



SolarDex AMM Smart Contracts

Review Assessment

SolarDex AMM / Solana Deployment






31 May 2022

Version: 1.0

Table of Contents

1. Audit Details	2
2. Overview	3
3. Scope	4
4. Analysis	5
5. Issues Status	6
6. Observation	7
7. General Remarks	8
7.1. Long files and functions	8
7.2. Code cleaning	8
8. Technical Analysis	9
9. Findings	10
9.1. Warnings	10
9.2. Failed Test	11
9.3. Code Formating	12
9.4. Bad Code Practice	13
Appendix B: The Classification of identified problems	14
Appendix A: AUDIT METHODOLOGY	15
Appendix C: Disclaimer	16

Audit Details

Project Name	SolarDex
 Audited project	Solardex AMM
 GitHub Code Repo	https://github.com/bloctech/solar.git
 Client Contact	SolarDex Team
 Blockchain	Solana
 Project Website	SolarDex.finance

Overview

Bloctech Solutions was engaged by the SolarDex Team to conduct a Smart Contracts Review Assessment in the form of a security code review of the SolarDex AMM Smart Contracts on the Solana blockchain. The assessment was conducted on the code pushed on Github Private Repo shared by the SolarDex Development team and focused on the following objectives:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract and include improvement recommendations based on the result of our tests.
- Ensuring contract logic meets the specifications and intentions of the client.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improving the security. 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.

Scope

The code review is based on the whitepaper and code provided by SolarDex Team. The code resides in a private repository at

<https://github.com/bloctech/solar.git>

All third-party libraries were deemed out of scope for this review and are expected to work as designed.

Analysis

This engagement was comprised of a code review including reviewing how the architecture has been implemented as well as any security issues. The architecture implementation review was based on the documentation. The implementation review concluded that the application implementation is well thought out, designed well, and uses the Solana features as designed.

The code review was conducted by the Bloctech Audit team on the code provided by SolarDex, in the form of a Github repository. The code review focused on handling secure and private information in the code.

As a result of our work, we identified 0 High, 0 Medium, 2 Low, and 2 Informational findings. The findings revolve around Code bases and file test units.

Issues Status

#	Issues Description	Status	Severity
01	Warnings	● Low	Resolved
02	Failed Test	● Low	Resolved
03	Code formatting	● Informational	Resolved
04	Bad Code Practice	● Informational	Acknowledged

Observations

We appreciate development team is using standard code provided by the Solana Core team for Swap Programs. We suggest that some refactoring is made to remove the challenges from large files and functions, a more consistent error handling model, and that duplicated code is refactored in such a way that the future maintenance of the code base will be less error-prone. We could not find any obvious memory leaks or any unsafe concurrency conditions.

This is a good sign of code maturity. Some design choices are suboptimal but functional based on the environment and limitations of the application. One such example is the handling of three month locking period of Liquidity.

During our initial review, it was noted by the review team that we assumed that the liquidity Locking would be handled outside of the SolarDex infrastructure. Still, after further discussion with the core team, this assumption was changed, and we agree with the recommendation that the Liquidity Locking thing will be handled with a Smart Contract. There will be no front-end or third-party dependency to execute the condition.

General Remarks

7.1 Long files and functions

Description:

Code files are very large, and functions are very long.

Recommendation:

- Refactor large functions by extracting code into separate functions.
- Move code to different files.

7.2 Code cleaning

Description:

The source files contain a lot of out-commented code. The lines that are ignored take up space and confuse other programmers about the intention of the code.

Recommendations:

Remove all out-commented code lines

Technical Analysis

Based on the source code validity of the code and correct implementation of the intended functionality has been verified to the extent that the state of the repository allowed. Further investigations were made which concluded that they did not pose a risk to the application. They were:

- No potential panics were detected
- No potential errors regarding wraps/unwraps, expect, and wildcards
- No internal unintentional unsafe references

Based on formal verification we conclude that the code implements the documented functionality to the extent of the code reviewed

Findings

9.1 Warnings:

The compiler generates warnings when building the code. Compiler warnings often indicate programming mistakes and should therefore be fixed at once.

```
warning: `solar-swap` (lib) generated 6 warnings
  Finished dev [unoptimized + debuginfo] target(s) in 1.28s
smart@bloctech:~/Documents/solana/solar/program/swap/program$
```

Recommendations:

Correct the code so the compiler does not produce any warnings. It should always be a priority that the code can be compiled without errors or warnings

Status :

Resolved

```
Checking curve25519-dalek v3.2.0
Checking sha3 v0.9.1
Checking digest v0.10.1
Checking hmac-drbg v0.3.0
Compiling proc-macro-crate v0.1.5
Compiling proc-macro-crate v1.1.0
Compiling borsh-derive v0.9.3
Compiling num_enum_derive v0.5.6
Compiling borsh-derive v0.8.2
Compiling libsecp256k1-gen-genmult v0.2.1
Compiling libsecp256k1-gen-ecmult v0.2.1
Compiling libsecp256k1 v0.6.0
Checking borsh v0.9.3
Checking num_enum v0.5.6
Checking spl-token v3.3.0
Checking spl-math v0.1.0
Checking solar-swap v2.1.0 (/home/smart/Documents/solana/solar/program/swap/program)
Finished dev [unoptimized + debuginfo] target(s) in 1m 52s
smart@bloctech:~/Documents/solana/solar/program/swap/program$
```

9.2 Failed Test:

When running tests for SolarDex AMM Two of the tests failed out of seventy-one. A failing test indicates that some functionality is broken. In this specific case, it seems that the functionality that the test is verifying has not yet been implemented. It should always be a priority that all tests pass. Automatic testing should be built into the continuous integration pipeline along with building to automatically ensure the high quality of the code.

```
state::tests::swap_v1_pack
test result: FAILED. 69 passed; 2 failed; 0 ignored; 0 measured; 0 filtered out; finished in 1.42s
error: test failed, to rerun pass '--p solar-swap --lib'
```

Recommendations:

Implemented the missing functionality to make the test pass

Status :

Resolved

```
test curve::stable::tests::constant_product_swap_no_fee ... ok
test result: ok. 71 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 1.53s
```

9.3 Code Formatting:

Not following a set of standard formatting rules makes code harder to read and understand by other developers than the author. Hence, it creates a risk if other developers should be able to maintain the code.

Recommendations:

Review and apply code formatting changes from rustfmt as a part of the development process. The code format check should be built into the continuous integration pipeline along with building and testing to automatically ensure high quality of the code.

Status :

Resolved

9.4 Bad Code Practice:

A linter is a tool that inspects the code for bad code practices of language constructs that lead to programming errors or incomprehensive code.

Some bad code practice have been noticed during test run some funtions have too many arguments that can lead to some potential errors.

```
warning: this function has too many arguments (11/7)  
--> swap/program/src/processor.rs:155:5
```

Recommendations:

It should always be a priority to remove all errors and warnings from indicated by the linters.

Status :

Acknowledged

Appendix A: AUDIT METHODOLOGY

1. Design Patterns:

We inspect the structure of the smart contract, including both manual and automated analysis.

2. Static Analysis:

The static analysis is performed using a series of automated tools, purposefully designed to test the security of the contract. All the issues found by tools were manually checked (rejected or confirmed).

3. Manual Analysis:

Contract reviewing to identify common vulnerabilities. Comparing requirements and implementation. Reviewing of a smart contract for compliance with specified customer requirements.

Appendix B: The Classification of identified problems

Issues is listed from most critical to least critical. Severity is determined by an assessment of the risk of exploitation or otherwise unsafe behavior.

Severity Levels:

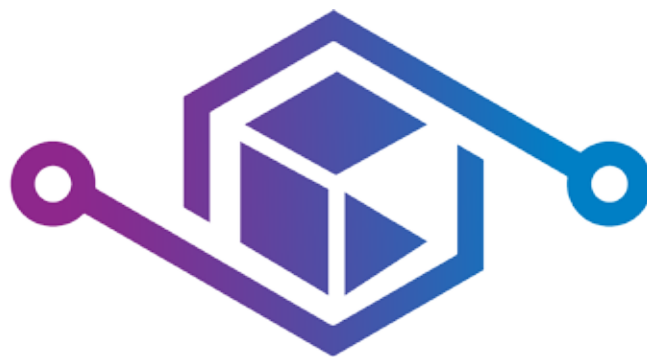
- Critical - Funds may be allocated incorrectly, lost, or otherwise result in a significant loss.
- Medium - Affects the ability of the contract to operate.
- Low - Minimal impact on operational ability.
- Informational - No impact on the contract.

Appendix C: Disclaimer

BlocTech Solutions team conducts security assessments on the provided source code exclusively. To get a full view of our analysis, you must read the full report. We have done our best in conducting our analysis and producing In this report, it is important to note that you should not rely on this report and cannot claim against us based on what it says or does not say, or how we produced it, and you need to conduct your independent investigations before making any decisions. The audit documentation is for discussion purposes only. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

This material is offered solely for informational reasons and is not intended to be relied upon as investment advice. BlocTech Solutions and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) owe no duty of care to you or anyone else, and BlocTech Solutions makes no warranty or representation to anyone about the report's reliability or completeness.

BlocTech Solutions hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose, and the application of reasonable care and skill) that could affect the report if this provision were not included. Except to the extent prohibited by law, BlocTech Solutions hereby disclaims all liability obligation, and neither you nor any other person shall have any claim against BlocTech Solutions for any amount or kind of loss or damage that may arise to you or any other person (including, without limitation, any direct, indirect, special, punitive, consequential, or pure economic loss or damages, or any loss of income, profits, data, contracts, use of money, or business disruption). In any way arising from or connected with this report and the use, inability to use, or results of the use of this report, and any reliance on this report, whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent), or otherwise under any claim of any nature anywhere in any jurisdiction). This cyber immunity-bearing report is evaluated on the basis of smart contracts. There were no security checks performed on any apps or activities. There hasn't been a review of any product codes.



BlocTech
SOLUTIONS