New issue

Jump to bottom

ComponentBytes is unsound #35

Closed Tracked by #327 **HeroicKatora** opened this issue on Jun 14, 2020 · 7 comments

HeroicKatora commented on Jun 14, 2020 • edited

The trait assumes that an arbitrary type `T: Copy + 'static` can be viewed (and even modified!) as a byte slice. That is super unsound. Here's an example of causing UB with it:

```
let component: &'static str = "Hello, World!";
let mut not_rgb = [component; 3];
let bytes = FromSlice::as_rgb_mut(&mut not_rgb[..]).as_bytes_mut();
// Just write over this reference internals, lol.
bytes[0] += component.len() as u8;
// XXX: on most architectures this points after the original static now
// e.g. into some different static or executable memory
println!("{}", not_rgb[0]);
```

kornelski commented on Jun 15, 2020 Owner

Thanks for reporting this.

I suspected it's dodgy to rely just on `Copy + 'static`, but forgot that `_mut` makes it so much worse.

I've changed it to require `Plain` bound from https://lib.rs/plain crate. It's a bit unfortunate to pull in a dependency for this, but I don't want to invent my own, and there isn't a clearly better crate for this (e.g. bytemuck is a bit more popular, but it also seems larger in scope).

kornelski commented on Jun 15, 2020 Owner

I plan to release it with the same semver version. It is technically breaking, but it affects only a special case of using a combination of custom wrapper type and `as_bytes()`. OTOH I want existing users to get the fix, and don't want to cause "libcpocalypse" breaking interoperability.

HeroicKatora commented on Jun 15, 2020 • edited Author

The unsoundness of inspecting arbitrary uninitialized padding exists in every version since `0.2`. If this were to avoid a break from `1.0` to `2.0` or similar I would understand it more but the same interop apocalypse is a large risk. There are a *lot* of users that could all be potentially relying on the trait bound, and requiring a switch shows serious effort to fix this long-standing soundness issue.

Also, seriously consider (for more than one day) publicly depending on a pre-`1.0` crate in your public interface. It's very much an interop risk on its own.

And lasty, consider having a full PR with discussion of the soundness fix because I'm not convinced it is fully fixed based on the commit itself and requiring *another* potentially breaking interface change will be more devastating.

kornelski commented on Jun 15, 2020 • edited Owner

I have bumped the version a few times in the past when the crate was used just by me, but now it has enough users (including major projects that are not public) that version bump would be disruptive.

I have missed the boat in calling it 1.0 sooner. At this point if I release 1.0, it will be just an empty crate re-exporting 0.8. That could be another chance to improve the as_bytes trait.

HeroicKatora commented on Jul 1, 2020 Author

So after a good bunch of consideration, I think I've come to terms with the breaking change at some point. At the very least it will actively flag potential UB in dependents even if it introduces some breakage. We sadly don't have `std` powers of phasing things out. (I agree the `std` reserves the right for breaking unsoundness fixes but most of these issues are either discovered fairly quickly or receive focussed attention with transition plans, crater runs, etc.)

There is a potential alternative, step-by-step plan: First add a deprecation marker for the whole trait, and instead of replacing it add a new one. Then you could at a later release, after hopefully some dependencies that used it have taken note, change the method implementations to *panic* (this is a sound implementation with no risk of UB anymore but keeps the interface), and only then in yet another release remove it. You of course don't have to take that whole marathon but maybe it provides some fresh consideration for managing this situation.

HeroicKatora mentioned this issue on Jul 3, 2020

DecompressScanlines::read_scanlines is unsound ImageOptim/mozjpeg-rust#10

Closed

Shnatsel mentioned this issue on Jul 3, 2020

File advisories for vulnerabilities with upcoming fixes rustsec/advisory-db#327

Closed

5 tasks

kornelski referenced this issue on Jul 6, 2020

`Use Plain trait for ComponentBytes soundness` 2691083

kornelski commented on Jul 6, 2020 Owner

Fixed in 0.8.20

kornelski closed this as completed on Jul 6, 2020

carnil commented on Aug 29, 2020 • edited

This issue (https://rustsec.org/advisories/RUSTSEC-2020-0029.html) has been assigned CVE-2020-25016

Repository owner deleted a comment on Sep 11, 2020

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

3 participants