SMART CONTRACT AUDIT

April 26th 2023 | v. 1.0

Security Audit Score

PASS

Zokyo Security 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 Ethlas smart contract evaluated by the Zokyo Security team.

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

Contract Status

There was no critical issue found during the audit. (See Complete Analysis)

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract but rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that can withstand the Ethereum network's fast-paced and rapidly changing environment, we recommend that the Ethlas team put in place a bug bounty program to encourage further active analysis of the smart contract.

Table of Contents

Auditing Strategy and Techniques Applied 3

Executive Summary 4

Structure and Organization of the Document 5

Complete Analysis 6

Test Results for all files written by Zokyo Security 8

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the Ethlas repository: https://github.com/ethlas-1/ELS-IDO

Last commit: 50c5ba4c039bc1c2ef7c1e16f6abcc27ae70ae75

Within the scope of this audit, the team of auditors reviewed the following contract:

- ELS20.sol

During the audit, Zokyo Security ensured that the contract:

- Implements and adheres to the existing standards appropriately and effectively;
- The documentation and code comments match the logic and behavior;
- Follows best practices, efficiently using resources without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the most resent vulnerabilities;
- Meets best practices in code readability, etc;
- Distributes tokens in a manner that matches calculations.

Zokyo Security has followed best practices and industry-standard techniques to verify the implementation of Ethlas smart contract. To do so, the code was reviewed line by line by our smart contract developers, who documented even minor issues as they were discovered. Part of this work includes writing a test suite using the Truffle. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

01	Due diligence in assessing the overall code quality of the codebase.
02	Cross-comparison with other, similar smart contracts by industry leaders.
03	Thorough manual review of the codebase line by line.

Executive Summary

Zokyo auditing team has run a deep investigation of the Ethlas smart contract. During the auditing process, there were no issues found. The contract is in excellent condition, well written and structured.
Based on the conducted audit, we give a score of 100 to the aforementioned contract. Zokyo auditing team can state that the contract is full production ready and bear no security or operational risk.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For the ease of navigation, the following sections are arranged from the most to the least critical ones. Issues are tagged as “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

DURING THE AUDITING PROCESS (MANUAL PART) NO ISSUES WERE IDENTIFIED.

	ELS20.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

TEST RESULTS FOR ALL FILES

Echidna Results

crytic_totalSupply_consistant_ERC20Properties: passed! 🎉
crytic_approve_overwrites: passed! 🎉
crytic_self_transferFrom_to_other_ERC20PropertiesTransferable: passed! 🎉
crytic_zero_always_empty_ERC20Properties: passed! 🎉
crytic_burn_allowance: passed! 🎉
crytic_self_transfer_ERC20PropertiesTransferable: passed! 🎉
crytic_self_transferFrom_ERC20PropertiesTransferable: passed! 🎉
crytic_revert_transferFrom_to_zero_ERC20PropertiesTransferable: passed! 🎉
crytic_revert_transfer_to_user_ERC20PropertiesTransferable: passed! 🎉
crytic_self_burn: passed! 🎉
crytic_revert_transfer_to_zero_ERC20PropertiesTransferable: passed! 🎉
crytic_transfer_to_other_ERC20PropertiesTransferable: passed! 🎉
crytic_less_than_total_ERC20Properties: passed! 🎉
Unique instructions: 3179
Unique codehashes: 1
Corpus size: 16
Seed: 6587217447334300947

ELSERC20

✓ The total supply is correctly initialized. (143ms)
✓ The initial supply is owner's balance. (135ms)
✓ Owner's balance is correctly initialized. (166ms
✓ User's balance is correctly initialized. (138ms)
✓ Attacker's balance is correctly initialized. (137ms)
✓ All the users have a positive balance. (234ms)
✓ The total supply is the user and owner balance. (207ms)

✓ The address 0x0 should not receive tokens. (161ms)
✓ Allowance can be changed. (297ms)
✓ Balance of one user must be less or equal to the total supply. (233ms)
✓ Balance of the crytic users must be less or equal to the total supply. (182ms)
✓ Using transfer to send tokens to the address 0x0 will revert. (1346ms)
✓ Using transferFrom to send tokens to the address 0x0 will revert. (431ms)
✓ Self transferring tokens using transferFrom works as expected. (243ms)
✓ Transferring tokens to other address using transferFrom works as expected. (269ms)
✓ Self transferring tokens using transfer works as expected. (241ms)
✓ Transferring tokens to other address using transfer works as expected. (306ms)
✓ Transferring more tokens than the balance will revert. (234ms)
✓ Burning from own account works (210ms)
✓ Burning from another account works (176ms)

20 passing (7s)

We are grateful for the opportunity to work with the Ethlas team.

The statements made in this document should not be interpreted as an investment or legal advice, nor should its authors be held accountable for the decisions made based on them.

Zokyo Security recommends the Ethlas team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo