



Products

Services

Publications

Resources

What's new

Follow @Openwall on Twitter for new release announcements and other news

[<prev] [next>] [<thread-prev] [thread-next>] [day] [month] [year] [list]

Message-ID: <877d0cl4md.wl-neal@walfield.org>

Date: Thu, 03 Nov 2022 11:22:18 +0100

From: "Neal H. Walfield" <neal@...field.org>

To: oss-security@...ts.openwall.com

Subject: Re: Re: OpenSSL X.509 Email Address 4-byte Buffer Overflow (CVE-2022-3602), X.509 Email Address Variable Length Buffer Overflow (CVE-2022-3786)

On Wed, 02 Nov 2022 13:03:31 +0100,

Alex Gaynor wrote:

> In Rust, assuming you wrote normal safe Rust[0], and you had code that  
> overran a buffer on the stack, you'd get a panic() -- which is roughly  
> an abort (there's even a mode where it literally is an abort. By  
> default it unwinds and runs destructors and such). As a general rule,  
> bounds check issues aren't caught at compile time (in contrast with  
> temporal safety, which mostly is enforced at compile time.)

If you are careless, then this can indeed result in a panic, but that is not inevitable. Here's how I'd copy a buffer in Rust:

```
fn main() -> Result<(), anyhow::Error> {
    let mut dest = vec![10; 0];
    let source = vec![10; 1];
    let dest_offset = 0;
    let source_offset = 0;
    let len = 11;

    dest.get_mut(source_offset..source_offset+len)
        .ok_or(anyhow::anyhow!("Index out of bounds"))?
        .copy_from_slice(
            source.get(dest_offset..dest_offset+len)
                .ok_or(anyhow::anyhow!("Index out of bounds"))?
        );

    Ok(())
}
```

<https://play.rust-lang.org/?version=stable&mode=debug&edition=2021&gist=b7981319d0629bfc83cc29f23d43a3be>

This returns an error when you attempt to read past the end of the source buffer (the first `ok\_or`) or write beyond the end of the destination buffer (the second `ok\_or`). The caller can catch or propagate any error in the usual Rust way (e.g., by using the `?` operator to return the error to its caller).

Unfortunately, `copy\_from\_slice` will still panic if the slices are not the same length.

[https://doc.rust-lang.org/stable/std/primitive.slice.html#method.copy\\_from\\_slice](https://doc.rust-lang.org/stable/std/primitive.slice.html#method.copy_from_slice)

:) Neal

Powered by [blists](#) - more mailing lists

Please check out the [Open Source Software Security Wiki](#), which is counterpart to this [mailing list](#).

Confused about [mailing lists](#) and their use? [Read about mailing lists on Wikipedia](#) and check out these [guidelines on proper formatting of your messages](#).