Why fuzz Rust code

Pierre C

July 2, 2019

Parsers everywhere

- Parsers: your #1 CVE provider since the 70s
- A kitten dies for every C parser that is written
- Use case: you need to parse an insane format (DER)
  - You've written a Rust parser
  - Language is (mostly) memory safe
  - (Of course) You have unit tests
- Are we safe yet?

TLDR; no

- Memory safety is not sufficient
- For ex: handling incomplete reads, error handling, etc.
- Most safe languages will panic (abort) on some errors
  - e.g: invalid array index
- Crashing is *not* an elegant method to handle an error
- Lazy solution: fuzzing!

fuzzing in rust

- **Very** easy in Rust
- Two main projects: `cargo-fuzz` (based on libFuzzer) and `honggfuzz`
- Based on instrumentation + coverage

Fuzzing tools

```
$ cargo install cargo-fuzz
$ cargo +nightly fuzz add fuzzer_parse-der
$ vi fuzz/fuzz_targets/fuzzer_parse_der.rs
```

Heat the Planet

Just call the targetted function:

```
#[export_name="rust_fuzzer_test_input"]
pub extern fn go(data: &[u8]) {
    let _ = der_parser::parse_der(data);
}
```

Run with 24 processes:

```
$ cargo +nightly fuzz run --jobs 24 --release
    fuzzer_parse_der
```

Pro tip: don't use your laptop. Even less if it's on your knees

Enjoy

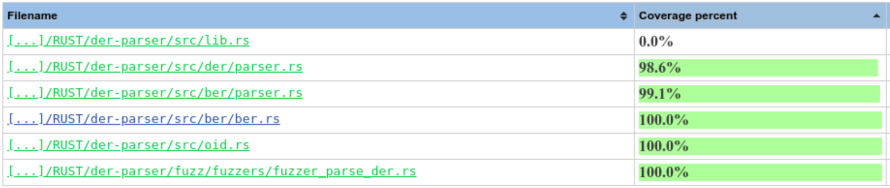
- Artefacts (crashing inputs) goes to a separate directory
- Process: run, fix bug, run again, . . .
- Lots of other tips (see link at last slide)

Common errors

- Debug or unfinished code, like unimplemented! and panic! calls
- Out of range accesses, like `array[i]`
- Integers overflows/underflows, like `base + offset`
- Stack overflows, unbound recursions
- Crashes in `unsafe` code
- Direct calls to `std::process::exit`
- Timeouts and functions that take too long

Bonus: visualize code coverage

Use `kcov` with all corpus elements:



Filename	Coverage percent
[...]/RUST/der-parser/src/lib.rs	0.0%
[...]/RUST/der-parser/src/der/parser.rs	98.6%
[...]/RUST/der-parser/src/ber/parser.rs	99.1%
[...]/RUST/der-parser/src/ber/ber.rs	100.0%
[...]/RUST/der-parser/src/oid.rs	100.0%
[...]/RUST/der-parser/fuzz/fuzzers/fuzzer_parse_der.rs	100.0%

Conclusion

- Please stop writing parsers in C (or C++, etc.). It's 2019!
- Please *do* test your programs (unit tests etc.)
- Fuzzing is useful, even for memory safe languages
- Share the corpus
- Remember that fuzzing is *not* enough to prove absence of bugs
- `https://www.wzdftpd.net/blog/rust-fuzzers.html`