

Summary

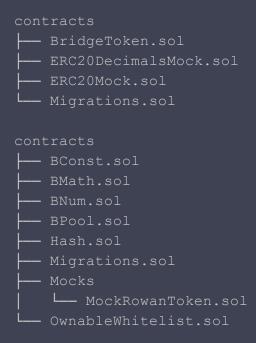
Audit Report prepared by Solidified covering the Sifchian token sale smart contracts (and their associated components).

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The debrief took place on November 9th, 2020, and the results are presented here.

Audited Files

The following contracts were covered during the audit:



Supplied in the following source code repositories:

https://github.com/Sifchain/balancer

https://github.com/Sifchain/eRowan-ERC20

Notes

The audit was based on commit numbers 0afb5d17e8c49d0ce10ba36438a5888b468a88ca and 28717430a3574b8ee1f0d2f880f6c0be62d16c08



Fixes were submitted in commit 773b8487701b2c9997f66678c41d200baad8e95c

Intended Behavior

The smart contract implements a token sale based on a modified version of the Balancer liquidity pool smart contract. In order to function as a token sale, the following modifications have been applied:

- Only a controller can provide liquidity to the pool
- Swap fees will are set to 0
- Users can only swap one-way (intended for buying eRowan with USDT)
- Only whitelisted addresses can buy



Executive Summary

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than a security audit and vice versa.

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	Medium-High	-
Level of Documentation	Medium	-
Test Coverage	Medium-High	-



Issues Found

Solidified found that the Sifchain token sale contracts contain no critical issue, no major issues, and 1 minor issue, in addition to 3 informational notes.

We recommend all issues are amended, while the notes are up to the team's discretion, as it refers to best practices.

Issue #	Description	Severity	Status
1	Anyone can provide liquidity (in contrast to specification)	Minor	Resolved
2	Zero fees cause unnecessary calculations	Note	Resolved
3	Unnecessary factory address	Note	-



Critical Issues

No critical issues have been found.

Major Issues

No major issues have been found.

Minor Issues

1. Anyone can provide liquidity (in contrast to specification)

Anyone can send eRowan or USDT directly to the contract and then use gulp() to absorb the tokens into the balance. This may not have a direct impact but circumvents the desired behavior as specified.

There may be a non-obvious impact on price calculations.

Recommendation

Limit gulp() function access to the controller.

Update

Fixed

Notes

2. Zero fees cause unnecessary calculations

SWAP_FEE and EXIT_FEE are hardcoded to 0 for this particular use case. There are, nevertheless, used for calculations and EXIT_FEE is even used to make an ERC-20 transfer of 0 tokens

Another result of this is that getSpotPriceSansFee() and getSpotPrice() provide duplicate functionality.

Recommendation

Consider removing fees from the calculation.



Update

The recommendation has been applied.

3. Unnecessary factory address

The _factory address in BPool.sol is unnecessary, since the original BFactory contract from the Balancer codebase has been removed. It is set to be the same as the controller address and only used for pushing making a 0 amount ERC-20 token transfer (see above).

Recommendation

Consider removing the variable for readability.



Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of Black Hole Industried or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Solidified Technologies Inc.