SMART CONTRACT AUDIT

ZOKYO.

March 28th, 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

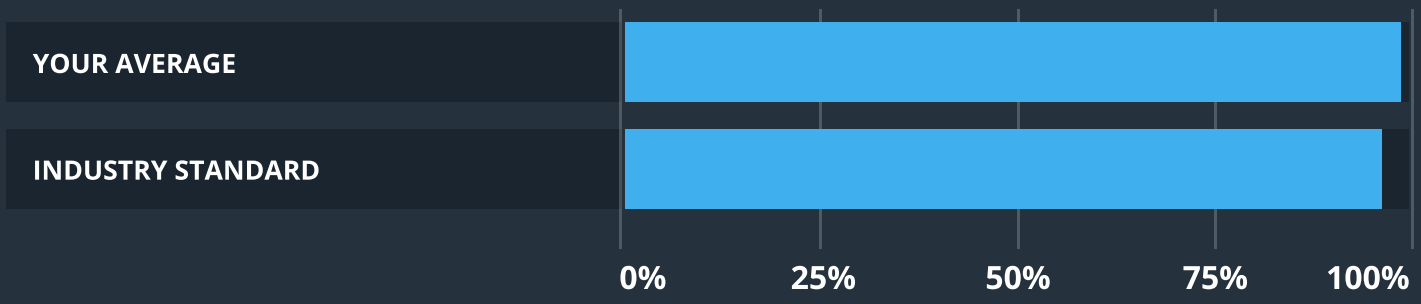
This document outlines the overall security of the Wonderman Token smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the Wonderman 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 98% 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 Wonderman Token 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 Wonderman Token repository.

Repository - https://github.com/GOATi/WondermanNation_Token_BSCLast
commit - 5264d6f1002c388323c30cdb1f5aa62c939cdad3

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

- WondermanToken.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;
- Distributes tokens in a manner that matches calculations;
- 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 Wonderman Token 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 issues found during the audit. All the mentioned findings may have an effect only in case of specific conditions performed by the contract owner.

Contracts are well written and structured. The findings during the audit have no impact on contract performance or security, so it is fully production-ready.

Despite the fact, the expected logic is managing all vestings by the owner, it should be careful with parameters to avoid mistakes during the vesting process.

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. Issues tagged “Verified” contain unclear or suspicious functionality that either needs explanation from the Customer’s side or it is an issue that the Customer disregards as an issue. 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

LOW | RESOLVED

The WondermanToken contract have a pragma version of ^0.8.0, that means it can be compiled with any compiler bigger then 0.8.0, however the openzeppelin contracts that it inherits also have ^0.8.0 expect the Address contract which have the pragma version of ^0.8.1, because the solidity compiler in the truffle.js is also declared as ^0.8.0, if the version used in compilation will be 0.8.1 the compilation will fail because of the Address contracts that gets inherited.

Recommendation:
Change the pragma version of the WondermanToken to ^0.8.1 or a fixed bigger version like 0.8.3 etc..

LOW | RESOLVED

There is a typo in the contract name, the contract file is called WondermaToken.sol and the contract is called WonermanToken, because there is only 1 letter difference we assume it's a typo, also it's best practice to name the contract the file with the same name.

Recommendation:
Change the contract name from WondermanToken to WondermanToken.

	WondermanToken
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 Wonderman Token team

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

Contract: WondermanToken

- ✓ The address 0x0 should not receive tokens (46ms)
- ✓ Allowance can be change (87ms)
- ✓ Balance of one user must be less or equal to the total supply (61ms)
- ✓ Balance of the crytic users must be less or equal to the total supply (61)
- ✓ Using transfer to send tokens to the address 0x0 will revert
- ✓ Using transferFrom to send tokens to the address 0x0 will revert
- ✓ Self transferring tokens tokens using transferFrom works as expected
- ✓ Transfering tokens to other address using transferFrom works as expected
- ✓ Self transferring tokens using transfer works as expected.
- ✓ Transfering tokens to other address using transfer works as expected
- ✓ Transfering more tokens then the balance will revert
- ✓ The address 0x0 should not receive tokens
- ✓ Allowance can be changed
- ✓ Balance of one user must be less or equal to the total supply
- ✓ Balance of the cryptic user must be less or equal to the total supply
- ✓ Using transfer to send tokens to the address 0x0 will revert
- ✓ Using transferFrom to send tokens to the address 0x0 will revert
- ✓ Self transferring tokens using transferFrom works as expected
- ✓ Transfering tokens to ther address using transferFrom works as expected.
- ✓ Self transferring tokens using transfer works as expected
- ✓ Transfering tokens to other address using transfer works as expected
- ✓ Transfering more tokens than the balance will revert

22 passing

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	% UNCOVERED LINES
WonermanToken	100.00	100.00	100.00	100.00	
All files	100.00	100.00	100.00	100.00	

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

ZOKYO.