Sovryn Security Review

by Pessimistic

This report is public.

Published: October 7, 2020

Abstract....2
Disclaimer....2
Summary....2
General recommendations....2
Procedure....3
Project overview....4
Project description....4
Manual analysis....5
Critical issues....5
New contract setup....5
Medium severity issues....5
Low severity issues....6
Code style....6

Abstract

In this report, we consider the quality of four pull requests for smart contracts of the Sovryn project. Our task is to find and describe security issues in these requests.

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, security audit is not an investment advice.

Summary

In this report, we considered the security of four pull requests for smart contracts of Sovryn project without reviewing the project itself. We performed our audit according to the procedure described below.

The initial audit of 4 pull requests showed a vulnerability (critical issue) and a few code style issues.

In the latest pull requests, the vulnerability and two code style issues were fixed.

However, the project includes contracts of enormous size, up to a point where it causes `out of gas` and `stack too deep` issues. This affects how new changes are implemented, complicates development, and creates a long-term risk for the project.

General recommendations

We recommend refactoring the code. All large smart contracts should be split into smaller parts, their logic should be simple and transparent. We also recommend using linters and static code analyzers

Procedure

In our audit, we consider the following crucial features of the code:

1. Whether the code is secure.
2. Whether the code meets best practices.

We perform our audit according to the following procedure:

- Manual audit
  - We manually analyze code base from the scope for security vulnerabilities.
- Report
  - We reflect all the gathered information in the report.

Project overview

Project description

In our analysis we consider smart contracts from two GitHub repositories of Sovryn project:

- Sovryn-smart-contracts
- oracle-base-amm

The scope of the audit included 4 pull requests. Initial repositories were not audited:

- https://github.com/DistributedCollective/Sovryn-smart-contracts/pull/28
- https://github.com/DistributedCollective/Sovryn-smart-contracts/pull/30
- https://github.com/DistributedCollective/Sovryn-smart-contracts/pull/42
- https://github.com/DistributedCollective/oracle-based-amm/pull/1

Latest pull requests

After initial audit, the developers applied fixes and made new pull requests:

- https://github.com/DistributedCollective/Sovryn-smart-contracts/pull/48
- https://github.com/DistributedCollective/oracle-based-amm/pull/8

Manual analysis

Four pull requests were manually analyzed, their logic was checked. Here were describe all the issues found.

Critical issues

Critical issues seriously endanger smart contracts security. We highly recommend fixing them.

New contract setup (fixed)

In **oracle-based-amm** repository, pull request 1, there are two functions for new contract deployment in **ConverterRegisry** contract:

- `newConverter()` function that performs the deployment.
- `setupConverter()` function that performs the setup newly deployed contract.

`setupConverter()` function can be called only once for newly created contract. It sets `msg.sender` as an owner of the contract. However, this function can be called by anyone and is vulnerable to the front-running attack. Thus, anyone can become an owner of the deployed contract.

The issue has been fixed in pull request 8.

Medium severity issues

Medium issues can influence project operation in current implementation. We highly recommend addressing them.

The audit showed no issues of medium severity.

Low severity issues

Low severity issues can influence project operation in future versions of code. We recommend taking them into account.

Code style

In pull request 30 for Sovryn-smart-contracts repository, there are two code style issues in **modules/LoanClosings.sol**:

- Time intervals are set in seconds:

  `.add(86400)` or `(backInterestTime >= 2628000)`.

  Consider using interval `1 days` and declaring a constant `uint constant internal MONTH = 365 days / 12;` instead to improve code readability.

  The issue has been fixed in pull request 48.

- At line 386, both sides of assignment use `destTokenAmountReceived`:

```
(destTokenAmountReceived , ,) = _swapBackExcess(
  loanLocal,
  loanParamsLocal,
  destTokenAmountReceived - interestAmountRequired,  //amount to be swapped
  loanDataBytes);
```

  Consider declaring a new variable instead.

> *Comment from developers: This change would lead to a stack too deep error. This function will need to be refactored along with most of the code and we're planning to do it step by step in the coming weeks.*

The following issue was found during general overview of the project. It is located out of the scope of the audit. However, to bring more use to the developers, we included the issue into the audit report:

- In **contracts/swaps/SwapsUser.sol** of **Sovryn-smart-contracts** repository, lines 127–157, there are 23 lines of commented code. We recommend removing commented code and replace if-else statement with `require()` to reduce nesting and thus improve code readability.

  The issue has been fixed in pull request 48.

This analysis was performed by Pessimistic:

Evgeny Marchenko, Senior Security Engineer
Boris Nikashin, Analyst
Alexander Seleznev, Founder

October 7, 2020