New issue

Unsound issue while converting bytes to utf8 str #60

Closed #61

CXWorks opened on Oct 4, 2024

Hi, thanks for your time to read this issue. Our static analysis tool found there might be an unsound issue in your `unsafe_as_str` converting the bytes to utf8 str:

obfstr/src/lib.rs
Lines 202 to 208 in `ec1a20b`

```
202     pub fn unsafe_as_str(bytes: &[u8]) -> &str {
203         // When used correctly by this crate's macros this should be safe
204         #[cfg(debug_assertions)]
205         return str::from_utf8(bytes).unwrap();
206         #[cfg(not(debug_assertions))]
207         return unsafe { str::from_utf8_unchecked(bytes) };
208     }
```

As mentioned in the comments, this may introduce invalid utf8 conversion and producing an invalid value, which is considered as undefined behaviors in Rust. We expect either to mark the whole function as `unsafe` or leverage the safe verison to convert because this library can take raw bytes from user and missed the validation of utf8. As a reference, the safe version of the utf8 conversion in std is:

https://github.com/rust-lang/rust/blob/3002af6cb643138839537f6fd0265162610fdbbe/library/core/src/str/converts.rs#L131-L140

Could you please help us double check the potential probelm? Thanks again for your time.

CasualX on Oct 5, 2024 Owner

I suspect this is an AI bot?

This function is intended to be used by the crate's macros only. Users of the crate should never use this function, hence why it is hidden from the docs. It is required to be public due to how macros work.

CasualX closed this as completed on Oct 5, 2024

CXWorks on Oct 5, 2024 Author •••

Hi,

Thanks for your quick response. I am not an AI bot 😄 but I do use a template to report bugs.

For this unsound issue, it's because user has the access to control the input, and in the debug mode, an utf8 validation is done but in release mode, this validation is skipped. So following code will have different behaviors in debug mode and release mode, and in release mode it's considered as an undefined behavior(invalid value) in Rust:

```
#[forbid(unsafe_code)]
use obfstr::unsafe_as_str;

struct A{}
impl A {
    pub const fn as_bytes(&self) -> &[u8] {
        // an invalid utf-8 encoding
        [0xC0, 0x80].as_slice()
    }
}

fn main() {
    println!("{:?}", obfstr::obfstr!(A{}))
}
```

Thanks again for your patience.

CasualX on Oct 5, 2024 Owner •••

Oh I see, I overlooked that you could use a custom type.

It is important that no string validation is done at runtime for performance, the check being done when debug_assertions is to catch any accidents just in case. And it will catch your example, you have to *really* go out of your way to trigger this UB 😅

The bytes version asserts that the type is `&[u8]` with this line: `_OBFBYTES_STRING: &[u8] = $s;` .

Would it be enough that the input `$s` string passed to the macro is asserted to be a `&str` ? You could still call the hidden exported `unsafe_as_str` (it has unsafe in the name after all...) but it will be harder to misuse.

👍 1

CasualX reopened this on Oct 5, 2024

CasualX mentioned this on Oct 5, 2024

Restrict obfstr! argument type to string slices. #61

CXWorks on Oct 5, 2024 Author

Thanks for your help, the patch looks great.

CXWorks closed this as completed on Oct 5, 2024

CasualX on Oct 5, 2024 Owner

Published, thanks for the report!

❤️ 1

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

Code with Copilot Agent Mode

Restrict obfstr! argument type to string slices.
CasualX/obfstr

Participants