VRust

Security Assessment

O2Lab VRust Team

03/24/2022 21:21:53

Contents

Summary

This report has been prepared for O2Lab VRust Team to discover issues and vulnerabilities in the source code of the O2Lab VRust Team project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques. The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	O2Lab VRust Team
Platform	Ethereum
Language	Solana
Crate	bridge
GitHub Location	https://github.com/parasol-aser/vrust
sha256	Unknown

Audit Summary

Delivery Date	03/25/2022
Audit Methodology	Static Analysis
Key Components	

Vulnerability Summary

Vulnerability Level	Total
Critical	6
Major	0
Medium	0
Minor	0
Informational	0
Discussion	0

Findings

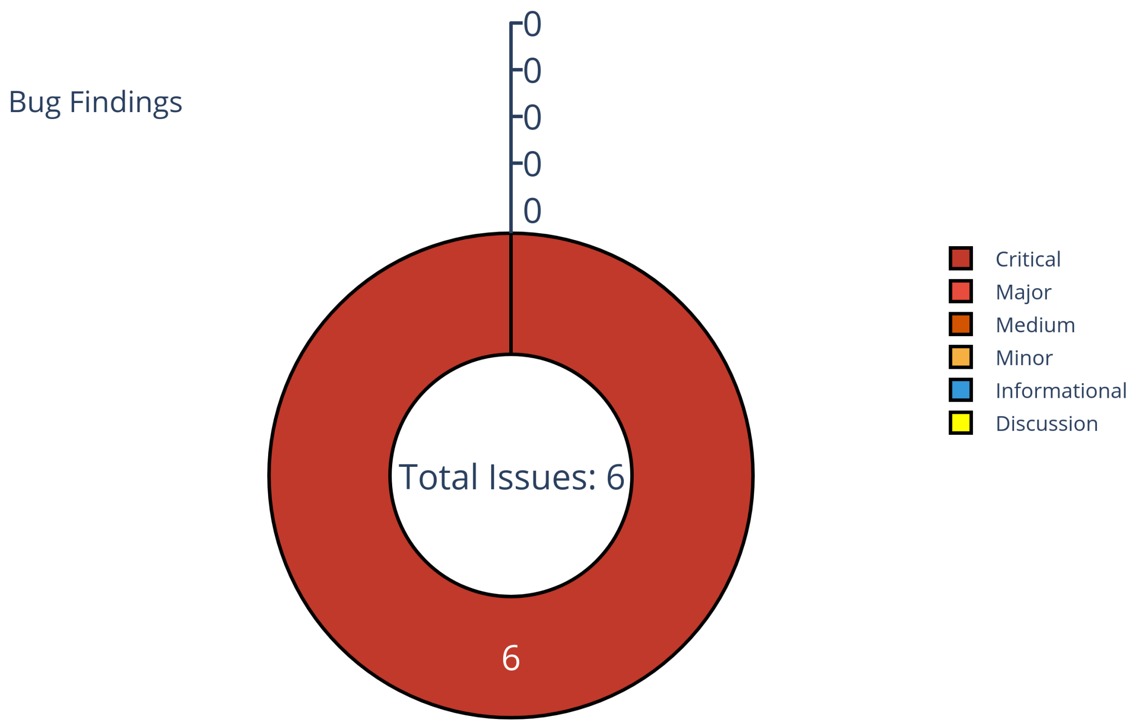

Figure 1: Findings

Finding Statistic

Category	Count
IntegerFlow	5
MissingKeyCheck	1

ID	Category	Severity	Status
0	IntegerFlow	Critical	UnResolved
1	IntegerFlow	Critical	UnResolved
2	IntegerFlow	Critical	UnResolved
3	IntegerFlow	Critical	UnResolved
4	IntegerFlow	Critical	UnResolved
5	MissingKeyCheck	Critical	UnResolved

Issue: 0: IntegerFlow

Category	Severity	Status
IntegerFlow	Critical	UnResolved

- Location

program/src/api/verify_signature.rs:100:25: 100:50

```
(current_instruction - 1)
```

- Code Context

Vulnerability at Line: 100

```
    if current_instruction == 0 {
        return Err(InstructionAtWrongIndex.into());
    }

    // The previous ix must be a secp verification instruction
    let secp_ix_index = (current_instruction - 1) as u8;
    let secp_ix =
    ↪   solana_program::sysvar::instructions::load_instruction_at_checked(
        secp_ix_index as usize,
        &accs.instruction_acc,
    )
```

- Call Stack

```
fn entrypoint(){// /home/ubuntu/.cargo/registry/src/github.com-
↪   1ecc6299db9ec823/solana-program-1.9.4/src/entrypoint.rs:120:9: 127:10
↪   }
    fn instruction::solitaire(){// /home/ubuntu/VRust/wormhole/wormhole-
    ↪   2.7.3/solana/solitaire/program/src/macros.rs:101:13: 108:14
    ↪   }
        fn instruction::dispatch(){// /home/ubuntu/VRust/wormhole/wormhole-
        ↪   2.7.3/solana/solitaire/program/src/macros.rs:89:13: 99:14
        ↪   }
```

```
            fn instruction::VerifySignatures::execute(){//
            ↪  /home/ubuntu/VRust/wormhole/wormhole-
            ↪  2.7.3/solana/solitaire/program/src/macros.rs:68:21: 74:22
            ↪  }
                    fn api::verify_signature::verify_signatures(){//
                    ↪  program/src/api/verify_signature.rs:68:1: 219:2 }

```

- description:

A mild bug. This int overflow involves a function call to "'let current_instruction = solana_program::sysvar::instructions::l
&accs.instruction_acc,)?; (https://docs.rs/solana-program/1.9.1/solana_program/sysvar/instructions/fn.load_current_i
Load the current Instruction's index in the currently executing Transaction. (Constrain: current_instruction>=0 (not general enough to model)). And it has a check at line 95: current_instruction != 0 (this could be modeled into the overflow checker.)

- link:
- alleviation:

Checker could be updated for `x - 1` and a check on `x == 0` or `x >= 0`, add constrains handling. (Needs a solver to handle the case where the instruction is `current_instruction - 5` or `x - y (a variable).`)

Issue: 1: IntegerFlow

Category	Severity	Status
IntegerFlow	Critical	UnResolved

- Location

/home/ubuntu/.cargo/registry/src/github.com-1ecc6299db9ec823/solana-program-1.9.4/src/message/legacy.rs:466:2
466:29

```
466 index * 2
467
```

- Code Context

Vulnerability at Line: 466

```
461         if index >= num_instructions as usize {
462             return Err(SanitizeError::IndexOutOfBounds);
463         }
464
465         // index into the instruction byte-offset table.
466         current += index * 2;
467         let start = read_u16(&mut current, data)?;
468
469         current = start as usize;
470         let num_accounts = read_u16(&mut current, data)?;
471
```

- Call Stack

```
fn entrypoint(){// /home/ubuntu/.cargo/registry/src/github.com-
↪   1ecc6299db9ec823/solana-program-1.9.4/src/entrypoint.rs:120:9: 127:10
↪   }
    fn instruction::solitaire(){// /home/ubuntu/VRust/wormhole/wormhole-
    ↪   2.7.3/solana/solitaire/program/src/macros.rs:101:13: 108:14
    ↪   }
        fn instruction::dispatch(){// /home/ubuntu/VRust/wormhole/wormhole-
        ↪   2.7.3/solana/solitaire/program/src/macros.rs:89:13: 99:14
        ↪   }
```

```
            fn instruction::VerifySignatures::execute(){//
            ↪  /home/ubuntu/VRust/wormhole/wormhole-
            ↪  2.7.3/solana/solitaire/program/src/macros.rs:68:21: 74:22
            ↪  }
                fn api::verify_signature::verify_signatures(){//
                ↪  program/src/api/verify_signature.rs:68:1: 219:2 }
                  fn
                  ↪  solana_program::sysvar::instructions::load_instruction_at_ch
                  ↪  /home/ubuntu/.cargo/registry/src/github.com-
                  ↪  1ecc6299db9ec823/solana-program-
                  ↪  1.9.4/src/sysvar/instructions.rs:71:1: 86:2
                  ↪  }
                    fn
                    ↪  solana_program::message::Message::deserialize_instructio
                    ↪  /home/ubuntu/.cargo/registry/src/github.com-
                    ↪  1ecc6299db9ec823/solana-program-
                    ↪  1.9.4/src/message/legacy.rs:455:5: 497:6
                    ↪  }
```

- description:

Built-in library for instruction serialization and `deserialize_instruction`.

- link:
- alleviation:

Not a real bug. The parameter `index` is calculated as `secp_ix_index` from `solana_program::sysvar::inst` Another argument is an external argument (can be fake). However, the `instruction id` variable is also with a check on `if index >= num_instructions as usize` (Line 461 in the report), and therefore, it is hard to reason about the value of the condition to revise the checker.

Issue: 2: IntegerFlow

Category	Severity	Status
IntegerFlow	Critical	UnResolved

- Location

/home/ubuntu/.cargo/registry/src/github.com-1ecc6299db9ec823/solana-program-1.9.4/src/serialize_utils.rs:25:21: 25:33

```
25 *current + 1
26
```

- Code Context

Vulnerability at Line: 25

```
24 pub fn read_u8(current: &mut usize, data: &[u8]) -> Result<u8, SanitizeError> {
25     if data.len() < *current + 1 {
26         return Err(SanitizeError::IndexOutOfBounds);
27     }
28     let e = data[*current];
29     *current += 1;
30
```

- Call Stack

```
1 fn entrypoint(){// /home/ubuntu/.cargo/registry/src/github.com-1ecc6299db9ec823/solana-program-1.9.4/src/entrypoint.rs:120:9: 127:10 }
2     fn instruction::solitaire(){// /home/ubuntu/VRust/wormhole/wormhole-2.7.3/solana/solitaire/program/src/macros.rs:101:13: 108:14 }
3         fn instruction::dispatch(){// /home/ubuntu/VRust/wormhole/wormhole-2.7.3/solana/solitaire/program/src/macros.rs:89:13: 99:14 }
4             fn instruction::VerifySignatures::execute(){// /home/ubuntu/VRust/wormhole/wormhole-2.7.3/solana/solitaire/program/src/macros.rs:68:21: 74:22 }
```

```
                    fn api::verify_signature::verify_signatures(){//
                    ↪  program/src/api/verify_signature.rs:68:1: 219:2 }
                        fn
                        ↪  solana_program::sysvar::instructions::load_instruction_at_ch
                        ↪  /home/ubuntu/.cargo/registry/src/github.com-
                        ↪  1ecc6299db9ec823/solana-program-
                        ↪  1.9.4/src/sysvar/instructions.rs:71:1: 86:2
                        ↪  }
                            fn
                            ↪  solana_program::message::Message::deserialize_instructio
                            ↪  /home/ubuntu/.cargo/registry/src/github.com-
                            ↪  1ecc6299db9ec823/solana-program-
                            ↪  1.9.4/src/message/legacy.rs:455:5: 497:6
                            ↪  }
       fn solana_program::serialize_utils::read_u8(){//
       ↪  /home/ubuntu/.cargo/registry/src/github.com-
       ↪  1ecc6299db9ec823/solana-program-
       ↪  1.9.4/src/serialize_utils.rs:24:1: 31:2
       ↪  }

```

- description:

Not real. There is a check at line 25: `data.len() < *current + 1`. If `*current += 1;` overflows, the if condition would fail.

- link:
- alleviation:

Similar to the first case, we can implement something specific to this case (if condition(has x + 1); some stmts; x += 1), but this is not generalize enough.

Issue: 3: IntegerFlow

Category	Severity	Status
IntegerFlow	Critical	UnResolved

- Location

/home/ubuntu/.cargo/registry/src/github.com-1ecc6299db9ec823/solana-program-1.9.4/src/serialize_utils.rs:35:21:
35:35

```
*current + len

```

- Code Context

Vulnerability at Line: 35

```
pub fn read_pubkey(current: &mut usize, data: &[u8]) -> Result<Pubkey,
↪   SanitizeError> {
    let len = std::mem::size_of::<Pubkey>();
    if data.len() < *current + len {
        return Err(SanitizeError::IndexOutOfBounds);
    }
    let e = Pubkey::new(&data[*current..*current + len]);
    *current += len;

```

Other Use Case for Variable: *current + len

```
    let e = Pubkey::new(&data[*current..*current + len]);
```

- Call Stack

```
fn entrypoint(){// /home/ubuntu/.cargo/registry/src/github.com-
↪   1ecc6299db9ec823/solana-program-1.9.4/src/entrypoint.rs:120:9: 127:10
↪   }
    fn instruction::solitaire(){// /home/ubuntu/VRust/wormhole/wormhole-
    ↪   2.7.3/solana/solitaire/program/src/macros.rs:101:13: 108:14
    ↪   }
```

```
        fn instruction::dispatch(){// /home/ubuntu/VRust/wormhole/wormhole-
        ↪  2.7.3/solana/solitaire/program/src/macros.rs:89:13: 99:14
        ↪  }
            fn instruction::VerifySignatures::execute(){//
            ↪  /home/ubuntu/VRust/wormhole/wormhole-
            ↪  2.7.3/solana/solitaire/program/src/macros.rs:68:21: 74:22
            ↪  }
                    fn api::verify_signature::verify_signatures(){//
                    ↪  program/src/api/verify_signature.rs:68:1: 219:2 }
                        fn
                        ↪  solana_program::sysvar::instructions::load_instruction_at_ch
                        ↪  /home/ubuntu/.cargo/registry/src/github.com-
                        ↪  1ecc6299db9ec823/solana-program-
                        ↪  1.9.4/src/sysvar/instructions.rs:71:1: 86:2
                        ↪  }
                            fn
                            ↪  solana_program::message::Message::deserialize_instructio
                            ↪  /home/ubuntu/.cargo/registry/src/github.com-
                            ↪  1ecc6299db9ec823/solana-program-
                            ↪  1.9.4/src/message/legacy.rs:455:5: 497:6
                            ↪  }
       fn solana_program::serialize_utils::read_pubkey(){//
       ↪  /home/ubuntu/.cargo/registry/src/github.com-
       ↪  1ecc6299db9ec823/solana-program-
       ↪  1.9.4/src/serialize_utils.rs:33:1: 41:2
       ↪  }

```

- description:

Similar to ID 2

- link:
- alleviation:

Similar to ID 2

Issue: 4: IntegerFlow

Category	Severity	Status
IntegerFlow	Critical	UnResolved

- Location

/home/ubuntu/.cargo/registry/src/github.com-1ecc6299db9ec823/solana-program-1.9.4/src/serialize_utils.rs:59:21: 59:40

```
*current + data_len

```

- Code Context

Vulnerability at Line: 59

```
pub fn read_slice(
    current: &mut usize,
    data: &[u8],
    data_len: usize,
) -> Result<Vec<u8>, SanitizeError> {
    if data.len() < *current + data_len {
        return Err(SanitizeError::IndexOutOfBounds);
    }
    let e = data[*current..*current + data_len].to_vec();
    *current += data_len;

```

Other Use Case for Variable: *current + data_len

```
    let e = data[*current..*current + data_len].to_vec();
```

- Call Stack

```
fn entrypoint(){// /home/ubuntu/.cargo/registry/src/github.com-
↪ 1ecc6299db9ec823/solana-program-1.9.4/src/entrypoint.rs:120:9: 127:10
↪ }
    fn instruction::solitaire(){// /home/ubuntu/VRust/wormhole/wormhole-
     ↪ 2.7.3/solana/solitaire/program/src/macros.rs:101:13: 108:14
     ↪ }
```

```
        fn instruction::dispatch(){// /home/ubuntu/VRust/wormhole/wormhole-
         ↪  2.7.3/solana/solitaire/program/src/macros.rs:89:13: 99:14
         ↪  }
            fn instruction::VerifySignatures::execute(){//
             ↪  /home/ubuntu/VRust/wormhole/wormhole-
             ↪  2.7.3/solana/solitaire/program/src/macros.rs:68:21: 74:22
             ↪  }
                    fn api::verify_signature::verify_signatures(){//
                     ↪  program/src/api/verify_signature.rs:68:1: 219:2 }
                        fn
                         ↪  solana_program::sysvar::instructions::load_instruction_at_ch
                         ↪  /home/ubuntu/.cargo/registry/src/github.com-
                         ↪  1ecc6299db9ec823/solana-program-
                         ↪  1.9.4/src/sysvar/instructions.rs:71:1: 86:2
                         ↪  }
                            fn
                             ↪  solana_program::message::Message::deserialize_instructio
                             ↪  /home/ubuntu/.cargo/registry/src/github.com-
                             ↪  1ecc6299db9ec823/solana-program-
                             ↪  1.9.4/src/message/legacy.rs:455:5: 497:6
                             ↪  }
       fn solana_program::serialize_utils::read_slice(){//
        ↪  /home/ubuntu/.cargo/registry/src/github.com-
        ↪  1ecc6299db9ec823/solana-program-
        ↪  1.9.4/src/serialize_utils.rs:54:1: 65:2
        ↪  }

```

- description:

Similar to ID 2

- link:
- alleviation:

Not relevant to this case, but some new heuristics: we could develop something to filter out overflow that on the LHS of "<" and RHS of "<", or underflow on the RHS of "<" and LHS of ">" with an error reported afterwards (if the added number is small, therefore, the result is small enough to trigger the error). For example: if x + 1 < y { return Err(SanitizeError::IndexOutOfBounds); }, if x+1 may overflow, it will trigger the error.

Issue: 5: MissingKeyCheck

Category	Severity	Status
MissingKeyCheck	Critical	UnResolved

- Location

/home/ubuntu/VRust/wormhole/wormhole-2.7.3/solana/solitaire/program/src/processors/peel.rs:214:52:214:80

```
214 ctx.info().data.borrow_mut()
215
```

- Code Context

– Function Definition:

```
192 fn peel<I>(ctx: &'c mut Context<'a, 'b, 'c, I>) -> Result<Self>
193
```

Vulnerability at Line: 208

```
203                     return
                         ↪  Err(SolitaireError::AlreadyInitialized(*ctx.info().key));
204                 }
205                 (false, T::default())
206             }
207             AccountState::Initialized => {
208                 (true, T::try_from_slice(&mut
    ↪  *ctx.info().data.borrow_mut())?)
209             }
210             AccountState::MaybeInitialized => {
211                 if **ctx.info().lamports.borrow() == 0 {
212                     (false, T::default())
213
```

Other Use Case for Variable: ctx.info().data.borrow_mut()

```
                    (true, T::try_from_slice(&mut *ctx.info().data.borrow_mut())?)
```

- Call Stack

```
fn entrypoint(){// /home/ubuntu/.cargo/registry/src/github.com-1ecc6299db9ec823/solana-program-1.9.4/src/entrypoint.rs:120:9: 127:10 }
    fn instruction::solitaire(){// /home/ubuntu/VRust/wormhole/wormhole-2.7.3/solana/solitaire/program/src/macros.rs:101:13: 108:14 }
        fn instruction::dispatch(){// /home/ubuntu/VRust/wormhole/wormhole-2.7.3/solana/solitaire/program/src/macros.rs:89:13: 99:14 }
            fn instruction::PostVAA::execute(){// /home/ubuntu/VRust/wormhole/wormhole-2.7.3/solana/solitaire/program/src/macros.rs:68:21: 74:22 }
                fn <api::post_vaa::PostVAA<'b> as solitaire::FromAccounts<'a, 'b, 'c>>::from(){// program/src/api/post_vaa.rs:54:10: 54:22 }
                    fn <solitaire::Data<'b, T, IsInitialized> as solitaire::Peel<'a, 'b, 'c>>::peel(){// /home/ubuntu/VRust/wormhole/wormhole-2.7.3/solana/solitaire/program/src/processors/peel.rs:192:5: 236:6 }

```

- description:

It does have `ctx.info().data.borrow_mut`, but no transaction involved.

- link:
- alleviation:

We could prioritize the bug reported with a transaction, transfer, or any other cirical functions involved.

Appendix

Copied from https://leaderboard.certik.io/projects/aave

Finding Categories

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum” command against the target file.

Disclaimer