SMART CONTRACT AUDIT

ZOKYO.

Nov 15th, 2021 | v. 1.0

PASS

Zokyo Security Team has concluded that this smart contract passes security qualifications and bear no security or operational risk

TECHNICAL SUMMARY

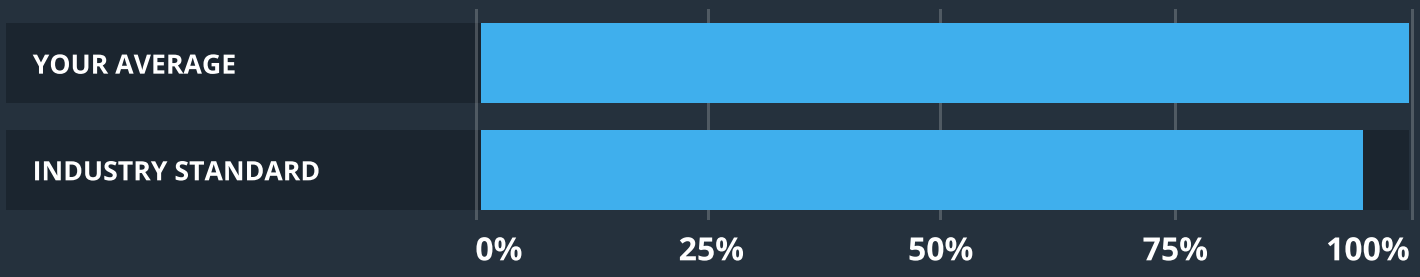
This document outlines the overall security of the DomFi smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the DomFi 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 DomFi 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 DomFi archive file.

Archive hash:
904b788c7e1fb49964d544c4b1b6c849be830e97

Last commit:
ddaa6279862cad612cca5d9af6587dcc9af0260e

Contracts:

- Vester;
- Vault.

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 DomFi 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.

SUMMARY

Zokyo security team has conducted a security audit for the given list of smart contracts. The contracts are in good condition and are well written. All the issues and vulnerabilities found are presented in the “Complete Analysis” section of this report.

There were no critical issues found during the audit, just 2 informational issues were spotted. All of them were resolved by the DomFi team.

Based on the results of the audit and the quality of the codebase, we can give a score of 100 and state that the audited smart contracts are 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

Useless import of IERC777

INFORMATIONAL | RESOLVED

Vester.sol imports IERC777 from openzeppelin which isn't used.

Recommendation:
Remove import:

```
@openzeppelin/contracts/token/ERC777/IERC777.sol;
```

Variables DomToken and Delegator can be immutable

INFORMATIONAL | RESOLVED

VesterFactory contract has two variables DomToken and Delegator which don't change after initialization in the constructor so they can be immutable.

Recommendation:
Set variables DomToken and Delegator as immutable.

	Vester	Vault
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 DomFi 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 DomFi 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	
VesterFactory.sol	100.00	100.00	100.00	100.00	
Vester.sol	100.00	98.00	100.00	96.00	4
Vault.sol	91.00	95.00	100.00	96.00	7
All files	**100.00**	**95.00**	**100.00**	**95.00**	

Test Results

```
Contract: Vester,VesterFactory
  deploying VesterFactory
    ✓ deploying VestingFactory (125ms)
    ✓ calling tokensReceived function with wrong token address (132ms)
    ✓ calling tokensReceived function (362ms)
  deploying Vester contract
    ✓ vesting with vestingBegin time early (113ms)
    ✓ vesting with vestingCliff less than vestingBegin time early (119ms)
```

```
    ✓ vesting with vestingEnd less than vestingCliff time early (123ms)
    ✓ deploying vesting contract (146ms)
  setRecipient function
    ✓ calling setRecipient with UnAuthorized recipient address (165ms)
    ✓ calling setRecipient with recipient address as zero (170ms)
    ✓ calling setRecipient (276ms)
  claim
    ✓ claiming before the vestingcliff time (106ms)
    ✓ claiming during cooldowwn (131ms)
    ✓ claiming the token (435ms)

Contract: Vault
  Deposit
    ✓ depositing after the deadline expired (124ms)
    ✓ depositing with longstaking address as zero (625ms)
    ✓ depositing with longstaking address as Non-zero (712ms)
    ✓ depositing with shortstaking address as zero (836ms)
    ✓ depositing with shortstaking address as Non-zero (790ms)
  withdraw and other functions
    ✓ calling arbitrage function (536ms)
    ✓ calling deposited for (725ms)
    ✓ calling rescue (345ms)
    ✓ withdrawing with withdraw mode as Reddem (917ms)
    ✓ withdrawing with withdraw mode as Settle (987ms)

23 passing (9s)
```

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

ZOKYO.