

[Tools](#)[Documentation](#)[Team](#)

# Welcome to Rust Cleaner!

Solana Software as a Service - Host of multiple static analysis tools for the Solana blockchain

[Start Analyzing](#)[Learn more](#)

### Software Analysis Tools:

☐ Cargo Audit

☐ Clippy

☒ Soteria

☐ Mirai

Enter GitHub URL

Or upload files to be analyzed (.zip)

📁 Rust-Website.zip

Submit

## Loading Results...



This might take awhile. Sign up for email notification?

Send email

Receive email on completion

## WARNINGS:

8

Correctness:

0

Suspicious:

0

Style:

4

Complexity:

1

Performance:

0

Low priority:

3

## Warnings from Clippy

Filter by:

Priority

Files

▼ style

▼ ./client/src/client.rs

▼ redundant pattern matching, consider using `is_err()`

Severity: "warning"

Lint Rule: "clippy::redundant\_pattern\_matching"

Error Line Numbers: 103 - 103

```
98     program: &Keypair,  
99     connection: &RpcClient,  
100 ) -> Result<()> {  
101     let greeting_pubkey = utils::get_greeting_public_key(&player.pubkey(), &program.pubkey());  
102  
103     if let Err(_) = connection.get_account(&greeting_pubkey) {  
104         println!("creating greeting account");  
105         let lamport_requirement =  
106             connection.get_minimum_balance_for_rent_exemption(utils::get_greeting_data_size())?;  
107  
108         // This instruction creates an account with the key
```

## Warnings from Cargo Audit:

### Crates:

> chrono

▼ hyper

▼ Lenient `hyper` header parsing of `Content-Length` could allow request smuggling

**Description:** `hyper`'s HTTP header parser accepted, according to RFC 7230, illegal contents inside `Content-Length` headers. Due to this, upstream HTTP proxies that ignore the header may still forward them along if it chooses to ignore the error. To be vulnerable, `hyper` must be used as an HTTP/1 server and using an HTTP proxy upstream that ignores the header's contents but still forwards it. Due to all the factors that must line up, an attack exploiting this vulnerability is unlikely.

**Solution:** Upgrade to  $\geq 0.14.10$

> Integer overflow in `hyper`'s parsing of the `Transfer-Encoding` header leads to data loss

> regex

> time

> tokio

- .editorconfig
- .gitattributes
- > • .github
- .gitignore
- .markdownlint.json
- Cargo.lock
- Cargo.toml
- CONTRIBUTING.md
- LICENSE.txt
- README.md
- > • results
- ✓ • src
  - ✓ • bin
    - cleanup.rs
    - hacktoberfest.rs
    - main.rs

```
1 // Cleans up `README.md`
2 // Usage: cargo run --bin cleanup
3
4 use std::fs;
5 use std::fs::File;
6 use std::io::Read;
7
8 fn fix_dashes(lines: Vec<String>) -> Vec<String> {
9     let mut fixed_lines: Vec<String> := Vec::with_capacity(lines.len());
10
11     let mut within_content = false;
12
13     for line in lines {
14         if within_content {
15             fixed_lines.push(line.replace(" - ", " - "));
16         } else {
17             if line.starts_with("## Applications") {
18                 within_content = true;
19             }
20
21             fixed_lines.push(line.to_string());
22         }
23     }
24
25     return fixed_lines;
26 }
27
28 fn main() {
29     // Read the awesome file.
30     let mut file = File::open("README.md").expect("Failed to read the file");
31
32     let mut contents = String::new();
33     file.read_to_string(&mut contents)
34         .expect("Failed to read file contents");
35 }
```