





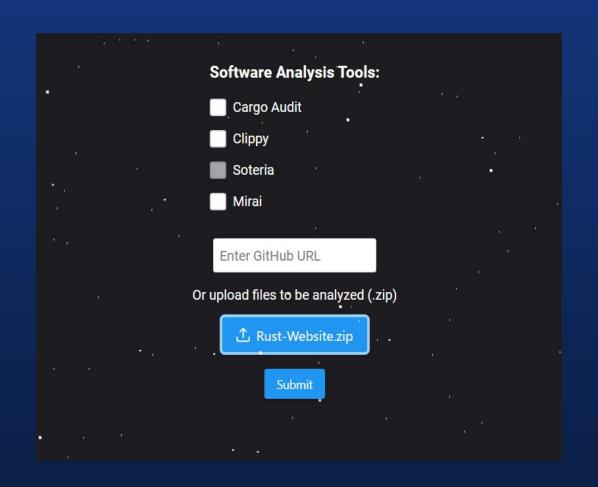


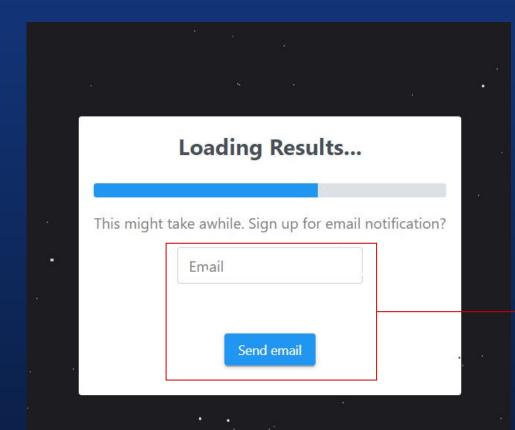
Welcome to Rust Cleaner!

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



Learn more





Receive email on completion





WARNINGS:

Correctness 0

Suspicious:

Style:

Complexity
1

Performance: 0

Low priority: 3 **Warnings from Clippy**

Filter by: Priority Files

```
✓ ./client/src/client.rs
```

✓ style

✓ 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 >=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
```

```
2 // Usage: cargo run --bin cleanup
   use std::fs;
 5 use std::fs::File;
 6 use std::io::Read;
   fn fix_dashes(lines: Vec<String>) -> Vec<String> {
        let mut fixed lines: Vec<String> = Vec::with capacity(lines.len());
10
        let mut within_content = false;
        for line in lines {
            if within content {
               fixed_lines.push(line.replace(" - ", " - "));
15
           } else {
               if line.starts with("## Applications") {
                    within_content = true;
20
               fixed lines.push(line.to string());
        return fixed_lines;
28 .fn main() {
       // Read the awesome file.
30
       let mut file = File::open("README.md").expect("Failed to read the file");
        let mut contents = String::new();
        file.read_to_string(&mut contents)
            .expect("Failed to read file contents");
```