RUSTSEC-2020-0072

GenericMutexGuard allows data races of non-Sync types across threads

Reported	October 31, 2020
Issued	November 19, 2020 (last modified: October 19, 2021)
Package	futures-intrusive (crates.io)
Type	INFO Unsound
Categories	memory-corruption thread-safety
Keywords	#concurrency
Aliases	CVE-2020-35915
Details	https://github.com/Matthias247/futures-intrusive/issues/53
CVSS Score	5.5 MEDIUM
CVSS Details	

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

CVSS Vector	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H
Patched	`>=0.4.0`

Description

`GenericMutexGuard<T>` was given the `Sync` auto trait as long as `T` is `Send` due to its contained members. However, since the guard is supposed to represent an **acquired lock** and allows concurrent access to the underlying data from different threads, it should only be `Sync` when the underlying data is.

This is a soundness issue and allows data races, potentially leading to crashes and segfaults from safe Rust code.

The flaw was corrected by adding a `T: Send + Sync` bound for `GenericMutexGuard`'s `Sync` trait.

This bug is similar to one in `std::sync::Mutex`.