SMART CONTRACT AUDIT

ZOKYO.

July 2nd, 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

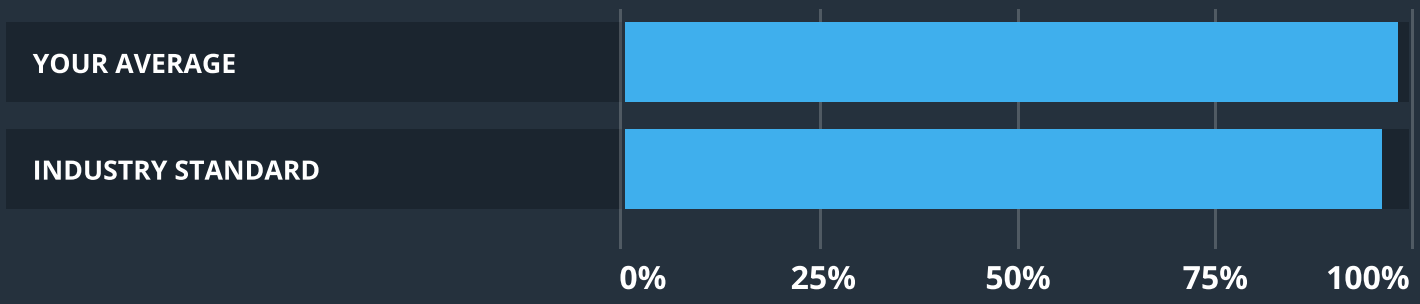
This document outlines the overall security of the Pontoon token smart contracts, evaluated by Zokyo's Blockchain Security team.

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

Contract Status

There were critical and medium issues found during the audit.

Testable Code



The testable code is 97%, 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 Pontoon 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 Pontoon Token repository.

Repository: https://github.com/pontoonfi/pontoon-token

Commit: 23d5c96d9c218c51f860c7d2dfc2ced612e1d43b
Last commit: d7671566e5c8a5a53d4a7b47c7ebde8bf30ac97e
Within the scope of this audit Zokyo auditors have reviewed the following contract(s):
PontoonToken.sol
PontoonTokenVesting.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 Pontoon 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 was one critical issue found related to incorrect Vesting initialization. Though, the functionality was verified after the conversation with the Pontoon team.
Also there were found several issues connected to incorrect or missing documentation, and gas optimization with SafeMath omitting. All together these mentioned findings may have an effect only in case of specific conditions performed by the contract owner or are connected to the code style, extra variables and minor optimizations.

All issues were successfully fixed or verified by the Pontoon 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 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

CRITICAL | RESOLVED

Incorrect initialization
PontoonTokenVesting.sol, line 113, 126, 151, 154, 157, constructor()
Incorrect parameters are passed to the _addTeam() for Ecosystem and for Community - vesting time and N of vestings are at the wrong places.
The same is applicable for all _addRound() calls.

Recommendation:
Correct the order of the parameters and verify the initialization.

Post-audit: marked as resolved after the conversation with Pontoon team and verification.

MEDIUM | RESOLVED

Incorrect documentation (or parameter)
PontoonTokenVesting.sol, line 126, 156, constructor()
The documentation comment states for 35.5% of supply for the Community rewards. Though, 35% is passed to _addTeam() function.
Potentially either incorrect comment (which is more likely) or incorrect parameter.
The same for the Strategic round - 15% stated in the comment, 4% passed to the method.

Recommendation: Verify the functionality and correct the comment or the documentation.

Post-audit: The functionality is verified, though comment is still incorrect.

LOW | RESOLVED

Gas optimization with SafeMath omitting

Solidity 0.8.4 has integrated support of safe arithmetical operations with overflow and underflow handling. So in such case SafeMath library can be omitted in order to perform gas savings.

Recommendation:
SafeMath library can be omitted.

LOW | RESOLVED

Method should emit event

PontoonToken.sol, setGovernance()

It is recommended to emit events for the governance (or any other access control value) change.

Recommendation:

Emit event for the governance change.

INFORMATIONAL | RESOLVED

Move “Magic numbers” to constants

PontoonTokenVesting.sol, line 116, 117, 120, 129, 130, 133, 142, 151, 154, 159, 161, 169, 321, 355, 449, 470

A lot of contract’s central constants are kept as “magic numbers”. For example:

- the accuracy (10000)
- the percent per team and per round, vesting time (constructor)
- the number of rounds (_massUpdateCliffEndTime())

Consider usage of named constants or variables to keep those values. This will increase the readability of the code and will help with further development.

Recommendation:

Use constants instead of “magic” numbers.

	PontoonToken	PontoonTokenVesting
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 team

As part of our work assisting Pontoon 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 Pontoon contract requirements for details about issuance amounts and how the system handles these.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts\	97.81	88.89	100	100	
PontoonToken.sol	100	83.33	100	100	
PontoonTokenVesting.sol	97.67	89.39	100	100	
All files	97.81	88.89	100	100	

```
Testing PontoonToken
  √ Should set governance (649ms)
  √ Only governance can call (473ms)
  √ Should revert token equal destination (396ms)
  √ Should recover tokens (863ms)

Testing TokenVesting
  Test constructor
    √ Should revert if startAfter equals zero (566ms)
    √ Should revert if addresses are zero (588ms)
  Test admin functions
    √ Should update start time (1214ms)
    √ Should not update start time if startAfter is 0 (375ms)
    √ Should not update start time if vesting has started (505ms)
    √ Should add and update single investor (2188ms)
    √ Should revert if investor is zero address (396ms)
    √ Should reverts if amount exceeds total supply (411ms)
    √ Should add several investors (1495ms)
    √ Should recover tokens (854ms)
  Test team claim
    √ Should revert if vesting has not started (316ms)
    √ Should not start vesting before team cliff end (612ms)
    √ Should claim initial tokens and revert if unlocked is zero (1338ms)
    √ Should revert if caller is not authorized (489ms)
    √ Should claim tokens and revert if all tokens are collected (1064ms)
  Test investor claim
    √ Should revert if caller is not investor (242ms)
    √ Should not claim tokens before round cliff end (1275ms)
    √ Should claim initial tokens and revert if unlocked is zero (1580ms)
    √ Should claim tokens and revert if all tokens are collected (997ms)
  Test view functions
    √ Should return investor claimable + initialClaim (281ms)
    √ Should return 0 for investor if vesting not started yet (162ms)
    √ Should return only initialClaim for investor (265ms)
    √ Should return claimable for investor without initialClaim (1026ms)
    √ Should return team claimable + initialClaim (338ms)
    √ Should return 0 for team if vesting not started yet (184ms)
    √ Should return only initialClaim for investor (306ms)
    √ Should return claimable for investor without initialClaim (921ms)
    √ Should return investors (189ms)


32 passing (46s)
```

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

ZOKYO.