Interviewer
What have been your motivations for learning Rust and choosing to use it?

Participant
What happened was, I realized I needed to do something in, in WebAssembly, because we want some, again, this is, I'm going to, this is what I, it would be so much easier if I could talk about the details of what I'm working on. What happened was, an application we needed to build, had to be in WebAssembly, had to use C++ libraries, and had to run in the web browser. And what happened was, I went and, at first, we were going to just do emscripten to build some C++ stuff, run it in JavaScript. And then when I took the Rust WebAssembly tutorial, I immediately realized that I was like, this Rust thing is something we need to take seriously. Because what happened, I was like, I looked, I, I learned a little bit about WebAssembly, specifically the WebAssembly JavaScript boundary. It was really obvious to me why it's like this, because WebAssembly is very much like a virtual machine running in your browser with its own address space. And, and as, you know, an old school C/C++ developer, it was really obvious to me why I was like this. And then I was like, I just immediately realized that JavaScript calling Emscripten C++ libraries was not going to cut it. It's like, I was not going to go and marshal things across a boundary all the time.

15:1 What happ... Web Application Devel

15:35 An... Emscripten

Interviewer
Gosh, yeah, I've worked with emscripten and before I can relate to what you're talking about directly, I think. Yeah. I asked my question. So with both Emscripten and Rust, you have this, I mean, the, what you're compiling to is WebAssembly. Were there any particular difficulties, I guess, related to C and C++ in that context,

Participant
Oh, yes. versus Rust when you're compiling for that target. We needed to use some specific open source libraries, and I will not mention which ones they are. There are Rust attempts to rewrite them. And frankly, I did not trust them.

Interviewer
Is there some why you didn't trust them?

Participant
What happened is, the specific library, actually, I could probably say, the specific library in question is the half buzz library for interpreting true type. And I, and I could probably say this because I've contributed with my work email address to half buzz before, very trivial stuff. There's a thing called Rusty Buzz out there, which is an attempt to rewrite it. And it says, Oh, this is great because we rewrote it in Rust. It runs 98% of the benchmarks correctly. And I was like, I don't want 98%. I want 100%. And furthermore, I use half buzz. I know that I'm using the industry standard

15:36 There'... Rust Rewrite is Incomp

library out there that everybody and their mother uses. And there's in that I never have to doubt that this thing is this Rusty Buzz thing is off to snuff. I will say I spent about a month figuring out how to get the C++, that particular library and other libraries to successfully link with WebAssembly.

15:... Rust Rewrite is Incomp

Interviewer
Okay, gotcha. So to clarify, it's you're still relying on these potentially C and C++ libraries that have the original standard implementation, but you're linking them against Rust, and then you're coming out into WebAssembly.

Participant
Yes.

Interviewer
Gotcha

Participant
And that was painful because the only WebAssembly target that and by the way, I have no idea how like, like how much you know, and you have no idea how much I know. So I'm just going to start stuff off and

Interviewer
That's perfectly fine.

Participant
Okay. So anyway, what happened was the only supported, the only supported target for for Rusty WebAssembly is wasm32-unknown-unknown. It's essentially raw WebAssembly, no standard libraries, nothing like that.

Participant
I tried to get things like Wasi and the Emscripten targets to work and, you know, I actually like found a few bug reports against the WESem, things like wasm-bindgen and got a big, you know, flows immediately. We only support WASM unknown under WASM32-unknown-unknown. We don't like Wasi, whatever. And what I ended up doing is I figured out how to compile. I used the Wasi tools to compile the C plus plus libraries. I linked them. I told, I used the target option to say target wasm32-unknown-unknown. And because I did not have the standard C plus plus libraries available, I had to reimplementing a subset of the standard C library in unsafe Rust so that I could successfully link to this stuff.

15:5 I... WASI
wasm-bindgen
15:4 A... No Other Choice
Rewrite it in Rust

Interviewer
Gotcha. Wow. That's fairly in depth.

Participant
That was my first month of Rust.

Interviewer
That was your first month of Rust running that tricky scenario, huh?

Participant
Yeah. Gotcha. So I'm a big fan of you solve the hard problems first. I can learn a new language.

Interviewer
So I guess the sort of to summarize to make sure I'm on the same page here with so you have the requirement that you need to use these original C and C plus plus libraries in order to meet that reliability.

Participant
Yeah.

Interviewer
And you also have the requirement that you need these to function to some extent as a WebAssembly application communicating with JS in the browser.

Participant
Yes.

Interviewer
And you have two choices, which one is Emscripten for the entire thing.

Participant
Yeah.

Interviewer
And the other is using Rust and working with Wasi to compile the.

Participant
Yeah. I was essentially, I probably wasn't even, this is probably not even the best way to go about that. I just grabbed the Wasi version of Clang and said target wasm32-unknown-unknown and it worked. I'm willing to bet that the Clang people would probably laugh at me and said, oh, if you just grabbed this off the shelf thing, you could have done this or whatever. Like, I'm going to be honest. I hate it. That was one of the most miserable months of my life, just figuring out, like, I'm good at getting tools and things to build, but I hate it. And that's maybe that's why I'm good at it.

15:37… Bindings are a Fiddly M

Interviewer

That's very relatable in terms of, yeah, I've definitely been in build system hell before. So I guess my question is then, what was so difficult and challenging about Emscripten where this was preferable to that?

Participant
What happened is I realized that again, I'm just going to speak in riddles because I don't want to talk, go into details about the nature of my application. I realized that because of the mode of interaction I was going to have with the C++ libraries, I was going to be calling these open source libraries like a hundred times went and crossing the Wasm Web Assembly JavaScript boundary way more than I thought it was going to be acceptable. And I was like, this is unacceptable. We can't do this. And I want to, I'm going to need to rewrite a lot more of my logic in something that lives on the web assembly side so that I can have a more of a heavy duty chatty interaction with these C++ libraries and cross the Web Assembly JavaScript boundary a bit more, I'm not sure what the word I should use is, deliberately and less more as an overt transaction as opposed to some chatty interaction.

15:6 I was going to be... Increase Performance

Interviewer
Gotcha. So originally, you'd just be linking these libraries directly with the JavaScript, but you were concerned about the performance. So instead, you add this layer of Rust that allows you to cross the sort of keep that encapsulated.

Participant
I was actually in a project to rewrite something on the server on the client. And oh, you rewrite client side logics written in JavaScript, you write it in JavaScript. And we'll have to use Emscripten and run these libraries. And as someone that wrote the original server side application, I knew that I was like, I was like, I know the mode of interaction I was going to have with these libraries. And I was like, as soon as I saw what the JavaScript web assembly boundary was like, I do not think it's going to be acceptable to have something crossing this all the time.

Interviewer
And those are decisions you made based on your prior experience, but you didn't have any formal profiling or performance data with that.

Participant
I knew that I was like, I do not want a situation where I press a button and the Web assembly JavaScript boundary is crossed 100 times, the user press a button, and the interactive application did its thing.

Interviewer
That makes sense.

Participant

Yeah.

Interviewer
So my next question targets more just unsafe Rust. You mentioned one use case was re-implementing parts of the C standard library to be able to get your builds through configuration to work. I'm guessing there was also plenty of unsafe in terms of communicating with those wasm targets. Could you describe just all the areas where you use unsafe Rust?

Participant
Essentially, I just wrote wrappers for these C++ libraries in unsafe Rust that provided and that just provided a Rust that just provided a Rust safe wrapper. And honestly, it wasn't that hard. It's just no different than any other interop layer than such say between .NET code and native code or anything like that. And it has the unsafe keyword on this. You pass a slice of something, I get the pointer and then the
length and then pass it to this a C++ wrapper generated with I forget the name of the thing, bindgen something.

15:12... Exposing a Safe API

15:7 And... rust-bindgen
Simple FFI

Interviewer
Oh, wasm-bindgen?

Participant
I'm not wasm-bindgen, but there's another, there's something for C++ bindgen.

Interviewer
Oh, okay, gotcha. CXX?

Participant
Yes, let me just.

Interviewer
No problem. I've got your survey responses here too ...

Participant
Yeah, actually, it's just called, yeah, it's just called Rust bindgen.

Interviewer
Okay, gotcha.

Participant
Yeah, this, yeah. Yeah. And since these libraries are well established, heavily used open source things, I think that from a rust-bindgen perspective, they were probably the easiest things to add. And I think I'm pretty sure rust-bindgen is used to wrap the LLVM libraries for the Rust

15:46 An... rust-bindgen

compilers. And I'm on the bet that compared to that, like, this was like, this was small potatoes.

15:... rust-bindgen

Interviewer
So you were interacting with the, like the way that you're creating this safe wrapper, you mentioned taking in slices, which then give you this information about what's safe to access and you can do something like that. Were there cases where you had to pass memory that had less guarantees, like taking just a Rust reference and taking that into a raw pointer and passing it in? Or was it always something?

Participant
There was nothing like that. I had to do some tricks to get around. There was one thing where I had to make like something, in my opinion, one of the biggest challenges of Rust, and I feel like they need to figure it out, figure out a way to handle it better, is it's really hard to have an interest struct reference. You have a struct that has, let's say, a foo and then a bar and then the bar needs to reference the foo. And Rust does not make that easy. And just the nature of these libraries, I had to have a foo which wrapped this library and then a bar that wrapped this library with some references between them. And I could not completely express the semantics to the Rust. I guess what ended up happening, I guess a good way to articulate what happened is that the Rust is that the wrappers I created in Rust had borrows, had reference semantics with each other that were not completely, it could not completely express to the Rust borrow engine or whatever they call it.

15:9 I ha... No Other Choice

Interviewer
Gotcha.

Participant
And I just kind of had to say, hey, there's this, you create one of these things, or you create another one of these things that references this, you better make sure that the second object lasts as long as the first object. So it's like, I did not create wrappers that are fully kosher from a Rust perspective, but then I just looked at that as a caveat of how I'm using these libraries. And when I started writing my real business logic in Rust, there's just this one little caveat, a big comment saying, don't you better make sure that this thing survives longer than this thing or else you'll crash. And given that I'm an old school CC plus plus developer where this where you do this stuff all the time, I'm having this one little caveat in this piece of code that I wrote and then probably ignored for like, the next year was just not that big a deal.

15:11 So it's like... Contract or Invariant
Documented...ract or
15:16 An... Audience-specific

86 **Interviewer**
Gotcha. And to clarify the situation, it's, is it that you have one parent struct, and then you have these two members and each reference
97 each other.
98

15:13... Unsafe Data Structure

99 **Participant**
That's exactly what happened where one reference is the other.

100

101 **Interviewer**
Yep. I recently ran into that same problem on an application that I was writing. And yeah, I can confirm that that is not something that
102 Rust really does not want you to do.

103

104 **Participant**
That is a very good way to articulate it. In fact, we start, like me and a coworker that I was working with, when we jump on this problem, like we quickly ran into this and it became our interview question for Rust. You have a situation where you want to do this and, and the sort of the correct answer that, you know, someone that that was a really good Rust developer said is, this is just something that you just avoid in Rust and you, and you just, you just stay away from. And, you know, and if you're writing Greenfield code, I mean, okay, you do this, you, you just, you avoid shooting yourself in the foot and, or you avoid the specific scenario and you architect your application in a way that, that, that completely avoids that problem.

15:14 You ha...
Preference for Safety

105

106 **Interviewer**
Gotcha. So, so is the, do you have any particular, like, what's your confidence in that the solution you implemented to do this is sound? Like, do you have any, any evidence that you are free of undefined behavior here, or is this something where the code works?

107

108 **Participant**
So it's okay, but it's kind of a sticky spot. I mean, it's a sticky spot because the promise of Rust is 100% undefined behavior you don't have to deal with. But it's like, I look at this and I'm just like, okay, I just need to have my bar exist as long as the foo exists. And since they're in the same structure, that's fine. And, and to me, it's just small potatoes. I mean, C plus plus, you do this all the time. And, and I'm just like, like, I know this is not a problem. And I'll say that one of the things I've seen on, you know, the Rust forms on Reddit that I start, there's this sort of phobia of undefined behavior that goes on. And it's almost at the degree of you have even the slightest concession to undefined behavior, and you've committed a crime against humanity. And, you know, and I'm kind of like, you know, and I honestly roll my eyes at this, like I've been in the C and C plus plus world, I know why I don't want to do it. But I'm sort of like, hey, I can go in, if I write a dozen lines of code that maybe not doesn't have this caveats, I mean, and that doesn't line the code and then as opposed to, you know, 50,000 lines of code application, and to me, it's just not a big deal.

15:15 I look...
It's not a problem
Tacit Knowledge

15:38 But I'...
Local, Minimal Unsafe

109

110 **Interviewer**
Gotcha. Yeah. So in your creation of these wrappers, is it ever necessary for, like, let's say I'm going to use this library, I know nothing about how

you implemented it under the hood. Are there things I'm going to need to check and make sure of myself to be able to call your wrappers safely? Like, is there ever some sort of precondition that I have to meet?

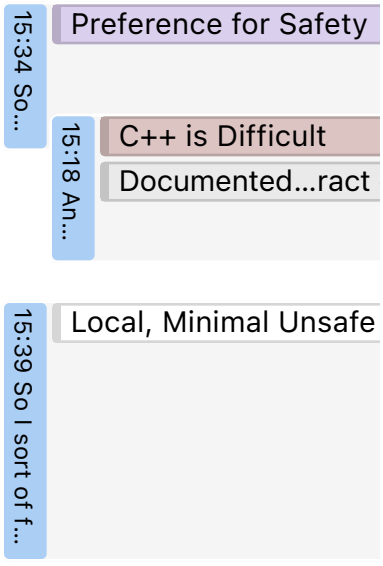
Participant
Just that you create a foo, or you create a bar, you pass it a reference to a foo, you better keep the foo around as long as the lifetime as the lifetime of our and you're actively using. And in my code base, there's only one place where I used a foo and one place I use a bar, and they're both in the same structure.

Interviewer
Gotcha.

Participant
So it's like, this is just, this is for me an extremely minor problem.

Interviewer
I guess outside of just that problem or two, I guess for any case where you're, where you have some sort of safe rust function that you've written that has unsafe within it. Are there situations where like, I guess the example I'm thinking of is you could have a safe rust function that takes in like an integer as a parameter, and that integer might be used to index into memory. So there's some sort of burden on the person who's calling that function to make sure that that integer is within a certain range. Is that type of situation something that ever happens, or is your interface written in a way?

Participant
I sort of, I feel that, and I might not be doing a good job at articulating this, I feel that a lot of those problems, I want to contain them at some level, whether they're in, whether they're in the whether they're in the specific wrappers that I created to hide all the unsafe stuff, or maybe some caveats on top of those parameters. So I guess what happened is I created a bunch of wrappers for C++ code, just so that I'm not just randomly calling unsafe, you know, like get pointer, get length, and calling this other stuff. And in some cases, those wrappers themselves had caveats. And I just put big comments on them and said, when you use these wrappers, here are some caveats. Again, you know, compared to the C++ world, this is nothing. And then in the few places where I use those wrappers, which are not, like, pervasive in my code base, I just made sure that I kind of abided by those guarantees. So I sort of feel, so for me, using Rust was not so much about eliminating 100% of undefined behavior from my code, but it's rather the ability to coordinate to this 2% of my code base or whatever, that, you know, that where I'm dealing with this stuff, whether it's in the wrappers that directly access it, or some of some caveats based on those wrappers and from those wrappers themselves.



122 **Interviewer**
Gotcha. Gotcha. Okay. So I guess the next question is a bit more specific on some of the types that you mentioned. You just selected Box and Arc. Could you describe the situations where each one has been used in today's context? I think I didn't,

123

124 **Participant**
I must admit, I didn't fully understand what you were asking there. Oh, gosh, I'll just explain what I was doing.

125

126 **Interviewer**
Yeah, sure. Yeah.

127

128 **Participant**
In the case of the Harfuz library, Harfuz is a low-level library that interprets true type and takes care of the most arcane aspects of text shaping. I got a ligature to look up the true type file and do this. And there's a notion of fonts and then there's also a notion of faces and a font, indirectly references the face it was based on and when an app happening is that my logic that where I had to work with these fonts and faces got put into another object that then was, I put an Arc around. Okay. So it's just by the time I got to either Rc/Arc or bot, like the wrapper code I wrote does not have a single Rc/Arc or Box in it.

15:19 And there's a...
Arc<T>/Rc<T>
Box<T>

129

130 **Interviewer**
Okay, gotcha.

131

132 **Participant**
So, and the code that actually used those wrappers itself does not have an Rc/Arc or Box on it. But that latter object is often referenced with an Rc/Arc or Box.

15:20...
Arc<T>/Rc<T>
Box<T>

133

134 **Interviewer**
Gotcha.

135

136 **Participant**
It's like, I actually tried to not check anything on that form and it didn't let me. So I was like, I'm probably like, if this ends up being interesting, it will show up in the conversation. I'm hoping what I'm saying makes sense.

137

138 **Interviewer**
Yeah, I think so. And I think all after this meeting, I'll probably change that so that it is an optional question.

139

140 **Participant**
Okay.

Interviewer
Because yeah, it's totally possible that someone could, here, let me back up, to confirm it's that you use these types to contain your wrapper objects, yes, but that you never use them to define operations that happened within the wrapper.

Participant
I don't even know how you would do that. Like, or maybe there's, maybe there's a specific pattern that you're referencing that I don't quite understand.

Interviewer
Oh, gotcha. So like one, one pattern that will sometimes occur is that someone will use Box to allocate something on the heap in Rust, and then they'll take that Box and convert it into a raw pointer, and then pass that raw pointer as a pointer to this allocation into C or C++.

Participant
So I think I might have done something like, let me, let me check. Gotcha. It is so unfortunate that I can't really talk about what I'm doing, but yeah. Yeah, actually, I'm not really doing anything like that. And this is just because PowerFuzz and these other libraries that I'm working with, like they have an API and they don't, and just like a well-crafted C and C++ API, they don't generally like, like, this is an API that's been around for a while and they don't want you go, and I don't have to do that for the same reason that when you build a C API, you don't necessarily have to do that when calling that or, or even things like, you know, the Zlib libraries or anything like that.

15:40 And this is just...
Simple FFI

Interviewer
Gotcha. Gotcha. Yeah. So then the, the next question is related to your use of bindgen. Could you describe what your experience was and using that with these libraries and whether or not you've had any, any problems with that that made it difficult to create these bindings correctly or that caused bugs or errors when you tried to use them?

Participant
I didn't really have any problems with bindgen itself.

...
bindgen

Interviewer
Gotcha.

Participant
I mean, I just ran the thing and it spat out a bunch of wrappers and I feel I spent more time just messing with the compiler and getting it to link and the getting, and getting a, getting a version of these libraries building with

15:41 I...
Generation VS Validati

WebAssembly and then, and, and just getting things to link. So I feel any problems I had with bindgen were, were completely drowned out by just general build system stuff.

15:41 I... Generation VS Validati

Interviewer
Gotcha. Gotcha. And then, so I, this might be, so was this, I guess, just one question to confirm about the way you're interacting with this C API, was it ever something where you had to wrestle with the differences between C and C plus plus memory and how likely the assumptions that are made with that and what assumptions Rust makes with memory or was the API structured in a way that you didn't need to worry about the memory models?

Participant
For the most part, I didn't need to do that.

Interviewer
Gotcha.

Participant
I, I mean, it's a well-crafted C and C plus plus API just gives you opaque pointers and like, I don't, and for the same reason, you wouldn't want to be as a C developer though and, you know, reaching into some instructors. I did not have to do that here.

15:22 I, I... Opaque Pointer
Simple FFI

Interviewer
Gotcha. Gotcha. Okay. And then double check this. This next question is much more broad. Describe a bug that you've faced that involved unsafe Rust.

Participant
Oh, my, my re-implementation of one of the standard C functions was incorrect. And I don't remember which one it was. It might have been like strchar or something like, something like that. And debugging that was a real pain in the ass.

Interviewer
Gotcha.

Participant
Because I, because it happened on WebAssembly and then I had to go and do the usual stuff of debugging. I had, when my, when I built these open source libraries, I had to go and I had to essentially debug those libraries. I spat out a bunch of printfs and, and then found out that, and that found out from the perspective of those libraries, the standard C library was misbehaving. And then I realized my plugin in the C lot and my re-implementation of one of the C functions and, and fix that. But it's

15:23... Printf-Style Debuggin

like, you know, hey, I'm from the C plus plus world, I've been there and done that.

Interviewer
Gotcha.

Participant
Yeah. Yeah. In some ways, that Rust wasn't even involved there at all. Because since this happened in the bowels of these other libraries, like the fact that, you know, Rust was, it's, Rust was almost just not a factor. It was like, I was debugging these open source libraries and seeing this malfunction and looking at, and looking at memory dumps and printfs to figure out when something changed and saw some classic C++ memory thrashing. And in some ways, the Rust stuff was as irrelevant as, you know, like the, you know, the build system was.

15:...

Interviewer
Okay. So it's, it's, it's more that this error was just like, there wasn't anything intrinsically related to unsafe Rust in this. Gotcha.

Participant
I'll tell you the other thing about the unsafe Rust code is that it's definitely was not quote unquote, ergonomic, like I'm doing, grabbing a pointer and then doing an offset on it, you know, and like, you know, I could probably, I could probably just look at one of these things and I have C lib, I have clibimpl.rs. Okay. In fact, what I did is I had my clibimpl.rs built, my application also builds to native code too. And I build the C library implementation. But then because I don't have to go and link, you know, do these shenanigans outside of Web Assembler, I wanted, I wanted that code to be built and be unit tested alongside my native code. And then I just had some like wrappers that basically activate the entry points only when, only when I'm compiling on Web Assembler. Okay. Yeah, it's like, like here is an implementation of, of the realloc function in C plus plus.

15:... Pointer Arithmetic

Interviewer
Gotcha.

Participant
Yeah. I mean, oh, yeah, this is what blog got me. There's this weird thing about how, there's this weird thing about, yeah, okay, this is something
176 that got me. Um, I'm just going to show it to you. Gotcha. There's this stuff about Rust memory management where you, when you allocate memory, you also have to pass in the alignment. I'm not quite sure why I'm guessing it's just that some of the more lower level allocators probably sidestep a lot of the, a lot of the stuff that normally happens in, in, in the C and C plus plus world, you know, shave a few bites off. But I was like, hey, I need to go and for some reason when I allocate memory, I need to pass in this alignment and it's just part of the rules. And I went along with it. I don't know. I'm sure if someone blogged about why exactly you need

to, I'm sure I'd find it to be an interesting blog entry, but I just like, hey, the, the Rust document tells, tells me that I have to do this and I have to do this.

Interviewer
So I guess the 16, was that just taken from the standard alignment that malloc would use?

Participant
Yes, I believe so.

Interviewer
Gotcha. Um, gotcha. Okay. Um, yeah, I'm, it's okay for me to copy that code down.

Participant
Yeah. That's fine. Um, this is not interesting code.

Interviewer
Gotcha. Yeah. Um, and then let me double check. I need to have your survey again. Gotcha. So with seven or with, so one of the survey questions was, which of the bug, the following bug finding tools did you use with code bases containing unsafe Rust, but since you checked other, was that sort of your like a N/A? So I guess are there reasons why you haven't been able to use bug finding tools in this situation?

Participant
I guess it just didn't occur to me to do so. Um, maybe I'm just, I was like, there's this thing going on. I don't know why it's going on and I'm going to throw down some print f's and I'm, I'm kind of, maybe I'm kind of a Luddite, but it just didn't occur to me that there was something out there that might have helped me find this. And if there was, I mean, I would like to know just from my future benefit, but it's like, I was like, I'm sure, I suspected that there was something wrong with my C-lib re-implementation, which, so the fact that I found that there was a bug there and was not an overall surprise.

15:26 I d...
Printf-Style Debuggin
Unaware of Tools

188 **Interviewer**
Gotcha.

203

204 **Participant**
But it's like, I'm, I just, you know, like what am I going to do? Go to Google for, um, I'm writing unsafe code and some random memory is being thrashed.

205

206 **Interviewer**
Um, maybe I'm, I don't know what out there would have helped me with

that. I guess just so I can, I can think about what might, um, could you refresh me again what the specific problem was in the, the one C-lib function?

Participant
I honestly don't remember. Gotcha. It was just, I think I'm looking at this, maybe something. Um, actually, let's get long. Um, get long. C-lib. Yeah, it was, um, okay, that's right. Okay, I'm looking at my comments. Um, yeah, I just did not, I did not understand the semantics of the layout, the std, std alloc layout object and exact, and the pre and post conditions for, um, for like how I use that and, and in the traditions of, of unsafe code, um, when I first wrote it, it mostly worked until I had a bug.

15:27 Okay, I...

Interviewer
Gotcha. Was there like a particular pre and slash post condition that was, was the issue there?

Participant
I think it had to do with me passing the alarm, reporting the alignment and passing it.

15:... Allocation & Alignmen
Improper Alignment

Interviewer
Gotcha. So was it just like the wrong alignment value then?

Participant
Yeah, I think, yeah, I think I, I think what I did is I had some code that, my code for figuring out that value was incorrect and I was passing some undefined value to it. And it worked most of the time and then until it didn't.

Interviewer
Gotcha. Gotcha. Okay. Yeah. There is a tool called Miri that is a Rust interpreter. And what, one of the things it can do is it tracks the alignment of every memory allocation. And then whenever you try to access something with improper alignment, it can record it and then you can find like a back trace of like where that allocation is created. But one of the things about Miri that might make it difficult to use in your situation is that it has no support for foreign function calls. So you wouldn't be able to deploy it on a code base, like the one that you described where you have this Rust component and then this component and then this other Rust component and they're like bridged together. That, yeah, that would be difficult to, like you just cannot do that right now.

Participant
Yeah. And that, and in a lot of ways, that's fine. It's like, as soon as I under, and using something like that, essentially required that I understand the bug I was making. And because there's literally like only one place in my code base where I was doing that, it's like, it just, like, I

15:... Local, Minimal Unsafe

feel it's something like that in practice would not be useful because it's like, hey, I'm doing this thing once. And, and then once I get this code working right, it's I ignore it and for the rest of my life. And it's not this, I have a feeling that the way I might be a weirdo for you is that in the code base I've written, the unsafe Rust is this, like, I got it working and then, and then moved on. It's not this pervasive theme in my development. Maybe there'll be some time when I either upgrade the, you know, upgrade half bars or these other open source libraries, and then I need to call a new function and I have to go and open that unsafe code. But it's like these, the times that I touch this stuff are like probably, you know, months and, you know, months in between each other.

15:42 And beca...

Local, Minimal Unsafe

Interviewer
Gotcha. Gotcha. Okay. Yeah. That makes sense. I think, oh, there's one particular question that I had. Sorry, I just lost my train of thought. Gotcha. So was the, I guess the alignment problem, when you first were noticing it, that was it immediately obvious that it was due to that particular subset of your code? Or was that something that you just had to figure out eventually through?

Participant
Yeah, I think it was the latter. I mean, essentially, I just went through the process of tracing down the code and then noticing that this thing, you know, was happening. And I'm like, and then when I saw this garbage value was being passed to this, you know, I googled these, you know, layout objects and I was like, oh, okay. And then the codes, the documentation says what you pass in when you allocate memory, better be what you pass in when you're deallocating memory. And I'm like, oh, I'm not doing that. So that's probably the problem.

Interviewer
Gotcha. So the docs were what helps you figure out the specific thing. Gotcha.

Participant
So yeah, it's, I will say it was an adjustment. Like, I'm used to met in the C++ in the C world, I'm used to malloc and free. And you don't have to go and like, you know, and when you deallocate memory, you don't necessarily have to go and, you know, have this information like the information that is around that, when when you allocated memory, in fact, I'm like, did I'm when you free memory? Yeah, it's like, I see I'm looking at my implementation of free. And it's like, I have to go in. Yeah, I have to pass in the layout object that I used to create the memory at the beginning. And I guess, and it better be the same and I ended up in just my wrappers just caching out along with everything else.

15:43 Like, I'm used to met i...

Feature Disparity

Interviewer
Gotcha. Gotcha. Have you a, I guess, just in general, what's been your experience with the Rust documentation about certain functions like this,

where it's giving you the sort of requirements that you have to follow in order to use something correctly? Has that generally been helpful for you?

Participant
Yeah, I think it's been excellent. Gotcha. And like literally just, hey, there's a piece of unsafe code. Here's a function that's unsafe. Here's why it's unsafe. And, and, and I know to do what I'm told. And probably the only thing where it's there was some unintuitive, where something was unintuitive, and I forget where I had to do this, was when you want to return a string from something, you have to create like a vector, and then you have to resize the vector. And then or you have to, what's the, you have to reserve space on the vector, and then, and then pass that buffer to something that fills it in. And then there's an unsafe function to go and, and, and, and fill that and, you know, set the vector of the appropriate size. And I think, and I will say I found the blog entries about why you have to do that and not memory on the to be pretty interesting because it went into pretty, into pretty in-depth details about the assumptions that, the assumptions that you're allowed to make and the ones that you're not allowed to make.

15:44 And I t... Engaging with...st Com

Interviewer
Gotcha. So like, is this just official rest of the documentation or are there also been just blog posts that you found that have been helpful from the community?

Participant
Yeah, there was a particular one that I found interesting.

Interviewer
Yeah, I think you can send me the link though, it'd be great.

Participant
Yeah, maybe on it. Rust, let me see if I can. Yeah, it's, I think actually this might have been a, I just remember, I don't remember exactly what I looked at. I, I just remember Googling for MaybeUninit and then seeing a lot of discussions about when it's appropriate to use that and when it's not appropriate to use that and why this reserve and, and I don't even remember the unsafe resize in the vector was like the proper way to go about it. And it's, there's all this stuff about, you know, the, and it sounds, and I remember the material, I don't remember this, I can't find this physical material I read, but it really went into sort of the details about what Rust expects and what it doesn't. Again, coming from the C++ world, this was all like, I mean, the material itself was like not something that I was, it's not, I wasn't saying da too, but it's like, I get it. Like, I know, I, I know about, I've worked with undefined behavior and C++ and I get it.

15:29 I just r... MaybeUninit<T>

15:45 Ag... Shared Experiences

230

242 **Interviewer**
Yeah, sure. So yeah, if you can find like a, if it, if it, if it ended up being

like a specific blog post, no worries, like no pressure right now, you can always just send me an email at a, that works.

Participant
Yeah, I'll see if I can, I might not be able to find the exact article I read, but I can, I'm sure there's some discussion out there about, about like the proper way to I'm calling something that wants a buffer that fills in a buffer. I don't want to do something stupid, like filling this buffer with zeros just because, you know, like, when you've created a vector in, in C and C++, you are in Rust that you have to do that. And I was like, I mean, I could do that, but that's just wasted code. It's like, I don't want to go and just fill a buffer and be just because I can.

Interviewer
Yeah, yeah. That makes sense. So I think that covers every single question that I have. I think, right. Thanks a ton for your time. Really appreciate talking.

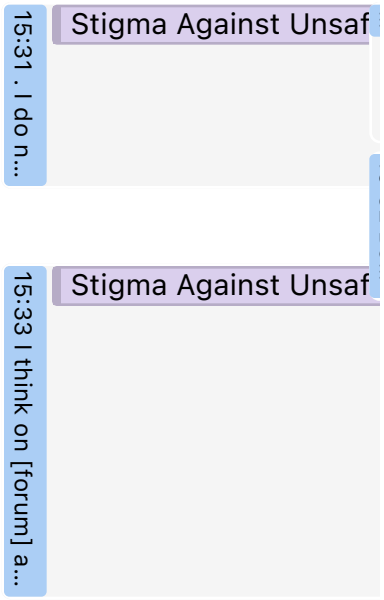
Participant
Yeah, no problem. I was, I was sort of wondering if my experiences are going to be interesting to you because I get the impression that you're typical, that the person that you're trying to engage are people that, for which unsafe Rust is this ongoing thing. And for me, it was a very specific challenge that when I, when I encapsulated it, you know, I had to do some really nasty stuff to encapsulate the problem. But now it's just this thing that is just there. And I don't think about it much.

Interviewer
I think like what we're really looking for is just diversity of experiences because there is a, there are particular like hypotheses about how unsafe Rust can be used safely and about studies of how it is used in terms of like, let's look at the code on crates.io. But there haven't been studies that just speak directly to developers who engage in unsafe Rust. So our goal for this is to be sort of a counterpoint to some of the more quantitative studies and be like, okay, we have all of these like very formal discussions about how unsafe is used and what its purposes are. Now let's compare that to what developers are actually saying. So yeah, it's, yeah, that is very valuable and very interesting. I think there's one quote in particular that you used to describe the FFI that I might want to reach back for. Well, if we ever need to quote something specifically from like in the, like, well, the transcripts will be published within the paper if we need to quote something, then we will reach out to you again to make sure that we get the words ....

Participant
So yeah, anyway. A little piece of commentary that the Rust in general is, I love it. I love being able to successfully reduce the, you know, unsafe stuff to this extremely small amount. I love it, changing it from something I have to deal with 100% of the time to something I have to deal with 1% of

15:30 A li...
Fanboyish
Local, Minimal Unsafe
Memory Safety

the time. I do not understand in the community about the obsession of getting that down to zero, because it's like, like one thing I tried to do, because I was like, hey, you would be really nifty. I could probably save a few bytes on my web assembly if I go and, if I go and find an option to make all panics be undefined behavior. Because, hey, like, I'm telling the compiler, you see a panic, you can optimize the hell out of this. And I was just like, hey, I can save a few dozen bytes. I was amazed that there's not an option to do so. And when I asked about this on, I think on [forum] a few months ago, I got a lot of, like, it was like I was proposing slaughtering children or something like this. It's like this phobia of undefined behavior is like so great that I'm like, and I'm thinking in my mind, like, I'm running in web assembly, I'm running in an environment hardened against malware. If I could go and get rid of my panics and save a few bytes, then I want to do so. And I get the benefit when I run my stuff on native code, which I'm doing anyway. And I just find it weird that there's this allergy to that out there.



Interviewer
Gotcha.

Participant
And it's just, I don't get it.

Interviewer
Gotcha. Yeah, no, that's, that's a really helpful, helpful perspective to have. So yeah, thanks.

Participant
Okay, I'll see how I can find some material about the maybe and it and it stuff and I'll send it to you.

Interviewer
Yeah, just email me at any point, no worries.

Participant
Okay, great. Thank you.

Interviewer
Yep, thank you. Bye.