Contribution ID: **215** Type: **not specified**

Barriers to in-tree Rust

Thursday, August 27, 2020 7:35 AM (30 minutes)

What would it take to have in-tree support for writing kernel code in Rust? What should Kbuild integration look like? What APIs should be the initial priorities to expose in Rust? Let's figure out if any other other questions remain (e.g., can we safely link against GCC-built kernels, and do we need to) about how to get in-tree support for Rust.

Rust is a systems programming language that is particularly well-suited to the kernel: it is a "better C" in a way that matches the kernel's needs (no GC, kernel-style OO, etc.) Rust can also be of significant benefit for security - safe Rust protects against entire classes of vulnerabilities such as use-after-frees, buffer overflows, and use of uninitialized memory, which form a large percent of kernel vulnerabilities.

(This session will not be an intro to the Rust language. See last year's Linux Security Summit NA talk "Linux Kernel Modules in Rust" video / slides for an overview of Rust for kernel hackers and a demo of Rust modules.)

I agree to abide by the anti-harassment policy

I agree

Primary authors: BAUBLITZ, John; DESAULNIERS, Nick (Google); GAYNOR, Alex; THOMAS, Geoffrey; TRIPLETT, Josh; OJEDA, Miguel

Presenters: BAUBLITZ, John; DESAULNIERS, Nick (Google); GAYNOR, Alex; THOMAS, Geoffrey; TRIPLETT, Josh; OJEDA, Miguel

Session Classification: LLVM MC

Track Classification: LLVM MC