SMART CONTRACT AUDIT

ZOKYO.

Jan 7th, 2022 | v. 1.0

PASS

Zokyo Security team has concluded that this smart contract pass security qualifications and is fully production-ready

TECHNICAL SUMMARY

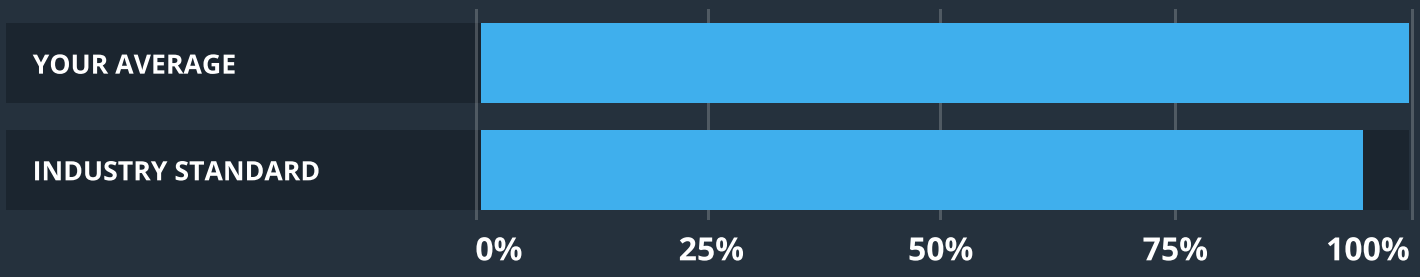
This document outlines the overall security of the ShapeShift smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the ShapeShift 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 96%, 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 ShapeShift 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:
https://github.com/shapeshift/yearn-router/tree/5c08b4c69bcce48be63980513179476744b9b09

Last commit –
https://github.com/ferrumnet/open-staking/commit/cc9ca9b0b9edc90ccd0534b83519e9ab21bb46d9

Contracts:

- ShapeShiftDAORouter.sol

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

The Zokyo team has conducted a security audit of the given codebase. The contract provided for an audit is well written and structured. All the findings within the auditing process are presented in this document.

During the auditing process, our auditor's team found 1 issue with a high severity level and 1 issue with a low severity level, which were successfully resolved by the ShapeShift team.

Based on the results of the audit, we can give a score of 99 and state that the audited contract is fully production-ready.

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

Function totalVaultBalance() doesn't use firstVaultId variable

HIGH | RESOLVED

Cycle "for (uint256 i = 0; i <= _lastVaultId; i++)" always starts from 0 but must start from the firstVaultId variable.

Recommendation:
Change "uint256 i = 0" to "uint256 i = firstVaultId"

Re-audit:
Fixed.

Functions _migration and _withdraw have unused statements migrator != address(this) and withdrawer != address(this)

LOW | RESOLVED

Function _withdraw can be called by functions: _migrate, withdraw(address,address), withdraw(address, address, uint256), withdraw(address, address, uint256, uint256, uint256). They all set the withdrawer as _msgSeder().

The same situation is with migrator. Function _migrate can be called by functions: migrate(address), migrate(address, uint256), migrate(address, uint256, uint256, uint256). They all set the migrator as _msgSeder().

Comment:
2 unused statements were removed but one was left on line 365.
Commit – 3b65405d97655236b3174f4318ad1c413f01e3f2.

Re-audit:
Fixed.

	ShapeShiftDAORouter.sol
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 Security team

As part of our work assisting Ferrum Network 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 Ferrum Network 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\	96.47	89.47	100.00	96.39	3
ShapeShiftDAORouter	96.47	89.47	100.00	96.39	3
All files	**96.47**	**89.47**	**100.00**	**96.39**	3

Test Results

```
Contract: ShapeShiftDAORouter
  Methods
    constructor
      ✓ Must be deployed correctly (248ms)
    setRegistry
      ✓ Must fail if caller isn't owner (586ms)
      ✓ Must fail if new registry has different governance than previous registry (274ms)
      ✓ Must set new registry correctly (362ms)
    numVaults
      ✓ Must return 0 if passed wrong address of token (489ms)
      ✓ Must return correct number of vaults with specific token (2173ms)
```

latestVault
- ✓ Must return correct address of latest vault (1403ms)

vaults
- ✓ Must return zero address if passed wrong arguments (173ms)
- ✓ Must return correct vault object (173ms)

totalVaultBalance
- ✓ Mustn't change the variables in storage (454ms)
- ✓ Must return correct vault balance of user (3098ms)
- ✓ Must fail if firstVaultId is bigger than lastVaultId (122ms)
- ✓ Must get user's vault balance from all vaults (1389ms)

deposit
- ✓ Must fail if depositor hasn't approved tokens to the router (635ms)
- ✓ Must fail if user doesn't have enough tokens to deposit (601ms)
- ✓ Mustn't change the variables in storage (893ms)
- ✓ Must deposit all user's tokens (1445ms)
- ✓ Must deposit ERC20 token to last vault correctly (1344ms)
- ✓ Must deposit ERC20 token to specific vault correctly (1484ms)
- ✓ Must deposit token two times a row (2212ms)

totalAssets
- ✓ Must fail if firstVaultId bigger than lastVaultId (82ms)
- ✓ Mustn't change the variables in storage (505ms)
- ✓ Must return total assets of vaults (2098ms)
- ✓ Must return assets of specific vault (1962ms)

withdraw
- ✓ Must fail if firstVaultId is bigger than lastVaultId (239ms)
- ✓ Must pass if caller hasn't deposited yet (354ms)
- ✓ Mustn't change the variables in storage (2311ms)
- ✓ Must withdraw tokens from vaults correctly (3998ms)
- ✓ Must withdraw all tokens from vaults to another user correctly (2789ms)
- ✓ Must withdraw tokens from specific vault correctly (3430ms)

migrate
- ✓ Must return 0 if number of migrated tokens equals to zero or latestVaultId equals to zero (289ms)
- ✓ Must fail if old vault doesn't have enough tokens to migrate (317ms)
- ✓ Mustn't change the variables in storage (2707ms)
- ✓ Must migrate all tokens from old vault to new vault (4557ms)
- ✓ Must migrate all tokens from old vault to new vault with 2 and more users (5761ms)
- ✓ Must migrate tokens from old vault to new vault (4054ms)

```
    ✓ Must migrate tokens from old vault to new vault with 2 and more users (6209ms)

  38 passing (1m)
```

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

ZOKYO.