



Rust

Memory Thinking

Version 2

Dmitry Vostokov
Software Diagnostics Services

Prerequisites

- **Rust** development experience

and (optional)

- Basic memory dump analysis (Windows or Linux)

Training Goals

- ⦿ Review fundamentals of **Rust**
- ⦿ Review **Rust** specifics from a memory analysis perspective
- ⦿ Use WinDbg and GDB for learning **Rust** internals

Warning

This is not a Rust course for C or C++ developers. But knowledge of C and C++ helps.

Training Principles

- Talk only about what I can show
- Lots of pictures
- Lots of examples
- Original content and examples

Schedule

- ⦿ `let sessions: Vec<Session> =
 vec![Session::default(); 10];`
- ⦿ `assert_eq!(sessions.len(), 10);`
- ⦿ `assert!(sessions.capacity() >= 10);`

Training Idea

- Similar C and C++ courses for Windows and Linux
- System programming language role
- Memory dump analysis training courses
- Debugging training courses

General Rust Aspects

- ◉ Philosophy of unsafe pointers
- ◉ Philosophy of values
- ◉ Rust: a Copernican revolution
- ◉ Rust philosophy of values
- ◉ Ownership
- ◉ Lifetime
- ◉ Rust philosophy of pointers
- ◉ References
- ◉ Static, stack, and heap memory
- ◉ Memory and pointers
- ◉ Basic types
- ◉ Size and alignment
- ◉ Conversion
- ◉ Tuples and tuple-like structs
- ◉ Structs
- ◉ Source code and symbols
- ◉ Free functions
- ◉ Function pointers and references
- ◉ Associated functions
- ◉ Type-associated functions
- ◉ Trait functions and objects
- ◉ Trait object memory layout
- ◉ Dynamic dispatch
- ◉ Constructors and destructors
- ◉ Clone and Copy
- ◉ Parameters by value
- ◉ Parameters by reference/pointer
- ◉ Closures and captures
- ◉ Pinning
- ◉ Smart pointers
- ◉ Return values
- ◉ Trait object parameters
- ◉ Enums
- ◉ Const values

What We Do Not Cover

We promise to include these topics in the third edition

Linux **Rust** Aspects

- ◉ Necessary x64 and A64 disassembly
- ◉ Parameter passing
- ◉ Implicit parameter
- ◉ Useful GDB commands

Windows **Rust** Aspects

- Necessary x64 disassembly
- Parameter passing
- Implicit parameter
- Useful WinDbg commands

Why Rust?

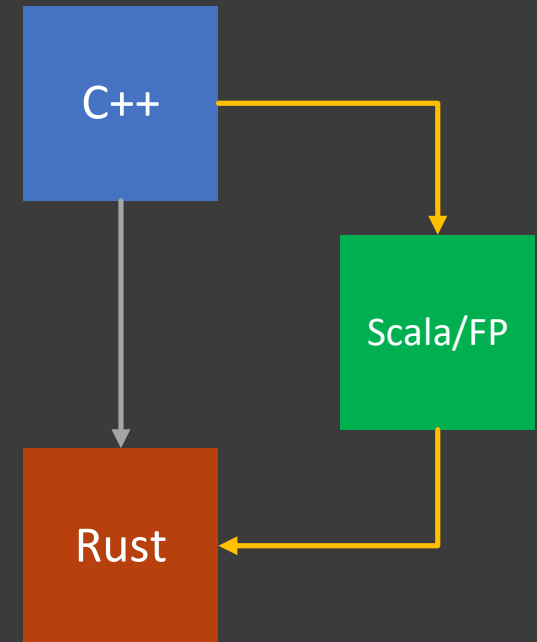
- ⦿ Better Rust developer and maintainer

Unsafe now / Rust Runner, 2049

- ⦿ Interfacing
- ⦿ Malware analysis
- ⦿ Vulnerability analysis and exploitation
- ⦿ Reversing
- ⦿ Diagnostics
- ⦿ Low-level debugging
- ⦿ OS Monitoring
- ⦿ Memory forensics
- ⦿ Crash and hang analysis
- ⦿ Secure coding
- ⦿ Static code analysis
- ⦿ Trace and log analysis

My Genealogy of Rust

- C from 1987 and C++ from 1989 ([Old CV](#))
- C++98/03/STL from 2001
- C++/STL semantics (2001-2003)
- First heard about Rust in 2015
- C++11/14 from 2016
- C++17 from 2017
- C++20 from 2023
- Scala/FP from 2020
- Functional programming from 2020
- Rust from 2022 (Windows API book)
- Rust included in Linux API book (2023)
- Rust included in Windows memory dump analysis book (2023)
- Rust included in Windows Debugging book (2024)
- Rust included in Linux Debugging book (2024)
- Rust Windows memory dump analysis book (2024)



Rust Mastery Process

Coding



Mental Compiling



Thought Process

◎ Rust, C, C++

Memory

◎ Rust, Scala/FP

Functions

◎ Rust, Python

Data