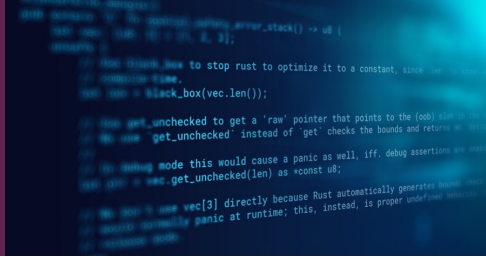
CHERIoT and Rust on ICENI™



Compile Time Guarantees with Hardware Enforced Protection

Rust and CHERI both address memory safety, but in different ways.

Rust enforces correctness at compile time. CHERIoT enforces memory safety at runtime in hardware.

Together in SCI Semiconductor's ICENI devices, they provide system wide protection that works for real embedded firmware.

Rust improves the reliability of new code through its ownership model and type system. Many defect classes become compile errors rather than runtime failures, including invalid lifetimes, unintended mutable aliasing, and common concurrency mistakes.

However, embedded systems still rely on low-level operations. Rust enables this through unsafe code, and mature embedded systems may represent significant investments in well-tested C/C++ components. Defects in these low-level layers can compromise the safety guarantees provided by the language, even on conventional hardware.

CHERIoT addresses this gap.

CHERI capabilities replace raw pointers with bounded, permission-carrying references enforced directly by the processor. Every memory access is checked against these bounds and permissions. Pointers cannot be forged, and invalid accesses trap deterministically instead of silently corrupting memory.

This enforcement applies across Rust safe code, Rust unsafe code, and non-Rust components.

CHERI

Prevention + Enforcement

Rust eliminates many classes of defects at build time

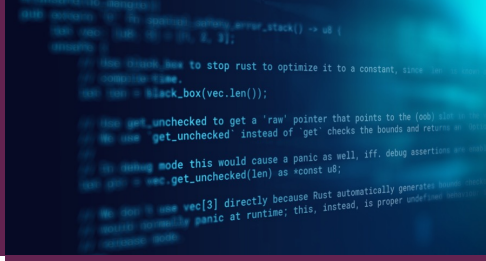
CHERIoT ensures that remaining defects cannot escalate into arbitrary memory corruption at runtime

Together they provide end to end protection across mixed language firmware, not just isolated Rust modules

Secure. Resilient. Uncompromising

SCI Semiconductor

CHERIoT and Rust on ICENI



Rust + CHERIoT Architecture

Complete rewrites of legacy firmware may not be practical. Rust promotes code reuse, which is great for rapid low-cost development but still requires the kinds of defence against supply-chain attacks that CHERIoT was designed to provide. On conventional systems, calling C from Rust weakens overall safety, since a memory error in C can compromise the entire process. **CHERIoT** enables a different model based on hardware-enforced compartmentalisation.

With CHERIoT, libraries can be placed in isolated compartments with access restricted to explicitly granted capabilities. These compartments cannot access unrelated memory or escalate their authority.
This enables teams to:

- Develop new functionality in Rust
- Retain proven C and C++ components
- Isolate higher risk libraries
- Contain the impact of defects

Capabilities also carry fine-grained permissions, such as bounds and read-only access. This enables least-privilege design at the object level, rather than relying solely on coarse process boundaries.

For third-party code, including external Rust crates, compartmentalisation reduces the trusted computing base by strictly constraining what that code can access, even in the presence of defects.

Built for real firmware teams

Write new features in Rust. Keep proven C and C++ components

Use hardware enforced isolation to ensure that defects remain contained

Rust improves correctness.
CHERIoT enforces safety.
ICENI enables both in a single embedded platform

WHY ICENI

SCI Semiconductor's ICENI devices implement the CHERIoT architecture for embedded systems, providing:

- Hardware enforced capability checking
- Strong isolation between firmware components
- Safe coexistence of Rust and C
- Practical least privilege designs

With **Rust** plus **CHERIoT** on **ICENI**, memory safety becomes a property of the platform, enabling incremental modernisation without weakening system security.

Further Reading

The CHERIoT Programmers' Guide: https://cheriot.org/book/
The CHERIoT Rust project: https://rust.cheriot.org

Secure. Resilient. Uncompromising

SCI Semiconductor