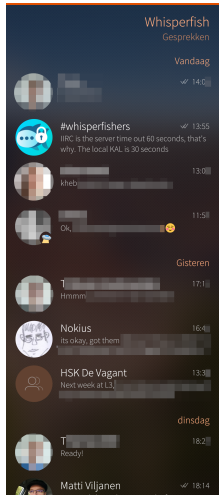
RustIEC - Vulnerabilities in Rust programs

RustIEC team

December 12, 2023

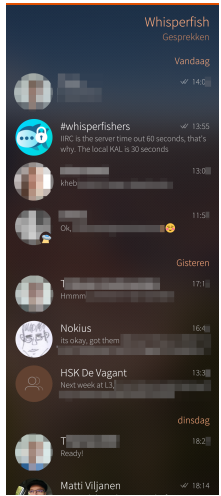
CONTEXT



Whisperfish:

- Client for Signal
- Written in Rust

CONTEXT

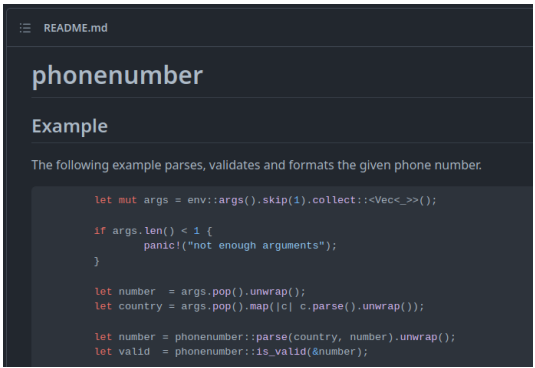


Whisperfish:

- Client for Signal
- Written in Rust
- Requires phonenumber parsing
- Requires blurhash parsing

SOME CRATES

PHONENUMBER-RUST


README.md

phonenumber

Example

The following example parses, validates and formats the given phone number.

```
let mut args = env::args().skip(1).collect::<Vec<_>>();

if args.len() < 1 {
        panic!("not enough arguments");
}

let number  = args.pop().unwrap();
let country = args.pop().map(|c| c.parse().unwrap());

let number = phonenumber::parse(country, number).unwrap();
let valid  = phonenumber::is_valid(&number);
```

- Parse phonenumbers from different formats
- Format phonenumbers in different formats

SOME CRATES

PHONENUMBER-RUST

README.md

phonenumber

Example

The following example parses, validates and formats the given phone number.

```
let mut args = env::args().skip(1).collect::<Vec<_>>();

if args.len() < 1 {
        panic!("not enough arguments");
}

let number  = args.pop().unwrap();
let country = args.pop().map(|c| c.parse().unwrap());

let number = phonenumber::parse(country, number).unwrap();
let valid  = phonenumber::is_valid(&number);
```

- Parse phonenumbers from different formats
- Format phonenumbers in different formats

The Whisperfish team took over maintenance in 2023.

SOME CRATES

BLURHASH-RS

SOME CRATES

BLURHASH-RS

README.md

blurhash-rs

Encoding

```
use blurhash::encode;
use image::GenericImageView;

fn main() {
  // Add image to your Cargo.toml
  let img = image::open("octocat.png").unwrap();
  let (width, height) = img.dimensions();
  let blurhash = encode(4, 3, width, height, &img.to_rgba().into_vec());
}
```

Decoding

```
use blurhash::decode;

let pixels = decode("LBAdAqof00WCqZj[PDay0.WB}pof", 50, 50, 1.0);
```

- Parse blurhash strings
- Generate blurhash strings

SOME CRATES

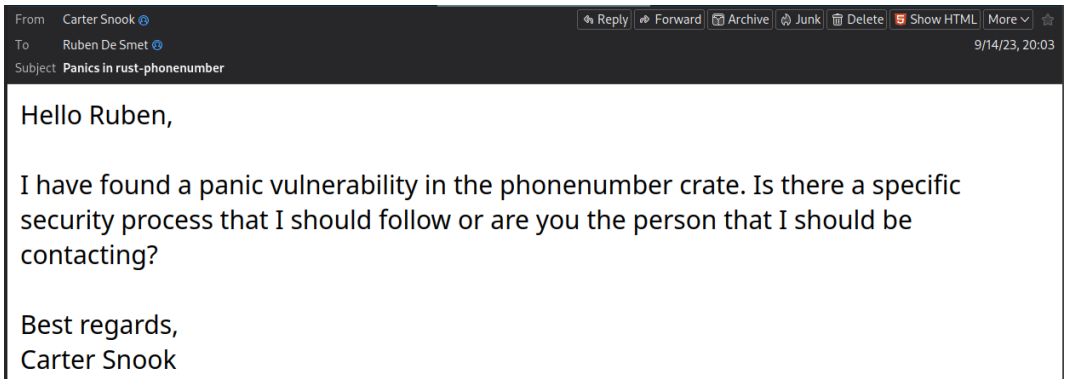
BLURHASH-RS

- Parse blurhash strings
- Generate blurhash strings

The Whisperfish team took over maintenance in 2023.

A TROUBLING EMAIL

SEPTEMBER 14 2023



From Carter Snook
To Ruben De Smet
Subject **Panics in rust-phonenumber**

Reply | Forward | Archive | Junk | Delete | Show HTML | More

9/14/23, 20:03

Hello Ruben,

I have found a panic vulnerability in the phonenumber crate. Is there a specific security process that I should follow or are you the person that I should be contacting?

Best regards,
Carter Snook

A TROUBLING EMAIL (CONT.)

SEPTEMBER 14 2023

```rust
#[test]
fn vuln0() {
    // Just make sure these don't panic.
    let _ = rfc3966::phone_number(".;phone-context=");
    let _ = crate::parse(None, ".;phone-context=");
}
```

This code panics...

A TROUBLING EMAIL (CONT.)

SEPTEMBER 14 2023

```rust
#[test]
fn vuln0() {
    // Just make sure these don't panic.
    let _ = rfc3966::phone_number(".;phone-context=");
    let _ = crate::parse(None, ".;phone-context=");
}
```

This code panics… and since `rust-phonenumber` is meant for untrusted input,

this is a vulnerability.

A PARSING VULNERABILITY

Parsing RFC3966-style phone number URI's like

`tel:863-1234;phone-context=+1-914-555`

```
params
    .as_ref()
    .and_then(|m| m.get("phone-context"))
    .map(|&s| if s.as_bytes()[0] == b'+' { &s[1..] } else { s })
```

What happens with this?

`tel:863-1234;phone-context=`

FIXING THE VULNERABILITY

Old:

```
params
  .as_ref()
  .and_then(|m| m.get("phone-context"))
  .map(|&s| if s.as_bytes()[0] == b'+' { &s[1..] } else { s })
```

FIXING THE VULNERABILITY

Old:

```
params
  .as_ref()
  .and_then(|m| m.get("phone-context"))
  .map(|&s| if s.as_bytes()[0] == b'+' { &s[1..] } else { s })
```

New:

```
params
  .as_ref()
  .and_then(|m| m.get("phone-context"))
  .map(|&s|
    if s.as_bytes().get(0) == Some(&b'+') { &s[1..] } else { s })
```

FIXING THE VULNERABILITY

Old:

```
params
  .as_ref()
  .and_then(|m| m.get("phone-context"))
  .map(|&s| if s.as_bytes()[0] == b'+' { &s[1..] } else { s })
```

Newer:

```
params
  .as_ref()
  .and_then(|m| m.get("phone-context"))
  .map(|&s| s.strip_prefix('+').unwrap_or(s))

```

LET'S DO SOMETHING FUN

LET'S PARSE IT IN C

```
param_t *params = ...;
```

LET'S DO SOMETHING FUN

LET'S PARSE IT IN C

```
param_t *params = ...;
param_t phone_context;
res = find_param(params, &phone_context, "phone-context");
```

LET'S DO SOMETHING FUN

LET'S PARSE IT IN C

```
param_t *params = ...;
param_t phone_context;
res = find_param(params, &phone_context, "phone-context");
const char *phone_context_stripped;
if (res) {
    if (*phone_context.s == '+') {
        phone_context_stripped = phone_context.s + 1;
    } else {
        phone_context_stripped = phone_context.s;
    }
}
```

What could possibly go wrong?!

VULNERABILITY IMPACT

Rust version

- Trusted input?
  No vulnerability.
- Untrusted input?
  Panic, (probably)
  crash.

VULNERABILITY IMPACT

Rust version

- Trusted input?
  No vulnerability.
- Untrusted input?
  Panic, (probably) crash.

C version

- Trusted input?
  No vulnerability.
- Untrusted input?

best case	SIGSEG (segmentation fault, core dumped)
worse case	Unexploitable read-out-of-bounds, program corruption, late crash, …
worst case	Exploitable read-out-of-bounds

REST OF THE TIMELINE

September 14 Carter Snook reports the vulnerability with suggested fix

September 15 Whisperfish team initiates the security advisory on Github[1] and prepares the patches

September 15 Whisperfish team requests a CVE through Github

September 18 Github assigns CVE-2023-42444

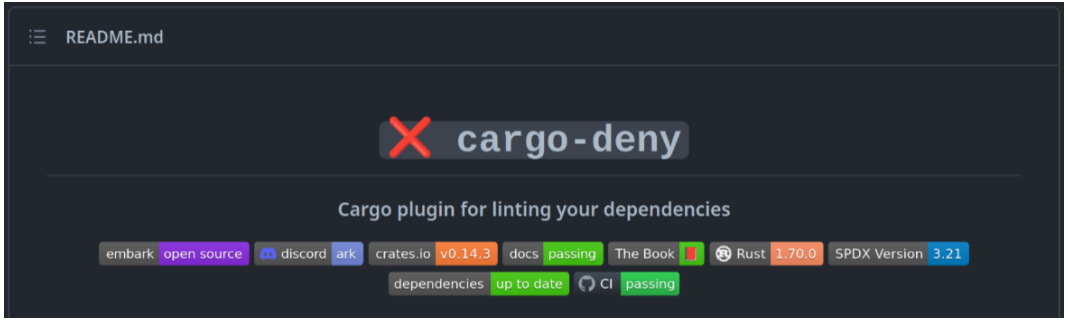
September 19 Whisperfish team publishes the advisory and patches

September 19 Whisperfish files in Rust's advisory db[2]

[1] `https://github.com/whisperfish/rust-phonenumber/security/advisories/GHSA-whhr-7f2w-qqj2`

[2] `https://github.com/rustsec/advisory-db/pull/1785`, still unmerged!

CARGO DENY & ADVISORY DB



Check your dependencies:

- License compliance
- Known vulnerabilities (advisory-db)
- Disallow/whitelist e.g. git-dependencies

Run it in CI!

HOW DID CARTER SNOOK FIND IT?

Fuzzing!

FUZZING BLURHASH-RS

…so what about `blurhash-rs`?

[3] With `cargo-fuzz`

FUZZING BLURHASH-RS

…so what about `blurhash-rs`?

Fuzzing Rust code[3] is very easy:

```
1 fuzz_target!(|data: &str| {
2     blurhash::decode(data, 50, 50, 1.0);
3 });
```

[3] With `cargo-fuzz`

FUZZING BLURHASH-RS

…so what about `blurhash-rs`?

Fuzzing Rust code[3] is very easy:

```
1 fuzz_target!(|data: &str| {
2     blurhash::decode(data, 50, 50, 1.0);
3 });
```

… and sure enough, after some seconds …

```
Output of `std::fmt::Debug`:

  "1RXRRL\nJ"
```

[3] With `cargo-fuzz`

FILING CVE'S FOR BLURHASH-RS

…here we go again!

[4] `https://github.com/whisperfish/blurhash-rs/security/advisories/GHSA-cxvp-82cq-57h2`

[5] `https://github.com/rustsec/advisory-db/pull/1786`, still unmerged!

FILING CVE'S FOR BLURHASH-RS

…here we go again!

September 16 Ruben finds vulnerabilities in blurhash-rs

September 16 Whisperfish team initiates the security advisory on Github[4] and prepares the patches

September 16 Whisperfish team requests a CVE through Github

September 18 Github assigns CVE-2023-42447

September 19 Whisperfish team publishes the advisory and patches

September 19 Whisperfish files in Rust's advisory db[5]

[4] `https://github.com/whisperfish/blurhash-rs/security/advisories/GHSA-cxvp-82cq-57h2`

[5] `https://github.com/rustsec/advisory-db/pull/1786`, still unmerged!

CONCLUSIONS

- Advisories and CVEs in free software are very easy (on Github)
- Rust prevented out-of-bounds read
- Fuzz your code
- Use cargo deny in your CI!

CONCLUSIONS

- Advisories and CVEs in free software are very easy (on Github)
- Rust prevented out-of-bounds read
- Fuzz your code
- Use cargo deny in your CI!

Questions are welcome on rubedesm@vub.be!