Secure Rusty Systems

Who are we?

Deian Stefan

- Faculty working on Security & PL
- Work on secure systems research
    - Sandboxing, Wasm, JITs, Web
- Co-founder of Cubist
    - Key management platform

Why Rust?

- Use it to build real systems (Cubist)
- We're doing some research touch Rust
- Annoyed by some of the koolaid

Office hours: M 5PM

Evan Johnson

- PhD student working on Security/Pl/Systems
- Works on practical verification for systems
    - Verified sandboxing
    - Verified Rust embedded OS
- On the job market: needs to learn to teach

Why Rust?

- Verifying C/C++ is a nightmare
- Rust community cares about safety
- It's fun to write :)

Office hours: W 3pm

Who are you?

Class structure

- Lectures
    - Meet 2x week
    - Read ~1 paper/meeting, we discuss paper
    - **Goal:** You understand the paper (and background) so we can talk about the interesting parts
- Project
    - Groups of 2-3
    - **Goal:** hardcore research or implementation
        - Work on a research project that can end in at least a workshop publication
        - Work on an implementation project that lands in a real system
    - Every week (M+F): Post project updates (real detail, not “still working on X”)
- Course site + slack
    - Every reading will be posted here: https://plsyssec.github.io/cse291k-fall24/
    - Slack for discussions and announcements: #cse291k-fall24
    - No canvas, piazza, etc.

Project ideas

- Verify a driver written in Rust in Tock/Linux
- Exploit rustc type-unsoundness bugs (or show they’re not exploitable?)
- Extend WaVe (verified sandboxing runtime) with a WASI 2.0 proposal
- Use WaVe to build a verified serverless platform
- Extend a Rust verifier (Flux) with a useful feature
- Try to verify inline assembly in a Rust program (hard!)
- Take existing specifications (e.g., from Flux) and generate a fuzzing harness/input validation

Project ideas

- Rust – C/C++ binding layer: what kinds of bugs are introduced at this layer
- Where does Rust fail?
    - Rust is not good for everything. Fundamentally, where is it bad?
- Rust supply chain analysis and attacks
    - Extend cargo scan with more complex analyses
    - What can we really do about unsafe code?
    - How should we audit proc macros?
- Rust bug landscape
- Where are people misusing Rust
    - E.g., if you’re compiling untrusted code code rust and think you’re getting isolation: you are not
- Port a serious driver to Rust

Is Rust the “right” systems language?

Go read “The Rise of Worse is Better”

Rust: a brief history

- 2006: Graydon Hoare starts work on Rust after his building’s elevator segfaults
- 2009: Mozilla adopts Rust as an official project with ~1 dozen engineers
- 2010: Rust transitions from Ocaml compiler to self-hosted llvm-based compiler

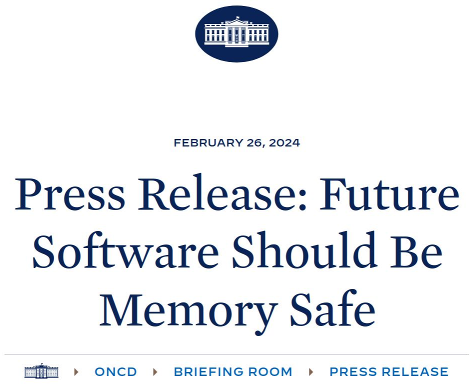
Rust: the early days

```
io fn f(chan[int] c)
{
 type t = tup(int,int,int);

 // Allocate an exterior.
 let @t x = tup(1,2,3);

 // Signal parent that we've allocated an exterior.
 c <| 1;

 while (true) {
   // spin waiting for the parent to kill us.
   log "child waiting to die...";
   c <| 1;
 }
}
```

Rust: path to release

- 2011: first class modules and actor-based concurrency removed
- 2012: classes, interfaces, and oo-style inheritance -> trait system
- 2013: gc -> ownership system

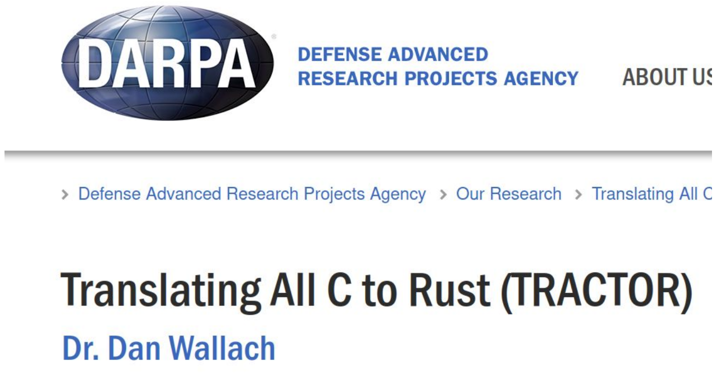
Rust (the modern era)



FEBRUARY 26, 2024

Press Release: Future Software Should Be Memory Safe

ONCD › BRIEFING ROOM › PRESS RELEASE

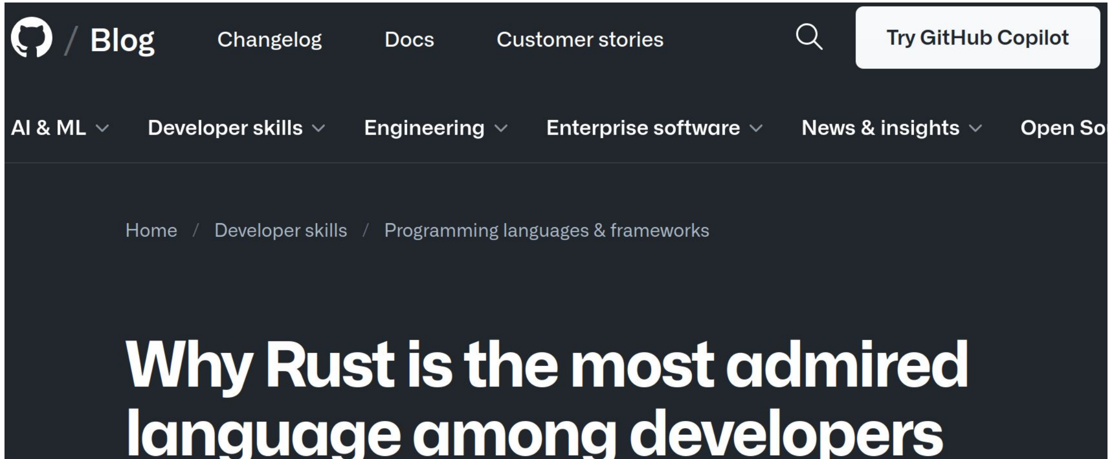


DARPA DEFENSE ADVANCED RESEARCH PROJECTS AGENCY ABOUT US

› Defense Advanced Research Projects Agency › Our Research › Translating All C

Translating All C to Rust (TRACTOR)

Dr. Dan Wallach



Home / Developer skills / Programming languages & frameworks

Why Rust is the most admired language among developers

Rust is a good systems language

- Good performance (no garbage collection!)
- Memory safety
- No data races
- No null references
- Tooling that actually works!

Rust is a good systems language, but…

- Not designed for embedded systems
  - What happens when Rust code interacts with hardware?
  - How do Rust features affect binary size?
- Rust can’t always guarantee safety when interoperating with other languages
  - What happens when Rust makes a reference out of a C pointer?
- “Fearless concurrency” only covers data races, not general race conditions
  - Can still have deadlocks
- Sometimes you need to do unsafe stuff…

For next time

- Read “Engineering the Servo Web Browser Engine using Rust” by Brian Anderson et al.
- Start looking for a project group
- Think about aspects of Rust security/Rusty systems/Rust formal methods that you might be interested in (we have some free paper slots)