# Spool

## SMART CONTRACT AUDIT

ZOKYO.

January 21st, 2022 | 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

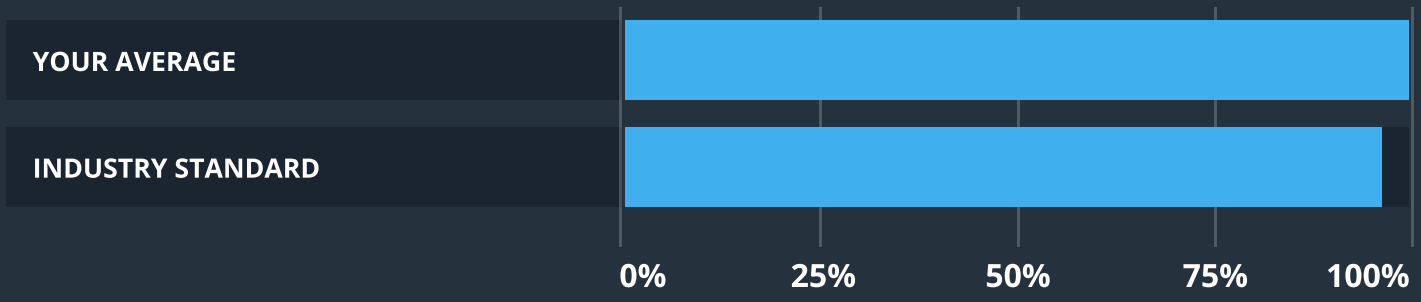
This document outlines the overall security of the Spool smart contracts, evaluated by Zokyo's Blockchain Security team.

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

## Contract Status


There were no critical or high 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 security of the contract we at Zokyo recommend that the SpoolFi DAO 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 Spool smart contract's source code was taken from the repository provided by SpoolFi DAO: https://github.com/SpoolFi/spool-dao-contracts

Audited commit: d623650aca76bb21aa072ecbf7dd3bcb764aea8f

Last commit (post-audit): 7a8b7735322ee8bed367016d4ab4e3df0f62617c

Within the scope of this audit Zokyo auditors have reviewed the following contract(s):

- BaseVesting.sol
- SpoolBuildersVesting.sol
- SpoolPreDAOVesting.sol
- voSPOOL.sol

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

- Implements and adheres to existing standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Follows best practices in efficient use of resources, 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 Spool 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. 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 or high issues found during the audit. Although certain number of medium and low issues were discovered, they relate to the following:

- Function can be called more than once
- Subtraction might revert.

After recommendations by Zokyo auditors, all issues that influence security and efficiency were fixed by SpoolFi DAO.

# 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 contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

### 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

### SpoolBuildersVesting.sol

MEDIUM | RESOLVED

**Function can be called more than once.**

line 41, function setVests(). It is said in comments, that function allows one to set vests once, however there are no restrictions to that, so that the function can be called more than once.

**Recommendation:**
Restrict function from being called more than once or change the comments to function.

### SpoolPreDAOVesting.sol

MEDIUM | RESOLVED

**Function can be called more than once.**

line 45, function setVests(). It is said in comments, that function allows one to set vests once, however there are no restrictions to that, so that the function can be called more than once.

**Recommendation:**
Restrict function from being called more than once or change the comments to function.

**BaseVesting.sol**

LOW | RESOLVED

**Subtraction might revert.**

line 250, function _setVest(). Since there is no restrictions to re-set amount of user's tokens, it is possible that the new amount will be less than previous. In this case, subtraction will underflow.

**Recommendation:**
Verify that a passed value can’t be less, than previously recorded one.

| | voSPOOL | BaseVesting |
|---|---|---|
| Re-entrancy | Pass | Pass |
| Access Management Hierarchy | Pass | Pass |
| Arithmetic Over/Under Flows | Pass | Pass |
| Delegatecall Unexpected Ether | 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 |

| | SpoolBuildersVesting | SpoolPreDAOVesting |
|---|---|---|
| Re-entrancy | Pass | Pass |
| Access Management Hierarchy | Pass | Pass |
| Arithmetic Over/Under Flows | Pass | Pass |
| Delegatecall Unexpected Ether | 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 Secured team

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

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

**Contract: SpoolBuildersVesting**
- Setting vests
  - ✓ Owner should set vests (338ms)
  - ✓ Owner should set vests twice with different member sets (535ms)

**Contract: SpoolPreDAOVesting, BaseVesting**
- ✓ Should revert if deployment with zero spool token address (48ms)
- Setting vests
  - ✓ Owner should set vests (121ms)
  - ✓ Owner should set vests twice with different amounts (478ms)
  - ✓ Owner should set vests twice with different member sets (195ms)
  - ✓ Owner should set vests if the negative difference from previous member totals (413ms)
  - ✓ Owner should not set vests if numbers of participants and amounts are not equal
  - ✓ Owner should not set vests if the user is passed twice
  - ✓ Owner should not set vests if a vesting has already started (263ms)
- Beginning a vesting
  - ✓ Owner should begin a vesting (150ms)
  - ✓ Owner should not begin a vesting if it has already started (148ms)
- Claiming
  - ✓ Should claim (486ms)
  - ✓ Should not claim if zero amount
- Vest transfer
  - ✓ Owner should transfer a vest (591ms)
  - ✓ Owner should transfer a vest with a new member adding (866ms)
  - ✓ Owner should not transfer a vest if no previous vested amount for an address
- If a vesting has not started
  - ✓ Should not get the amount which a user can claim
  - ✓ Should not claim
  - ✓ Owner should not transfer a vest
- Only owner can
  - ✓ Transfer a vest

✓ Begin a vesting
✓ Set vests

**Contract: voSPOOL**

Authorization for an address to mint or burn tokens
✓ Owner should authorize an address
✓ Owner should not authorize a zero address
✓ No owner should not authorize an address

Minting
✓ Authorized user should mint (39ms)
✓ No authorized user should not mint

Burning
✓ Authorized user should burn (96ms)
✓ No authorized user should not burn

Prohibited actions
✓ Transfer
✓ Transfer tokens from a sender to a recipient
✓ Approval
✓ Allowance increasing
✓ Allowance decreasing

36 passing (8s)

| FILE | % STMTS | % BRANCH | % FUNCS |
|---|---|---|---|
| voSPOOL | 100.00 | 100.00 | 100.00 |
| BaseVesting | 100.00 | 100.00 | 100.00 |
| SpoolBuildersVesting | 100.00 | 100.00 | 100.00 |
| SpoolPreDAOVesting | 100.00 | 100.00 | 100.00 |
| **All files** | **100** | **100** | **100** |

We are grateful to have been given the opportunity to work with SpoolFi DAO.

**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 SpoolFi DAO put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.