QuillAudits

Audit Report
July, 2024

For

Table of Content

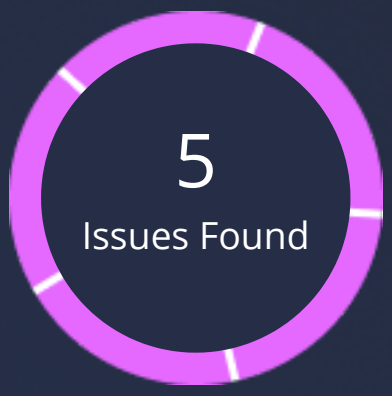
Executive Summary

Project Name	Degen Labs LTD
Overview	DegenFund is a fair launch platform similar with extra features. Users can instantly create and trade tokens with virtual liquidity using a bonding curve, stopping trading and migrating liquidity to Raydium once the desired amount is reached. Key features include SPL and SPL22 pool creation, a dev locker, antibot protection, customizable quote configurations, a fee collector, and the option for unsellable pools.
Timeline	2nd July 2024 - 11th July 2024
Updated Code Received	5th July 2024
Second Review	6th July 2024 - 11th July 2024
Method	Manual Review, Functional Testing, Automated Testing, etc. All the raised flags were manually reviewed and re-tested to identify any false positives.
Audit Scope	The scope of this audit was to analyze the Degen Labs LTD Contract for quality, security, and correctness.
Source Code	https://github.com/ArcSys-Labs/degenfund-program-v2
Contracts In-Scope	degen_fund
Branch	main
Contracts out of Scope	In-scope contracts have been audited by QuillAudits. However, these contracts inherit functionality from out-of-scope Smart contracts that were not audited. Vulnerabilities in unaudited contracts could impact in-scope Smart Contracts functionality. QuillAudits is not responsible for such vulnerabilities. Below are Out of Scope Contracts: metaplex metadata program, Solana native token2022 program,instructions/seed_spl22.rs, instructions/seed_spl.rs

Executive Summary

Fixed In	172f2aef8c4db6215eba4fdc0d74018c4b721d8e
Note	The fixed codebase repo has been moved to another github account on 16th July 2024. Here is the new github link: https://github.com/xastrobunnyy/degenfund-program-v2

Number of Security Issues per Severity



High
Medium
Low
Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	4
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	0	1

Checked Vulnerabilities

We have scanned the solana program for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Signer authorization
- Account data matching
- Sysvar address checking
- Owner checks
- Type cosplay
- Initialization
- Arbitrary cpi
- Duplicate mutable accounts
- Bump seed canonicalization
- PDA Sharing
- Incorrect closing accounts
- Missing rent exemption checks
- Arithmetic overflows/underflows
- Numerical precision errors
- Solana account confusions
- Casting truncation
- Insufficient SPL token account verification
- Signed invocation of unverified programs

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- Implementation of Token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Hardhat, Foundry.

Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Informational Issues

1. Centralization Risk Due to Emergency Withdrawal Function Allowing Admin to Withdraw All Funds

Path

programs/degen-fund/src/instructions/admin/emergency_withdraw.rs

Function

emergency_withdraw

Description

It was observed that the DegenFund program includes an emergency withdrawal function that allows the admin to withdraw all funds from user pools. This introduces a significant centralization risk, as it grants the admin unilateral control over the funds deposited by users. In case of misuse or compromise of the admin account, users' funds could be at risk.

Recommendation

To mitigate the centralization risk and enhance transparency, the following steps are recommended:

1. **Communicate Clearly**: Ensure that users are well-informed about the existence of the emergency withdrawal function and the conditions under which it may be used.
2. **Implement Multi-Signature Approval**: Introduce a multi-signature approval mechanism for invoking the emergency withdrawal function. Require multiple trusted parties to sign off on the withdrawal to reduce the risk of misuse.

Status

Acknowledged

2. Unused Treasury Fee Concept

Path

programs/degen-fund/src/states/fee_collector.rs

Function

get_treasury_fee_bps

Description

It was observed that the get_treasury_fee_bps function and the rev_share_bps parameter are defined in the DegenFund program but are not utilized in any part of the code. This results in dead code that can cause confusion and maintainability issues. Furthermore, unused functions and parameters might indicate incomplete implementation or missed functionalities, which could affect the overall performance and integrity of the program

Recommendation

To address this issue, it is recommended to remove unused variables.

Status

Acknowledged

3. Hardcoded Admin Address Limits Flexibility in Non-Upgradable Deployments

Path

programs/degen-fund/src/lib.rs

Function

admin module

Description

The DegenFund program contains a hardcoded admin address, which presents a potential issue if the program is deployed as final (non-upgradable). In such a scenario, the admin address cannot be changed. This inflexibility could be problematic if administrative control needs to be transferred or if the original admin address becomes compromised.

Recommendation

To enhance the flexibility and security of the program, it is recommended to replace the hardcoded admin address with a configurable admin address that can be updated through a secure procedure.

Status

Resolved

QuillAudits Team's Comment

The admin key was included in the configuration account and is not hardcoded anymore.

4. Unused Vault in FeeCollector

Path

programs/degen-fund/src/states/fee_collector.rs

Function

FeeCollector

Description

It was observed that the vault in the FeeCollector structure is defined but not utilized within the DegenFund program. This results in redundant code and potential confusion about the intended functionality of the FeeCollector. The presence of unused variables can also make the codebase harder to maintain and understand.

Recommendation

To improve code clarity and maintainability, it is recommended to either remove the unused vault from the FeeCollector structure if it is not needed or integrate it properly into the program's logic if it was intended for future use. Ensuring that every defined element in the code serves a clear purpose will help maintain a clean and efficient codebase.

Status

Acknowledged

5. Devbox Lock Vault Token Account Not Closed After Claim

Path

programs/degen-fund/src/instructions/dev_locker_claim.rs

Function

dev_locker_claim

Description

It was observed that the Devbox lock vault token account is not closed after tokens are claimed. As a result, the associated lamports remain locked in this account, preventing users from reclaiming them. This can lead to unnecessary resource usage and inefficiencies in the Solana network, as these accounts continue to consume storage and lamports.

Recommendation

To ensure efficient resource usage and allow users to reclaim their lamports, it is recommended to implement functionality that automatically closes the Devbox lock vault token account after tokens have been claimed. This will release the locked lamports and free up storage on the network, contributing to a more efficient and user-friendly platform.

Status

Acknowledged

Functional Tests Cases

1. Dev Locker

Test Case 1.1: Successful Dev Locker Initialization

- **Expected Results**: Dev Locker is initialized successfully with the specified lock duration.

Test Case 1.2: Dev Locker Claim Post Unlock Timestamp

- **Expected Results**: Tokens are successfully claimed from the dev locker.

2. Antibot Protection

Test Case 2.1: Successful Antibot Activation

- **Expected Results**: Antibot protection is enabled for the pool.

Test Case 2.2: Transaction Blocking by Antibot

- **Expected Results**: Transaction is blocked, requiring antibot server signature.

3. Fee Collector

Test Case 3.1: Fee Collector Initialization

- **Expected Results**: Fee collector is initialized successfully.

Test Case 3.2: Fee Collector Claim

- **Expected Results**: Fees are successfully claimed and transferred to the treasury.

4. Unsellable Pools

Test Case 4.1: Enable Unsellable Pool

- **Expected Results**: Unsellable pool feature is enabled, preventing token sales on the curve.

Test Case 4.2: Attempted Token Sale in Unsellable Pool

- **Expected Results**: Token sale is blocked, with an error indicating the pool is unsellable.

5. General Security

Test Case 5.1: Emergency Withdrawal

- **Expected Results**: Emergency withdrawal is executed successfully, allowing immediate fund withdrawal.

Test Case 5.2: Initialize User ATA

- **Expected Results:** User's Associated Token Account (ATA) is initialized successfully.

Test Case 5.3: Quote Configuration Initialization

- **Expected Results**: Quote configuration is initialized successfully, allowing pairing of quote tokens.

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the Degen Labs LTD V2 codebase. We performed our audit according to the procedure described above.

Some issues of Medium, Low and informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in Degen Labs LTD V2 smart contracts. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats.
QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of Degen Labs LTD V2 smart contracts. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of the Degen Labs LTD V2 to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With six years of expertise, we've secured over 1000 projects globally, averting over $30 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.

1000+
Audits Completed

$30B
Secured

1M+
Lines of Code Audited

Follow Our Journey

Audit Report
July, 2024

For

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com