

unsafe & traits

29th of January 2020
Rust Graz, lab10

PROLOGUE

PROLOGUE: IS UTF-8 ROBUST W.R.T. TO BIT FLIPS?

Consider the following [contrived] notion of robustness.

Let $\{X\}$ be a valid UTF-8 string. Is $[A, \{X\}, A]$ a valid UTF-8 string if one arbitrary bit flip occurs?

PROLOGUE: IS UTF-8 ROBUST W.R.T. TO BIT FLIPS?

```
// Consider a byte array like [0b1000001, 0b11110xxx,  
// 0b10xxxxxx, 0b10xxxxxx, 0b10xxxxxx, 0b1000001]  
// where x denote arbitrary bits.  
// Continue only if the array is a valid UTF-8 string.  
// Now introduce one arbitrary bit flip in any bit of  
// the 4 mid bytes. Does it give a valid UTF-8 string?  
// Returns the total number of tests and invalid strings.
```

PROLOGUE: IS UTF-8 ROBUST W.R.T. TO BIT FLIPS?

```
fn flips_in_4_bytes() -> (u64, u64) {  
    let mut total = 0u64;  
    let mut invalid = 0u64;  
  
    let mut text = [  
        'A' as u8,  
        0b1111_0000u8, 0b10_000000,  
        0b10_000000, 0b10_000000,  
        'A' as u8,  
    ];  
    // TODO next slide  
  
    (total, invalid)  
}
```

PROLOGUE: IS UTF-8 ROBUST W.R.T. TO BIT FLIPS?

```
// base gives x-values in 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
for base in 0..=2097152u32 {
  // bit_flip defines the index of the bit to flip
  for bit_flip in 0..32 {
    text[1] = 0b1111_0000 | (base & 0b111) as u8;
    text[2] = 0b1000_0000 | ((base >> 3) & 0b0011_1111) as u8;
    text[3] = 0b1000_0000 | ((base >> 9) & 0b0011_1111) as u8;
    text[4] = 0b1000_0000 | ((base >> 15) & 0b0011_1111) as u8
    if str::from_utf8(&text).is_err() {
      continue
    }
    // TODO next slide
    total += 1;
  }
}
```

PROLOGUE: IS UTF-8 ROBUST W.R.T. TO BIT FLIPS?

```
// apply bit flip
if bit_flip < 8 {
    text[1] = text[1] ^ (1 << bit_flip);
} else if bit_flip < 16 {
    text[2] = text[2] ^ (1 << (bit_flip - 8));
} else if bit_flip < 24 {
    text[3] = text[3] ^ (1 << (bit_flip - 16));
} else {
    text[4] = text[4] ^ (1 << (bit_flip - 24));
}
match str::from_utf8(&text) {
    Ok(_) => {},
    Err(_) => invalid += 1,
};
```


PROLOGUE: IS UTF-8 ROBUST W.R.T. TO BIT FLIPS?

bytes	strings	turned invalid	%
1	1024	128	12.5 %
2	30,720	10,112	33.0 %
3	1,474,560	512,000	34.7 %
4	33,554,432	13,107,200	39.1 %
	35,060,736	13,629,440	38.9 %

PROLOGUE: IS UTF-8 ROBUST W.R.T. TO BIT FLIPS?

Is $[A, \{X\}, A]$ a valid UTF-8 string if one arbitrary bit flip occurs? Yes, with 61.6 % chance.

DIALOGUE: *actix*

ACTIX

<https://actix.rs/>

- “rust's powerful actor system and most fun web framework”
- Uses [tokio](#) (“asynchronous run-time for Rust”)
- Uses [futures](#) (“zero-cost asynchronous programming in Rust”)
- `async/sync` actors
- Actor supervision
- Typed messages
- HTTP 1 / HTTP 2 support \Rightarrow `actix_web`

ACTIX_WEB

“[Actix web](#) is a small, pragmatic, and extremely fast rust web framework.”

- routing
- GET/POST parameters
- form handling
- `serde` for serialization
- many dependencies, but [fastest web framework](#)

ACTIX_WEB



Nikolay Kim
@fafhrd91



Actix web 1.0.0 is released - a small, pragmatic, and extremely fast web framework for [@rustlang](#)



[actix/actix-web](#)

Actix project postmortem. Contribute to actix/actix-web development by creating an account on GitHub.

[github.com](#)

7:38 AM · Jun 5, 2019 · [Twitter Web Client](#)

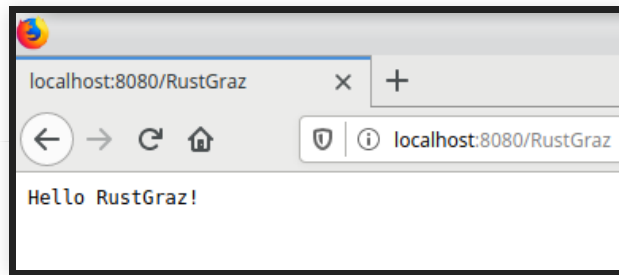
ACTIX_WEB EXAMPLE

```
use actix_web::{web, App, Responder, HttpServer};

async fn index(info: web::Path<String>) -> impl Responder {
    format!("Hello {}!", info)
}


#[actix_rt::main]
async fn main() -> std::io::Result<()> {
    HttpServer::new(|| App::new().service(
        web::resource("/{name}").to(index))
    ).bind("localhost:8080")?.run().await
}
```


ACTIX_WEB EXAMPLE



DIALOGUE: THE actix-web INCIDENT

GITHUB PAGE (2019-12-17)

 Why GitHub? ▾ Enterprise Explore ▾ Marketplace Pricing ▾

Search 

Sign In Sign up

actix / **actix-web**

Watch 191 Star 6k Fork 434

[Code](#) [Issues 48](#) [Pull requests 2](#) [Projects 0](#) [Security](#) [Insights](#)

Join GitHub today

Dismiss

GitHub is home to over 40 million developers working together to host and review code, manage projects, and build software together.


Sign up

Actix web is a small, pragmatic, and extremely fast rust web framework. <https://actix.rs>


[rust](#) [web](#) [web-development](#) [websockets](#) [actix](#) [actix-web](#) [async](#)

[2,993 commits](#) [8 branches](#) [0 packages](#) [180 releases](#) [177 contributors](#) [View license](#)

Branch: **master** ▾ [New pull request](#) [Find file](#) [Clone or download ▾](#)

 **avranju** and **fahrd91** Fix poll_ready call for WebSockets upgrade (#1219) ...

Latest commit 3b860eb on Dec 17, 2019

 **actix-cors**

update deps

last month

GITHUB PAGE (2019-12-17)

README.md

Actix web

Actix web is a small, pragmatic, and extremely fast rust web framework


build **passing**  **codecov** **80%**  **crates.io** **v1.0.7**  **chat**  **on github**  **docs** **2.0.0**  **downloads** **822k**  **rustc** **1.39+**  **license** **MIT/Apache-2.0**

[Website](#) | [Chat](#) | [Examples](#)

Actix web is a simple, pragmatic and extremely fast web framework for Rust.

- Supported *HTTP/1.x* and *HTTP/2.0* protocols
- Streaming and pipelining
- Keep-alive and slow requests handling

GITHUB PAGE (2020-01-17)

 Why GitHub? ▾ Enterprise Explore ▾ Marketplace Pricing ▾

Search / Sign In Sign up

actix / **actix-web**

Watch 13 Star 197 Fork 30

<> Code

Issues 17

Pull requests 0

Security

Insights

Join GitHub today

Dismiss

GitHub is home to over 40 million developers working together to host and review code, manage projects, and build software together.

Sign up

Actix project postmortem

6 commits

1 branch


0 packages

0 releases


1 contributor

Branch: master ▾ New pull request

Find file Clone or download ▾


 fahrd91 Update README.md

Latest commit 6e6bc3f 2 days ago

 README.md

Update README.md

2 days ago

 README.md

GITHUB PAGE (2020-01-17)

README.md

Actix project postmortem

Another day, another "unsafe shitstorm", I guess I get used to it.

It is interesting how easy to move comment out of context and how hard to comment with very clear intention (especially if you are not native speaker) What was the patch? It was very straightforward, simple, uncreative change, intention was just to remove unsafe not to fix existing code. I believe software development is one of the most creative work we do, and creativity is part of why we love software development, why it is fun. Especially if you combine it with real world projects constraints. "creative constraints" could be source of very interesting solutions. Being on the edge of your abilities is super fun. So uncreative change felt boring (oh! And author gave up copyright claims for that patch (a bit irony and sarcasm)). I've never used unsafe unintentionally, I use it because I believe usage is safe. There was no any malicious intentions. I believed it held mutable aliasing invariant and I was very happy that someone found real problem. I wanted to solve the problem, just with a bit of creativity. And use RefCell solution only if it would be not possible to solve it with any other way. Btw, I like the solution I found, it is in master and solves the problem at least one from the issue. If you want to push boundaries you have to touch this boundaries and sometimes you push too hard.

Be a maintainer of large open source project is not a fun task. You always face with rude and hate, everyone knows better

TWEET BY @FAFHARD91



Nikolay Kim
@fafhrd91



I am done with open source.

12:37 PM · Jan 17, 2020 · [Twitter Web App](#)

README TEXT EXCERPTS

Another day, another "unsafe shitstorm", I guess I get used to it.

README TEXT EXCERPTS

*I've never used unsafe unintentionally,
I use it because I believe usage is safe.
There was no any malicious intentions.
I believed it held mutable aliasing
invariant and I was very happy that
someone found real problem.*

README TEXT EXCERPTS

Be a maintainer of large open source project is not a fun task. You always face with rude and hate, everyone knows better how to build software, nobody wants to do home work and read docs and think a bit and very few provide any help.

README TEXT EXCERPTS

For example, async/await took three weeks 12 hours/day work stint, quite exhausting, and what happened after release, I started to receive complaints that docs are not updated and i have to go fix my shit.

README TEXT EXCERPTS

[...] But damage to the project's reputation is done and I don't think it is possible to recover. Actix always will be “shit full of UB” and “benchmark cheater”.

README TEXT EXCERPTS

(Btw, with tfb benchmark I just wanted to push rust to the limits, I wanted it to be on the top, I didn't want to push other rust frameworks down.)

README TEXT EXCERPTS

Everything started with actix, then actix-web and then actix-net. It took a lot of time to design api and architecture. Each of this projects was rewritten from scratch at least 4-5 time

README TEXT EXCERPTS

Nowadays supporting actix project is not fun, and be part of rust community is not fun as well.

*I am done with open source. [...]
Everything has to come to the end. It was fun trip but now is time to move on. Life should be fun.*

SUMMARY


- Somehow related to some shitstorm
- Shitstorm was somehow related to some unsafe feature of rust
- github repo was wiped (incl. issues) and README with post mortem text appeared
- @fafhrd91 used to be incredibly active and supportive towards users

→ An attempt of a reconstruction

2018/06/08

- [github issue #289](#)
- title “Unsound uses of Unsafe in API”
- “Right now the actix-web code contains 100+ uses of unsafe. Presumably this is in order to achieve the best possible performance in hot parts of the code.” ...

2018/06/12

- [github issue #301](#)
- issue is related to a memory-safety bug. Technical discussion is started.
- [comment](#) by @fafhrd91 “@seanmonstar do you think I don’t understand impl Send? :)” 11× 

2018/07/06

- reddit thread references issue #289
- title “actix-web has removed all unsound use of unsafe in its codebase. It's down to less than 15 occurrences of unsafe from 100+”

2020/01/17

- reddit thread “Actix-net unsoundness patch "is boring"”
- by user Code-Sandwich
- brings github issue #83 to reddit user's attention (is deleted these days)
- Summarizes issue #83
- Asks “I hope it's an objective summary. Any thoughts?”

ISSUE #83 2020-01-08 AND LATER

[@Shnatsel opens the issue](<http://archive.is/TokNs#selection-1603.1-1609.0>):

The following code is unsound: [actix-net/actix-service/src/cell.rs Lines 34 to 36 in 7c5fa25] This uses Rc::as_ref() to obtain a reference to the underlying data, which does not guarantee uniqueness. It is possible to obtain several mutable references to the same memory location by calling this function repeatedly: [...]

ISSUE #83 EXCERPTS

@fafhrd91

This is internal code. There is no repeated call to get_mut() anywhere in code

Closes issue.

@fafhrd91

Please, don't start

ISSUE #83 EXCERPTS

@Shnatsel

These two references do not need to exist in the same function to trigger undefined behavior, they only need to exist at the same point in time. An easy way to see if this is a problem in practice is replace `.as_ref()` with `.get_mut().unwrap()` and see if anything panics.

ISSUE #83 EXCERPTS

@cdbattags

I don't think "unsound" was meant to be personal or offensive. Why close this so quickly?

@fafhrd91

I need unit test that shows UB. Unit test must use public api.

ISSUE #83 EXCERPTS

@Nemo157

[provides such a unit test]

@JohnTitor

I think it's worth to fix, re-opening.

ISSUE #83 EXCERPTS

@fafhrd91

*@Nemo157 @repnop nice! finally some
real code!*

[provides patch]

should be fixed in master

ISSUE #83 EXCERPTS

Discussion about incompleteness of patch. Nemo157 provides patch using `Rc<RefCell<A>>`, not `Cell<A>` to ensure borrow safety at runtime.

@fafhrd91

this patch is boring

ISSUE #83 EXCERPTS

@CJKay

*quote[this patch is boring] So is
resolving silent data corruption.*

ISSUE #83 EXCERPTS

@bbqsrc

@fafhrd91 seriously? Please just stop writing Rust. You do not respect semver, you do not respect soundness, so why are you using a language predominantly based around doing these things right?

COMMUNITY RESPONSES

Posted by u/OkRaise6 2 days ago 🟡

Actix dev once again demonstrates no care for sound Rust code



Sorry, this post has been removed by the moderators of r/rust.

Moderators remove posts from feeds for a variety of reasons, including keeping communities safe, civil, and true to their purpose.

 44 Comments  Share  Save  Hide  Report

69% Upvoted

binkarus 117 points · 2 days ago · edited 2 days ago

I was going to see if there was some defense for the words he used, and so I read the thread. But then I went back 30 minutes later and he deleted and redacted everything in response to this article. <https://github.com/actix/actix-net/issues/83>

I thought you were overreacting at first, but now I can safely say **avoid actix if you can help it, the main developer is unprofessional, uncooperative, and untrustworthy**. We could use plenty of other more opinionated words to describe that behaviour, but I think those are enough.

He missed one, but [just look at this carnage](#)

E: He's deleted the entire issue, now.

[Continue this thread →](#)



git yeet HEAD~1 --hard

@mgattozzi



Reason #18723 why I hate Actix and think the Rust community really should abandon ship on it and work on alternatives like warp, tower, gotham or rocket
[twitter.com/whitequark/sta...](https://twitter.com/whitequark/status/1218723123123123123)

This Tweet is unavailable.

12:36 AM · Jan 17, 2020 · [Twitter Web App](#)



tomaka
@tomaka17



As much as I complain a lot, I barely ever actually link to things because there's inevitably a mob of angry people who will jump in and start trashing the maintainer.

I also think it's very irresponsible to do so if you have more than ~1k followers.



actix/actix-web

Actix project postmortem. Contribute to actix/actix-web development by creating an account on GitHub.

github.com

1:24 PM · Jan 17, 2020 · [Twitter Web App](#)



Stu Small
@thetusmall



What just happened with actix is a perfect example of the idea that not all software is made for you.

Don't like the level of unsafe? Don't like how issues are triaged? Fine, don't use the project. That doesn't mean you should harass the maintainer and bandwagon issues in Reddit.

3:10 PM · Jan 17, 2020 · [Twitter for Android](#)



HenkPoley @HenkPoley Jan 17 08:19

Why is the main maintainer of Actix rejecting memory safety bug fixes?

- https://www.reddit.com/r/rust/comments/epoloy/ive_smoketested_rust_http_clients_heres_what_i/feksq2i/
- https://www.reddit.com/r/rust/comments/epszt7/actixnet_unsoundness_patch_is_boring/



Nikolay Kim @fafhrd91 Jan 17 08:20

Because it was fixed in non boring way

But it seems everybody listen those who mist vocal and don't want to check actual code

Most vocal



Paul Dix
@pauldix



Wow, Actix maintainer quit and took his code home. I'm actually using it in a new project. Guess it's time to refactor to Hyper. I understand the creator's frustration, but it would have been better to just hand over to someone else. [#rustlang news.ycombinator.com /item?id=220739...](https://news.ycombinator.com/item?id=220739...)

3:19 PM · Jan 17, 2020 · [Twitter Web App](#)



Nikolay Kim @fafhrd91 Jan 17 08:52

Nobody wants to do any work, they just prefer relies on others work
That is reality of open source



Frank Denis

@jedisc1



The Rust community is toxic. Early adopters who used to love the language eventually quit.



actix/actix-web

Actix project postmortem. Contribute to actix/actix-web development by creating an account on GitHub.

github.com

3:25 PM · Jan 17, 2020 · [Tweetbot for Mac](#)



Restioson @Restioson Jan 17 21:05

@Dowwie why refactor away the unsafe blocks? @fafhrd91 made them with caution and wisdom.

some of them are alright but some are legitimately unsound (like the one in [#83](#) before it was deleted) so those would need to be fixed up. but you are right, unsafe is not the issue - unsound use of unsafe is the issue. the community handled it terribly.



steveklabnik
@steveklabnik



It's a sad day for [#rustlang](#): actix-web is dead. I am not feeling very good about our community today.

A sad day for Rust

actix-web is dead. This situation is bad, from all sides. When Rust was a tiny, tiny community, I thought to myself, "wow, I ..."
[words.steveklabnik.com](#)

3:26 PM · Jan 17, 2020 · [Twitter Web App](#)



enziuous @enziuous Jan 17 16:50

I know you're upset, and in a lot of ways you're justified, but people have invested time into Actix in their own way. I've spent 2 years developing a project on Actix. The least you could do is leave it up. This leaves me just completed dumbfounded. What the fuck have I been doing? What will I do now?



Vlad Filippov
@vladikoff



Replying to [@fafhrd91](#)

Thank you for your excellent work and time 🙌. I recently gave a Rust workshop including actix and everyone really enjoyed using it! I hope the community can step up and evolve the project from here.

4:23 PM · Jan 17, 2020 · [Twitter Web App](#)



Guillaume Balaine @lgosuki Jan 17 16:29

@fafhrd91 You don't need to go alone, every big open source project requires multiple maintainers and a leader



Armin Ronacher ✓
@mitsuhiko



Replying to [@bascule](#)

what frustrates me about the entire actix conversation is that most of the participants in all these conversations had fuck all to do with actix.

5:00 PM · Jan 17, 2020 · [Twitter Web App](#)



Joshua Barretto 🌹

@jsbarretto



Replying to [@ManishEarth](#) and [@steveklabnik](#)

I think what's particularly sad is that if Actix was written in C++, this would definitely be an acceptable level of UB anyway. Our standards are so high in the Rust community that we don't properly appreciate pragmatic stances that other languages have to deal in.

6:05 PM · Jan 17, 2020 · [Twitter Web App](#)



Sean Griffin

@sgrif



Folks who are claiming "The discussion on Reddit about Actix" was fine should consider what it feels like to have 3 separate front page posts all saying you're doing it wrong, each with hundreds of comments discussing the nuances of how wrong you are. Empathy helps, people.

6:53 PM · Jan 17, 2020 · [Twitter Web Client](#)



Dear Nikolay Kim (a.k.a. fahrd91),

We are users, contributors, and followers of your work in the Rust community. Over the past three years you have worked on Actix, and have had to endure repeated harassment from a small minority for your contributions to Rust and open source. We are extremely disappointed at the level of abuse directed towards you.

Working on open source projects should be rewarding, and your work has empowered thousands of developers across the world to build web services with Rust. It's incredibly tragic for someone who has contributed so much to the community, to be made to feel so unwelcome that they feel that they have no other choice than to leave. This is not the kind of community we want. Everyone should be able to contribute and work on their projects without fear of being harassed. What happened to you has shown that we still have a lot to do to achieve that.

Thank you so much for bringing Actix to the world and your contributions to the Rust community.

Signed,

- [Corey Farwell \(@frewsxcv\)](#)
- [Dylan DPC \(@Dylan-DPC\)](#)



Jehan
@JTremback



Unpopular opinion: the demise of Actix is good for Rust

10:01 PM · Jan 17, 2020 · [Twitter for iPhone](#)



Mingshen Sun
@MingshenSun



As a security engineer working on Rust, it's very sad to see what happened on Actix. We should do more on Rust unsafe: providing unsafe guidelines, helping people to understanding relation of unsafe and soundness, fuzzy testing unsafe code, providing static analysis tools, etc.

5:25 AM · Jan 18, 2020 · [Twitter for iPhone](#)



Saoirse Shipwreckt

@withoutboats



I also want to say that the root of this problem **is** with the Rust project! Not because we every encouraged this behavior explicitly, but because we allowed it to be done to us



Saoirse Shipwreckt @withoutboats · Jan 18

What we need is a culture which promotes psychological safety - encouraging and empowering, not scornful, bullying, shaming, mocking

[Show this thread](#)

12:37 PM · Jan 18, 2020 · [Twitter for iPhone](#)



Ashton Kemerling

@ashton



Having read all the stuff around the Actix brew up, I'm really not convinced that the author was harassed in anything other than a single comment (which others called out).

What instead I see is incredibly poor community management, and the negative consequences thereof.

1:38 AM · Jan 19, 2020 · [Twitter Web App](#)



Raph Levien
@raphlinus



My response to the recent Actix drama about unsafe, with a modest proposal how to hopefully make things better:



The Soundness Pledge

Lately there has been considerable drama around Actix-web, for which I'll point to Steve Klabnik's A sad day for Rust to ...

raphlinus.github.io

6:26 AM · Jan 18, 2020 · [Twitter Web App](#)

→ “The Soundness Pledge”

THE SOUNDNESS PLEDGE

“The intent of this crate is to be free of soundness bugs. The developers will do their best to avoid them, and welcome help in analyzing and fixing them.”

Particular_Bobcat 1 point · 21 hours ago · edited 19 hours ago

My take here is that the maintainer was childish.

- The issue opener is polite and professional.
- The maintainer made the first stab with " the patch is boring"... seriously who says this kind of things???
- The nasty comment came from a random person that wasn't related to the issue at hand.

Nobody agrees with this nasty comment. Everyone thinks it's nasty and over the line.

Still, the maintainer used this nasty comment as the reason to close down the project (and also not solve the original issue).

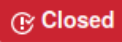
GITHUB ISSUES ON 2020-01-20

- lmao
- a sad day for rust ... and me
- actix-web is great
- jealous assholes
- Cheer up
- Please continue....
- Proud of your Awesome work!
- Don't take shit from anyone
- Condorcet

GITHUB ISSUES ON 2020-01-20

- It's fair
- fork?
- We love actix, thank you for creating it!
- Don't care about the prejudices of those few people, support you!
- Actix is still awesome
- Consider applying for GitHub sponsorship?
- Please archive the project instead of deleting it.
- Thank you for your service

Project future #1289



Closed fafhrd91 opened this issue 9 days ago · 131 comments



fafhrd91 commented 9 days ago

Member ...

I realized, a lot of people depends on actix. And it would be unfair to just delete repos. I promote **@JohnTitor** to project leader. He did very good job helping me for the last year. I hope new community of developers emerge. And good luck!



1310



4



109



304



795



234



32

... with purely positive replies.

RESOURCES

- [Wiped repo](#)
- [HackerNews thread](#)
- [Support letter repo for Nikolay](#)
- [Steve Klabnik's blog post](#)
- [“The Soundness pledge”](#)
- [reddit discussion](#)

QUESTIONS TO CONTEMPLATE

- What are the expectations/responsibilities of a community/maintainer?
- Is there a difference between a commercial and a hobby project?
- When shall I contribute or fork a project?
- How many maintainers/contributors does a project need?
- How much shall we consider differences in cultural norms?
- What non-technical aspects do we need to discuss/ensure in a project?

UNQUESTIONABLY GOOD ADVICES

- Be lenient in what you accept. Be strict in what you send out.
- Try to understand each other. Expect the other has good intentions. In short: be respectful.
- Ask for help if you need advice.
- “Code of Conduct” make is explicit that “do whatever is legally allowed” sets the bar too low

DIALOGUE: UNSAFE RUST

unsafe

See [Unsafe Rust](#) section in the Rust book and [nomicon](#).

1. Is a keyword in rust:

```
unsafe { let a = 42; }
```

2. Applies to blocks of code, incl. functions, traits.

3. Does not turn off borrow checker or alike

4. Semantics are basic: Some responsibilities move from rust to the programmer!

unsafe RATIONALE

1. *Static analysis is conservative.* When the compiler tries to determine whether or not code upholds the guarantees, it's better for it to reject some valid programs rather than accept some invalid programs.
2. *The underlying computer hardware is inherently unsafe.* Optimizations sometimes requires breaking strong guarantees.

unsafe SUPERPOWERS

1. Dereference a raw pointer
2. Call an unsafe function or method
3. Access or modify a mutable static variable
4. Implement an unsafe trait
5. Access fields of unions

RAW POINTERS

→ does not require unsafe rust. But we can't dereference raw pointers outside an unsafe block.

1. Are allowed to ignore the borrowing rules by having both immutable and mutable pointers or multiple mutable pointers to the same location
2. Aren't guaranteed to point to valid memory
3. Are allowed to be null
4. Don't implement any automatic cleanup

IN PRACTICE: RAW POINTERS

```
fn main() {  
    let addr = 0x00;  
    let ptr = addr as *const i32;  
    println!("{}", *ptr);  
}
```

`ptr` is a raw pointer. Does it compile?

```
error[E0133]: dereference of raw pointer is unsafe and
              requires unsafe function or block
```

```
--> unsafe_tests.rs:4:20
```

```
|
4 |     println!("{}", *ptr);
|                        ^^^^ dereference of raw pointer
```

```
= note: raw pointers may be NULL, dangling or unaligned;
       they can violate aliasing rules and cause data races:
       all of these are undefined behavior
```

```
error: aborting due to previous error
```

IN PRACTICE

```
fn main() {  
    let addr = 0x00;  
    let ptr = addr as *const i32;  
    unsafe {  
        println!("{}", *ptr);  
    }  
}
```

IN PRACTICE

```
meisterluk@gardner ~ % ./unsafe_block  
[1]      29915 segmentation fault  ./unsafe_block
```


IN PRACTICE

```
unsafe fn deref() {  
    let addr = 0x00;  
    let ptr = addr as *const i32;  
    println!("{}", *ptr);  
}  
  
fn main() {  
    unsafe {  
        deref();  
    }  
}
```

If `unsafe` block is omitted, “call to unsafe function is unsafe and requires unsafe function or block”

IN PRACTICE: `mut` STATIC VARIABLE

```
static MY_INT: u64 = 42;

fn main() {
    println!("{}", MY_INT);
}
```

→ type specifier is required

IN PRACTICE

```
static MY_INT: u64 = 42;

fn main() {
    MY_INT += 1;
    println!("{}", MY_INT);
}
```

IN PRACTICE

```
static MY_INT: u64 = 42;

fn main() {
    MY_INT += 1;
    println!("{}", MY_INT);
}
```

Does it compile?

IN PRACTICE

```
error[E0594]: cannot assign to immutable static item `MY_INT`
--> mut_static_var.rs:4:5
   |
4  |     MY_INT += 1;
   |     ^^^^^^^^^^^^^ cannot assign

error: aborting due to previous error
```

IN PRACTICE

```
static mut MY_INT: u64 = 42;

fn main() {
    MY_INT += 1;
    println!("{}", MY_INT);
}
```

Does it compile?

IN PRACTICE

```
error[E0133]: use of mutable static is unsafe
              and requires unsafe function or block
--> mut_static_var.rs:4:5
   |
4  |     MY_INT += 1;
   |     ^^^^^^^^^^^^^ use of mutable static
   |
= note: mutable statics can be mutated by multiple
       threads: aliasing violations or data races will
       cause undefined behavior

error[E0133]: use of mutable static is unsafe and
              requires unsafe function or block
--> mut_static_var.rs:5:20
   |
```

IN PRACTICE

```
static mut MY_INT: u64 = 42;

fn main() {
    unsafe {
        MY_INT += 1;
    }
    println!("{}", MY_INT);
}
```

Does it compile?

IN PRACTICE

```
error[E0133]: use of mutable static is unsafe and
              requires unsafe function or block
--> mut_static_var.rs:7:20
   |
7  |     println!("{}", MY_INT);
   |                   ^^^^^^^ use of mutable static
   |
= note: mutable statics can be mutated by multiple
      threads: aliasing violations or data races will
      cause undefined behavior
```

IN PRACTICE

```
static mut MY_INT: u64 = 42;

fn main() {
    unsafe {
        MY_INT += 1;
        println!("{}", MY_INT);
    }
}
```

IN PRACTICE: union

```
#[repr(C)]
union Ambiguous {
    my_int: u32,
    addr: *const u32,
}

fn main() {
    let a = Ambiguous { my_int: 0x00 };
    println!("{}", *a.addr);
}
```


DO I SEE UNSAFE RUST IN THE DOCUMENTATION?

Yes, because you need to call it unsafely.

Example: [Rc::assume_init](#)

```
[...] impl<T> Rc<MaybeUninit<T>>
```

[src]

```
[...] pub unsafe fn assume_init(self) -> Rc<T>
```

[src]



This is a nightly-only experimental API. (new_uninit [#63291](#))

Converts to Rc<T>.

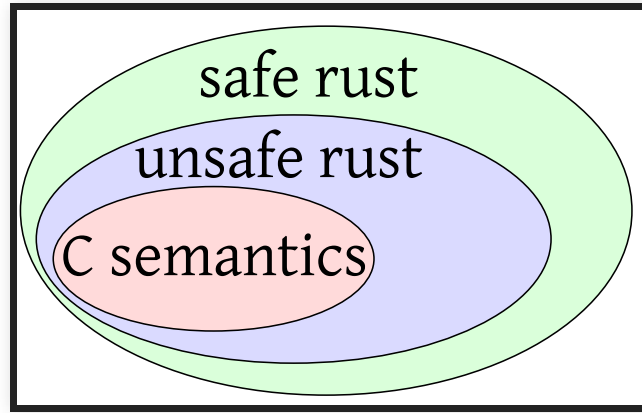
Safety

As with [MaybeUninit::assume_init](#), it is up to the caller to guarantee that the inner value really is in an initialized state. Calling this when the content is not yet fully initialized causes immediate undefined behavior.

Examples

**SO, HOW TO CONSIDER UNSAFE
RUST?**

COMPILER GUARANTEES



$C \subset \text{unsafe rust} \subset \text{safe rust}$



Lokathor commented 2 days ago

Contributor ...

Safety dance isn't about eliminating all unsafe code.

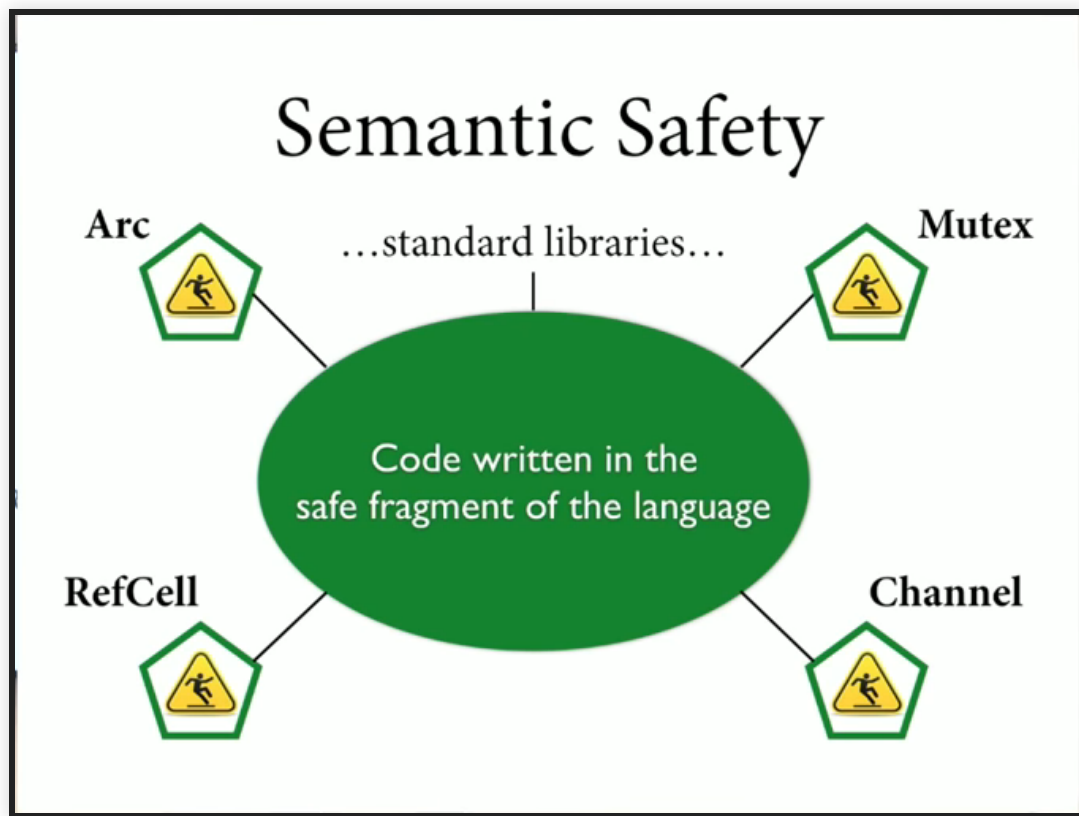
It's about auditing all unsafe code for correctness.

However, if you reduce the amount of unsafe code it obviously doesn't need to be audited any more.

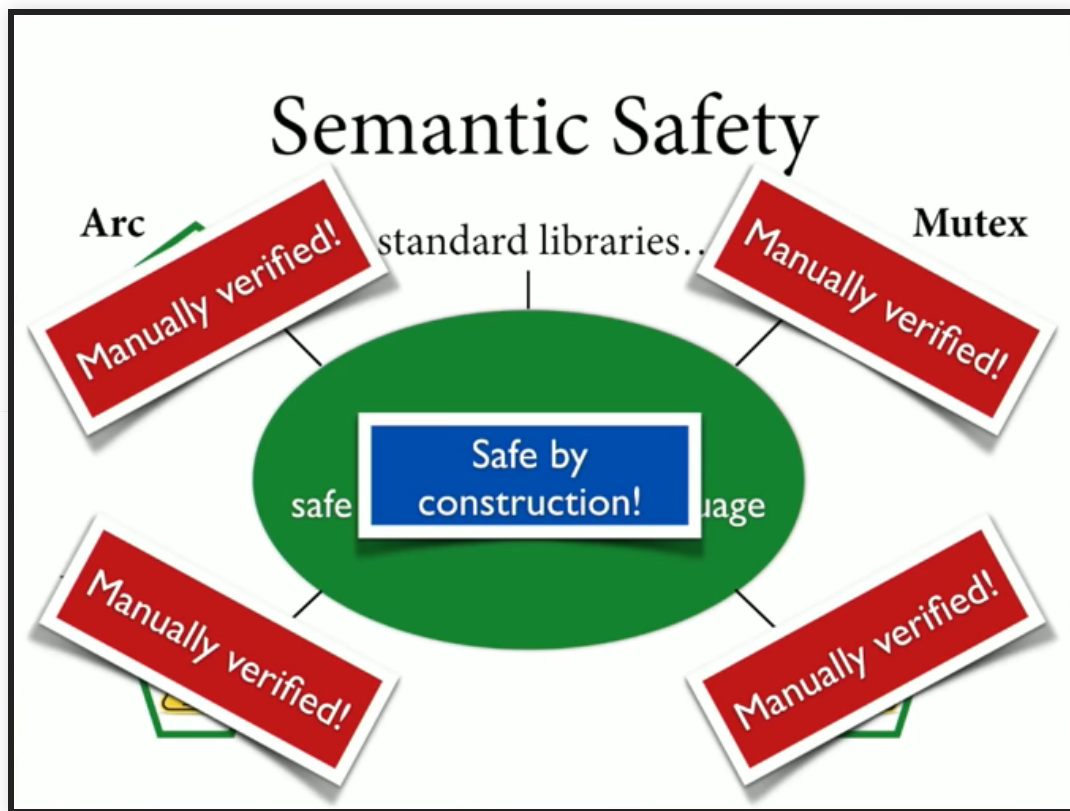
That is why reducing the amount of unsafe code helps the goal of auditing. Less unsafe code means less unsafe code to audit.

via [github: safety dance](#)

RustBelt: Logical Foundations for the Future of Safe Systems Programming by Derek Dreyer



RustBelt: Logical Foundations for the Future of Safe Systems Programming by Derek Dreyer



QUIZ

THANKS!