RUSTBELT:

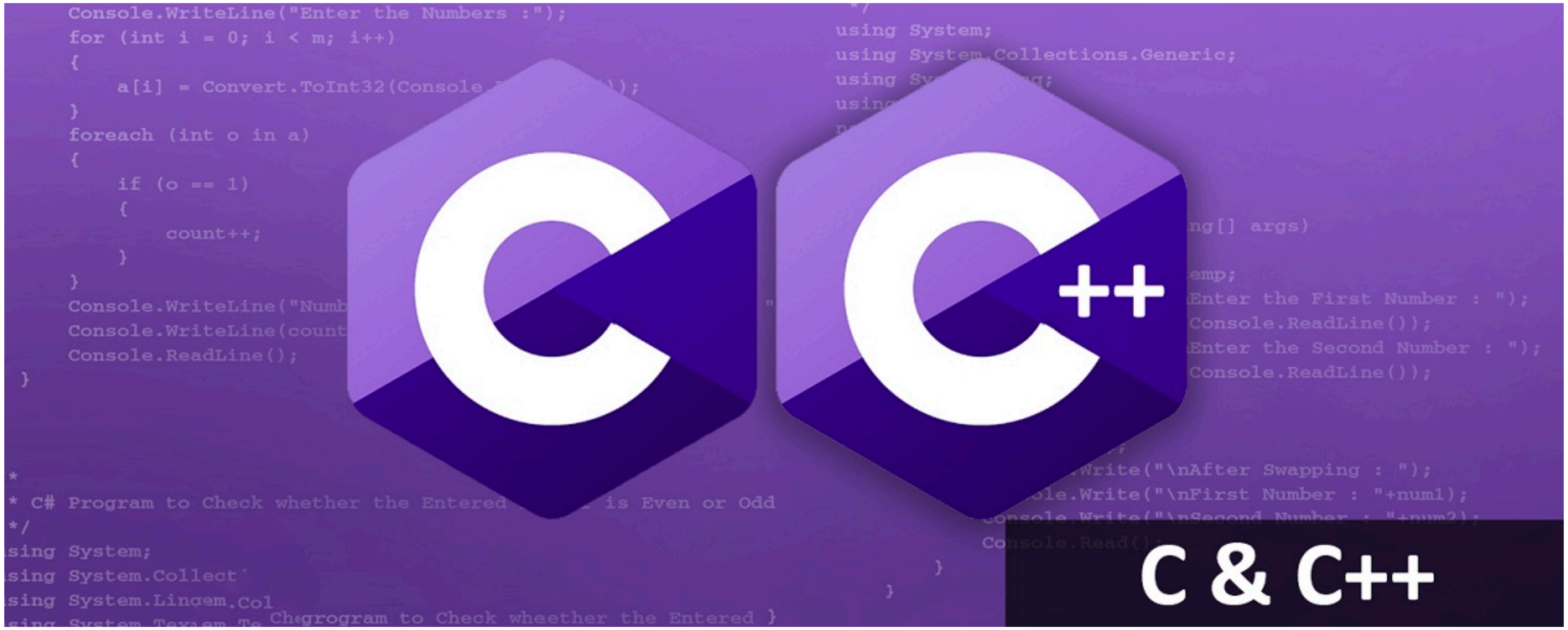
LOGICAL FOUNDATIONS FOR THE FUTURE OF SAFE SYSTEMS PROGRAMMING

Derek Dreyer
MPI for Software Systems

ENS Paris-Saclay Visit
November 2022

A Longstanding Problem

- Many core systems applications require low-level control over memory/resources
- Such applications are typically written in

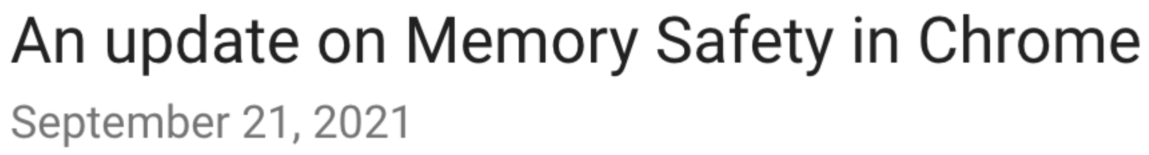


A Longstanding Problem

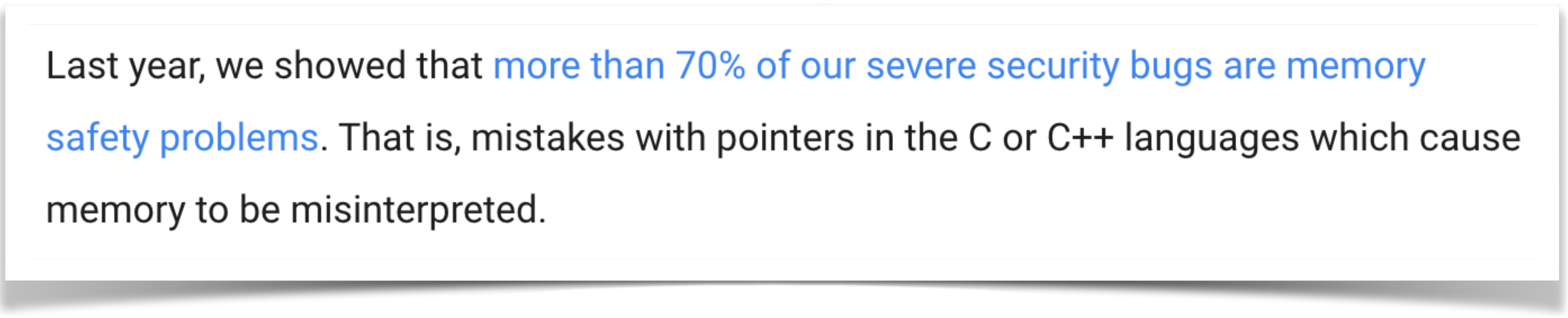
- Many core systems applications require low-level control over memory/resources
- Such applications are typically written in

UNSAFE!

from Google Security Blog

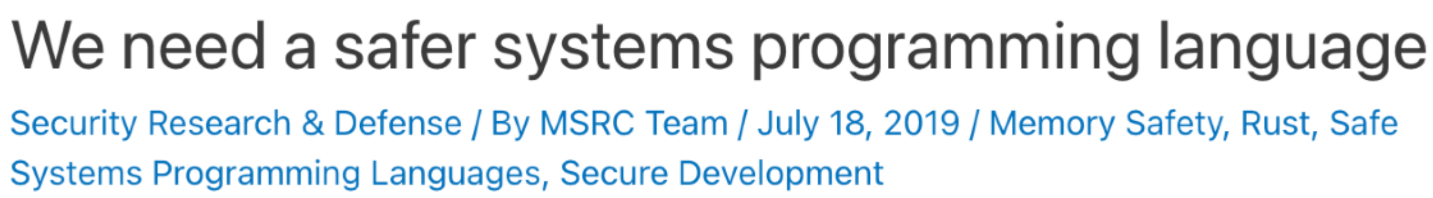

An update on Memory Safety in Chrome

September 21, 2021

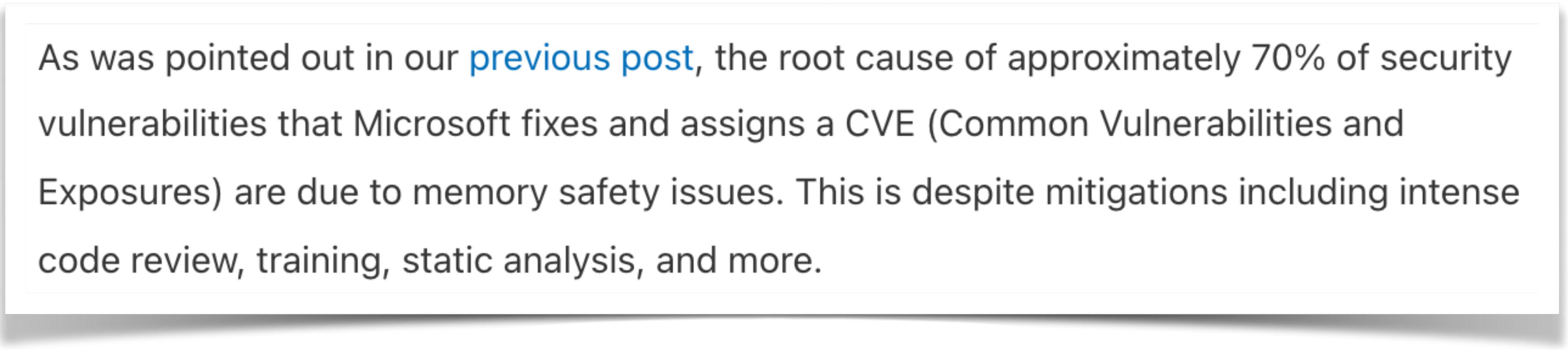

Last year, we showed that more than 70% of our severe security bugs are memory safety problems. That is, mistakes with pointers in the C or C++ languages which cause memory to be misinterpreted.

---

from Microsoft Security Response Center


We need a safer systems programming language

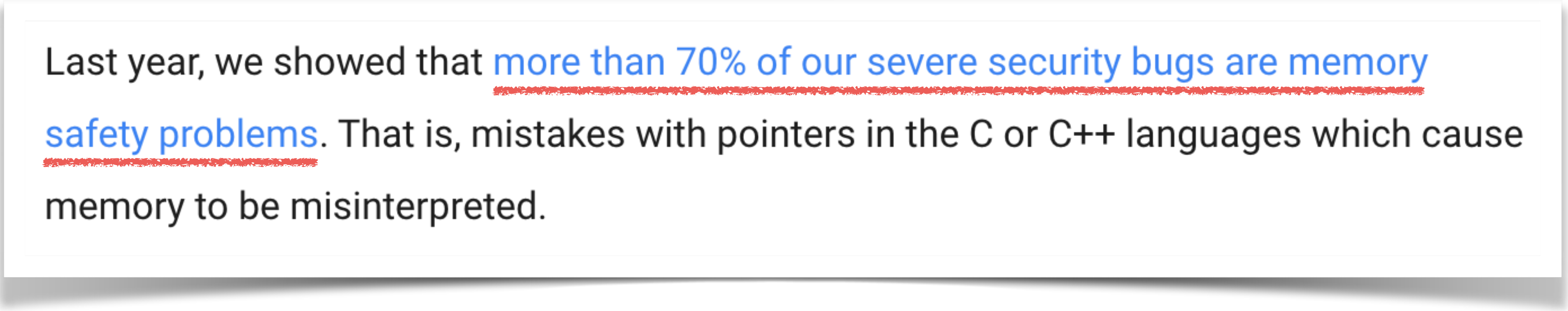
Security Research & Defense / By MSRC Team / July 18, 2019 / Memory Safety, Rust, Safe Systems Programming Languages, Secure Development


As was pointed out in our previous post, the root cause of approximately 70% of security vulnerabilities that Microsoft fixes and assigns a CVE (Common Vulnerabilities and Exposures) are due to memory safety issues. This is despite mitigations including intense code review, training, static analysis, and more.

from Google Security Blog

An update on Memory Safety in Chrome

September 21, 2021



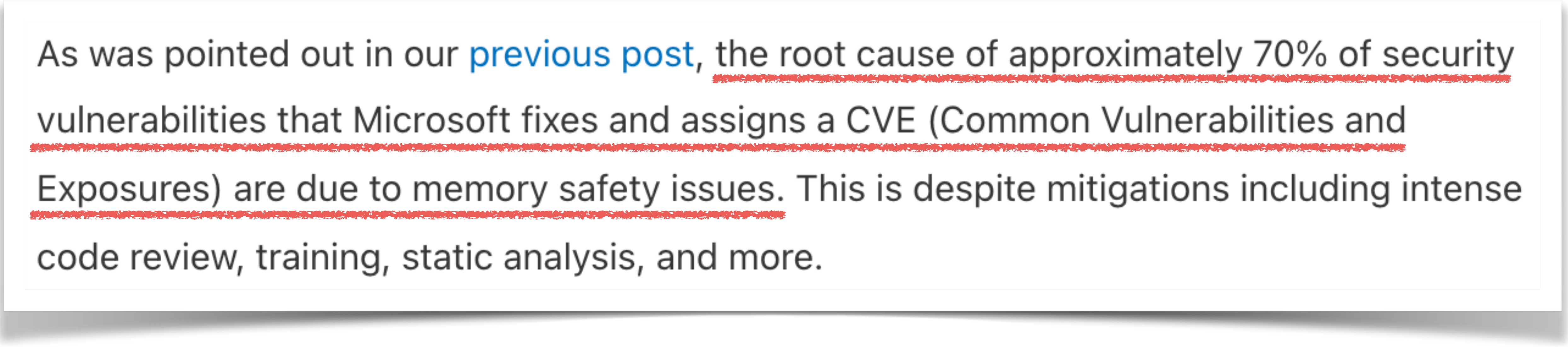
Last year, we showed that more than 70% of our severe security bugs are memory safety problems. That is, mistakes with pointers in the C or C++ languages which cause memory to be misinterpreted.

---

from Microsoft Security Response Center

We need a safer systems programming language

Security Research & Defense / By MSRC Team / July 18, 2019 / Memory Safety, Rust, Safe Systems Programming Languages, Secure Development

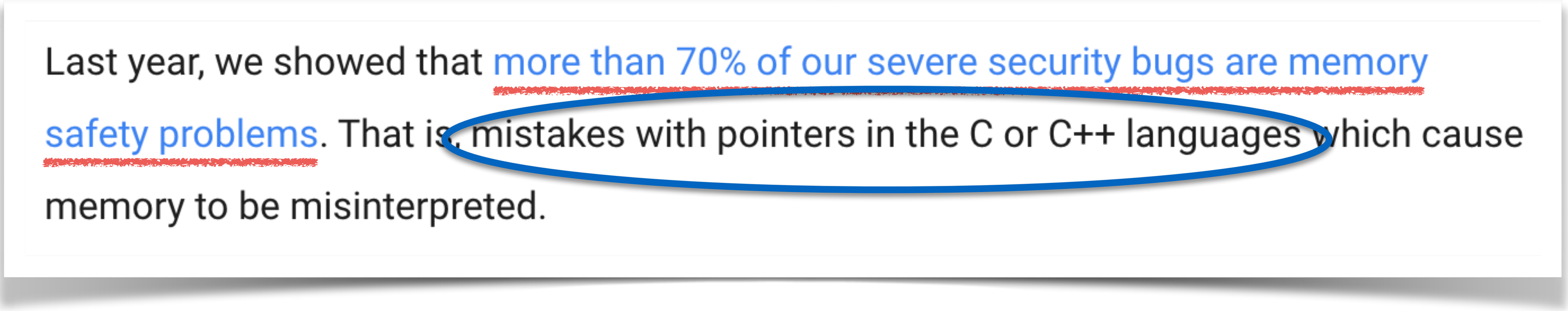


As was pointed out in our previous post, the root cause of approximately 70% of security vulnerabilities that Microsoft fixes and assigns a CVE (Common Vulnerabilities and Exposures) are due to memory safety issues. This is despite mitigations including intense code review, training, static analysis, and more.

from Google Security Blog

An update on Memory Safety in Chrome

September 21, 2021



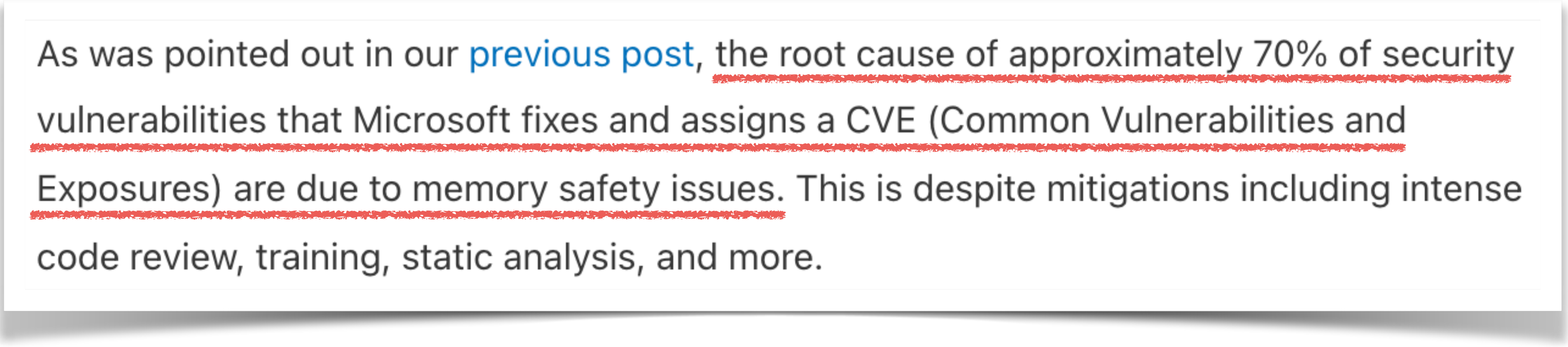
Last year, we showed that more than 70% of our severe security bugs are memory safety problems. That is, mistakes with pointers in the C or C++ languages which cause memory to be misinterpreted.

---

from Microsoft Security Response Center

We need a safer systems programming language

Security Research & Defense / By MSRC Team / July 18, 2019 / Memory Safety, Rust, Safe Systems Programming Languages, Secure Development

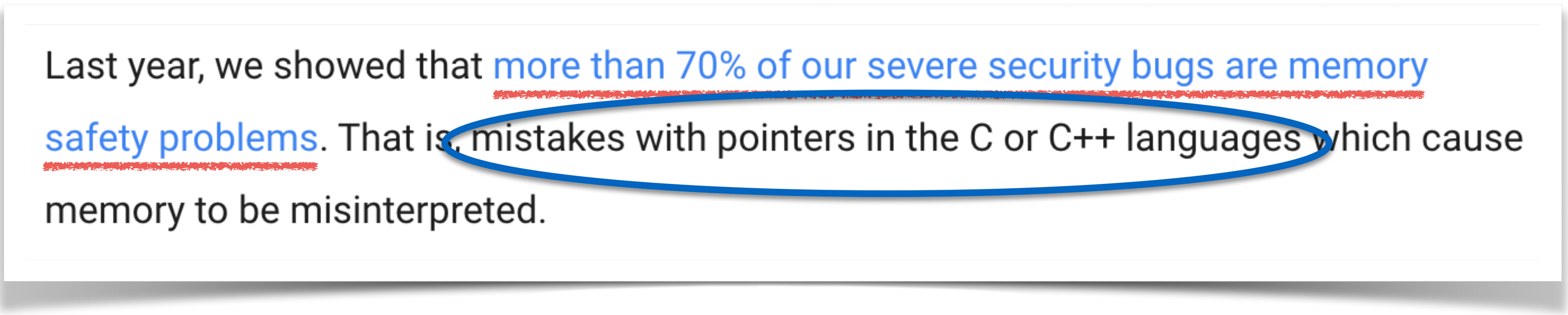


As was pointed out in our previous post, the root cause of approximately 70% of security vulnerabilities that Microsoft fixes and assigns a CVE (Common Vulnerabilities and Exposures) are due to memory safety issues. This is despite mitigations including intense code review, training, static analysis, and more.

from Google Security Blog

An update on Memory Safety in Chrome

September 21, 2021



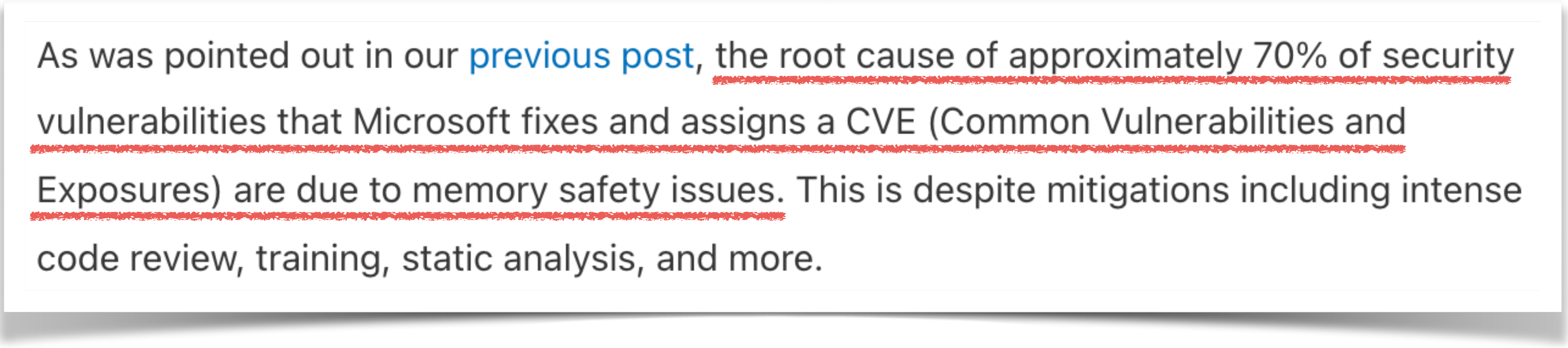
Last year, we showed that more than 70% of our severe security bugs are memory safety problems. That is, mistakes with pointers in the C or C++ languages which cause memory to be misinterpreted.

---

from Microsoft Security Response Center

We need a safer systems programming language

Security Research & Defense / By MSRC Team / July 18, 2019 / Memory Safety, Rust, Safe Systems Programming Languages, Secure Development



As was pointed out in our previous post, the root cause of approximately 70% of security vulnerabilities that Microsoft fixes and assigns a CVE (Common Vulnerabilities and Exposures) are due to memory safety issues. This is despite mitigations including intense code review, training, static analysis, and more.

Rust:

The Future of Safe Systems Programming?

In development since 2010, with 1.0 release in 2015

- Mozilla used Rust to build Servo, a next-gen browser engine, later incorporated into Firefox

Rust is the only "systems PL" to provide…

- Low-level control à la modern C++
- Strong safety guarantees
- Industrial development and backing

Many major companies using Rust in production

- In 2021, the **Rust Foundation** was formed, incl. Amazon, Google, Huawei, Meta, Microsoft, Mozilla

Rust:

The Future of Safe Systems Programming?

In development since 2010, with 1.0 release in 2015

- Mozilla used Rust to build Servo, a next-gen browser engine, later incorporated into Firefox

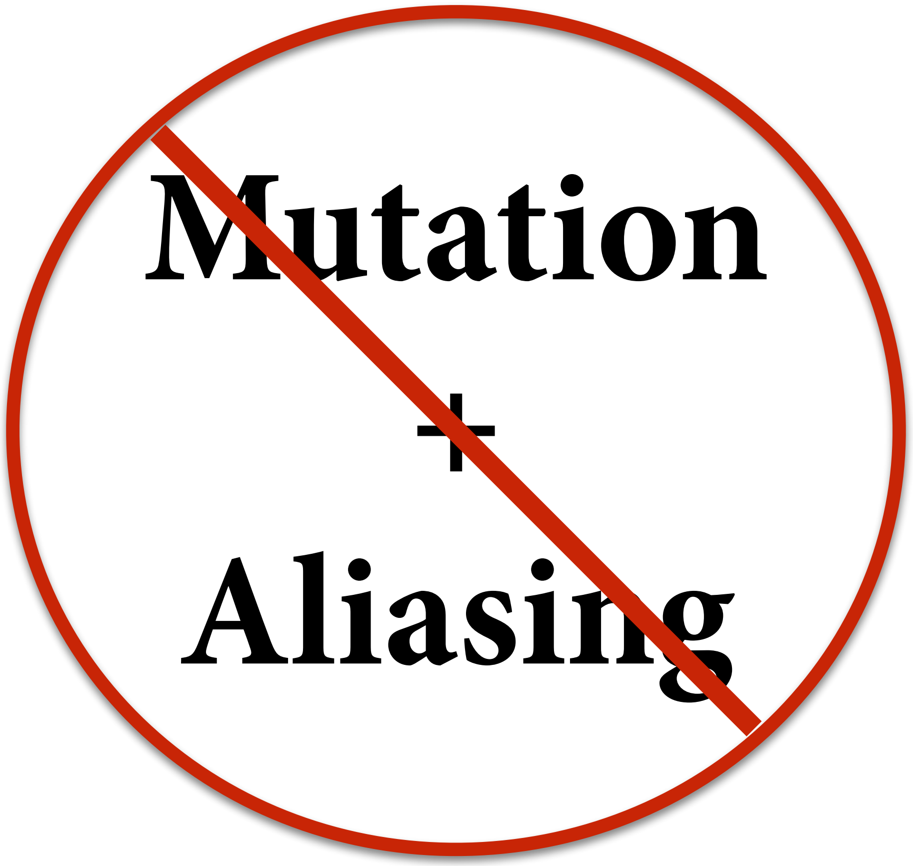
The "safety" of Rust is central to its promise.

But how do we know Rust is safe?

Many major companies using Rust in production

- In 2021, the **Rust Foundation** was formed, incl. Amazon, Google, Huawei, Meta, Microsoft, Mozilla

Core Idea of Rust

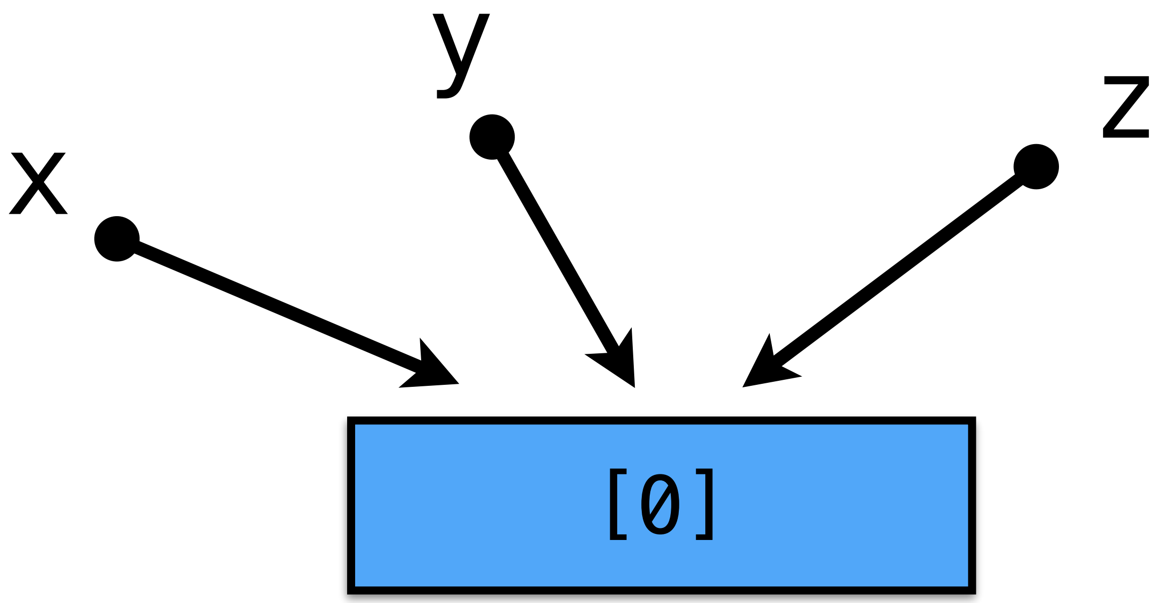


Mutation

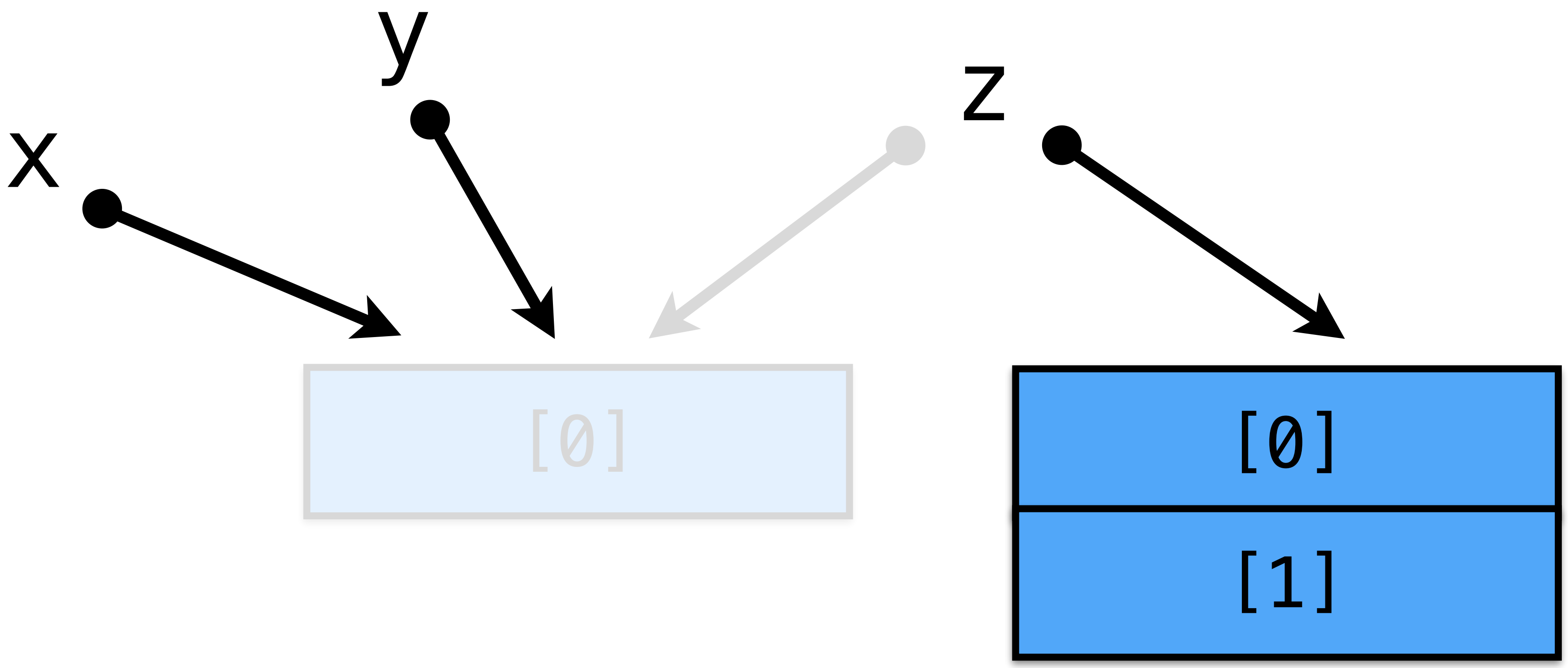
+

Aliasing

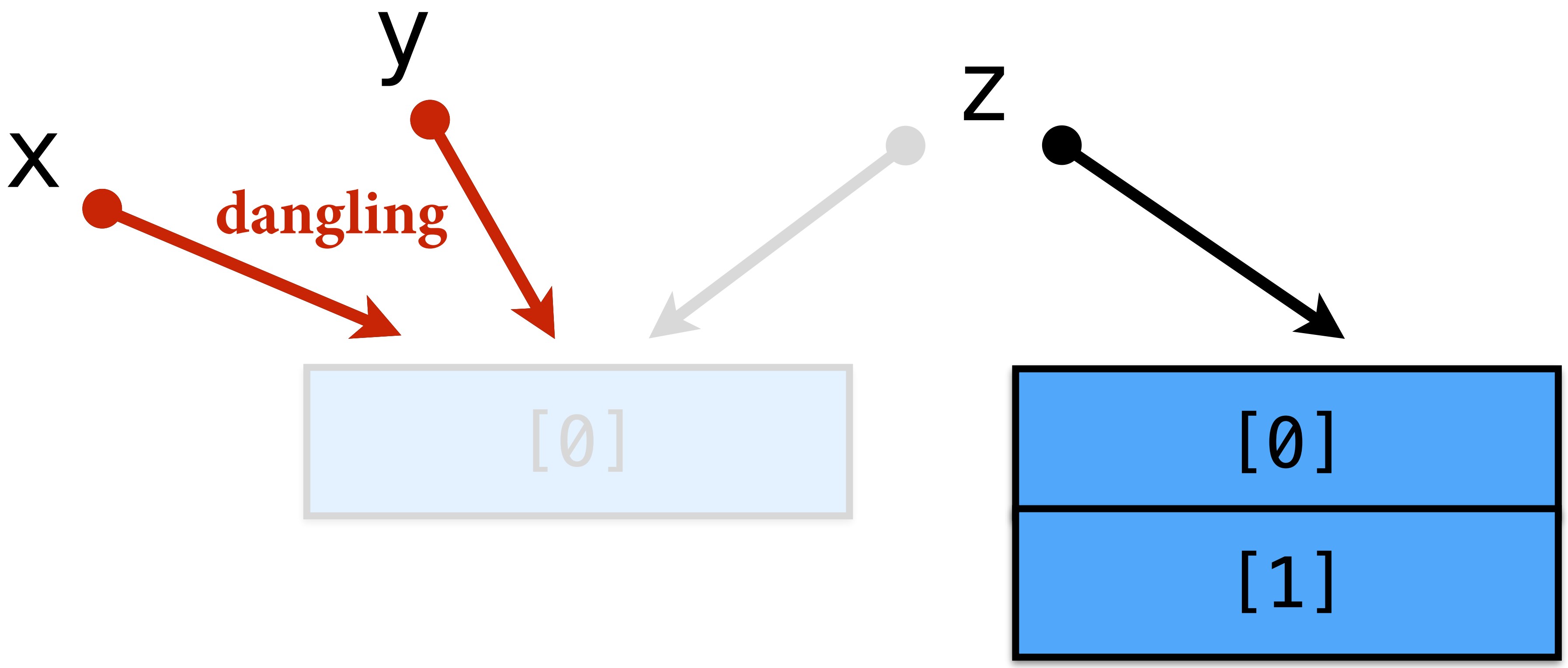
Core Idea of Rust



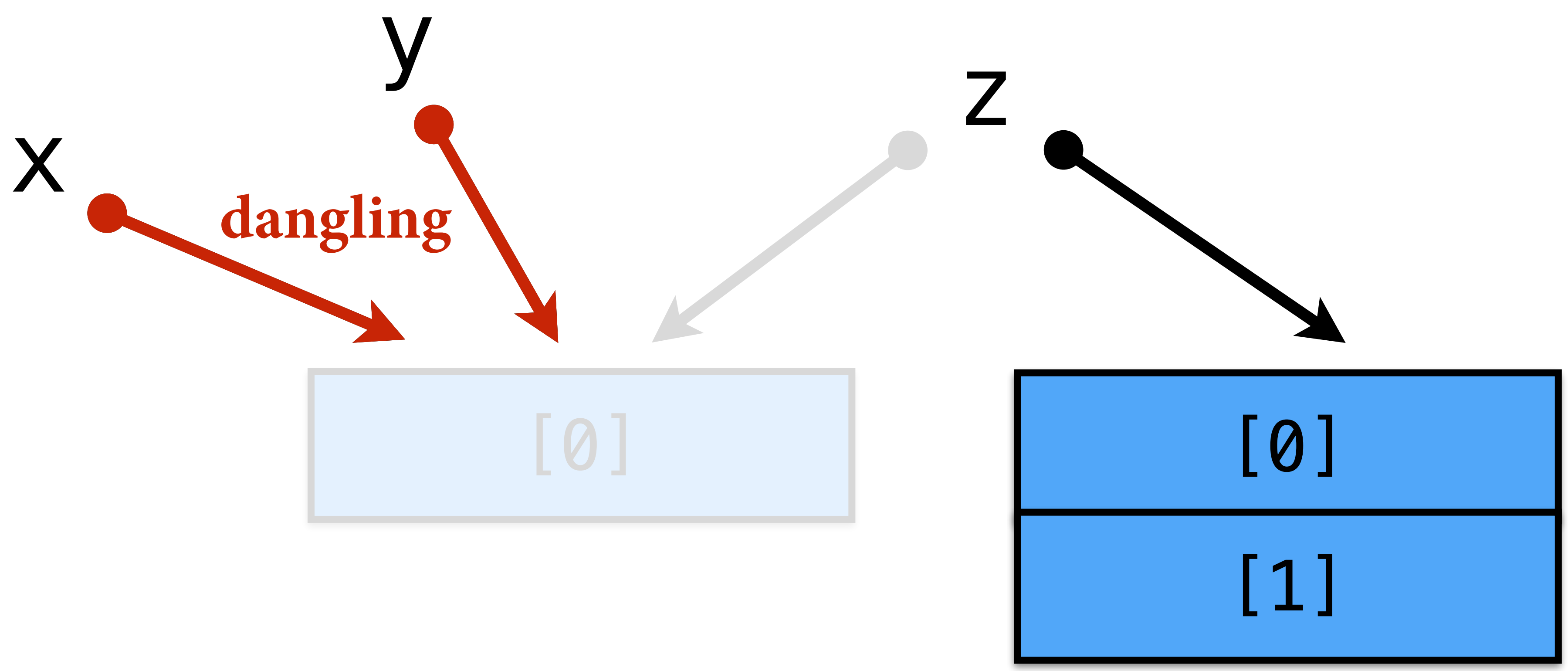
Core Idea of Rust



Core Idea of Rust



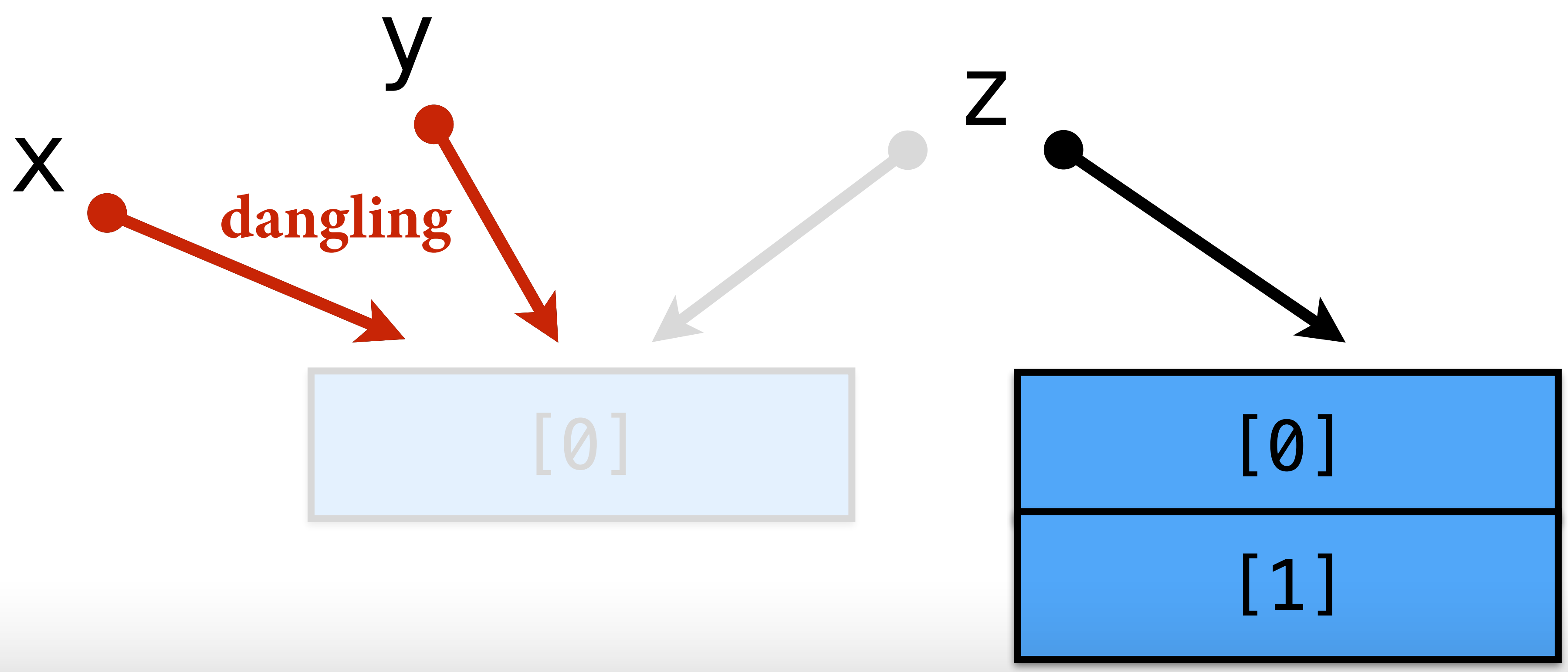
Core Idea of Rust



Unrestricted mutation and aliasing lead to:

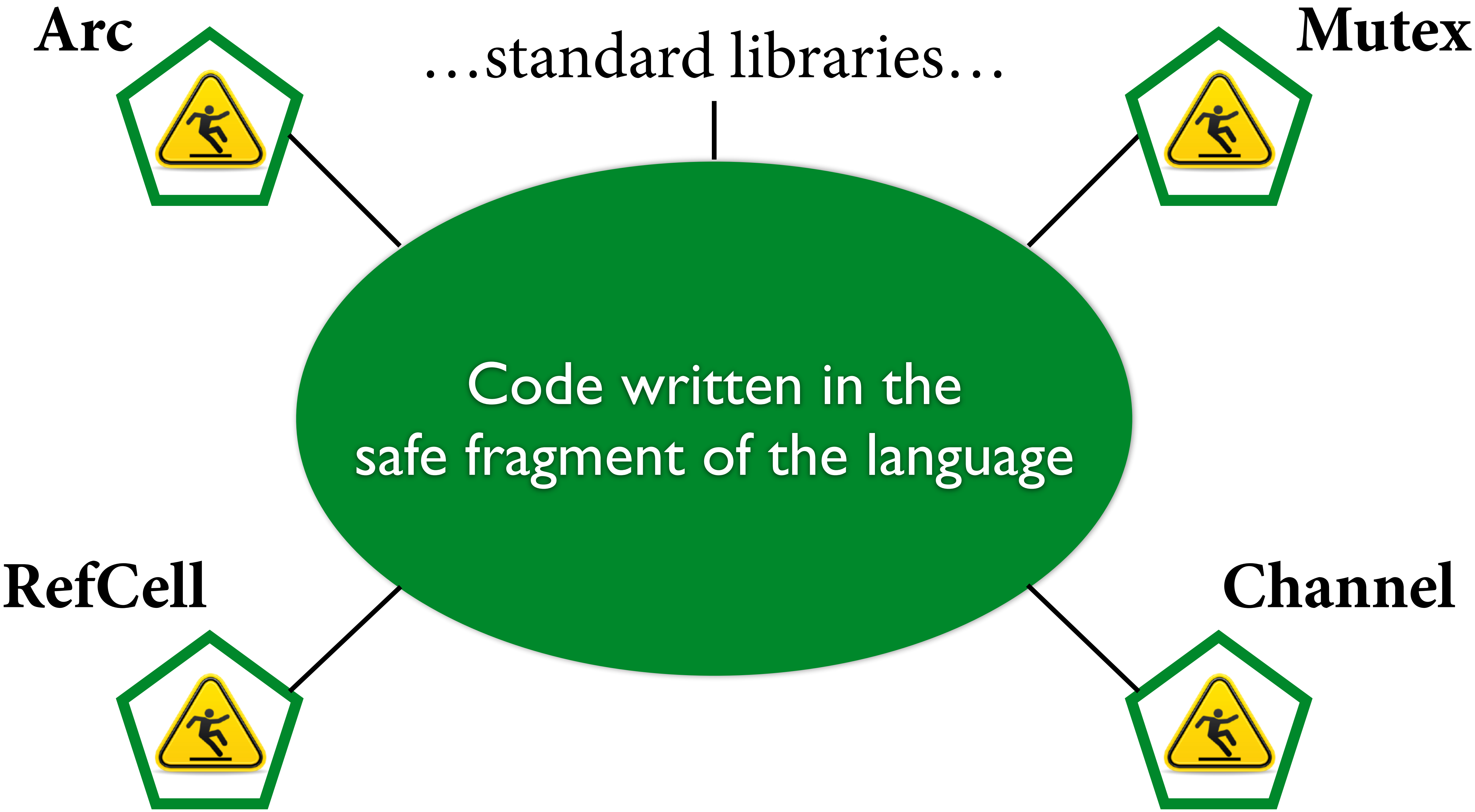
- use-after-free errors (dangling references)
- data races
- iterator invalidation

Core Idea of Rust



Rust prevents all these errors using a sophisticated “ownership” type system

The Reality of Rust



Arc

…standard libraries…

Mutex

Code written in the safe fragment of the language

RefCell

Channel

The Reality of Rust

Arc
Mutex
RefCell
nel

…standard libraries…

```
...
pub fn borrow(&self) -> Ref<T> {
  match BorrowRef::new(&self.borrow) {
    Some(b) => Ref {
      _value: unsafe { &*self.value.get() },
      _borrow: b,
    }, ...
  }
}
...
```

The Reality of Rust

Arc

…standard libraries…

Mutex

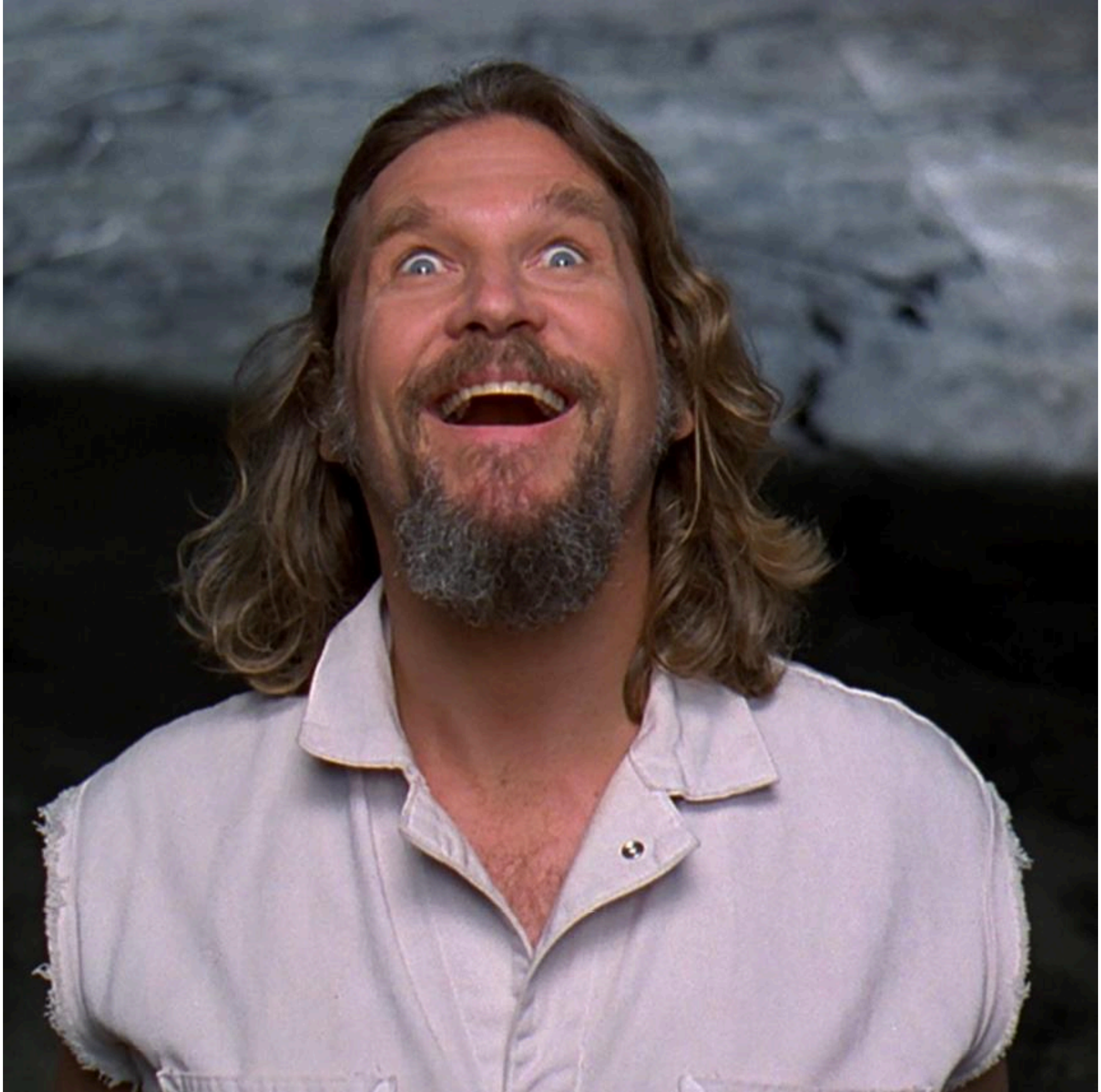
RefCell

…

nel

Claim library developers want to make:

Clients written in the safe fragment
will never observe any undefined behavior.



RUSTBELT

Goal: **Develop 1st logical foundations for Rust**

- Based on **Iris**, a new framework for higher-order concurrent separation logic in Coq
- Use these foundations to verify the safety of the Rust core type system and std libraries
- Give Rust developers the tools they need to safely evolve the language