RUSTSEC-2020-0059

MutexGuard::map can cause a data race in safe code

History · Edit

Reported	October 22, 2020
Issued	October 30, 2020 (last modified: October 19, 2021)
Package	futures-util (crates.io)
Type	Vulnerability
Categories	thread-safety
Keywords	#concurrency #memory-corruption #memory-management
Aliases	CVE-2020-35905
Details	https://github.com/rust-lang/futures-rs/issues/2239
CVSS Score	4.7 MEDIUM

CVSS Details

Attack vector	Local
Attack complexity	High
Privileges required	Low
User interaction	None
Scope	Unchanged
Confidentiality	None
Integrity	None
Availability	High

CVSS Vector	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H
Patched	`>=0.3.7`
Unaffected	`<0.3.2`

Affected Functions	Version
`futures_util::lock::MutexGuard::map`	`>=0.3.2`

Description

Affected versions of the crate had a Send/Sync implementation for MappedMutexGuard that only considered variance on T, while MappedMutexGuard dereferenced to U.

This could of led to data races in safe Rust code when a closure used in MutexGuard::map() returns U that is unrelated to T.

The issue was fixed by fixing `Send` and `Sync` implementations, and by adding a `PhantomData<&'a mut U>` marker to the `MappedMutexGuard` type to tell the compiler that the guard is over U too.