Shield Finance

SMART CONTRACT AUDIT

ZOKYO.

May 6, 2021 | v. 1.0

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

TECHNICAL SUMMARY

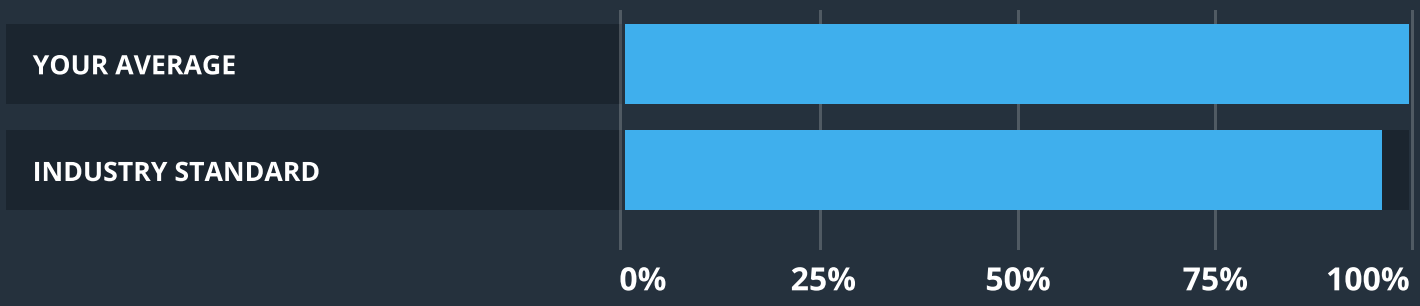
This document outlines the overall security of the Shileld Finance smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the Shield Finance token 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 Shield Finance 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 3

Executive Summary 4

Structure and Organization of Document 5

Complete Analysis 6

Code Coverage and Test Results for all files 9

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the Shield Finance repository.

Repository - https://github.com/ShieldFinanceHQ/contracts
Commit id - 65e9431eea2cd89608a9bbd9a771404b61c25c46

Last commit reviewed - 3cc6de851cc401b5c02d493950aac126dcbaea8c

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 Shield Finance 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.	3	Testing contract logic against common and uncommon attack vectors.
2	Cross-comparison with other, similar smart contracts by industry leaders.	4	Thorough, manual review of the codebase, line-by-line.

EXECUTIVE SUMMARY

There were no critical issues found during the audit. All the mentioned findings may have an effect only in case of specific conditions performed by the contract owner.

The findings during the audit have a slight impact on contract code style and only may have impact during further development.

Nevertheless, all findings were successfully resolved by the Shield Finance team.

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

HIGH | RESOLVED

TODO in pre-production code

Line 84
TODO statements in pre-production code for allocations calculations. This statement needs clarification, especially from the point of view of the non-audited code potentially added.

Recommendation:
Clarify the statement and finish the logic.

HIGH | RESOLVED

Missing total amount calculation

Line 83, No calculations and checks for the total amount added through the addAllocations() method.
There are no checks if the total amount exceeds the current supply, or if the new portion of allocations added to the same vesting type exceeds the allocations, or if the next vesting type amount added will exceed the total supply. This is the crucial point because there is no logic for changing the frozen wallet with the allocation.

Recommendation:
Finish the total amount checking logic.

Post-audit:
After the conversation with the Shield Finance team, auditors verified the functionality and its coverage with tests. The issue is marked as resolved.

MEDIUM | RESOLVED

Missing mint method

Line 95, _mint() function overloading
Token's initializer already provides mint for the full maximum supply (line 48, mint for getMaxTotalSupply()). Though the token can enable burn functionality, so more place for minting appears. Though, overloaded _mint() function is called from nowhere but from the initializer, which is called only once. Looks like either public mint() function is absent, or overload _mint() method is unnecessary.

Recommendation:
Clarify the minting logic and the necessity of _mint() and mint() methods.

LOW | RESOLVED

Extra variable

Line, 95, _mint() function.
Since there is overloaded totalSupply() method, prefix "super.totalSupply()" can be omitted so as the local variable. Require statement may be simplified to "getMaxTotalSupply() >= (totalSupply() + amount)". For now such statements are confusing and misleading,

Recommendation: simplify _mint() function.

LOW | RESOLVED

Commented code

Line, 228, 260, commented line.

Recommendation: remove commented code to prevent misleading.

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

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Secured team

```
Tests set for ShieldToken
  ShieldToken initialization
    √ Correct release time must be set (44ms)
    √ Can set release time on the past (571ms)
    √ Correct burnBefore option must be set as false (45ms)
    √ Correct token properties must be set (72ms)
    √ Correct amount of tokens must be minted (39ms)
  ShieldToken writing functionality
    √ Default users are not allowed to add vesting type (214ms)
    √ Adds vesting type correct (145ms)
    √ Must not add new allocations by default user (76ms)
    √ Must add new allocations after release (364ms)
    √ Must transfer to many users (207ms)
    √ Must withdraw ETH correct (187ms)
    √ Must withdraw ERC20 token (153ms)
    Security testset
      √ Must transfer after defence is over (351ms)
      √ Must disable defense (194ms)
      √ Disables transfers (157ms)
      √ Pause unpause check (107ms)
      √ Can't change release time after release (62ms)
    ShieldToken view functionality
      √ If getUnlockedAmount() gets no months
      √ getLockedAmount() returns proper value (83ms)
      √ User can transfer condition <true> (40ms)
      √ User can transfer condition <false> (106ms)
      √ If total transferable amount greater than frozen supply - returns frozen supply (491ms)


22 passing (5s)
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts\ Token.sol	100	97.83	100	100	
All files	100	97.83	100	100	

We are grateful to have been given the opportunity to work with the Shield Finance 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 Shield Finance team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.