Audit Whale

Smart contract code review and security analysis report

This document may contain confidential information about IT systems and the intellectual property of the Customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed — upon a decision of the Customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for Algebra Finance.
Approved by	Jameson | COO AuditWhale
Type	Decentralized exchange
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/cryptoalgebra/Algebra
Commit	7ceaaf3315213a1abc22d9d010cb5900e841d143
Technical Documentation	YES
JS tests	YES
Website	algebra.finance
Timeline	22 NOVEMBER 2021 – 13 DECEMBER 2021
Changelog	07 DECEMBER 2021 – INITIAL AUDIT 13 DECEMBER 2021 – SECOND REVIEW 15 DECEMBER 2021 – THIRD REVIEW

Table of contents

Introduction

AuditWhale (Consultant) was contracted by Client (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between November 22nd, 2021 – December 7th, 2021.

Second review conducted on December 13th, 2021. Third

review conducted on December 15th, 2021.

Scope

The scope of the project is smart contracts in the repository:

Repository:
https://github.com/cryptoalgebra/Algebra
Commit:
7ceaaf3315213a1abc22d9d010cb5900e841d143
Technical Documentation: Yes
(https://drive.google.com/file/d/1EYtf4YPniPtk7B8f2XsozlGZ9BaAdyMV/view) JS tests:
Yes

- https://github.com/cryptoalgebra/Algebra/tree/v0.1/src/core/test
- https://github.com/cryptoalgebra/Algebra/tree/v0.1/src/tokenomics/tes t

Contracts:
core/contracts/libraries\AdaptiveFee.sol
core/contracts/libraries\DataStorage.sol
core/contracts/libraries\PriceMovementMath.sol
core/contracts/libraries\TickManager.sol
core/contracts/libraries\TickTable.sol
core/contracts/libraries\TokenDeltaMath.sol
core/contracts/AlgebraFactory.sol
core/contracts/AlgebraPool.sol
core/contracts/AlgebraPoolDeployer.sol
core/contracts/DataStorageOperator.sol
tokenomics/contracts/AlgebraFarming.sol
tokenomics/contracts/AlgebraVirtualPool.sol
tokenomics/contracts/VirtualPoolDeployer.sol
tokenomics/contracts/libraries/RewardMath.sol
tokenomics/contracts/AlgebraTokenStaking.sol
tokenomics/contracts/libraries/FreezableToken.sol

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts are well-secured.

Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found 1 medium and 2 low severity issues.

After the second review security engineers received a commit with updated bootstrap scripts to compile, build and execute tests. Therefore medium issue was closed and there are still 2 low severity issues left.

After the third review security engineers found that all issues were addressed.

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution

Audit overview

Critical

No critical issues were found.

High

No high severity issues were found.

Medium

Some tests are failed

143 out of 851 tests of the “core” module, as well as 13 out of 26 tests of the “tokenomics” module, are failed. Those tests are cover the checks for gas fees, contract size for the core, and the problem to create a fixture loader for the tokenomics tests.

Recommendation: Please make sure your tests are up to date and successful or update contracts accordingly.

Status: Fixed

Low

1. Missing events access control

Updating the factory address in the AlgebraPoolDeployer should emit an event for better trackability off-chain.

Contracts: AlgebraPoolDeployer.sol, AlgebraFarming.sol

Function: AlgebraPoolDeployer.setFactory, AlgebraFarming.setIncentiveMaker

Recommendation: Please make sure that changes of important addresses in contracts emit events.

Status: Fixed

2. State variables that could be declared immutable

Constant state variables that are initialized in the constructor should be declared immutable to save gas.

Contracts: AlgebraVirtualPool.sol

Variables: desiredEndTimestamp, desiredStartTimestamp

Recommendation: Add the immutable attributes to state variables that never change and are initialized in the constructor.

Status: Fixed

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

As a result of the audit, security engineers found 1 medium and 2 low severity issues.

After the second review security engineers received a commit with updated bootstrap scripts to compile, build and execute tests. Therefore medium issue was closed and there are still 2 low severity issues left.

After the third review security engineers found that all issues were addressed.

Disclaimers

AuditWhale Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.