Rust, TEE, and OS (3)

Mingshen

Recap 1

- ***What is TEE?***
  - Security properties: data confidentiality, data integrity, code integrity, etc.
  - TEE implementations
  - Applications of TEE
- ***Why need an OS in TEE?***
  - TEE Development Model
  - Security vs Usability
- ***Does language matter?***
  - Memory safety
  - How does Rust guarantee the memory safety?
  - Why use Rust for TEE development?

Recap 2

- ***Rust's Ownership, Borrowing, and Lifetime***
  - Memory Safety: Mutable + Aliasing

Owership	`T`	"owned"
Exclusive Access	`&mut T`	"mutable borrow"
Shared Access	`&T`	"read-only borrow"

- ***Rust's Syntax***
  - basic, types, impl, trait
  - borrowing, lifetimes, closures, generics
  - use and mod
  - attributes
  - error handling

Recap: Homework

- Writing an OS in Rust: bare bones and interrupts
- Run sample code
- Rewrite on RISC-V
- Writing an OS in Rust: https://os.phil-opp.com
- The Adventures of OS: Making a RISC-V Operating System using Rust: http://osblog.stephenmarz.com/

Homework: Status Checked @ Sep 25

rustOS链接提交 ☆ 最近保存 21:35

插入 | 常规 | .0 | 默认字体 | 10

D16

	A	B	C
1	姓名	github链接	备注
2	邹泽桦	https://github.com/HuaHuaY/Rust_TEE_OS	
3	鲍志远	https://github.com/bzy-debug/blog_os	
4	彭宇科	https://github.com/sdww0/rustOS	
5	周翊澄	https://github.com/Zhou-Yicheng/rust_os	
6	张开源	https://github.com/imporoutco586/RUST-OS	
7	孙永康	https://github.com/Konata-CG/Rust_TEE-OS	

Most of the people has done the first two tasks.

Sadly, I didn't see the RISC-V rewrite.

Homework

The Adventures of OS: Making a RISC-V Operating System using Rust

Running title: RISC-V OS using Rust

26 September 2019

Purpose

RISC-V ("risk five") and the Rust programming language both start with an R, so naturally they fit together. In this blog, we will write an operating system targeting the RISC-V architecture in Rust (mostly). If you have a sane development environment for RISC-V, you can skip the setup parts right to bootloading. Otherwise, it'll be fairly difficult to get started.

This tutorial will progressively build an operating system from start to something that you can show your friends or parents -- if they're significantly young enough. Since I'm rather new at this I decided to make it a "feature" that each blog post will mature as time goes on. More details will be added and some will be clarified. I look forward to hearing from you!

The Road Ahead...

+ Chapter 0: Setup and pre-requisites (UPDATED 2020: Rust out-of-the-box!)
+ Chapter 1: Taking control of RISC-V
+ Chapter 2: Communications
+ Chapter 3.1: Page-grained memory allocation
+ Chapter 3.2: Memory Management Unit
+ Chapter 4: Handling interrupts and traps
+ Chapter 5: External interrupts
+ Chapter 6: Process memory
+ Chapter 7: System calls
+ Chapter 8: Starting a process
+ Chapter 9: Block driver
+ Chapter 10: Filesystems
+ Chapter 11: Userspace Processes

Next week
- Read the first three chapters.
- Implement by your self.

Two weeks later
- Come back to the original homework to rewrite into RISC-V.

Homework: Status Checked @ 11AM Oct 10

	姓名	GitHub 链接
✔	邹泽桦	https://github.com/HuaHuaY/Rust_TEE_OS
	鲍志远	https://github.com/bzy-debug/blog_os
	彭宇科	https://github.com/sdww0/rustOS
	周翊澄	https://github.com/Zhou-Yicheng/rust_os
✔	张开源	https://github.com/imporoutco586/RUST-OS
	孙永康	https://github.com/Konata-CG/Rust_TEE-OS
	王辰宇	https://github.com/BruceW959/Rust-TEE

Homework: Highlights

	姓名	GitHub 链接
✓	邹泽桦	https://github.com/HuaHuaY/Rust_TEE_OS

- cargo-make
- uart_16550

✓	张开源	https://github.com/imporoutco586/RUST-OS

- OpenSBI

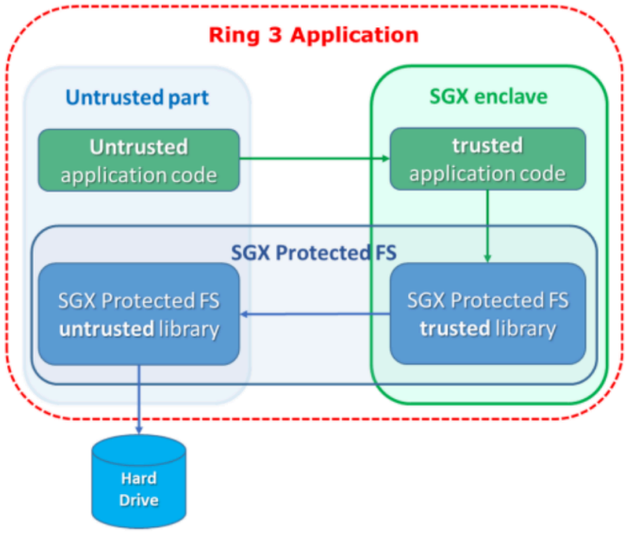
Projects

- Protected File System for TEE
- Side Channel Resistant Crypto Library for TEE
- `RustPython` in TEE
- `wasmtime` in TEE

Projects

- *Protected File System for TEE*
  - Rewrite Intel SGX Protected File System Library in Rust
  - Provide interfaces similar with fs module in Rust std.
  - Provide C APIs for compatibility.
  - https://github.com/intel/linux-sgx/tree/master/sdk/protected_fs

Language	Files	Lines	Code	Comments	Blanks
C Header	5	563	277	188	98
C++	9	3622	2370	624	628
Makefile	2	159	70	61	28
Total	16	4344	2717	873	754



Intel Protected File System Layout

Projects

- ***Side Channel Resistant Crypto Library for TEE***
  - Make existing Rust crypto library side channel resistant in TEE.
  - Learn and summarize existing side channel attacks in the crypto library.
  - Summarize method to mitigate the side channel attacks in the source code level (e.g., cache flush, branches, etc.)
  - Make several crypto algorithms (ECC, AES, etc.) side channel resistant.
  - https://github.com/RustCrypto
  - https://www.bsi.bund.de/EN/Topics/Cryptography/SideChannelResistance/side_channel_resistance_node.html

Projects

- ***wasmtime in TEE***
  - Port the wasmtime WebAssembly runtime to Intel SGX or ARM TrustZone (OP-TEE OS).
  - Understand the implementation of WebAssembly runtime.
  - Support a bare-mental WebAssembly interpreter.
  - Study the WebAssembly System Interface (WASI) and design "safe" WASI interfaces for TEE.
  - Finally, wastime in TEE. (potentially with JIT/AOT support)
  - https://github.com/bytecodealliance/wasmtime

Projects

- ***RustPython in TEE***
  - Port the RustPython Python interpreter to Intel SGX or ARM TrustZone (OP-TEE OS)
  - Understand the implementation of a Python interpreter.
  - Customize the interpreter for TEE to support some typical machine learning libraries/applications in TEE.
  - https://github.com/RustPython/RustPython

Projects

- Protected File System for TEE
- Side Channel Resistant Crypto Library for TEE
- `RustPython` in TEE
- `wasmtime` in TEE