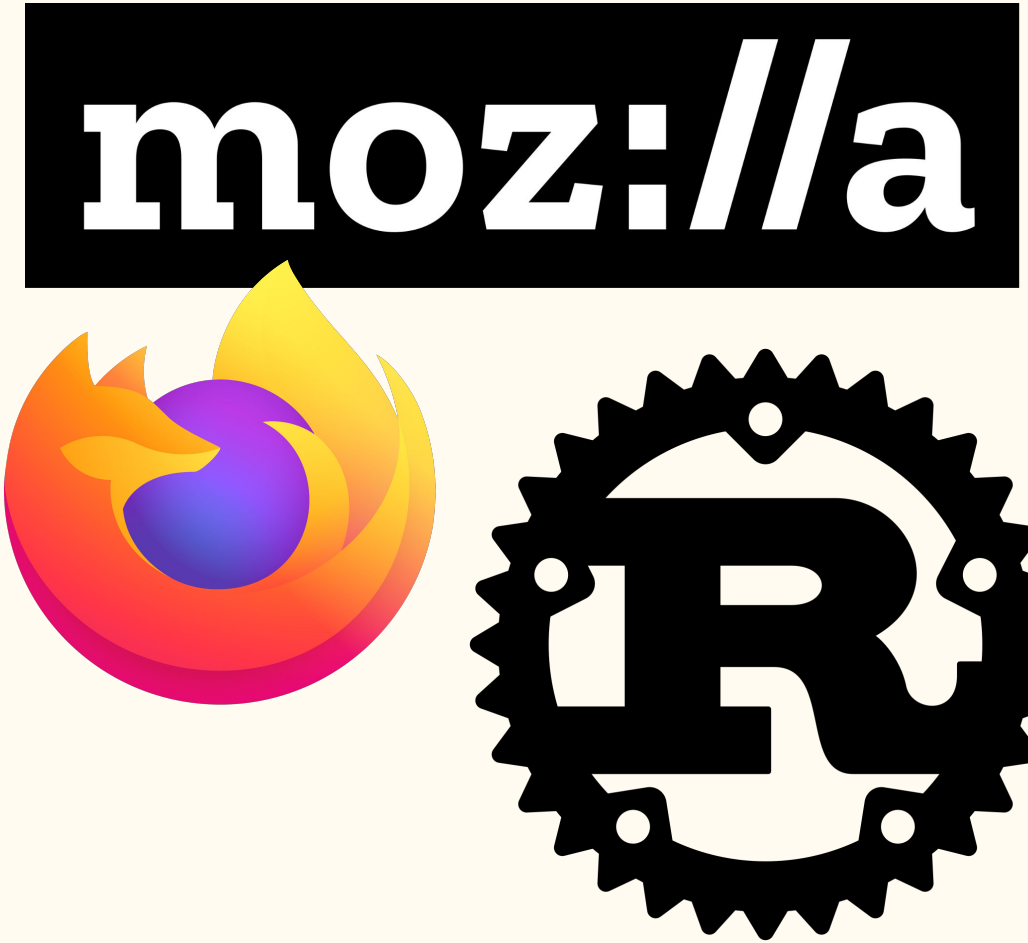
The Rust Programming Language

Evan Duffield, Katie Haney-Osborne, Dylan Morris

Introduction

Origins of Rust

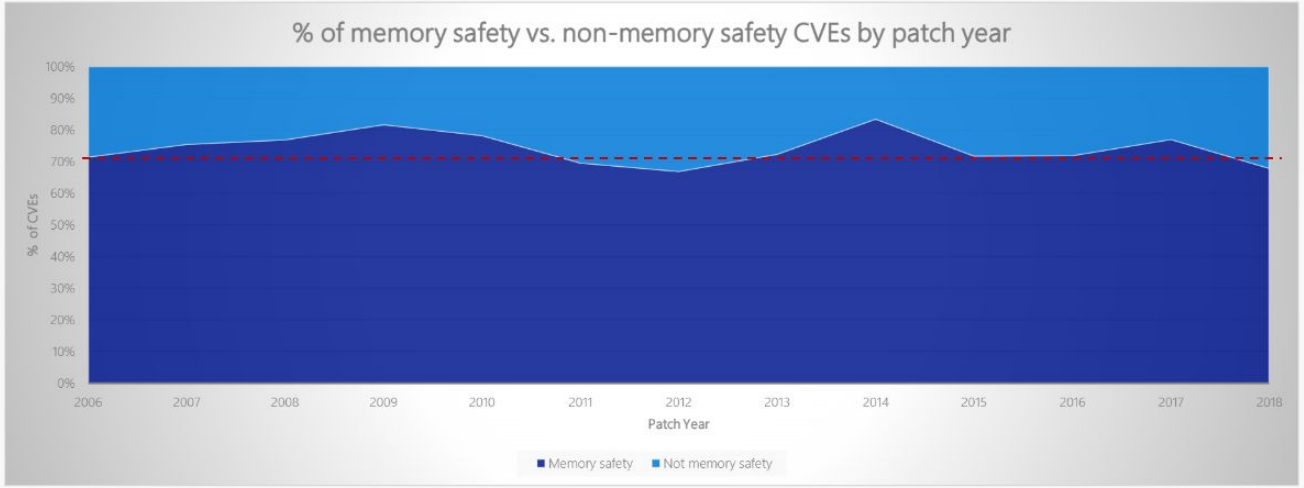
- Started by a Mozilla employee as a personal project in 2006, was officially sponsored in 2009.
- Memory errors from using C++ was one of the biggest sources of bugs during the development of Firefox.
- Rust was developed with features to minimize these errors without sacrificing the control and performance that is mandatory for systems programming.



Why Memory Safety Matters

Memory safety issues remain dominant

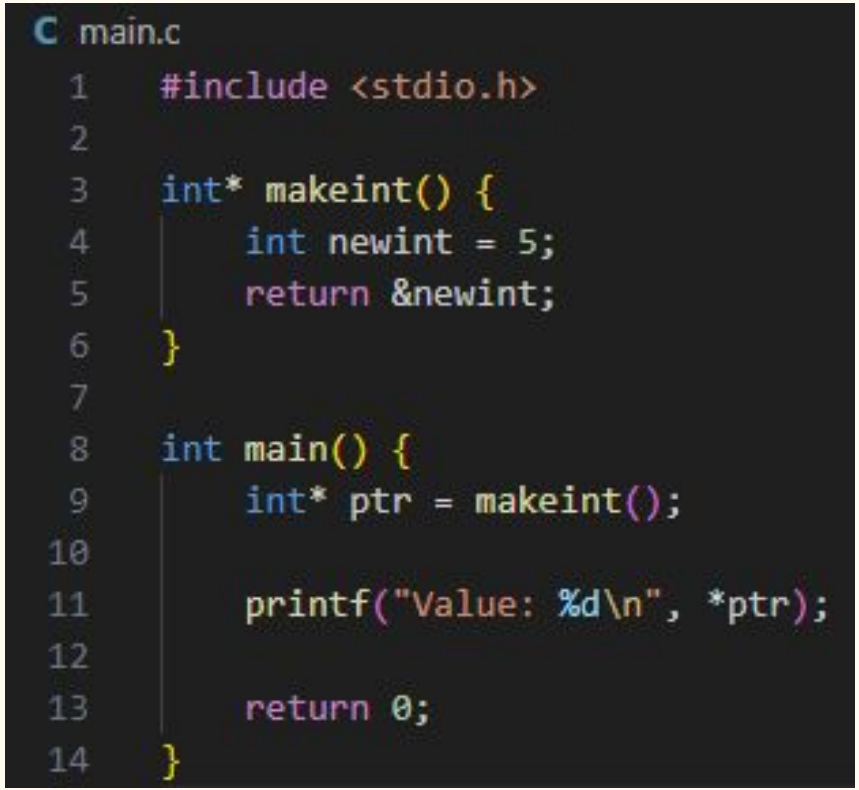
We closely study the root cause trends of vulnerabilities & search for patterns



~70% of the vulnerabilities addressed through a security update each year continue to be memory safety issues

An Example of Memory Error

- Systems programming languages like C/C++ pass-by-value, we have to use pointers that contain the memory address of a variable to access it instead of a copy.
- However, it is common for programmers to have pointers that eventually lead to data that has been destroyed. These almost always cause the program to crash if no preventions are in place.



How Rust Fixes This

- Rust has an Ownership system that allows it to efficiently manage the memory of the program without losing performance.
- Every value in the program has a single ‘owner’ (whatever scope it is being created/used in).
- Rust can imitate pass-by-reference by giving ownership of a value to another part of the program. This makes sure any data the program could potentially point to is protected from deallocation.

main.rs

```rust
fn makeint() -> i32 {
    let newint = 5;
    newint
}

fn main() {
    let newint = makeint();

    println!("Value: {}", newint);
}
```

What Sets Rust Apart?

Is Rust Object-Oriented?

- Objects contain data and behavior: Yes
  - `structs` and `enums` have data, and `impl` blocks provide methods on `structs` and `enums`
  - Not called objects, but have the same functionality
- Encapsulation that hides implementation details: Yes
  - Marking a `struct` as `pub` makes it usable by other code, but the fields within the struct are private
  - Methods can be implemented to update values within the struct

```
pub struct AveragedCollection {
    list: Vec<i32>,
    average: f64,
}
```

Listing 17-1: An `AveragedCollection` struct that maintains a list of integers and the average of the items in the collection

Is Rust Object-Oriented?

- Inheritance? No
  - Inheritance is at risk of sharing more code than necessary
  - Subclasses shouldn't always share all characteristics of the parent class, but do so with inheritance
- Reusing code like inheritance? Yes
  - Rust uses 'trait' objects rather than inheritance

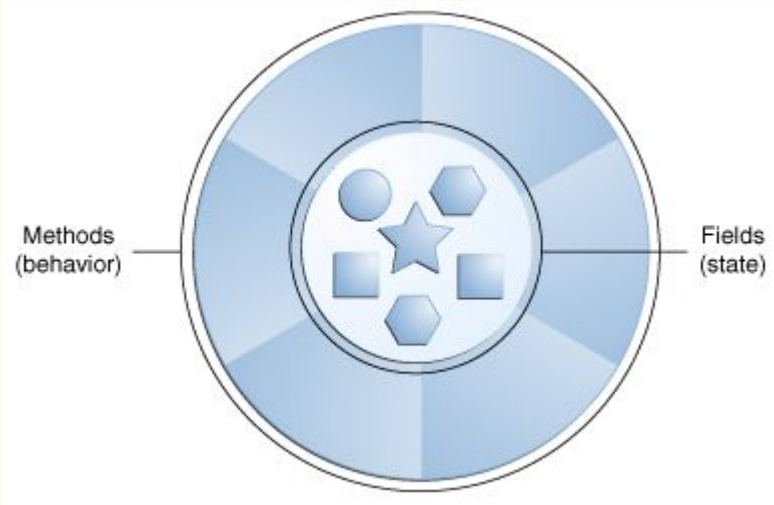
```
pub trait Draw {
    fn draw(&self);
}
```

Listing 17-3: Definition of the `Draw` trait

```
pub struct Screen {
    pub components: Vec<Box<dyn Draw>>,
}
```

Listing 17-4: Definition of the `Screen` struct with a `components` field holding a vector of trait objects that implement the `Draw` trait

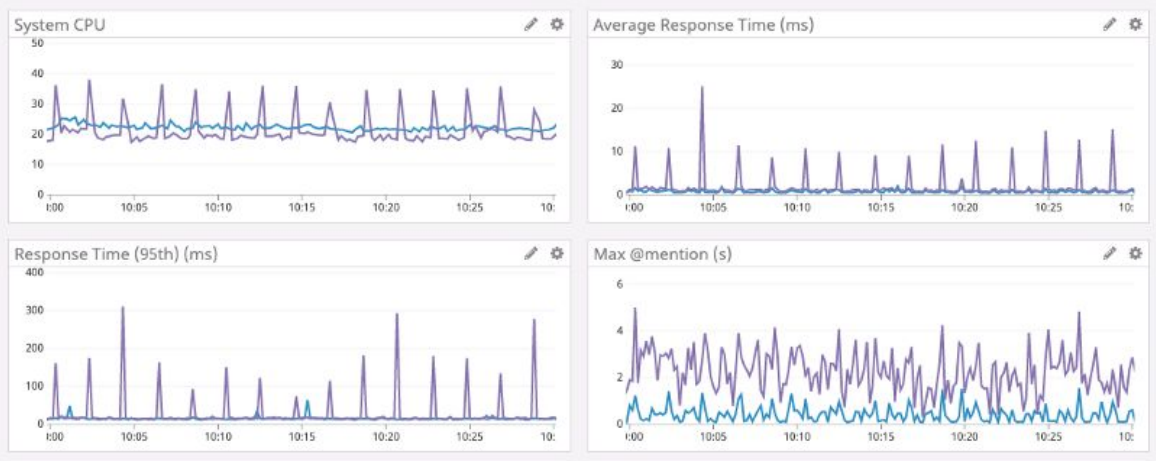
- Java has object-oriented programming using classes
- Objects have a state and behavior



- Go is not exactly an object-oriented language, similar to Rust
- Go has types and methods and allows for oop-style programming, there is no type hierarchy
    - Not identical to subclassing
- One of the strengths of Go

Rust compared to Go: Garbage Collection

- In Go, memory is not immediately freed
  - Garbage collection runs every so often to find any memory that has no references, then frees it
- Rust has no garbage collection and is very memory-efficient and fast
- Rust has memory “ownership”, keeping track of who can read or write to memory

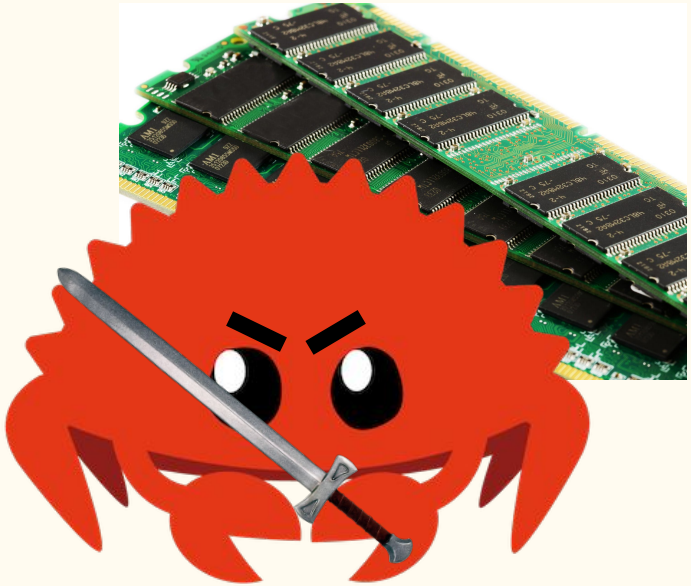


Go: Purple
Rust: Blue

Can Rust Replace C/C++?

Memory protection

- Rust's ownership system provides excellent security for computer memory, working to prevent memory corruption
- Since ownership works by telling the user when they've made a decision that puts memory in danger and putting it on them to fix it, it trains users to value memory safety in all programming languages
- The lack of protection in languages like C and C++ means that programmers are fully responsible for making sure memory is properly managed, which leads to tons of difficult bugs and unsafe memory

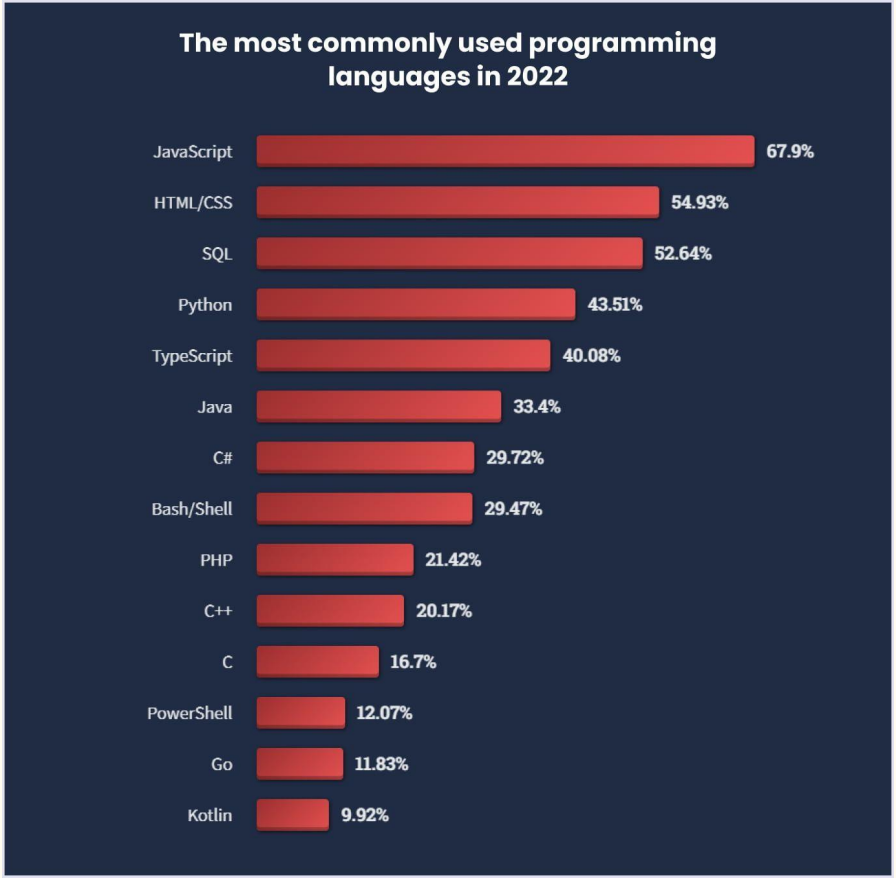


Proven to work

- Rust is already used by high profile applications like Firefox and Linux, becoming a well known name in the industry
- Many people are improving the language and working to integrate it with interfaces like Visual Studio
- Being developed by the programmers behind Firefox, it has a strong foundation from experienced industry programmers

An immobile foundation

- The vast majority of the programming languages used today are decades old- it's very hard for new languages to break in
- Replacing old programming languages means mass reprogramming or replacing old applications
- Old languages can be updated to meet new safety standards, though it's not as effective as developing a language with that safety from the ground up
- Essentially, the culture and design of programming doesn't accommodate large scale change in language use



Rust in Operating Systems

- Linux developers are working towards incorporating Rust as a second programming language for the Linux kernel, with safety being a primary factor in the decision
- Rust could potentially be used to run already developed C code in the kernel, allowing the ownership system to be used in maintaining and running it
- In addition, developers of Android have been considering Rust as a potential addition to the programming languages used by the mobile operating system, which would mean Rust being used by one of the biggest tech companies

Conclusion

Conclusion

- Rust has many features that improve the quality of life of the programmer, has the ability to make systems programming bearable again.
- Wide area of potential application compared to other new languages, is actively in use.
- Still in its infancy, has to compete with massive libraries written in C/C++.
- Google is working on Carbon, a language similar to Rust that allows direct interfacing with C++ programs.

References

- Gallardo, Raymond, et al. The Java Tutorial: A Short Course on the Basics. Addison-Wesley, 2015.
- Howarth, Jesse. “Why Discord is switching from Go to Rust.” Discord, 4 February 2020, https://discord.com/blog/why-discord-is-switching-from-go-to-rust. Accessed 22 January 2024.
- Klabnik, Steve, and Carol Nichols. The Rust Programming Language, 2nd Edition. No Starch Press, 2023.
- Miller, Matt. “Trends, challenges, and strategic shifts in the software vulnerability mitigation landscape.” GitHub, https://github.com/Microsoft/MSRC-Security-Research/blob/master/presentations/2019_02_BlueHatIL/2019_01%20-%20BlueHatIL%20-%20Trends%2C%20challenge%2C%20and%20shifts%20in%20software%20vulnerability%20mitigation.pdf. Accessed 22 January 2024.
- “Frequently Asked Questions (FAQ).” The Go Programming Language, https://go.dev/doc/faq. Accessed 22 January 2024.
- https://www.codica.com/blog/top-programming-languages-2023/ (Graph in slide 15)

Rustaceans