RUSTSEC-2021-0013

History · Edit

Soundness issues in `raw-cpuid`

Reported	January 20, 2021
Issued	January 24, 2021 (last modified: August 22, 2021)
Package	raw-cpuid (crates.io)
Type	Vulnerability
Categories	memory-corruption denial-of-service
Aliases	CVE-2021-26306 CVE-2021-26307
Details	https://github.com/RustSec/advisory-db/pull/614
Patched	`>=9.0.0`
Affected Architectures	`x86` `x86_64`

Description

Undefined behavior in `as_string()` methods

`VendorInfo::as_string()`, `SoCVendorBrand::as_string()`, and `ExtendedFunctionInfo::processor_brand_string()` construct byte slices using `std::slice::from_raw_parts()`, with data coming from `#[repr(Rust)]` structs. This is always undefined behavior.

See https://github.com/gz/rust-cpuid/issues/40.

This flaw has been fixed in v9.0.0, by making the relevant structs `#[repr(C)]`.

`native_cpuid::cpuid_count()` is unsound

`native_cpuid::cpuid_count()` exposes the unsafe `__cpuid_count()` intrinsic from `core::arch::x86` or `core::arch::x86_64` as a safe function, and uses it internally, without checking the safety requirement :

> The CPU the program is currently running on supports the function being called.

CPUID is available in most, but not all, x86/x86_64 environments. The crate compiles only on these architectures, so others are unaffected.

This issue is mitigated by the fact that affected programs are expected to crash deterministically every time.

See https://github.com/gz/rust-cpuid/issues/41.

The flaw has been fixed in v9.0.0, by intentionally breaking compilation when targeting SGX or 32-bit x86 without SSE. This covers all affected CPUs.