1Sol Audit Report

Version 1.0.0

Serial No. 2021111100012012

Presented by Fairyproof

November 11, 2021

FAIRYPROOF

01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the [1Sol](#) project.

Audit Start Time:

November 3, 2021

Audit End Time:

November 8, 2021

Audited Code's Github Repository:

https://github.com/1sol-io/1sol-protocol

Audited Code's Github Commit Number When Audit Started:

1349d2e9d3b9559723312059cdda32af4cdf1cc3

Audited Code's Github Commit Number When Audit Ended:

N/A

The goal of this audit is to review 1Sol’s Rust implementation for its cross-chain aggregator application, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

We make observations on specific areas of the code that present concrete problems, as well as general observations that traverse the entire codebase horizontally, which could improve its quality as a whole.

This audit only applies to the specified code, software or any materials supplied by the 1Sol team for specified versions. Whenever the code, software, materials, settings, enviroment etc is changed, the comments of this audit will no longer apply.

— Disclaimer

Note that as of the date of publishing, the contents of this report reflect the current understanding of known security patterns and state of the art regarding system security. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk.

The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. If the audited source files are smart contract files, risks or issues introduced by using data feeds from offchain sources are not extended by this review either.

Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit is recommended after the issues covered are fixed.

To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

— Methodology

The above files' code was studied in detail in order to acquire a clear impression of how the its specifications were implemented. The codebase was then subject to deep analysis and scrutiny, resulting in a series of observations. The problems and their potential solutions are discussed in this document and, whenever possible, we identify common sources for such problems and comment on them as well.

The Fairyproof auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Fairyproof to make sure we understand the size, scope, and functionality of the project's source code.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Fairyproof describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run the test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the source code to improve maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

— Structure of the document

This report contains a list of issues and comments on all the above source files. Each issue is assigned a severity level based on the potential impact of the issue and recommendations to fix it, if applicable. For ease of navigation, an index by topic and another by severity are both provided at the beginning of the report.

— Documentation

For this audit, we used the following sources of truth about how the cross-chain aggregator application should work:

https://1sol.io/

https://github.com/1sol-io/1sol-protocol

These were considered the specification.

— Comments from Auditor

Serial Number	Auditor	Audit Time	Result
2021111100012012	Fairyproof Security Team	November 3, 2021 - November 8, 2021	Medium Risk

Summary:

The Fairyproof security team used its auto analysis tools and manual work to audit the project. During the audit 1 risk of medium-severity and 5 risks of low-severity were discovered and 1 neutral suggestion was listed.

02. About Fairyproof

Fairyproof is a leading technology firm in the blockchain industry, providing consulting and security audits for organizations. Fairyproof has developed industry security standards for designing and deploying blockchain applications.

03. Introduction to 1Sol

1Sol Protocol is a cross-chain DEX aggregator for decentralized protocols on Solana, enabling the most seamless, efficient and protected operations in DeFi. With DeFi infrastructure rapidly growing, aggregators in high demand, cross-chain transactions being the future, 1Sol is born to bring together liquidity from both DeFi and CeFi (swaps, orderbook DEX(s), OTC, etc.) for multi-chains.

04. Coverage of issues

The issues that the Fairyproof team covered when conducting the audit include but are not limited to the following ones:

- Re-entrancy Attack
- Replay Attack
- Reordering Attack
- DDos Attack
- Transaction Ordering Attack
- Race Condition
- Access Control
- Integer Overflow/Underflow
- Timestamp Attack
- Gas Consumption
- Inappropriate Callback Function

- Unsafe External Call
- Function Visibility
- Implementation Vulnerability
- Uninitialized Storage Pointer
- Arithmetic Precision
- Scoping and Declaration
- Account Validity
- Data Serialization and Deserialization
- Tx.origin
- Fake Deposit
- Shadow Variable
- Design Vulnerability
- Token Issurance
- Admin Rights
- Inappropriate Proxy Design
- Inappropriate Use of Slots
- Asset Security
- Contract Upgrade/Migration
- Code Improvement

05. Severity level reference

Every issue in this report was assigned a severity level from the following:

Critical severity issues need to be fixed as soon as possible.

High severity issues will probably bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Neutral is not an issue or risk but a suggestion for code improvement.

08. List of functions audited

The Fairyproof security team analyzed the major functions during the audit and the result is as follows:

Function Name	Rent-Exemption Check	Account Check	Signer Check	Program_id check
process_swap_spltokenswap	N/A	15/15	Yes	Yes
process_swap_serumdex	N/A	19/19	Yes	Yes
process_initialize_amm_info	Failed	7/7	N/A	Yes
process_initialize_dex_mark_open_orders	Failed	4/7	N/A	Yes
process_swap_two_steps	N/A	N/A	Yes	Yes

09. Descriptions of function checkpoints

The Fairyproof security team analyzed the checkpoints of each of the functions listed in section 08 and the result is as follows:

- process_swap_spltokenswap

Index	Account Name	Writable/Signer	Checked
0	User Source Token Address	Writable	Yes
1	User Destination Token Address	Writable	Yes
2	User Source Token Authority	Signer	Yes
3	Spl-Token Program ID	N/A	Yes
4	OneSolProtocol AmmInfo Address	Writable	Yes
5	OneSolProtocol AmmInfo authority	N/A	Yes
6	OneSolProtocol AmmInfo token a account	Writable	Yes
7	OneSolProtocol AmmInfo token b account	Writable	Yes
8	TokenSwap swap_info account	N/A	Yes
9	TokenSwap swap_info authority	N/A	Yes
10	TokenSwap token_A Account	Writable	Yes
11	TokenSwap token_B Account	Writable	Yes
12	TokenSwap Pool token mint	Writable	Yes
13	TokenSwap Fee account	Writable	Yes
14	Token-Swap program id	N/A	Yes
15	Host fee account	Writable	Yes

- process_swap_serumdex

Index	Account Name	Writable/Signer	Checked
0	User Source Token Address	Writable	Yes
1	User Destination Token Address	Writable	Yes
2	User Source Token Authority	Signer	Yes
3	Spl-Token Program ID	N/A	Yes
4	OneSolProtocol AmmInfo Address	Writable	Yes
5	OneSolProtocol AmmInfo authority	N/A	Yes
6	OneSolProtocol AmmInfo token a account	Writable	Yes
7	OneSolProtocol AmmInfo token b account	Writable	Yes
8	Serum-dex market Address	Writable	Yes
9	Serum-dex request_queue	Writable	Yes
10	Serum-dex event_queue	Writable	Yes
11	Serum-dex market_bids	Writable	
12	Serum-dex market_asks	Writable	Yes
13	Serum-dex coin_vault	Writable	Yes
14	Serum-dex pc_vault	Writable	Yes
15	serum-dex vault_Signer	N/A	Yes
16	serum-dex open_orders	Writable	Yes
17	serum-dex rent_sysvar	N/A	Yes
18	serum_dex_program_id	N/A	Yes

- process_initialize_amm_info

Index	Account Name	Writable/Signer	Checked
0	New OneSolProtocol AmmInfo account	Writable, Signer	Yes
1	OneSolProtocol AmmInfo Authority	N/A	Yes
2	Owner account	N/A	Yes
3	Token_a_vault	Writable	Yes
4	Token_a_mint	Writable	Yes
5	Token_b_vault	Writable	Yes
6	Token_b_mint	Writable	Yes
7	Spl-Token program id	N/A	Failed

- process_initialize_dex_mark_open_orders

Index	Account Name	Writable/Signer	Checked
0	New OneSolProtocol DexMarket account	Writable, Signer	Yes
1	OneSolProtocol DexMarket account Authority	N/A	Yes
2	AmmInfo account	Writable	Yes
3	SerumDex Market account	Writable	Yes
4	SerumDex OpenOrders account	Writable	Failed
5	rend sysvar	N/A	Failed
6	SerumDex ProgramId	N/A	N/A

- process_swap_two_steps

Index	Account Name	Writable/Signer	Checked
0	TokenSwap swap_info account	N/A	Yes
1	TokenSwap swap_info authority	N/A	Yes
2	TokenSwap token_A Base Account	Writable	Yes
3	TokenSwap token_B Base Account	Writable	Yes
4	TokenSwap Pool token mint	Writable	Yes
5	TokenSwap Fee account	Writable	Yes
6	Token-Swap program id	N/A	Yes
7	Host fee account	Writable	Yes
8	serum-dex market	Writable	Yes
9	serum-dex request_queue	Writable	Yes
10	serum-dex event_queue	Writable	Yes
11	serum-dex market_bids	Writable	Yes
12	serum-dex market_asks	Writable	Yes
13	serum-dex coin_vault	Writable	Yes
14	serum-dex pc_vault	Writable	Yes
15	serum-dex vault_Signer	N/A	Yes
16	serum-dex open_orders	Writable	Yes
17	serum-dex rent_sysvar	N/A	Yes
18	serum-dex serum_dex_program_id	N/A	Yes
19	User Source Token Address	Writable	Yes
20	User Destination Token Address	Writable	Yes
21	User Source Token Authority	Signer	Yes
22	Spl-Token Program ID	N/A	Yes
23	OneSolProtocol AmmInfo2 account	Writable	Yes
24	OneSolProtocol AmmInfo2 authority	N/A	Yes
25	OneSolProtocol AmmInfo2 token a account	Writable	Yes
26	OneSolProtocol AmmInfo2 token b account	Writable	Yes

10. List of issues by severity

Index	Description	Issue/Risk	Severity	Status
N1	Data Deserialization Error	Data Serialization and Deserialization	Medium	
N2	Rent Exemption Not Checked	Design Vulnerability	Low	
N3	Spl-Token Program_id Not Checked	Design Vulnerability	Low	
N4	Inappropriate Comments	Design Vulnerability	Neutral	
N5	Validity of reference-only accounts not checked	Access Control	Low	
N6	Program_id not included	Design Vulnerability	Low	
N7	Reliability of Trust	Access Control	Low	

11. Issue descriptions

[N1] [Medium] Data Deserialization Error

Risk Severity: Medium

Issue/Risk: Data Serialization and Deserialization

Description:

Line 99 of the `src/program-rust/src/account_parse.rs` file has the following code section

```
let is_initialized = data[0];
```

With regard to data that is read from `SplTokenSwapInfo`, the first digit is its version number and the second digit is its initialization status.

Recommendation:

Consider changing the above statement to the following one:

```
let is_initialized = data[1];
```

Status:

[N2] [Low] Rent Exemption Not Checked

Risk Severity: Low

Issue/Risk: Logic Vulnerability

Description:

Neither the `process_initialize_amm_info` function in line 64 nor the `process_initialize_dex_mark_open_orders` function in line 124 of the `src/program-rust/src/processor.rs` file checks whether or not the initializing account is a rent exempt one.

Recommendation:

Consider using `rent.is_exempt` to compare lamports and data length

Status:

[N3] [Low] Spl-Token Program_id Not Checked

Risk Severity: Low

Issue/Risk: Design Vulnerability

Description:

The `process_initialize_amm_info` function defined in the `src/program-rust/src/processor.rs` file doesn't check whether or not the owner of `token_a_mint_info/token_b_mint_info` is the key of `spl_token_program_info`.

Recommendation:

Consider adding a statement to check the owner of `token_a_mint_info/token_b_mint_info` .

Status:

[N4] [Neutral] Inappropriate Comments

Risk Severity: Neutral

Issue/Risk: Design Vulnerability

Description:

The comment on `InitializeAmmInfo` in `OneSolInstruction` defined in the `src/program-rust/src/instruction.rs` file states that the pc vault Account is writable. In fact, `process::process_initialize_amm_info` doesn't update the data of pc vault Account, so it should not be marked as writable.The same applies to the pc_mint Account, coin_vault Account, coin_mint Account, and the ammInfo account of `InitDexMarketOpenOrders` as well.

Recommendation:

Consider revising these comments.

Status:

[N5] [Low] Validity of Reference-only Accounts Not Checked

Risk Severity: Low

Issue/Risk: Access Control

Description:

The `process_initialize_dex_mark_open_orders` function defined in the `src/program-rust/src/process.rs` file doesn't check whether or not the owner of amm_info_acc_info is the program itself. Therefore a malicious actor could create accounts with arbitrary data and pass these accounts to the program as valid accounts. The arbitrary data could be crafted in a way that leads to unexpected or harmful program behavior.

Recommendation:

Consider adding a statement to check the owner of amm_info_acc_info.

Status:

[N6] [Low] Program_id Not Included

Risk Severity: Low

Issue/Risk: Design Vulnerability

Description:

The statement `invoke_signed(&instruction, &swap_accounts[..], signers)?;` in line 790 of the `src/program-rust/src/process.rs` file doesn't include `token_swap_program_id` in `account_infos` , so a check of RefCell is missing. This applies to the call of `invoke_signed` in the `serum_dex_order.rs` file as well.

Recommendation:

Consider revising the code to include `token_swap_program_id` in `account_infos` .

Status:

[N7] [Low] Reliability of Trust

Risk Severity: Low

Issue/Risk: Access Control

Description:

The mainnet program account is upgradeable, the 1Sol team can upgrade the main network contract code and change the logic. This introduces reliability of trust.

Recommendation:

Consider deploying the program by running `solana program deploy <PROGRAM_FILEPATH> --final`

Status:

12. Recommendations to enhance the overall security

We list some recommendations in this section. They are not mandatory but will enhance the overall security of the system if they are adopted.

Consider adding a `size_of` statement for each type at the beginning of its `impl` section. For instance, in line 98 of the `state.rs` file, for the `AmmInfo` structure, consider adding `const DATA_LEN: usize = 280;`