Contribution ID: **407** Type: **not specified**

Rex and its integration with Rust-for-Linux

TL;DR We propose to present the Rex project (Rust-based kernel extension) and discuss its integration with Rust for Linux.

Rex is a Rust-based kernel extension framework (https://github.com/rex-rs/rex). It offers similar safety guarantees as eBPF. Different from eBPF, which verifies the safety of extension code via an in-kernel verifier, Rex builds its safety guarantees atop the language-based safety of Rust, combined with light-weighted runtime protection. Specifically, Rex enforces extension programs to be written in a safe subset of Rust; the Rex compiler performs safety checks and generates native code directly. This approach avoids the overly restricted verification requirements (e.g., program complexity constraints) and the resulting arcane verification errors in eBPF. Rex implements its own kernel crate that offers a safe kernel interface that wraps existing eBPF interface with safe Rust wrappers and bindings. At the same time, Rex employs a lightweight runtime that implements graceful Rust panic handling with resource cleanups, kernel stack usage checks, and extension program termination.

We plan to first go over the design of Rex, and then collect the feedback and answers to the following questions:
- How does the Rust-for-Linux community think about the idea of a new kernel extension mechanism that sits at the middle ground of Rust kernel modules and eBPF?
- Is there any aspect of Rex that is also useful for Rust-for-Linux and how can we contribute?
- Does the trust we put on the Rust toolchain make sense and how can we potentially make it more trustworthy?

Primary authors: WILLIAMS, Dan (Virginia Tech); JIA, Jinghao (University of Illinois Urbana-Champaign); QIN, Ruowen; XU, Tianyin (University of Illinois at Urbana-Champaign)

Presenters: JIA, Jinghao (University of Illinois Urbana-Champaign); QIN, Ruowen

Session Classification: Rust MC

Track Classification: Rust MC