mozilla / **authenticator-rs** ✓

Rust library to interact with Security Keys, used by Firefox

#u2f #hid #rust

194 commits | **3** branches | **6** releases | **14** contributors | MPL-2.0

Branch: **master** ▾ | v0.2.6 | Create new file | Find File | Clone or download ▾

jcjones Merge pull request #91 from Eijebong/forgotten-stuff … Latest commit 674fd0b on May 22

examples	Issue #50 - Rename project to `authenticator-rs` with the package nam…	6 months ago
fuzz	Issue #50 - Rename project to `authenticator-rs` with the package nam…	6 months ago
src	Add a missing constant and bump version	4 months ago
.clippy.toml	Fix #35 - Run Clippy at TravisCI, and clean up Clippy warnings (#70)	11 months ago
.gitignore	Fix #35 - Run Clippy at TravisCI, and clean up Clippy warnings (#70)	11 months ago
.travis.yml	clippy updates for rust 1.31.0	10 months ago
Cargo.toml	Add a missing constant and bump version	4 months ago
LICENSE	Remove white space at the beginning of the license	2 years ago
README.md	Update with new repo location	5 months ago
rustfmt.toml	Run rustfmt, and configure Travis to enforce rustfmt.	2 years ago

A Rust library for interacting with CTAP1/CTAP2 Security Keys

build passing | maturity release

This is a cross-platform library for interacting with Security Key-type devices via Rust.

- **Supported Platforms**: Windows, Linux, FreeBSD, and macOS.
- **Supported Transports**: USB HID.
- **Supported Protocols**: FIDO U2F over USB. CTAP2 support is forthcoming, with work being done in the **unstable** `ctap2` branch.

This library currently focuses on USB security keys, but is expected to be extended to support additional transports.

Usage

There's only a simple example function that tries to register and sign right now. It uses env_logger for logging, which you configure with the `RUST_LOG` environment variable:

```
cargo build --example main
RUST_LOG=debug cargo run --example main
```

Proper usage should be to call into this library from something else - e.g., Firefox. There are some C headers exposed for the purpose.

Tests

There are some tests of the cross-platform runloop logic and the protocol decoder:

```
cargo test
```

Fuzzing

There are fuzzers for the USB protocol reader, basically fuzzing inputs from the HID layer. There are not (yet) fuzzers for the C API used by callers (such as Gecko).

To fuzz, you will need cargo-fuzz (the latest version from GitHub) as well as Rust Nightly.

```
rustup install nightly
cargo install --git https://github.com/rust-fuzz/cargo-fuzz/

cargo +nightly fuzz run u2f_read -- -max_len=512
cargo +nightly fuzz run u2f_read_write -- -max_len=512
```