
  - Application linker script template
- Delivered by integrator for a specific board/project
- Single root of trust for configurable parameters

- A SDK is compiled for each kernel to enforce consistency
  - Tailored to chosen kernel configuration
  - Supports C and Rust (as a start and encouraged)
  - Provides pkg-config files and Cargo local registry for UAPI
  - Provides tools: Kconfig, metadata generation, signature, etc.

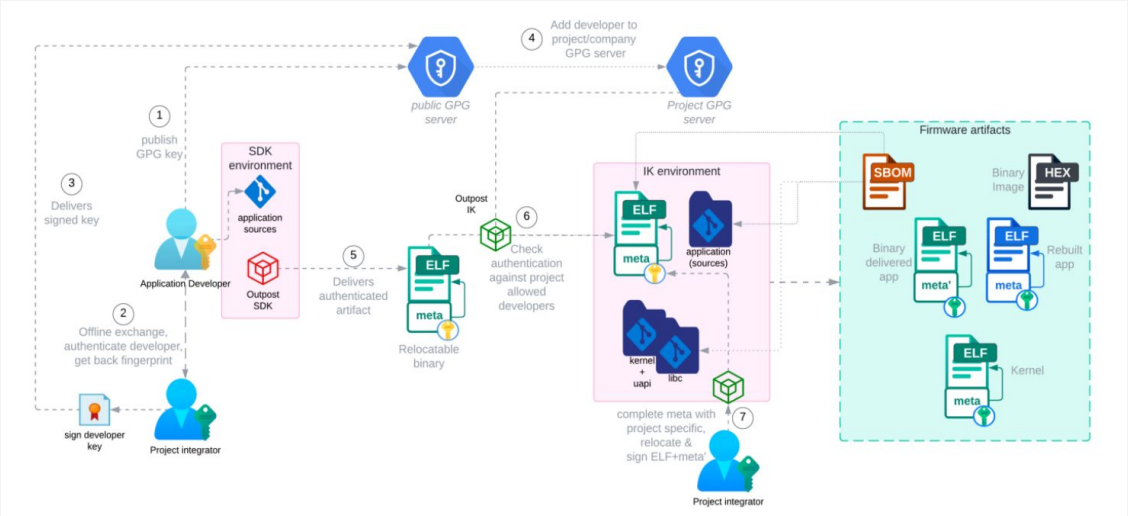
Integrator Kit

- Use the same kernel configuration as SDK
- Enforce consistency checks at build time
  - Resources ownership
  - Syscall Capabilities per application
- Relocate applications independently
  - No link step between applications
  - Increase isolation and allows licenses mix



Reproducible Build and Secure Development Chain

- Build system based on:
  - KConfig for configuration, Devicetree for board description.
  - Ninja build script with Meson and Cargo package support
- Handles GPG keys at each stage to enforce a trusted development chain.
- The Integrator Kit delivers a signed SBOM, build manifest and CPE, that:
  - Ensure traceability, authenticity for input artefacts
  - Allow automatic detection of vulnerability (CVE-xxx), COTS update, and licences management.



Comparison with other OSes

Comparison with other OSes:

- Where is Outpost in regards of main functionalities we were searching for?

Characteristic	Mbed	TockOS	FreeRTOS	Wookey	Zephyr
Highly secure and robust (isolation)	❌	~	❌	~	❌
Micro-Kernel	❌	✅	❌	✅	❌
Open-Source	✅	✅	~ (unsecure only)	✅	✅
SDK with C & Rust support	❌	✅	~	❌	~
Built components authentication	❌	❌	❌	❌	~ (west)
Integration Kit	❌	❌	❌	❌	❌
SBOM and SCPA generation[1]	❌	❌	❌	❌	❌

Notes:
1. SBOM: Software Bill Of Material; SCAP: Security Content Automation Protocol

Back to the future…

What’s next

What's next ?

Missing features

- logging and post-mortem checks
- Drivers framework & OSS drivers in SDK
- Cryptographic signature fully integrated in Integrator Kit
- Complete system upgrade mechanism
- Complete low-power management

Hardened product

- ARMv8-M Secure-boot integration
- Enforced in-depth fault injection counter-measures
- Integrator's Kit SCAP & CPE generator
- Integrator's Kit security compliance analysis tool ("aka product configuration security level")

OSS & licensing

- **Generic tooling** built for SDK and IK open-sourced (Apache-2.0)
  - here: svd2json, dts-utils 👀
- **SDK** and **IK** meta toolkit on the go for OSS (Apache 2.0)
- **Sentry kernel** cleanup and finalization on the go (Apache 2.0)
- **Userspace libraries & drivers** : discussions on Licensing (Apache vs BSD or dual-licensing model)

So, this is our journey

So this is our Journey:

- <u>Our departure:</u>
  - To address new fancy displays, we were searching for a secure OS for a MCU used as graphic coprocessor
  - Of our constraints, notably highly secure and auditable source code, our research has not yielded anything

⇒So we decided to develop one: **Outpost OS**

- <u>Our base camp today:</u>
  - Microkernel architecture with high security and robustness at core.
    - All applications and resources fully isolated.
  - A SDK supporting C and Rust memory-safe language.
    - A toolchain supporting independent development and secure integration processes.

MESON

python Package Index

- <u>Till our next destination:</u>
  - Support completely Cortex-M v7/8 and RISC-V architectures.
  - Once ready, apply Ledger open-sourcing philosophy to make it available for review and improvements

ARM Cortex-M

RISC-V

APACHE 2

This is the end!

We love questions 😎