wasder

SMART CONTRACT AUDIT

ZOKYO.

May 2, 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

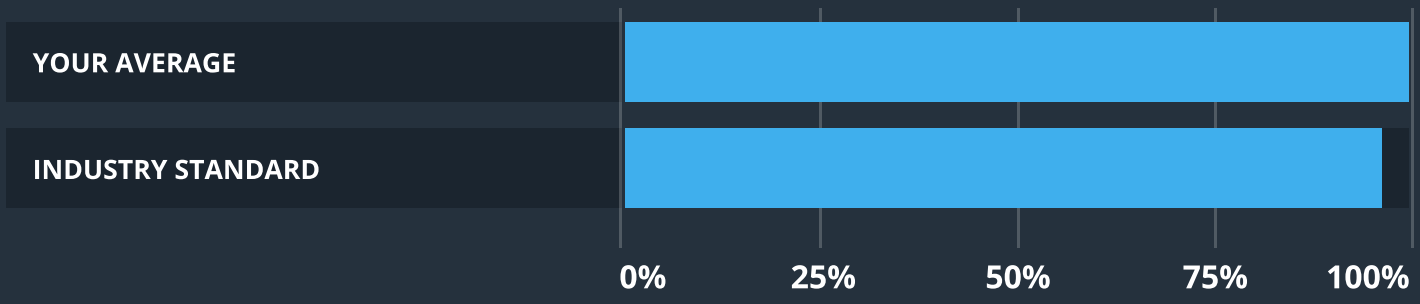
This document outlines the overall security of the Wasder smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the Wasder 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 Wasder 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 3

Executive Summary 4

Structure and Organization of Document 5

Complete Analysis 6

Code Coverage and Test Results for all files 9

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the Wasder repository.

Repository - https://github.com/WasderGG/wasder-token
Commit id - 111932cd8d62b3361e72f3e02351e4f5c1bb5b56

Last commit reviewed - ac657b7861ecb13a44db0840de76d7b955fd551d

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 Wasder 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 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.

The findings during the audit have a slight impact on contract code style and only may have impact during further development.

Nevertheless, all findings were successfully resolved by the Wasder 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 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

LOW | RESOLVED

Functions should be declared as external

Methods burn(uint256), snapshot() and transferAndCall(address,uint256,bytes) should be declared external.

Recommendation:
declare methods as external

LOW | RESOLVED

Add check for 0 amount

Line 29, Burn function has no checks for 0 amount. It can lead to accidental transactions, which will be confirmed and mined but will have no effect. Though they can be misleading, due to the errors of the caller, or errors connected to the frontend validation failure (these are some reasons for 0 amount passed to the transaction).

Recommendation:
add 0 amount check ('require()' statement) into the mint() function.

LOW | RESOLVED

Unused contract

Standard Migrations contract can be deleted.

Recommendation:
remove unused Migrations contract.

INFORMATIONAL | RESOLVED

Move number to constant

Line 21. To increase code readability and transparency we recommend to move the initial supply from the number to the constant.

Recommendation:
Move initial supply to the constant.

	WasderToken
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 Secured team

```
  Contract: Wasder token testset
    Token
      √ Should have correct attributes (115ms)
      √ should have correct supply (114ms)
      √ should add role to the owner (101ms)
    Roles and burn
      √ should grant burner role (185ms)
      √ should allow burn for a burner (251ms)
      √ should not allow burn for not a burner (719ms)
    Snapshot
      √ should create a snapshot (563ms)
    OnTransfer
      √ should call onTransfer function (293ms)


  8 passing (7s)

------------------|----------|----------|----------|----------|----------------|
File              |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
------------------|----------|----------|----------|----------|----------------|
 contracts\       |      100 |      100 |      100 |      100 |                |
  WasderToken.sol |      100 |      100 |      100 |      100 |                |
------------------|----------|----------|----------|----------|----------------|
All files         |      100 |      100 |      100 |      100 |                |
------------------|----------|----------|----------|----------|----------------|
```

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

ZOKYO.