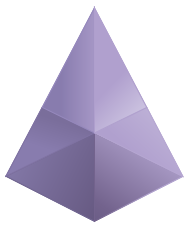

STAKEWISE

Stakewise Security Analysis

by Pessimistic

This report is public

September 16, 2022

Abstract

In this report, we consider the security of smart contracts of Stakewise project. Our task is to find and describe security issues in the smart contracts of the platform.

Disclaimer

The audit does not give any warranties on the security of the code. A single 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, a security audit is not investment advice.

Summary

In this report, we considered the security of Stakewise smart contracts. We performed our audit according to the procedure described below.

The audit showed one issues of medium severity: Non-typical vesting claiming which was addressed by the developers. Also, several low-severity issues were found.

The overall code quality is good.

After the initial audit, the codebase was updated to fix the issue discovered by developers.

General recommendations

We recommend addressing the remaining low severity issues and issues from the previous audit.

Project overview

Project description

For the audit, we were provided with Stakewise project on a public GitHub repository on the following commits:

- 21b15b28933c00894d46c3f1debd0ef28e9779bf for Ethereum blockchain
- 3a5d7c9fab8eea284f413e2e0b0a86d85936c990 for Gnosis blockchain
- 6b5ede8775b0fee634ba35b112e0792e5f90d507 for Portara project

The scope of the audit includes all contracts.

The project has detailed user documentation.

215 tests out of 217 pass successfully. The code coverage is unknown.

The total LOC of audited sources is 1737 (on `21b15b28933c00894d46c3f1debd0ef28e9779bf` commit).

Project development includes the CI process to run tests and calculate code coverage.

Codebase Update

After the initial audit, the codebase was updated. We were provided with the following commits:

- 81e12be91939215cf22909d99010437b3b0ad50d for Ethereum mainnet.
- 9ea26352ab31bf0137a7b61026bfb940306071a2 for Gnosis blockchain
- 8cb58d2c460f70e7e65c68468fd70745f494ec06 for Portara project

These changes fix the bug earlier discovered by the project team.

Previous audit

Previous audit of the codebase has been performed on commits `5e43ba4b820676dea0147fd2e212ff8658ae8c06` and `27d11f3d1b50bcaec66e60ec0df332c561523d44`. That audit showed no issues of critical or medium severity. Two low severity issues (erc-20 event emitting and gas optimization) from the previous report stay unaddressed in this version.

Procedure

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

1. Whether the code is secure.
2. Whether the code corresponds to the documentation (including whitepaper).
3. Whether the code meets best practices.

We perform our audit according to the following procedure:

- Automated analysis
  - We scan the project's codebase with the automated tools: Slither and Smartcheck.
  - We manually verify (reject or confirm) all the issues found by the tools.
- Manual audit
  - We manually analyze the codebase for security vulnerabilities.
  - We assess the overall project structure and quality.
- Report
  - We reflect all the gathered information in the report.

We specifically analyzed the changes in the codebase regarding the logic that implements receiving priority fees from validators to the Stakewise protocol. No issues were found in this functionality.

Inter alia, we verify that:

- No standard Solidity issues are present in the codebase
- The token reward logic is implemented correctly
- The system provides proper frontrunning protection
- All Ether obtained as MEV is accounted for in the rewards

Manual analysis

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

The audit showed no critical issues.

Medium severity issues

Medium issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

M01. Non-typical vesting claiming (addressed)

It is expected that the transfer in the `claim` would be sent to the recipient. However, the transfer goes to the beneficiary's address at line 123 in `claim` function of **VestingEscrow** contract.

> *Comment from the developers: This behavior is intentional*

M02. Inaccurate total supply update (fixed)

`updateTotalRewards` function of **RewardEthToken** contract accounts for Ether that the protocol receives as a priority fee through the **FeesEscrow** contract. However, the function did not update `totalRewards` value properly. This issue did not affect the system's consistency; however, it could cause integration problems since the reward token total supply depends on this variable.

The issue has been fixed and is not present in the latest version of the code.

Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

L01. EIP712 version is not updated

The `version` part of the signature should be changed when upgrading to a new version for all upgrades of ERC20PermitUpgradeable tokens (e.g., **RewardEthToken** contract).

L02. Overpowered role

Admin can set the number of activated validators along with the oracles in `setActivatedValidators` function of **Pool** contract.

L03. Redundant checks

Zero-checks at lines 129 and 175 in **RewardEthToken** make the code complicated and are not necessary. Consider including the zero amount case inside the `_calculateNewReward` function to simplify the codebase.

L04. Code duplication

The repetitive signature uniqueness check in **Oracles** contract might be moved to a separate internal function.

L05. Code quality

The hardcoded address at line 43 in the `upgrade` function of **MerkleDistributor** contract should be documented.

L06. Code quality

Consider declaring these functions as external instead of public:

- `nonces` and `permit` in **ERC20PermitUpgradeable** contract
- `transfer`, `transferFrom`, `approve`, `increaseAllowance`, `decreaseAllowance`, `allowance`, `name`, `symbol` and `decimals` in **ERC20Upgradeable** contract
- `updateRewardCheckpoint` and `updateRewardCheckpoints` in **RewardEthToken** contract
- `unclaimedAmount` in **VestingEscrow** contract

Notes

N01. Overpowered owner (addressed)

The owner of **PoolEscrow** contract can send assets to an arbitrary address. We recommend designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig.

> *Comment from the developers:* ***PoolEscrow*** *owner is DAO address, who only performs transactions upon voting.*

This analysis was performed by Pessimistic:

Evgeny Marchenko, Senior Security Engineer
Vladimir Tarasov, Security Engineer
Irina Vikhareva, Project Manager

September 16, 2022