POLLUX and CASTOR: Enforcing Separation for Mixed-Language Environments

Andrés Sánchez, Flavio Toffalini, Trent Jaeger, Mathias Payer

The menace of unsafe external libraries in safe code

```
extern "C" { fn f_c(addr_c: i64); }
fn f_rust() {
  let vals: [i64;3] = [1,2,3];
  let cb: fn(&mut i64) = fp1; // Function pointer
  let addr_r = &vals as *const i64 as i64;
  unsafe{ f_c(addr_r) }
  (cb)(&mut vals[0]);
}
```

Fig 1. Call from Rust to external C function

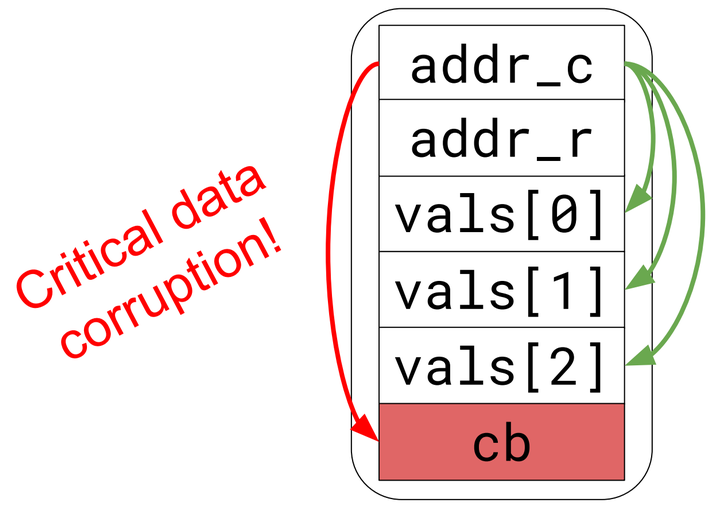
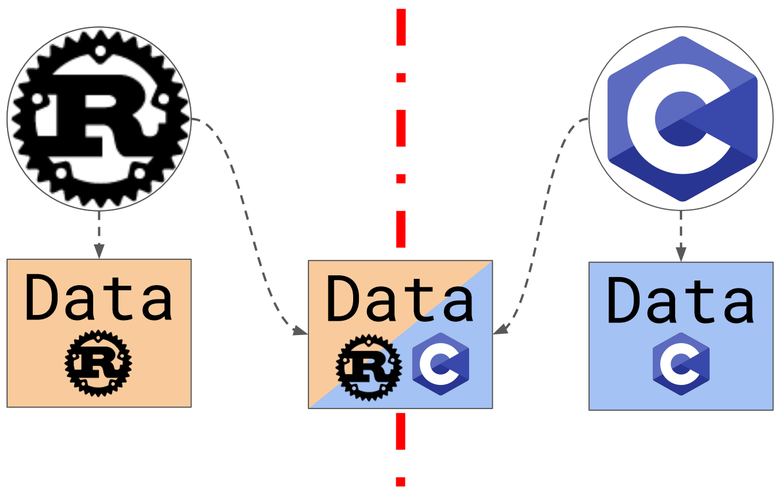


Fig 2. Stack-based cb corruption from C

Apply compartmentalization between languages data

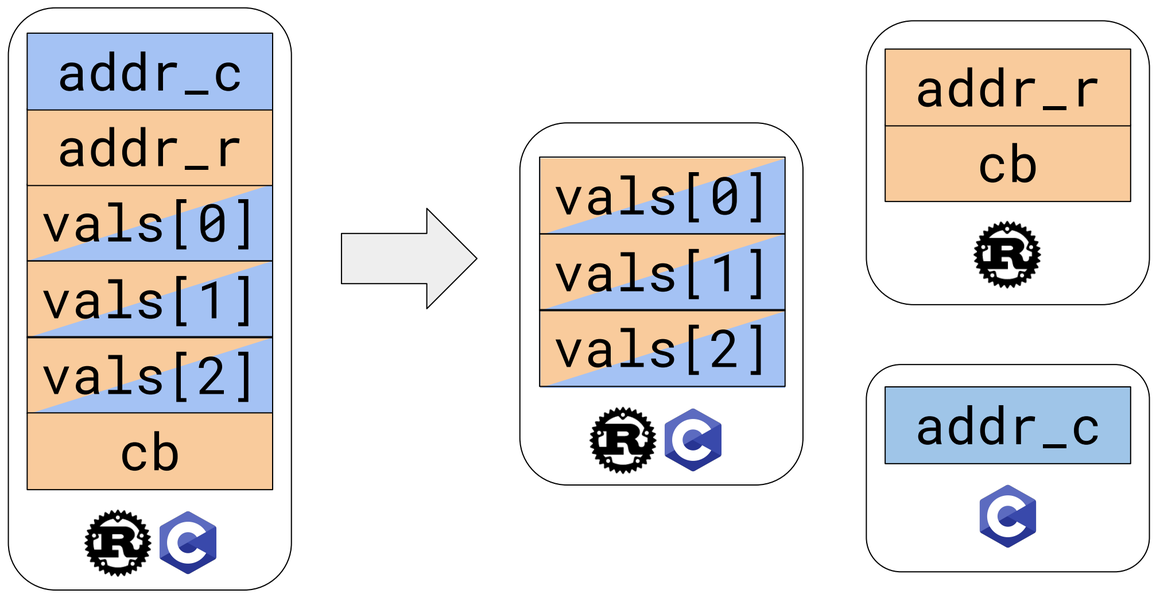
Data access is restricted to languages

(e.g., data shared between Rust and C)

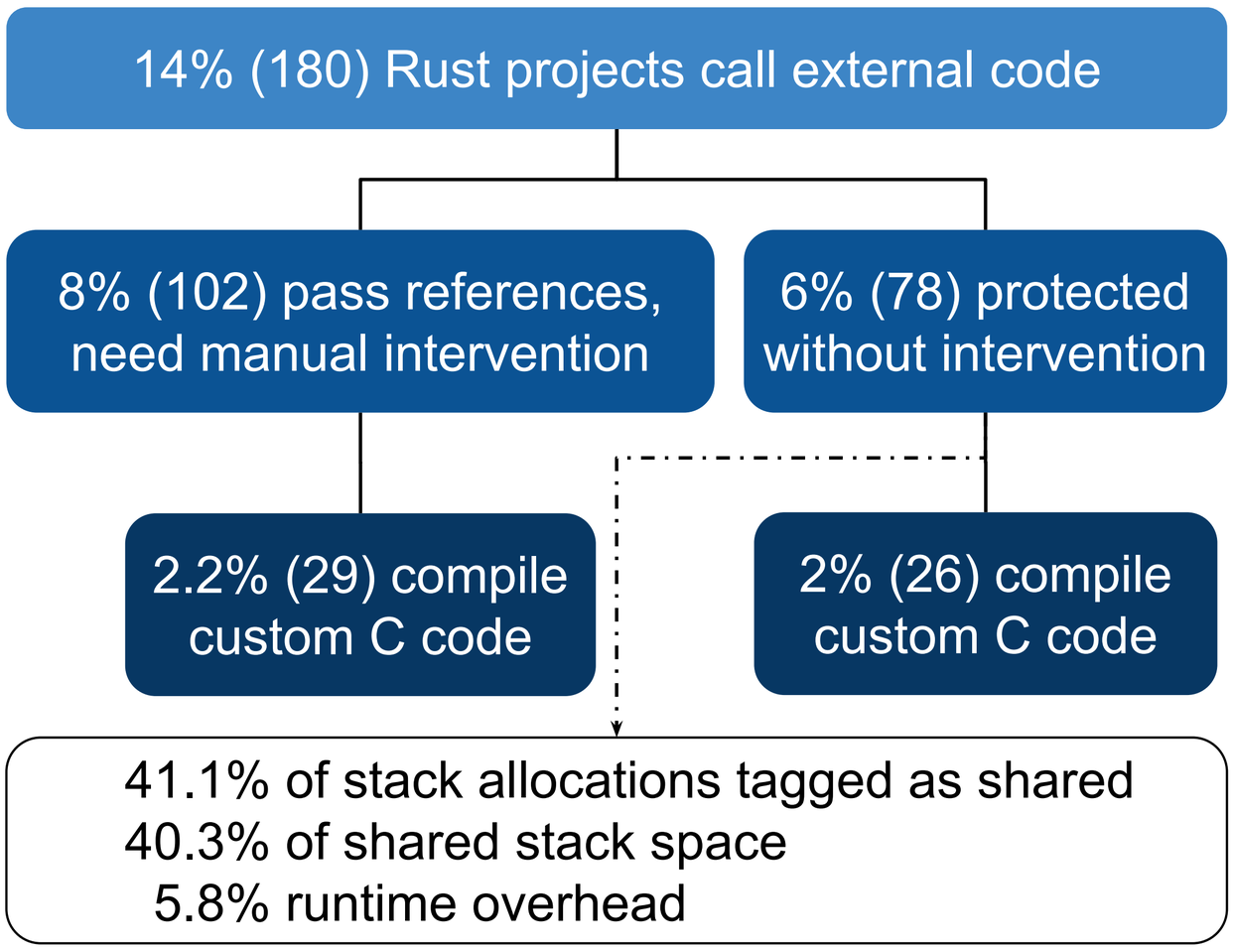


Stack address space is split in

two private stacks and a shared stack



Evaluation



Impact

- Compiler warns about unsafe patterns
- Developer informed about critical shared data (e.g., code pointers)
- No need to analyze unsafe code
- Leverages existing hardware isolation extensions (e.g., MPK, MTE, TEEs)
- Applies to other languages than C/Rust
- Challenge: performance cost due to frequent inter-language transitions