SMART CONTRACT AUDIT

zokyo

February 13th 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

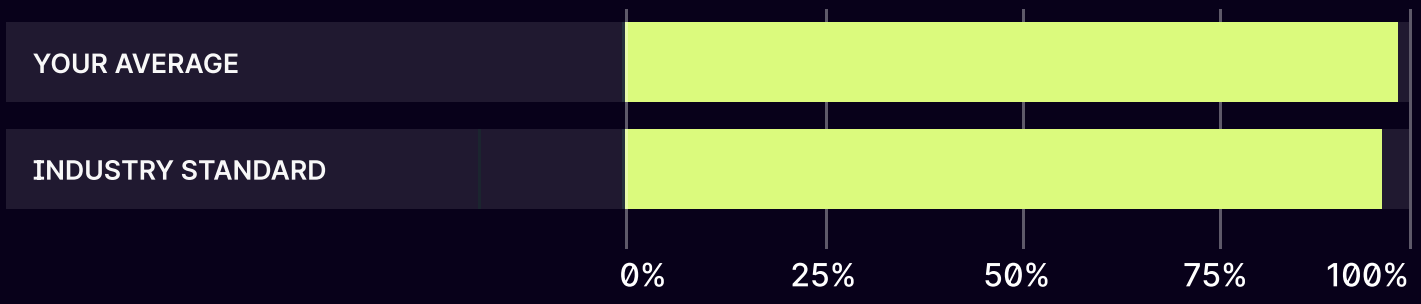
This document outlines the overall security of the AngelBlock smart contracts evaluated by the Zokyo Security team.

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

Contract Status

There were 0 critical issues found during the audit. (See Complete Analysis)

Testable Code



100% of the code is testable, 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 contracts 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 AngelBlock team put in place a bug bounty program to encourage further active analysis of the smart contracts.

Table of Contents

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the AngelBlock repository: https://github.com/angel-block/angelblock-contracts

Last commit: 2d68dde39e1fae1516d160b96b336293c65305c5

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

- allocation/VestedAlloc.sol
- interfaces/IVestedAlloc.sol
- utils/MultiBatch.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;
- Distributes tokens in a manner that matches calculations;
- 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.

Zokyo Security has followed best practices and industry-standard techniques to verify the implementation of AngelBlock smart contracts. 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 Hardhat testing framework. 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	Testing contracts logic against common and uncommon attack vectors.
04	Thorough manual review of the codebase line by line.

Executive Summary

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

STRUCTURE AND ORGANIZATION OF THE 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. The issues that are tagged as “Verified” contain unclear or suspicious functionality that either needs explanation from the Client or remains disregarded by the Client. 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 (both manual part and testing part) no issues were identified.

	allocation/VestedAlloc.sol interfaces/IVestedAlloc.sol utils/MultiBatch.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

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Security

As a part of our work assisting AngelBlock in verifying the correctness of their contracts code, our team was responsible for writing integration tests using the Hardhat testing framework.

The tests were based on the functionality of the code, as well as a review of the AngelBlock contracts requirements for details about issuance amounts and how the system handles these.

VestedAlloc

#constructor
- ✓ fails if total supply is incorrect
- ✓ fails if release type is invalid
- ✓ fails if release type is invalid

#addRecipients
- ✓ check event after adding recipients
- ✓ fails if called after configuration of contract
- ✓ fails if caller is not the owner
- ✓ fails if recipient sum is invalid

#configure
- ✓ check state and event after configuration
- ✓ fails without transfer erc20 token
- ✓ fails if balance does not equal totalSupply
- ✓ fails if caller is not the owner
- ✓ fails if called after configuration
- ✓ fails if token address is incorrect

#setDexPaths
- ✓ check state and event after setting dex paths
- ✓ fails if called before configuration of contract
- ✓ fails if caller is not the owner

#forward
- ✓ check event after forwarding
- ✓ fails if nothing to forward
- ✓ fails if cannot forward claimable funds
- ✓ fails if called before configuration
- ✓ fails if caller is not the owner

#claim
✓ check event after claiming
✓ fails if called bbefore configuration
✓ fails if nothing to claim
#getAvailable
✓ getAvailable
✓ fails without configure
#unlockReserve
✓ unlockReserve
✓ fails if price not met
✓ fails if called before configuration
✓ fails if release type is invalid
✓ fails if dex paths not set
#safeguard
✓ check event after safeguard
✓ fails if called before configuration
✓ fails if nothing to safeguard

MultiBatch

#multiread
✓ should execute
✓ should revert
#multicall
✓ should execute
✓ should revert

38 passing

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	% UNCOVERED LINES
allocation/VestedAlloc.sol	100	100	100	100	
interfaces/IVestedAlloc.sol	100	100	100	100	
utils/MultiBatch.sol	100	100	100	100	
All Files	100	100	100	100	

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

zokyo