Audit

Security Assessment

Date: **02. Oktober 2024**

Client: **FalXDex**

Index

FIRST LINE OF DEFENSE

Introduction

line1.io is a distinguished brand under the officially registered entity, FutureVisions Deutschland, based in Germany. Our expertise lies in Blockchain Security, providing comprehensive services including Smart Contract Audits and KYC verification for project teams. At line1.io, we rigorously evaluate smart contracts for security vulnerabilities, verify the alignment between the codebase and related whitepapers/documentation, and deliver detailed recommendations for improvement.

Disclaimer

line1.io reports are neither an endorsement nor a disapproval of any specific project or team. These reports do not reflect the economic viability or value of any product or asset created by any team. line1.io does not test or audit the integration with external contracts or services.

line1.io audits do not offer any warranty or guarantee regarding the absolute absence of bugs in the analyzed technology, nor do they provide any information about the proprietors of the technology. These audits should not be used as a basis for making investment decisions or for involvement in any particular project. The reports do not constitute investment advice and should not be relied upon as such.

line1.io reports represent a comprehensive auditing process designed to assist our clients in enhancing the quality of their code while mitigating the significant risks associated with cryptographic tokens and blockchain technology. Given the high level of inherent risk in blockchain technology and cryptographic assets, line1.io emphasizes that each company and individual must conduct their own due diligence and maintain continuous security. line1.io does not claim any guarantee of the security or functionality of the technologies we analyze.

Project Overview

Overview

Project Name	**FalXDex**
Website	https://falxdex.com/
About the project	FalXDex the blazing fast Dex on the Solana Blockchain
Chain	Solana
Language	Rust

Social Media Information

Telegram	https://falx.pro/FalXDexTelegram
X / Twitter	https://x.com/FalxDex
Discord	https://discord.gg/falxdexofficial
YouTube	https://www.youtube.com/@FalXDex

Audit Summary

Version	Date	Changelog
V1.0	03th October 2024	Initial Report

Note: This audit report provides a detailed security analysis of the solidity codebase used in the project, particularly focusing on potential vulnerabilities to external malicious interference with the program's functions. This analysis did not cover functional testing (or unit testing) of the program's logic. Therefore, we cannot assure complete logical correctness of the code, as we did not perform functional tests on it. This includes the internal calculations in the algorithms implemented in the codebase.

Scope of Work

Line1 will conduct a thorough and comprehensive audit of the provided Solidity Smart Contracts, focusing on identifying vulnerabilities, ensuring code integrity, and verifying adherence to security best practices. This audit is designed to provide assurance that the smart contract is secure, efficient, and performs as intended.

The auditing process follows a structured and methodical approach:

- Pre-Audit ReviewWe begin by reviewing the specifications, source code, and any supplementary documentation provided to Line1. This ensures a clear understanding of the smart contract's scope, functionality, and underlying design, setting the foundation for an in-depth analysis.
- Manual Code ReviewOur team manually inspects the source code, examining each line in detail to identify potential vulnerabilities, such as logic flaws, access control issues, and susceptibility to known exploits (e.g., reentrancy, overflows). The goal is to uncover weaknesses that could be exploited in real-world scenarios.
- Specification ComparisonWe verify that the code aligns with the provided specifications and functional requirements. This comparison ensures that the smart contract behaves as expected and that the implementation matches the intended design.
- Test Coverage AnalysisWe evaluate the test coverage to ensure the testing framework thoroughly exercises the contract. This involves analyzing how much of the code is covered by existing tests and identifying gaps where additional test cases may be necessary to ensure robust functionality.
- Symbolic Execution and Automated ToolsWe use advanced automated tools, including symbolic execution, to simulate a wide range of inputs and execution paths, ensuring that every part of the contract behaves securely under different conditions. This helps identify edge cases and potential failure points that might not be evident through manual inspection alone.
- Best Practices ReviewLeveraging industry-standard guidelines, academic research, and established best practices, we evaluate the smart contract's architecture to ensure it is efficient, maintainable, and secure. Our review focuses on improving clarity, maintainability, and resilience against future threats.
- Actionable RecommendationsUpon completion, Line1 will provide a detailed audit report, including specific, itemized recommendations. These actionable insights will help secure the smart contract and guide any necessary improvements or optimizations.

Zip-File	Hash
FalXDex.zip	d6672d55147936fd98e06a15f81f792961c889cd

Imported Packages

Used code from other Frameworks.

Dependency / Import Path	Version
anchor-lang	0.29.0
anchor-spl	0.29.0
spl-token	4
spl-transfer-hook-interface	0.5.1
solana-program	1.17
thiserror	1.0
uint	0.9.1
borsh	0.9.1
solana-security-txt	1.1.1
proptest	1.0
serde	1.0.117
serde_json	1.0.59

Note for Investors: We have only audited the dependencies listed in the above scope. We have not reviewed any additional dependencies related to the project that are not included in our audit scope, and we cannot comment on their security. We are not responsible for any security issues arising from these unreviewed dependencies.

Audit Information

Vulnerability & Risk Management Level

Risk represents the probability that a certain source threat will exploit vulnerability and the impact of that event on the organization or system. The risk Level is computed based on CVSS version 3.0.

Level	Value	Description	Required Action
CRITICAL	9-10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
HIGH	7-8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
MEDIUM	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
LOW	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
INFORMATIONAL	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the audit of the Cosmos SDK Layer 1 blockchain repository, meticulous attention was dedicated to identifying security vulnerabilities, ensuring code quality, and verifying adherence to both specifications and best practices. Our audit was conducted by a seasoned team of penetration testers and blockchain developers with extensive experience in the Cosmos ecosystem and smart contract security.

Our team conducted a thorough line-by-line review of the code, ensuring that no detail was overlooked. Every identified issue was meticulously documented to provide a comprehensive overview of potential vulnerabilities and areas for improvement. Each file within the repository was subjected to a thorough manual examination, maintaining a high level of scrutiny and precision.

While automated tools were employed, their usage was strategically limited to augment the efficiency and effectiveness of the manual review process rather than replace it. This combination of rigorous manual inspection and strategic use of automated tools allows us to deliver an in-depth and reliable assessment, ensuring the highest standards of security and code quality for our clients.

Methodology

The auditing process follows a structured series of steps to ensure thorough examination and assessment:

- Specification Review: Our team meticulously reviewed all relevant documentation, specifications, and guidelines provided initially. This step ensures a deep understanding of the blockchain's architecture, scope, and intended functionalities.

- Manual Code Inspection: The source code was examined line by line in an intensive review aimed at uncovering potential security vulnerabilities that could be exploited maliciously. This careful inspection ensures no detail is overlooked.

- Specification Conformance: The code was rigorously compared against the provided specifications to confirm its fidelity in performing as described. This ensures that the implementation accurately reflects the intended design and requirements.

During this audit, certain aspects, such as unused constants, functions, exported functions, global variables, and parameters, were excluded from the scope of our review. While these elements were not directly evaluated, we suggest revisiting these areas in future audits to ensure that they do not introduce security concerns or affect the overall quality of the codebase.

Overall Security

Checks	Privileges
Upgradeability	➢ The solana program is not upgradable as the upgrade authority got revoked

Audit Results

Critical Issues - [0]

No Critical Issues

High Issues - [0]

No High Issues

Medium Issues - [0]

No Medium Issues

Low Issues - [0]

No Low Issues

Informational - [3]

#I-1 Delete unused code		
Severity	**Location / Line**	**Status**
Informational	All	Open
Description	On multiple files and sections of the code, unused code is commented out instead of being deleted. For good readable code, it is recommended to delete that lines instead.	

#I-2 Using rust naming convention		
Severity	**Location / Line**	**Status**
Informational	All	Open
Description	On multiple files and sections of the code, functions names are not named after the rust naming convention.	

#I-3 Delete unused imports		
Severity	**Location / Line**	**Status**
Informational	All	Open

Description	On multiple files, there are unused imports, it is recommended to delete these unused imports.