zeroswap

SMART CONTRACT AUDIT

ZOKYO.

Nov 2nd, 2021 | v. 1.0

PASS

Zokyo Security team has concluded that the given codebase passes security qualifications

SCORE

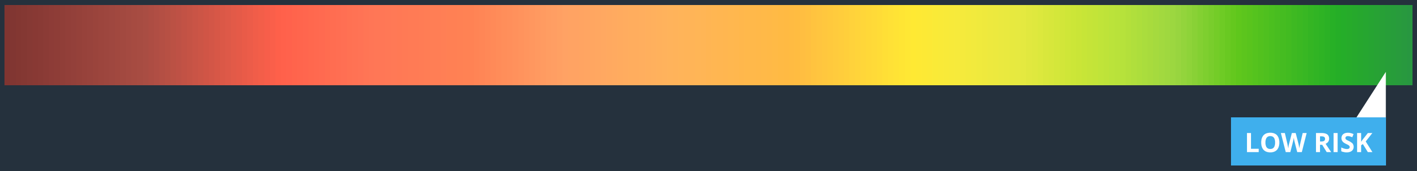
100

TECHNICAL SUMMARY

This document outlines the overall security of the ZeroSwap smart contracts, evaluated by Zokyo's Blockchain Security team.

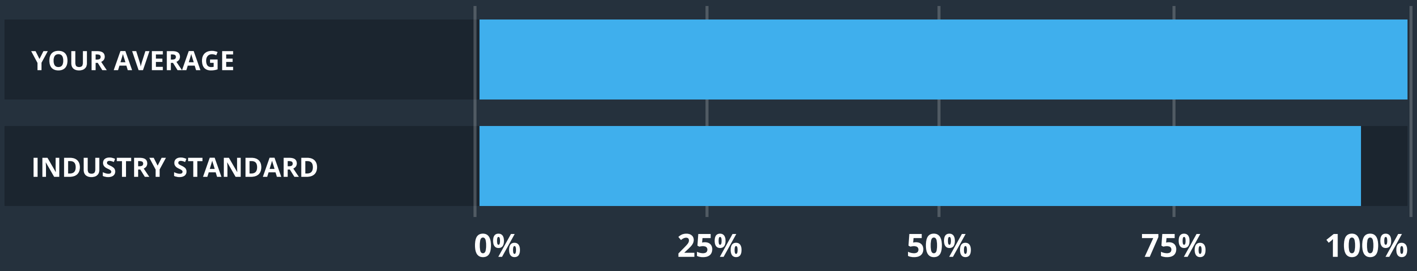
The scope of this audit was to analyze and document the ZeroSwap smart contract codebase for quality, security, and correctness.

Contract Status



There were no critical issues found during the audit.

Testable Code



The testable code is 100%, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the ZeroSwap team put in place a bug bounty program to encourage further and active analysis of the smart contract.

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the ZeroSwap archive file.

Repository:
https://github.com/ZeroSwapLabs/zeefarm-contract/
commit/2afe9010940c2d39bcfa97639482a5913ae134b8

Contracts under the scope:

- Utils;
- Vault;
- VaultFactory;
- ZeeFarm;
- ZeeFarmFactory.

Throughout the review process, care was taken to ensure that the token contract:

- Implements and adheres to existing Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1. Due diligence in assessing the overall code quality of the codebase.
2. Cross-comparison with other, similar smart contracts by industry leaders.
3. Testing contract logic against common and uncommon attack vectors.
4. Thorough, manual review of the codebase, line-by-line.

SUMMARY

Zokyo auditing team has run a deep investigation of ZeroSwap's smart contracts. The contracts are in good condition. They are well written and structured.

During the auditing process, there were two issues found. After the technical discussion with ZeroSwap team, we can state that these issues is not valid and marked in the doc accordingly.

Based on the conducted audit, we give a score of 100 to the aforementioned contracts.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

● Critical

The issue affects the ability of the contract to compile or operate in a significant way.

● High

The issue affects the ability of the contract to compile or operate in a significant way.

● Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

● Low

The issue has minimal impact on the contract’s ability to operate.

● Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

Change the state before transferring the funds

LOW | NOT VALID

In Zeefarm.sol file, in releaseRewards(), move the line-164 before the line-158, that is update the state before transferring the funds or value.

Recommendation:
Place the statement

```
user.rewardClaimed[index] += totalPending;
```

before

```
vault.safeRewardTransfer(rewardToken,msg.sender,totalPending);
```

Comment:
Since there is no chance of a reentrancy attack, the state change doesn’t affect it.

Change the state before transferring the funds

LOW | NOT VALID

In Zeefarm.sol file, in releaseRewards(), move the line-181 before line-176, that is update the state before transferring the funds or value.

Recommendation:
Place the statement

```
user.rewardClaimed[index] += pendingClaim;
```

before

```
vault.safeRewardTransfer(rewardToken,msg.sender,pendingClaim);
```

Comment:
Since there is no chance of a reentrancy attack, the state change doesn't affect it.

	Utils	Vault	VaultFactory
Re-entrancy	Pass	Pass	Pass
Access Management Hierarchy	Pass	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass	Pass
Unexpected Ether	Pass	Pass	Pass
Delegatecall	Pass	Pass	Pass
Default Public Visibility	Pass	Pass	Pass
Hidden Malicious Code	Pass	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass	Pass
External Contract Referencing	Pass	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass	Pass
Unchecked CALL Return Values	Pass	Pass	Pass
Race Conditions / Front Running	Pass	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass	Pass
Floating Points and Precision	Pass	Pass	Pass
Tx.Origin Authentication	Pass	Pass	Pass
Signatures Replay	Pass	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass	Pass

	ZeeFarm	ZeeFarmFactory
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Security team

As part of our work assisting ZeroSwap in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Truffle testing framework.

Tests were based on the functionality of the code, as well as a review of the ZeroSwap contract requirements for details about issuance amounts and how the system handles these.

Code Coverage

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts\	100.00	95.00	100.00	95.00	5
ZeeFarm.sol	100.00	95.00	100.00	95.00	
ZeeFarmFactory.sol	100.00	100.00	100.00	100.00	
VaultFactory.sol	100.00	100.00	100.00	100.00	
Vault.sol	100.00	100.00	100.00	100.00	
Utils.sol	100.00	100.00	100.00	100.00	
All files	**100.00**	**95.00**	**100.00**	**95.00**	

Test Results

```
Contract: ZeeFarmFactory,VaultFactory
  deploy function
    ✓ change the vaultFactory address (602ms)
    ✓ deploying of ZeeFarm should be failed (782ms)
    ✓ deploying the ZeeFarm contract (830ms)
    ✓ getting the count of deployed ZeeFarm contracts (150ms)
```

```
Contract: ZeeFarm
  Deploying the contract
    ✓ deploying ZeeFarm contract with rewardTokens and rewardPerBlock arrays
      as unequal (423ms)
    ✓ deploying ZeeFarm contract with blockDuration less than claimDelayDuration (378ms)
    ✓ deploying ZeeFarm contract with rewardTokens.length greater than 4 (437ms)
    ✓ deploying ZeeFarm contract with invalid vault address (326ms)
    ✓ deploying ZeeFarm contract with invalid stakingToken address (297ms)
    ✓ deploying ZeeFarm contract with invalid rewardToken addresses (346ms)
    ✓ deploying ZeeFarm contract (598ms)
  Staking and other functions
    ✓ trying to stake before the farming has started (713ms)
    ✓ trying to stake after the farming has ended (728ms)
    ✓ staking 20 tokens (1512ms)
    ✓ staking 10 tokens again (1741ms)
    ✓ getting the user data (115ms)
    ✓ getting the pending rewards (137ms)
    ✓ UnStaking the tokens greater than the existing (322ms)
    ✓ UnStaking the 10 tokens (1046ms)
    ✓ emergencyRescue with a token as stakingToken (854ms)
    ✓ Emergency unStaking (604ms)
    ✓ withdraw remaining rewards before farming has ended (297ms)
    ✓ withdraw remaining rewards after farming has ended (412ms)

23 passing (19s)
```

We are grateful to have been given the opportunity to work with the ZeroSwap team.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

Zokyo's Security Team recommends that the ZeroSwap team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.