Rust Cleaner

Rust Code Analysis as a Service

Arsh Kabarwal
Joseph Carpman
Kipp Corman
Olivia Ornelas

01

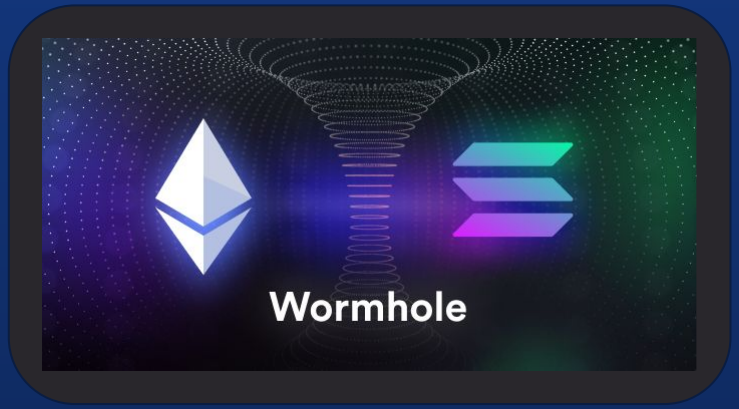
Wormhole Hack

Wormhole Hack - Overview

- On February 2nd, 2022, a hacker exploited a vulnerability in the Wormhole bridge between Solana and Ethereum.

- 120000 Ethereum was stolen, worth ~$320 million at the time.

Wormhole Hack - Causes



- Caused by a failure to verify validator accounts in the smart contract chain.
- Additionally, the “sysvar:instructions” API that Wormhole was using was deprecated.
- Patch was pushed just hours before the attack.

Wormhole Hack - Solution

- Cargo Clippy and free-use Soteria can warn the user of the deprecated API.

- Premium Soteria can warn the user of the unverified validator accounts.

```
==========This account may be UNTRUSTFUL!==========
Found a potential vulnerability at line 249, column 13 in bridge/src/processor.rs
The sysvar instructions API is unsafe and deprecated (wormhole exploit):

 243|        }
 244|
 245|        // The previous ix must be a secp verification instruction
 246|        let secp_ix_index = (current_instruction - 1) as u8;
 247|        let secp_ix = solana_program::sysvar::instructions::load_instruction_at(
 248|            secp_ix_index as usize,
>249|            &instruction_accounts.try_borrow_mut_data()?,
 250|        )
 251|            .map_err(|_| ProgramError::InvalidAccountData)?;
 252|
 253|        // Check that the instruction is actually for the secp program
 254|        if secp_ix.program_id != solana_program::secp256k1_program::id() {
 255|            return Err(ProgramError::InvalidArgument);
>>>Stack Trace:
>>>spl_bridge::processor::_$LT$impl$u20$spl_bridge..state..Bridge$GT$::process::hf71455780127ef67 [bri
>>>  spl_bridge::processor::_$LT$impl$u20$spl_bridge..state..Bridge$GT$::process_verify_signatures::he
or.rs:95]
```

```
*************** attack surface #8: sol.verify_signatures ***************
account: accs.instruction_acc
==========VULNERABLE: UnverifiedParsedAccount!==========
Found a potential vulnerability at line 103, column 9 in api/verify_signature.rs
The account is not validated before parsing its data:
 97|    }
 98|
 99|    // The previous ix must be a secp verification instruction
 100|    let secp_ix_index = (current_instruction - 1) as u8;
 101|    let secp_ix = solana_program::sysvar::instructions::load_instruction_at(
 102|        secp_ix_index as usize,
>103|        &accs.instruction_acc.try_borrow_mut_data()?,
 104|    )
 105|    .map_err(|_| ProgramError::InvalidAccountData)?;
 106|
 107|    // Check that the instruction is actually for the secp program
 108|    if secp_ix.program_id != solana_program::secp256k1_program::id() {
 109|        return Err(InvalidSecpInstruction.into());
>>>Stack Trace:
>>>sol.verify_signatures [lib.rs:94]
```

02

Problem Overview

Problem Background

Significant security bugs can’t be found from a compiler

Once code is deployed to the blockchain, code is immutable, so it can be difficult to change

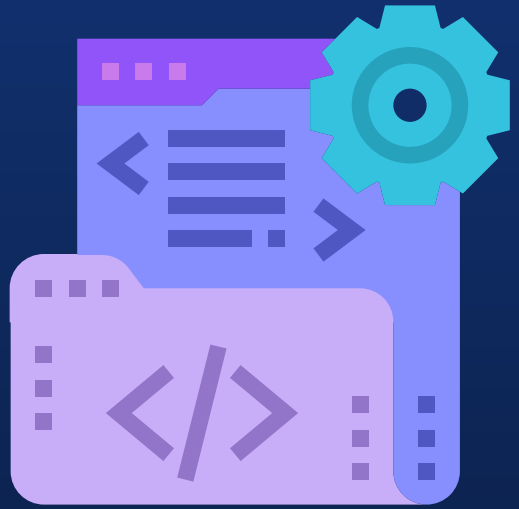
Other tools require quotes and download times which take extra time and money from the average user

Solution

- Software-as-a-Service tool which allows users to quickly and clearly review their code before deployment
- Allows the user to freely use the service
- Easy to understand results that makes debugging quick and hassle-free

03

User Cases



User case 1 : Blockchain Developer

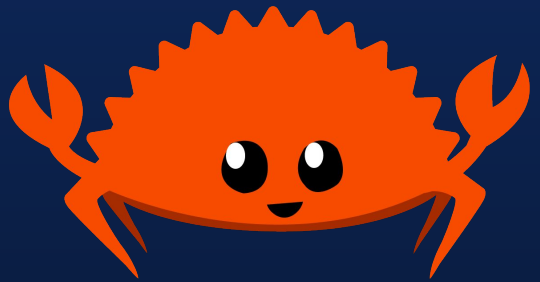
Code can be analyzed before deployed

Save time from manually debugging security flaws with a test network

Give the developer a peace of mind that a obscure bug isn’t present

User case 2 : Rust Hobbyist

- Find security and runtime flaws not found through compilation
- No time spent installing and configuring static analysis tools
- Easy to read format makes warnings more approachable
- Remove the need for powerful hardware



User case 3 : Rust-based Startup

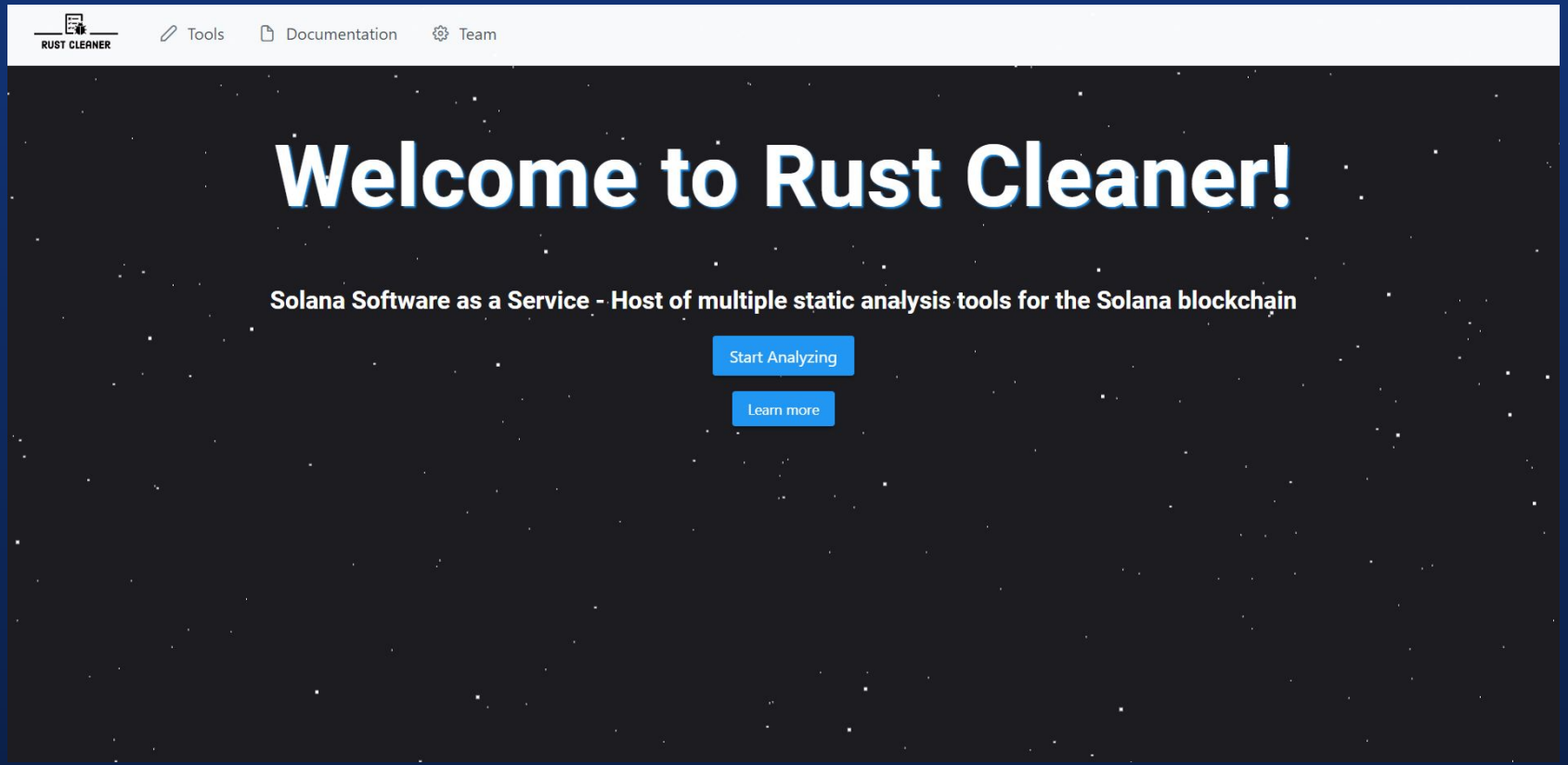
Run code review nightly without server infrastructure

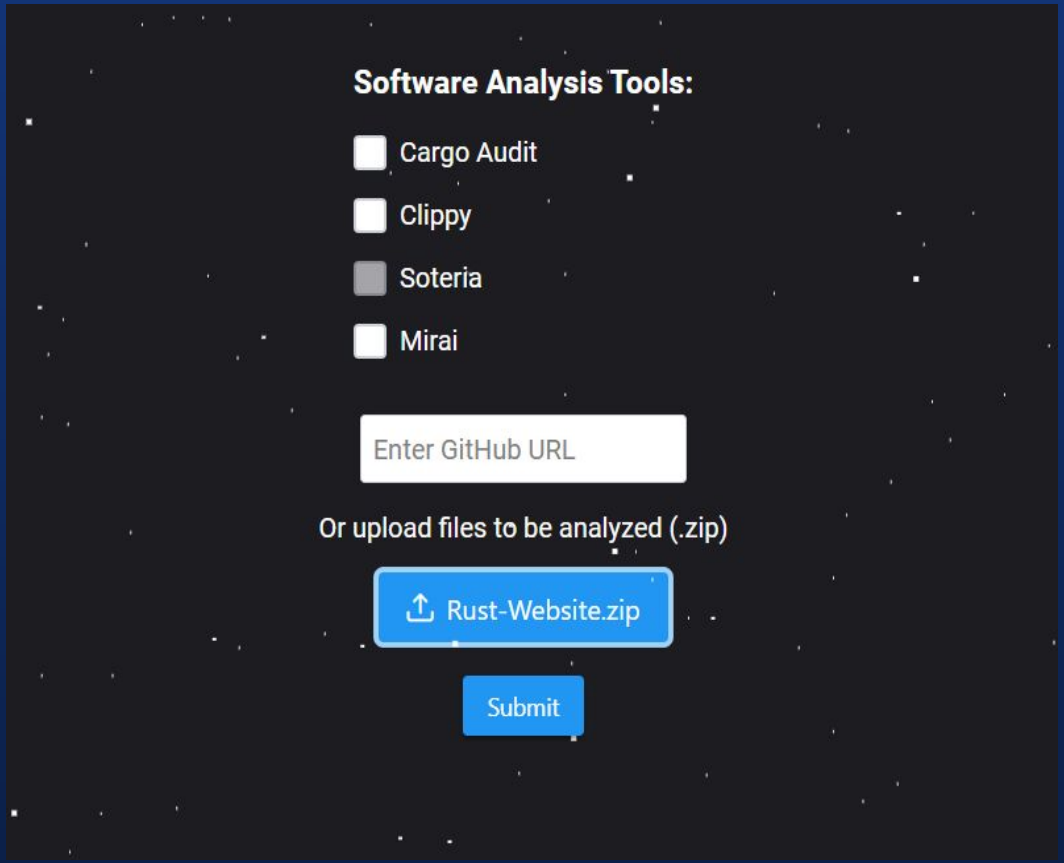
Hasten development by removing review infrastructure

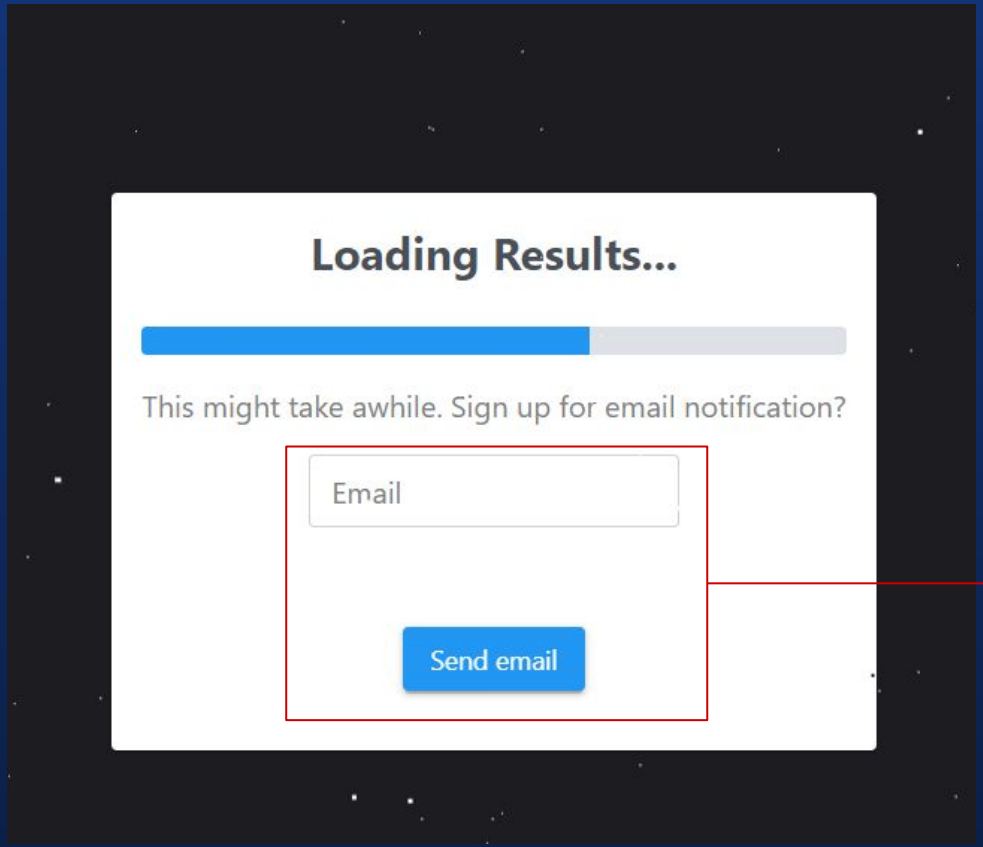
Easy to share results with colleagues

04

Product


RUST CLEANER
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



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:

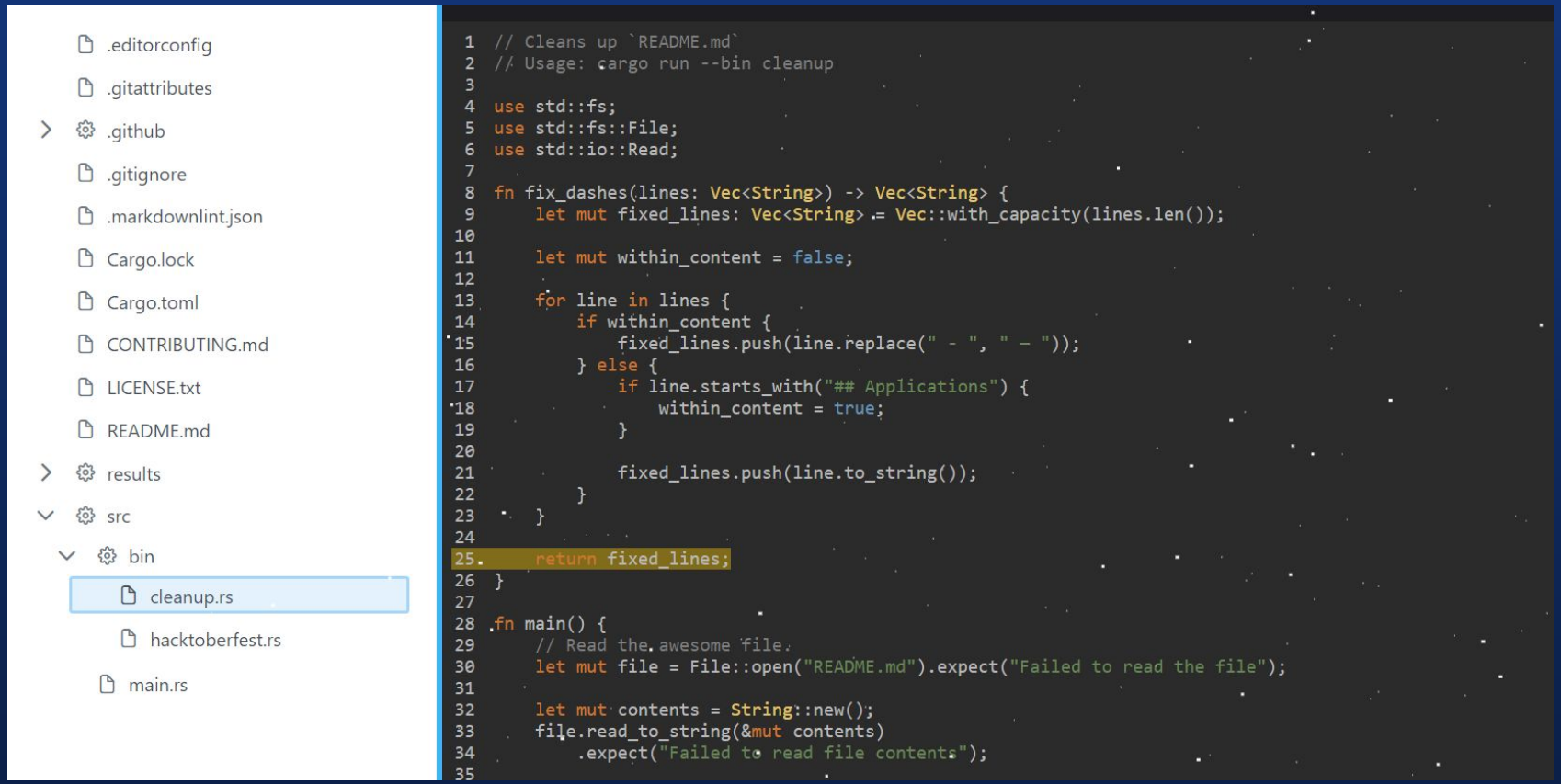
Crates:

- 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 >=0.14.10

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

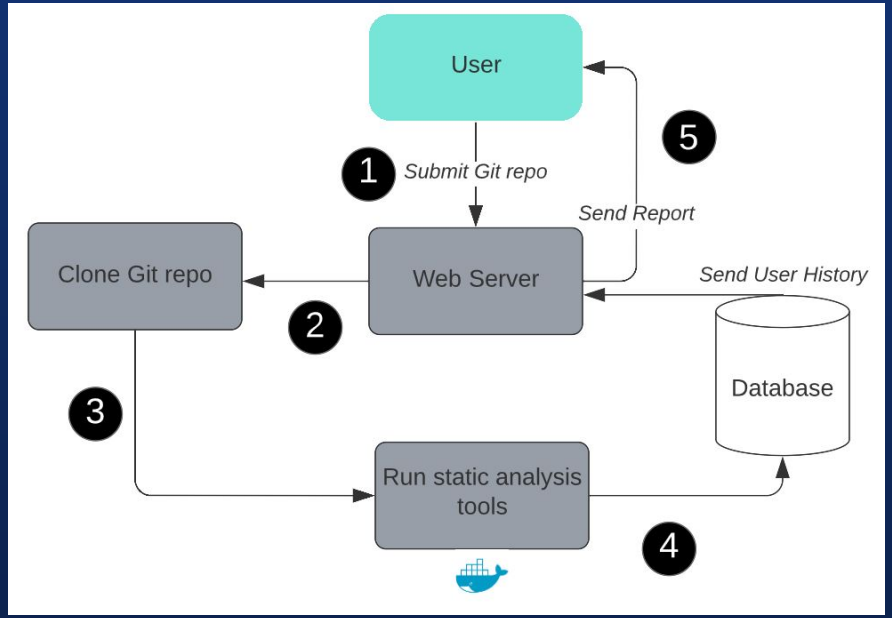

.editorconfig
.gitattributes
.github
.gitignore
.markdownlint.json
Cargo.lock
Cargo.toml
CONTRIBUTING.md
LICENSE.txt
README.md
results
src
bin
cleanup.rs
hacktoberfest.rs
main.rs
// Cleans up `README.md`
// Usage: cargo run --bin cleanup
use std::fs;
use std::fs::File;
use std::io::Read;
fn fix_dashes(lines: Vec<String>) -> Vec<String> {
let mut fixed_lines: Vec<String> = Vec::with_capacity(lines.len());
let mut within_content = false;
for line in lines {
if within_content {
fixed_lines.push(line.replace(" - ", " – "));
} else {
if line.starts_with("## Applications") {
within_content = true;
}
fixed_lines.push(line.to_string());
}
}
return fixed_lines;
}
fn main() {
// Read the awesome file.
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");

05

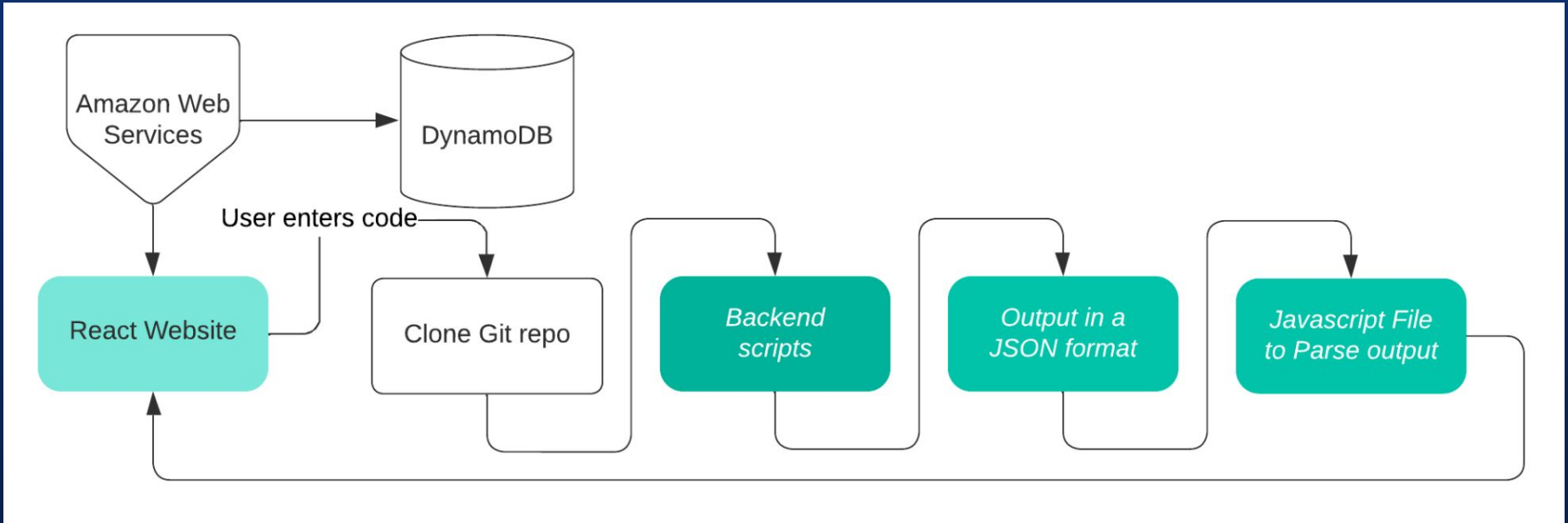
System Design

System Design

- Currently all on the same server
  - Could be split up for scalability
- Easy to add any tool in SARIF format



Subsystem Design



Future Improvements

Additional Tools 

 Additional Language Support

Submission History 

 Analyze On Code Change

QUESTIONS?