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

Participant
Not some years ago, what were my motivations? Was it about the time that Go was taking off? And I guess I was really struggling with finding good models for handling concurrency in code that I was using. I had this. It was kind of this book, a seven. Let me see if I can get it.

Interviewer
No worries. I was reading this book here. The concurrency models in seven weeks. And that kind of got me thinking about it. I was trying out Go and figured out that didn't really answer my questions. And then I started looking into Rust, maybe more of a coincidence. I guess the trigger was I saw it was always the most loved language in the survey. I thought I'm going to find out at least what this is about. And in time it did actually provide a lot of answers to the problems that have been struggling with for years where concurrency and multithreading have been involved.

8:1 I guess the t…
Fearless Concurrency

Interviewer
Sure.

Participant
That has been a problem that subsequently arisen because of that.

Interviewer
Oh, gotcha. Has it been mostly the concurrency motivation then or was memory safety also a big bonus?

Participant
I guess both are sort of analogous concerns. I've been writing C and C++ for decades and I sort of didn't have a lot of memory and safety issues, but where things would sort of fall apart would typically be when multithreading was involved. So of course memory safety. Yeah, it's excellent. But it wasn't really what drew my attention at the time.

8:36 I've…
C++ is Difficult

Interviewer
Gotcha, okay, gotcha. So I guess in addition to Rust, so you were considering Go or were you already a Go developer at that point?

Participant
Well, I bought the book. I thought, well, you know, this is something new, but I kind of found at that time I was working in seismic data processing. And it wasn't really, no, it was not the right language for the job. They had a good story with the concurrency model, but for a while there I just said,

…
Data Processi…& Seria

you know, it was just a Go language, a concurrency model wrapped in a new language. There's got to my critique of it, stuck with the with the channels and go routines and that is your very much hinges on that particular model.

Interviewer
Gotcha. So just to sort of summarize, you were mostly you mentioned mostly working in C and dealing with a lot of concurrency issues with that. And then you encountered Go.

Participant
C and C++ and I was.

Interviewer
Gotcha.

Participant
I had this kind of prolonged search for a better way of doing things, one could say.

8:3... C++ is Difficult

Interviewer
Oh, okay. Could you be a bit more specific about what aspects of concurrency that you found particularly difficult in C and C++ that you thought Rust was the solution for?

Participant
Well, it wasn't immediately clear that it was that it was a solution straight away because it took me a while to get to really get into it. It was kind of a back burner project and I did many things and Rust and then it grew on me. And so the kind of the watershed moment was we had some data, large amount of seismic data that we wanted to process change another format, but I wanted to do it on demand so that when people wanted files in this particular format, they could access them as if they were in the correct format. And what I wanted to do then was I'd use the fuse user space file system in Linux to sort of generate a view of the same that the files in one format and you generate a view of the files in a different format, which was what the client wanted. And that had to serve multiple users and I kind of mocked it up in C, but I did not dare to do this multi-treaded in C and C++.

8:3 lar... Data Processi...& Seria

Interviewer
Yeah, sure.

17
33 **Participant**
And the demand came that we needed to do it and I said, okay, well, if we're going to do it, I'm going to do it in Rust. There's just no other language that will give me the confidence. There's a funny anecdote to

33

this one though, because I had developed it and the time I was kind of short to get this. So I developed it, put it in production to a client and then the client asks or reports back that the performance was abysmal. What had happened was probably sometime during development, I just limited to one worker thread. I thought it was just single-threaded performance when you'd have multiple clients trying to read these files. That explained my performance was bad. But I realized I hadn't done a lot of testing in multi-threaded scenario, but I did have some test frameworks. I could do it and I was just random access to files kind of exposed a lot of these issues. I bumped the number of worker threads to 10 or whatever number it was, ran these tests concurrently, all tests passed, and I immediately shipped this version to the client. And it worked flawlessly. There was not a single concurrency bug or anything that you'd kind of, in C and C++, you'd kind of expect that someone's down the road to report funny issues that were impossible to debug. There was none of that.

8:2 I bumped th...
Fearless Concurrency

Interviewer
Just immediately and you didn't have, there weren't, I guess I'm thinking back to my own experiences of thinking that something was perfect and then realizing it isn't like a couple weeks later. There weren't any major follow-ups

Participant
Well, I just had a lot of, I just put a lot of faith in the promises of fearless concurrency. Of course, it wasn't just blind faith because I did have some, I mean, that was the premise of doing this in Rust to begin with was that Rust would allow me to make a multi-threaded implementation with confidence.

8:39 Well, I j...
Fearless Concurrency
Tacit Knowledge

Interviewer
Gotcha. Yeah, that makes sense. So then I guess along those lines, another more broad question is, what do you use unsafe for us for?

Participant
That's a good question. So this seismic thing did work on both the data acquisition hardware, which was STM-based, just microcontroller collecting data, writing to memory card. And that was written in C and that had all sorts of the concurrency issues. It wasn't written by me. And so I'd always be in there debugging the code, finding bugs that other people couldn't really pinpoint. It was a usual suspects, forgetting to take a mutex, that kind of thing, or using the wrong locking or not using any locking at all. All the stuff that Rust would prevent you from doing. So that kind of made me shift the limit into embedded because of it, because I saw that this stuff really needs Rust to make the code robust and to avoid all these problems that I was seeing. And so a lot of the unsafe uses, just because of the ecosystem on embedded, sometimes not all the hardware stuff is implemented. So like the most common use case is that there's some particular memory map device that I need to do, configure in a certain way. And the available hardware abstraction layer doesn't provide that functionality. And then I just have to go in there and hope get the little

8:4 S...
Operating & E...dded S

8:5 . And so...
Concurrency Bug
Data Races

8:6 And so a lot...
Interacting with hardw
No Other Choice

details, you know, sort of a bit here and there. I can't show you some code, you know, what it is, if that would.

Interviewer
Yeah, that would be amazing.

Participant
You can actually just grab for unsafe just to see. Usually recording the video or such maybe be careful with what I was going to say.

Interviewer
Yeah, I delete the videos after the interview, but I do save all audio so.

Participant
Yeah. Okay, I'll just show you a particularly ugly example.

Interviewer
Share this. Post disabled participant screen sharing.

Participant
Let me fix that quick for you.

Interviewer
Where's it you're based? I mean, where are you?

Participant
Oh, I'm in [redacted] right now.

Participant
So, okay, it's for [redacted]. Okay, so this is just like you even annotated
41 with hack. So it turns out that for this particular microcontroller, the ADC is not enabled. Unless I go with some bits and particular registers. So this is kind of this this file here just initializes all the all the hardware peripherals. And we could just do a search for unsafe, you know, get a get a feeling for why that.

60

61 **Interviewer**
Okay.

62

63 **Participant**
Yeah, so here is a problem with the ownership. I don't know this is in the Arctic framework, but there's an older version may hope maybe. I'm not sure who took them and tried to just steal them instead. Yeah, this is really low level stuff where you're actually writing into the re allocating a vector table for the CPU. What's going on here. I'm not enabling a DMA for some reason. Reboot magic. So this microcontroller, it will, it doesn't

8:7 Yeah,...
Contract or Invariant
Interacting with hardw

erase the RAM during reboot. So then I have certain sequences that okay, say you want to update you want to flash other firmware or you want to go into sleep mode. And this handle is just write a magic value to RAM. And then when I boot up by check, okay, what is this this magic value. So I just use this maybe on in it. I guess you need unsafe to access that. Yeah.

8:7 Yeah, this is...
Contract or Invariant
Interacting with hardw

Interviewer
So I noticed for most of these unsafe usages I'm seeing they are commented and you have some particular rationale for it, as well as like maybe whether or not it's it's you're in incredibly caught like the confident in it like there's one that was said kind of do be a see something. Do you feel that that's like a common pattern that you see in most unsafe in the code that you interact with, or is there like the reason it's unsafe is if you don't use those registers or stuff correctly then the whole thing is going to do come crashing down in flames.

Participant
Yeah. Or lock up or something like that so it was very good. I'm not happy with writing unsafe code. You know, I try and I try to keep it at a minimum but it just like, you know, certain things tend to be unavoidable. I guess we covered all the unsafe in this one. Yeah. Okay, why is this there's some I didn't write this. So this would kind of be typical of the unsafe code that you'd see in embedded. Let me just show you other code, which in general, it's cases where you're interfacing with the hardware on a level where I mean, once you start writing magic. The hardware abstraction layer should be, you know, in some cases the hardware abstraction layer is unfinished. And so I just go around it. In other cases, even if it is in the and you can use your hardware abstraction layer it is kind of unsafe what you're doing so putting it in an unsafe block I think is entirely appropriate.

8:8 I'...
Local, Minimal Unsafe
Preference for Safety

8:9 in gener...
No Other Choice

8:10 I...
As Documentation

Interviewer
Gotcha. Gotcha. Yeah. Just looking at. I just wanted to pull up your screening survey responses for a second to double check something.

Participant
There's another code. I guess when you described the hardware being unfinished. Yeah, the abstraction layer is being unfinished.

Interviewer
Could you talk a bit more about your relationship with that abstraction layer as a developer like what, what degree, can you trust the like are the
63 guarantees that you rely on that abstraction layer to give you in order for things to be safe.

74

75 **Participant**
I think embedded how I've done just a brilliant, the embedded how working groups have done a brilliant job of, of what should we say, expressing or giving access to the hardware through rust type system. It's

...
Fanboyish

just a game changer, complete game changer for embedded development, how they've really been very carefully thinking about how, how you can enforce expectations of the hardware into the type system. So, so for instance, you know, I have a microcontroller, it's got a bunch of pins and say I want to put a serial port, configure it to certain pins. If I do that incorrectly, I'm going to get a compilation at her. You can't use those pins for this combination of, you know, I can show, let me just show it to you how, how amazing. I'll just go back to, to the same. I'm going to share. Okay. So we're in this thing of taking a bit of a shot in the dark here. Of course. You can see what I'm doing here.

8:12 It...
Fanboyish

8:11 So, so f...
If it compiles, it works

Interviewer
Yeah, if I can see everything fine.

Participant
Oh, if I just go to a serial device. Where am I? Okay, so here. So I'm, I'm trying to use, I'm using you start one and that there's some limitations on what pins that can be configured to. So if I say, okay, well, I think I'm using you start one for something else. I want to use user two instead. And a [unknown].

Interviewer
Gotcha. So would you say that this is like how just to successfully help you at avoiding unsafe code and

Participant
No, it just neatly encapsulates all the, all the hardware so that and if it's done correctly, it just gives you pretty strong guarantees that you are using the hardware correctly.

8:13 N...
"Safe" API

Interviewer
Gotcha. You know, you can try, you can fiddle with the hardware, but if you do some stupid mistake and nobody's going to, nothing is going to complain about it. It just, it's not going to work.

Interviewer
Gotcha.

Participant
But regarding on, well, there's a lot of unsafe code in here. If I go to definition here. This is how So, you know, so the hall is full of unsafe stuff, but hopefully they know what they're doing. And so, you know, that, you know, I guess, you know, that's normally unsafe code in, you know, encapsulates data structures and stuff that the Rust compiler has issues with. Yeah. But in this, in this case, I mean, in the embedded world is a little bit different that you use it to provide safe abstractions for memory mapped IO and also white compiler guarantees. 75

Interviewer
Gotcha.

Participant
So you're so typically you have macros with these tables here that defines and so this one defines sort of the legal combination of the, of the IO pins that you can use if you want to do a you are to and if you do anything illegal with just not going to compile.

8:14 So y...
Macros

Interviewer
Gotcha.

Participant
Yeah. Okay, I was trying to give them a completely different get out of the embedded world.

Interviewer
Yeah, I guess are there any use cases you have for unsafe that are not in the embedded world that are in like a different area, or is it most Um,

Participant
Okay. Share another window here is shown. Try this one. Yeah. So like this is some log handling thing. Here's a. Now, why is that does that need to be unsafe. That's memory mapping a file. So, gotcha. Why? I'm not sure maybe I'm kind of puzzling that you can't have a safe abstraction for that. Oh yeah, I guess if you encoded the structure of the mapping in your type system. This might be, you know, I'm not, I'm not sure all all my unsafe are really necessary, but I don't really do advance. I mean, I've never implemented any new data structures that rely on unsafe for it to work. And I think, you know, that's kind of the classical use case of why you would want that. And now this is an interesting one. So in this case, there was a performance critical piece of code that I managed to speed up by using AVX to intrinsics on on the Intel.

8:15 Now...
Exposing an U...e API t

8:16...
When to Encapsula

8:17 S...
Increase Performance
Intrinsics

Interviewer
Oh, gotcha, yeah.

Participant
And then all of that is unsafe. And, and it's even, you know, it's even I have a warning here. This is this thing is going to read with 31 bytes out of power just because your data size isn't bigger than.

8:18 And then all of th...
No Other Choice

Interviewer
Yeah, yeah.

Participant
So sort of it's, it's unavoidable in that case.

8:1... No Other Choice

Interviewer
Yeah.

Participant
And, and in this case, you know, so I think in the server you asked, okay, though, can we use me to check that this is that stuff I haven't, I think that kind of maybe came around and matured a bit later than than this. But, you know, I know what the problem is because I did see it seg fault. If I use it incorrectly.

Interviewer
Gotcha.

Participant
But no. So, I think this is really rare. It's very rare that you're able to write code that's considerably faster than what your compiler would normally do. Sort of have to find some set of instructions that are perfect fit for the job.

Interviewer
Gotcha.

Participant
And in this case, because you have knowledge of the potential for hardware acceleration there and you're so you it's, you can reason about how to appropriately use these. But because that burden is on you then everything has to be unsafe, because you don't have any guarantees from the type system about whether or not you're, you're using the AVX. Yeah, it's all these, it's all these Intel intrinsics, but there are rates out there that are supposed to do SIMD. So I don't know if, if David will provide me with the same, which was like granularity of instructions that are good, because they try to abstract over different CPU types. And I'm using like, I think like this MM256 shuffle API, which is a kind of a byte swapping arrangement. I don't know if that would even be in that crate. Maybe it is.

8:2... SIMD Intrinsics
Volatile Intrinsics

Interviewer
Gotcha.

Participant
It could be, it could be their safe abstraction, but you know, this stuff here I'm just reading memory out of a pointer. That's That's normally not say that can normally not be done in a safe way. So there's some use cases here. Yeah, sharing and trying to find some others also.

Interviewer
I was also curious about you mentioned. So there are two types that you generally use an unsafe context. The first was Box. And then the second one you in other you mentioned embedded PC types could you could you talk a bit about each of the use cases you have for those.

Participant
Yeah. So the one one a common one you'll find in this code here. There might be So what have I done here I'll just share my screens with you. This looks like the one. And that's kind of just a transportation of data types. Which I think sometimes there are safe abstractions for it. And this is not typically how I would do it, but this is a transportation of allocated data one type to another. Yeah. If you want to do serializing the serializing of data at maximum speed. Yeah, we'd be tempted to do it. There might be. There might be crates that help you do it. But I really don't know how fast they are. This I know is going to be fast.

8:2… Transmute

Interviewer
Gotcha, so it's like if. Is it that you feel that you so it is sort of to summarize you, you know that you have a certain guarantee of performance when using just these baseline operations. And it's also extra investment to be able to use an additional library that has some different days. So you're not really sure if it's going to give you the performance. Gotcha.

Participant
So performance. And this is I think a lot of this stuff is related to translation of seismic data which was performance critical because we're processing. Yeah, we had hundreds of terabytes of data that Not sure how much the CPU was a bottleneck, but you know, everything was Do you want to take any chances of rating bottlenecks where they're less than any for it. Yeah, so it's you say it's it's easier to Just use a solution you know is performant, even if it is unsafe versus spending extra effort to invest in some framework that might give you a nice encapsulation but Might not this is also so this is also you know several years old so the ecosystem has was less mature at this time you know

8:22 Yeah, s… Difficult to Encapsulat

Interviewer
Gotcha.

Participant
It's possible that the sort of cost benefits right off with performance is different now than it was when you originally were writing that. Yeah, but but also the cost benefit thing you know it will be a one one a one time investment defined. If you know Just trying to find out when when these files are from At least 2020 maybe be earlier. But I would know it approaches okay. This way okay I don't, I don't like writing unsafe code. Just gives you doesn't give you good vibes. And then there's also this you

8:2… Stigma Against Unsafe

know if you, if there is a problem with your code the first place you start looking is in your unsafe. And you're all the better off for not using unsafe because that means there are even fewer places you have to look. Just I try and avoid it and a lot of my libraries and stuff I'll explicitly do deny unsafe. If I'm writing a library and I don't really see any need for this to be unsafe or I wanted. Yeah, I don't want to worry about problems arising from it so then rather do and it's, you know, it is reasonable that you will take some performance hit for just doing things safely. I've not tried to look for every optimization that's out there. And then that's, you know, that that would be my default position on that.

8:24 Just...

Exposing a Safe API

Interviewer
Gotcha. Yeah, makes sense. So, one other thing I wanted to ask about to you mentioned that you call foreign. So you, you operate sort of going across the FFI both directions it seems like so you mentioned that you call C functions from rust, but then you also call rust functions from C C plus plus and Python.

Participant
Yeah.

Interviewer
And in particular you've used like bind gen and pi o three for pi o three you can avoid a lot of the unsafe. I was curious like what's what's your experience using each of those those tools.

Participant
So I will have to say PyO3 is just brilliant. I know for a long time it was relying on some features that were only available in a nice way. And so it was kind of not. Yeah, the option there was some other option some other way of doing it but I just love the way higher three allowed you to bridge rust and and Python. And I used it for a number of things and it just just the performance is just amazing the performance increase you can get by a lot of it was log processing. Or are processing log data in particularly in the space efficient formats which also meant that they were slower to read and then Python really didn't do a good job of that. But you know if I could do all the bits manipulation stuff in in rust and then present the data in a pandas data frame or whatever you'd want to have at the higher level. Yeah, it's definitely a way of doing things and if Yeah, I would sort of be my, my go to tool for speeding things up in Python. Find out where where's Python burning all the CPU, kind of go in there, add some rust.

PyO3

Interviewer
Okay, so those were the particular spots. I guess what were you so you were you profiling and then using profiling data to inform where you add in rust, like what was your process for finding the hotspots.

Participant
More of gut didn't think that this.

Interviewer
Gotcha.

Participant
This is not something you want to do in Python you don't want to do lots of big manipulation and make sense. And then I guess with these tools. I think I have some of these tools. You're just curious now if I used any unsafe in it. Oh, just for the memory mapping. So just the memory mapping but everything else with that Python interoperation was all just a safe interface. Yeah.

Interviewer
Wow. So then with was it was that I'm assuming it was a bit different when you were doing foreign function calls to C though right.

Participant
See if I can show you some other. Like one, the most recent use case is. Yeah, it's basically a library that we were provided with just object code for. And then we have calling into it is. Gotcha. Safe block. And also, well, I can.

Interviewer
No worries if you can't find specific lines here.

Participant
Yeah, they're just like one line. That was just a one liner. But I do have something here. Let me show you. That's good. So this is a library that is provided by ST.

Interviewer
Gotcha.

Participant
And let me see. Should be unsafe in here. Or have I used bind might have used bind again for this. Let me see. No unsafe here. How did I do this? Yeah, awful transmute. Gotcha. Oh, I guess you're calling. Yeah, so I don't I don't one of my doing. Yeah, there's a. What in the world is going on here. I really. Don't know what I've done here.

8:26... Transmute

Interviewer
I've definitely been in the same place before.

Participant
I mean, it could be. I'm being passed a void pointer to a rust. Last in callbacks or something weird like that. I'm having problems. No worries.

Opaque Pointer

I'm curious if I did a generic binding. How did I do this? Yeah. I can't. I can't answer this one. No worries.

Interviewer
I guess I was also curious though with. Is your. You've had uses of foreign functions going both ways from rust and to rust. Have you had any particular challenges there since you're, you're working in rust, which has one set of assumptions about memory and then having that work with Python and C and C plus plus, which have their own set of assumptions. Have there been challenges that you face that have been unique to that interaction?

Participant
If you're approaching it at the level of a C program or, you know, then your expectations are really lowest. You're just a happy you can call into C and things doesn't just catch fire and explode in your face. Of the bat. I haven't tried C plus plus because that's an entirely different thing that I've faced interfacing with. So I would, you know, tend to avoid that. And maybe if I had to, maybe, well, there is a, there is this C plus plus bindgen thing. Maybe I would try it, but, you know, for at most five minutes and then I'd write a, a C wrapper or something. C plus plus or, you know, just to avoid all that horrible namespace or a man name mangling and all that fun.

Interviewer
Gotcha.

Interviewer
And then [unknown] up.

Interviewer
I guess another question was this one is a bit more broad, I think, and could describe any usage of unsafe, but, um, describe a bug that you faced related to unsafe rust or to like, in, in this case, the interaction between rust and some of the unsafe code you have and C or C plus plus and sort of just describe the bug and then maybe like how, how

Participant
I haven't, I haven't had. I haven't had any, any, uh, shows a lurking bugs.
Uh, you know, what you typically have and see, but it looks good.
Compiles, runs. Yeah. And if there are problems, you can just spot them
immediately. That's my, just my experience. I think maybe once I had a
problem where, uh, there was some alignment issue, but that was tied to
this. And that was tied to the, the AVX code. And so there were just
certain data structures that had some, I don't know if I could do that safer,
169 but I think I had to explicitly align, uh, data so they would. And that was a
bit, bit random, you know, because you don't know the, you, uh, you're
not always sure about what alignment your data is going to get when it's
allocated.

8:2... If it compiles, it works

8:29 . I think maybe o... Allocation & Alignmen
Improper Alignment

Interviewer
Gotcha.

Participant
Uh, I guess related to this, then, um, you also mentioned using, uh, just for the, for bug finding, um, both clippy and Valgrind. Could you talk a bit more about how those tools have been helpful for you?

Interviewer
Uh, well, clippy is not. Uh, I don't know. Once a clippy really helps you find bugs. Uh, I don't know why that, why I checked that it helps clean up your code. Uh, as, uh, what I find it most, uh, what I really like about clippy, now I'm trying to catch up on, on new features and, uh, they come with a different compiler releases, but I guess that don't have the capacity to do a deep dive in every release, but clippy kind of helps me to catch up with it. Sometimes, you know, there's a new feature out there and they'll make sure that they tell you about it. That, yeah, you can do this a little bit easier now with the, and so it's more like code hygiene. I would say, uh, clipped in and, uh, it doesn't, I don't think it ever tells you to do things, do something different. It just tells you a different way you can do the same thing. And I guess that's the point of any lint is not going to, it shouldn't change the behavior of your code. It's no longer a lint. Yeah. But the ball trend, uh, especially in these, uh, as good, especially in those cases where I, uh, I, uh, have C and rust interactions. Uh, to to make sure that there's no memory leaks. That's kind of a typical, typical problem. Gotcha.

8:30 Uh, as,...
Clippy for Learning Ru
Shifting Ground

Interviewer
So it's, it's have, have those been the typical errors that you found in, in having an interoperation between rust and other languages?

Participant
Well, I haven't been using open SS on the outside. I just like notorious for leaking. It's more, I guess it is, it's more like I do it out of habit. But when I've done a project like that and I have a, you know, I'm near to completion or feature complete, then I will also add some tests in Valgrind. Uh, it's very simple. And you know, you're just starting to stop the program and then you'll catch in it and memory leaks or, you know, illegal reads and writes. Um, so that's that's not, not much more to say to that.

8:31 But...
Dynamic Analysis Safe
Valgrind

Participant
Recently started using a fuzzing cargo fuzz, which I just had to say is amazing. That's really helped me find bugs that I wouldn't find any other way. Gotcha.

8:3...
Fuzzing

Interviewer
And, uh, like what, what, what, what's been your, I guess what do we,

what have you found really amazing about it? Like what kind of bugs has it shown you in, in code?

Participant
Well, the thing is, you know, it seems like it's actively trying to modify the data in a way that brings you into all your branches and nooks and crannies and kind of triggers. Uh, if you have a comp, you know, a piece of code with high cyclomatic complexity, it's somehow manages to navigate its way into all your little edge cases. And it's just amazing to see how fast it's able to, to generate fuzz input to, I mean, you can put it, put a bomb in there, just do a panic and it's going to find, find it no matter where in your branch tree you have it.

Interviewer
Gotcha. Okay. Yeah. So we have, have panic's been like is.

Participant
No, it's not panic. It's, uh, I had to, like the most reason when I found is where I'm parsing data from a piece of hardware. Uh, and, uh, there was sort of an edge case there. I'm not sure it would happen in real life, but it was like one of the, uh, because it would have been probably sanitized at the higher level, but at the lower level, it didn't out of bounds. Um, access. Gotcha. That was panic. And I caught that. And, uh, what was really nice about it is, uh, uh, that this piece of code, I can't, I actually have to rewrite the logic to see. Uh, because it's going into a part of the product that is going to be I, IEC 61508, a SIL 2 certified. And there's no certified rust compiler. Gotcha. All right. So when we are having all the, all this rigor, uh, uh, both what is required from the process, but in addition, I just, you know, throw everything and the kitchen sink at the code because I don't want to have to deal with bugs in it.

Interviewer
Yeah. Yeah.

Participant
And I even, even got my hand at, uh, playing with Kani. Uh, the formal, uh, just to see if that could help me because it was, uh, and the particular problem, the problem there was I, uh, there's some data that needs to be processed in an interrupt handler. Uh, it can do several passes. Uh, or it might process it in, in, you know, in many smaller chunks. It should only do it in two passes, do it in two passes. And we have to, uh, make a statement about that the execution time of this algorithm isn't longer or at least, you know, say, say, okay, this is only going to take five microseconds or whatever. And then Kani was actually kind of nifty there because I could relatively easy set up or have it formally prove that regardless of the input, you're not going to do this more than two times and your execution time is going to be bounded by that. But, but, um, yeah.

8:33 And...

Formal verification

196
207

Interviewer
Awesome. So I think the final question is, I guess, are there problems that you encounter related to unsafe code that you feel like your development tools can't solve for you and that you feel like a new tool or some improvement to an existing tool would help.

Participant
Yeah. No. Like, well, I mentioned, I mentioned, you know, if you're doing AVX to, uh, uh, Intel intrinsic code. Uh, I know the, the, so the Rust compiler, if you don't say that you have this particular hardware architecture that supports all of these instructions, exactly as [unknown] knows, it's going to implement it in emulated instructions or just do it the slow way. See, you have, have to actually, have to actually, uh, uh, make sure that your compiler options are correct to get the pure assembly code.

Interviewer
Oh, sure. Yeah. Okay.

Participant
Now that tells me, uh, that if I can do all of this in, in, uh, where all these instruction instructions are emulated and I don't know how it's done, maybe it should be possible for Miri to check the code because you're not really dependent on having, uh, the hardware instruction, uh, uh, implementation of all of these because there is a emulation. So that, you know, that's just a thought that came to me when we went through it and why didn't I check this in Miri? And I think it could be doable.

8:35 Now th...

Wants Solution