OXSCANS

ZkLock

AI Generated at 06:27 PM, +0000

March 29, 2024

OVERVIEW

This audit has been perpared for 'ZkLock' to review the main aspects of the project to help investors make an informative decision during their research process

You will find a summarized review of the following key points:

- Contract's source code
- Owner wallets
- Tokenomics
- Team transparency and goals
- Website's age, code, security and UX
- Whitepaper and roadmap
- Social media and online presence

Table of Content

General Information

ZkLock

Name	ZkLock

General Information

Tokenomics

Contract Address	0x96884fcaac082db4b32601ada5b177fd6cbffa88

General Analysis

Audit Review Process

1. Testing the smart contracts against both common and uncommon vulnerabilities
2. Assessing the codebase to ensure compliance with current best practices and industry standards
3. Ensuring contract logic meets the specifications and intentions of the client
4. Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders
5. Thorough line-byline AI review of the entire codebase by industry

Token Transfer Stats

Transactions (Latest Mine Block)	Token holders	Compiler
1	352	v0.8.24

Smart Contract Stats

Functions	Events	Constructor
21	3	1

Detail Analysis

Threat Level

● High Issues on this level are critical to the smart contract's performace/functionality and should be fixed before moving to a live enviroment

● Medium Issues on this level are critical to the smart contract's performace/functionality and should be fixed before moving to a live enviroment

● Low Issues on this level are minor details and warning that can remain unfixed

● Informational Informational level is to offer suggestions for improvement of efficacy or secruity for fratures with risk free factor

Threat Level

● High	1 threats found
● Medium	0 threats found
● Low	0 threats found
● Informational	0 threats found

Detail Analysis

Vulnerability Check 🟢 20 Passed 🔴 1 Fail

- 🟢 Reentrancy
- 🟢 Flash Loans
- 🟢 Unused Code
- 🟢 Sybil Attack
- 🔴 Front Running
- 🟢 Oracle Issues
- 🟢 Logical Issues
- 🟢 Compiler Issues
- 🟢 Improper Events
- 🟢 Race Conditions
- 🟢 Unbounded Loops
- 🟢 Signature Issues
- 🟢 Ether/Token Theft
- 🟢 Integer Over/Underflow
- 🟢 Overall Contract Safety
- 🟢 Centralization of Control
- 🟢 Outdated Compiler Version
- 🟢 Arbitrary Jump/Storage Write
- 🟢 Improper Authorization Scheme
- 🟢 Delegate Call to Untrusted Contract
- 🟢 Dependence on Predictable Variables

Detail Analysis

Detail Analysis

🟢 20 Passed 🔴 1 Fail

CATEGORY	STATUS	NOTES
Reentrancy	🟢	The contract uses modifiers to prevent reentrancy where necessary.
Flash Loans	🟢	The contract does not interact with flash loan functions, making it unaffected by flash loan attacks.
Unused Code	🟢	The contract's code does not contain redundant or unused code, ensuring efficiency and reducing attack surface.
Sybil Attack	🟢	The nature of the contract does not make it susceptible to Sybil attacks.
Front Running	🔴	The contract could be susceptible to front-running as it involves liquidity functions and token transfers without anti-front-running measures.

Detail Analysis

Detail Analysis 🟢 20 Passed 🔴 1 Fail

CATEGORY	STATUS	NOTES
Oracle Issues	🟢	The contract does not interact with oracles, thus not exposing it to oracle-related risks.
Logical Issues	🟢	No apparent logical issues or inconsistencies in the contract logic.
Compiler Issues	🟢	Compiled with a recent Solidity version (0.8.24) without known compiler bugs and with optimizations enabled.
Improper Events	🟢	All critical functions emit events correctly, providing transparency and traceability.
Race Conditions	🟢	No functions or patterns were found that could lead to race conditions.
Unbounded Loops	🟢	All loops in the contract have bounded conditions, avoiding risks of gas limit issues or denial-of-service.

Detail Analysis

Detail Analysis

🟢 20 Passed 🔴 1 Fail

CATEGORY	STATUS	NOTES
Signature Issues	🟢	The contract does not rely on external signatures, hence is not exposed to signature-related risks.
Ether/Token Theft	🟢	No functions are present that directly transfer Ether or tokens to arbitrary addresses in an unauthorized manner.
Integer Over/Underflow	🟢	The contract uses SafeMath for all arithmetic operations, ensuring no overflows or underflows occur.
Overall Contract Safety	🟢	The contract follows general best practices and does not exhibit critical vulnerabilities. However, improvements could be made to mitigate front-running risks.
Centralization of Control	🟢	No risk of centralization since the contract owner is a dead address, eliminating the concern of a single point of control.
Outdated Compiler Version	🟢	The contract uses a recent Solidity compiler version (0.8.24), which is not outdated.

Detail Analysis

Detail Analysis

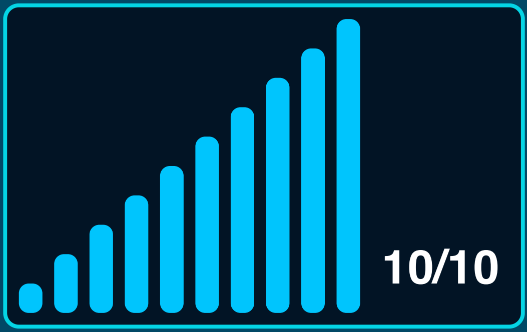
🟢 20 Passed 🔴 1 Fail

CATEGORY	STATUS	NOTES
Arbitrary Jump/Storage Write	🟢	The contract does not exhibit arbitrary jumps or storage writes, as it adheres to standard Solidity development patterns.
Improper Authorization Scheme	🟢	Given the contract owner is a dead address, there's no risk of improper authorization schemes revolving around owner privileges.
Delegate Call to Untrusted Contract	🟢	There is no use of delegatecall to an untrusted contract, mitigating risks associated with delegate calls.
Dependence on Predictable Variables	🟢	The contract does not rely on variables like block.timestamp or block.number in a way that affects core functionalities or security.

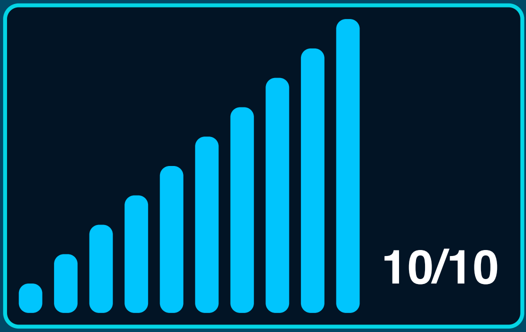
Market Analysis

Score

Total Audit Score



Security Score



Legal Disclaimer

0xscans operates as an automated system for smart contract due diligence, acknowledging the possibility of bugs or vulnerabilities impacting token values. We do not hold specific obligations regarding your trading outcomes or the utilization of audit content. Users release 0xscans from any liability associated with content obtained through the tool.

AI generated by 0xscans AI technology

Chat with us

Telegram

For more information. Visit below:

Twitter

Github