Building Secure Systems Using RISC-V and Rust

Arun Thomas
arun.thomas@acm.org

RISC-V Workshop Zurich
Talk Overview

- How do you build a **secure computer system**?

- My take:
  - **RISC-V Hardware**: Openness, Simplicity, and Flexibility
  - **Rust Software**: Safety, Performance, and Productivity
How do you build a secure computer system?
Hint: Antivirus is not the answer.

(Ask Project Zero.)
Securing Computer Systems

Security spans all abstraction layers

- Application
- Programming Language
- Operating System
- Hardware

A flaw in any layer can compromise system security requirements.
How Are We Doing?

Well, it turns out security is HARD.
Building Secure Systems in the 21st Century: RISC-V Hardware and Rust Software
RISC-V: Opening Up the Hardware Ecosystem
RISC-V: Opening Up the Hardware Ecosystem

• Having a simple, open, customizable ISA is a big deal

  • No more homegrown, custom, bespoke ISAs
RISC-V: Opening Up the Hardware Ecosystem

• Having a simple, open, customizable ISA is a big deal
  • No more homegrown, custom, bespoke ISAs
• Several open RISC-V implementation exist
  • LowRISC, PULP, Chips Alliance, OpenHW, …
RISC-V: Opening Up the Hardware Ecosystem

- Having a simple, open, customizable ISA is a **big deal**
  - No more homegrown, custom, bespoke ISAs
- Several open RISC-V implementation exist
  - LowRISC, PULP, Chips Alliance, OpenHW, …
- Industry and academia can collaborate more effectively
  - **No NDAs needed.** Just clone a git repo
- Critical for making progress on solving hard security problems
The RISC-V community has an **historic opportunity to “do security right”** from the get-go with the benefit of up-to-date knowledge.

-RISC-V Foundation
RISC-V as a Platform for Security Research
RISC-V as a Platform for Security Research

- Several groups exploring new security ideas on RISC-V
  - Tagged Architectures (LowRISC, PIPE)
  - Hardware enforced capabilities (CHERI)
  - Formally verified RISC-V implementations (Kami)
  - Secure enclaves (Keystone, MI6)
RISC-V as a Platform for Security Research

- Several groups exploring new security ideas on RISC-V
  - Tagged Architectures (LowRISC, PIPE)
  - Hardware enforced capabilities (CHERI)
  - Formally verified RISC-V implementations (Kami)
  - Secure enclaves (Keystone, MI6)
- More interesting work in the security session (post-break)
On Systems Programming

[A] systems programmer has seen the **terrors** of the world and understood the **intrinsic horror** of existence

-James Mickens, *The Night Watch*
The Challenges of Systems Software
The Challenges of Systems Software

• Systems software is **critical to enforcing security**

  • Operating systems, hypervisors, runtimes, drivers, firmware, browsers, web servers, …
The Challenges of Systems Software

• Systems software is **critical to enforcing security**
  
  • Operating systems, hypervisors, runtimes, drivers, firmware, browsers, web servers, ...
  
• Usually written in C or C++ for performance reasons
  
  • BUT C and C++ are **not memory-safe**
The Challenges of Systems Software

• Systems software is **critical to enforcing security**

  • Operating systems, hypervisors, runtimes, drivers, firmware, browsers, web servers, ...

• Usually written in C or C++ for performance reasons

  • BUT C and C++ are **not memory-safe**

• **Memory corruption vulnerabilities** abound (and are exploited)

  • Microsoft study estimates 70% of security bugs are due to memory safety issues
It turns out programming languages have evolved in the last 50 years.
Rust is a safe, performant systems programming language.
Rust and Mozilla
Rust and Mozilla

- Rust was originally created by Mozilla Research
  - Initial use case: Developing experimental Servo browser engine
Rust and Mozilla

- Rust was originally created by Mozilla Research
  - Initial use case: Developing experimental Servo browser engine
- Mozilla began shipping Rust components in Firefox 48 in 2016
  - **Oxidization** is Mozilla’s term for “Rusting out” components
Rust and Mozilla

• Rust was originally created by Mozilla Research
  • Initial use case: Developing experimental Servo browser engine
• Mozilla began shipping Rust components in Firefox 48 in 2016
  • Oxidization is Mozilla’s term for “Rusting out” components
• Rust code has improved Firefox’s security and performance
  • Security: Safe parsers (e.g., New MP4 metadata parser replaced libstagefright)
  • Performance: New parallel CSS engine speeds up page loads
“Systems Programming Can Have Nice Things”
“Systems Programming Can Have Nice Things”

- Performance on par with C/C++
“Systems Programming Can Have Nice Things”

- Performance on par with C/C++
- Memory safety without overhead of garbage collection
  - Programmer has fine-grained control over memory
“Systems Programming Can Have Nice Things”

- Performance on par with C/C++
- Memory safety without overhead of garbage collection
  - Programmer has fine-grained control over memory
- Provides thread safety ("Fearless Concurrency")
  - No concurrency bugs associated with multi-threaded code
“Systems Programming Can Have Nice Things”

- Performance on par with C/C++
- Memory safety without overhead of garbage collection
  - Programmer has fine-grained control over memory
- Provides thread safety (“Fearless Concurrency”)
  - No concurrency bugs associated with multi-threaded code
- Rust’s type system enforces memory/thread safety at compile time
“Systems Programming Can Have Nice Things”

- Performance on par with C/C++
- Memory safety without overhead of garbage collection
  - Programmer has fine-grained control over memory
- Provides thread safety (“Fearless Concurrency”)
  - No concurrency bugs associated with multi-threaded code
- Rust’s type system enforces memory/thread safety at compile time
- Excellent package manager and productive developer environment
Gaining Popularity in Systems Community
Gaining Popularity in Systems Community

- Rust-based OSes: Tock, Redox, intermezzOS, …
Gaining Popularity in Systems Community

• Rust-based OSes: Tock, Redox, intermezzOS, …

• Amazon and Google have developed Rust-based VMMs (crosvm, Firecracker)
Gaining Popularity in Systems Community

• Rust-based OSes: Tock, Redox, intermezzOS, …

• Amazon and Google have developed Rust-based VMMs (crosvm, Firecracker)

• Coreboot developers exploring Rust rewrite (oreboot - “fork of coreboot, with C removed”)
Gaining Popularity in Systems Community

- Rust-based OSes: Tock, Redox, intermezzOS, …

- Amazon and Google have developed Rust-based VMMs (crosvm, Firecracker)

- Coreboot developers exploring Rust rewrite (oreboot - “fork of coreboot, with C removed”)

- Projects exploring Rust OS components for Linux kernel, FreeBSD kernel, seL4, and Fuchsia OS
Rust/RISC-V
Rust/RISC-V (RV32)

- RISC-V 32-bit support ([rust-embedded/wg#36], [rust-lang/rust#52787], [rust-lang/rust#53822]) released in Rust 1.30
Rust/RISC-V (RV64)

- RISC-V 64-bit support (rust-embedded/wg#218, rust-lang/rust#58406) released in Rust 1.34
Bare Metal Rust for RISC-V
Bare Metal Rust for RISC-V

• There is a handy quickstart template for bare-metal Rust RISC-V development

  • https://github.com/riscv-rust/riscv-rust-quickstart
Bare Metal Rust for RISC-V

• There is a handy quickstart template for bare-metal Rust RISC-V development
  • https://github.com/riscv-rust/riscv-rust-quickstart

• Includes several example projects
  • E.g., “hello world” to UART
Bare Metal Rust for RISC-V

- There is a handy quickstart template for bare-metal Rust RISC-V development
- Includes several example projects
  - E.g., “hello world” to UART
- See my Oxidize ’19 talk for more details
What About OS Targets?
What About OS Targets?

- Linux/RV64 is up next
  - Important for Linux distributions and Firefox port
  - Help Wanted
What About OS Targets?

- Linux/RV64 is up next
  - Important for Linux distributions and Firefox port
  - Help Wanted
- Tock microcontroller OS port in progress
  - Tock is written in Rust and provides a secure foundation for IoT devices
Summary

- RISC-V and Rust provide a strong foundation for building secure systems by combining:
  - Safety, performance, and productivity of Rust
  - Openness, simplicity, and flexibility of RISC-V