Security Assessment

For Beehive
24 Nov 2024

Ascendant

Ascendant
@ascendantfi
www.ascendant.finance

Ascendant

Table of Contents

3	Disclaimer
4	Executive Summary
5	Overview
7	Findings Summary & Legend
8	Manual Review • Audit Findings • Omitted Results
15	Automated Review • Unified Model Language
17	Conclusion

DISCLAIMER

This independent audit has been conducted to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The programs audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the programs are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by the auditor.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and the auditor is under no covenant to the completeness, accuracy or solidity of the programs audited. In no event will the auditor or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the programs provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the programs are functioning as intended, specifically that the functions therein contained within said programs have the desired intended effects, functionalities and outcomes of the Project team.

Auditor retains full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Auditor is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. The auditor may, at its discretion, claim bug bounties from third-parties while doing so.

Executive Summary

The program reviewed in this audit were found to be **Well Secured**, meaning they contain no critical severity issues that would render them too unsafe to launch. However, it is recommended that the remaining issues found within this report be resolved or mitigated to ensure best user experience.

Security Level	
Well Secured	✓
Secured	
Poorly Secured	
Insecure	

We performed an independent technical audit to identify program uncertainties. This shall protect the code from illegitimate authorization attempts or external & internal threats of any type. This also ensures end-to-end proofing of the program from frauds. The audit was performed semi-manually. We analyzed the program code line-by-line and used an automation tool to report any suspicious code.

The following tools were used:

- Anchor
- Mocha
- Slither for Rust

Overview

This report has been prepared for Beehive for the Solana Network. This audit provides a user-centered examination of the program to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

Summary

Project Name	Beehive
Platform	Solana
Language	Rust

Program Info

Name	Location
lib.rs	Not Published
ProgramID	4JnqmqsP4Zbk86wz1seR8QjP28jZydwCawvmZfMC9zMc
Owner	6sGFmkLvs9FiyUszh7T6yT4soonUzB12tcwBLhtrEp7B

Program Summary

The Beehive program implements a reward pool management system with basic functionalities like pausing/unpausing, depositing, and distributing rewards. While the program passes the basic tests, there are areas for improvement in flexibility, security, and maintainability.

Roles

The program has an owner role. The owner has the authority to:

- Pause the reward distribution
- Unpause the reward distribution
- Distribute rewards

Users have the authority to:

- Deposit to the reward pool

Users DO NOT have the authority to:

- Pause the program
- Unpause the program
- Distribute rewards
- Access funds in the reward pool

Findings Summary

Severity	Found
High	0
Medium	1
Low	4
Informational	6
Total	11

Classification of Issues

High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the program and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax or style best practices, Generally pose a negligible level of risk, if any.

Manual Review

Severity	Medium
Program	lib.rs
Description	Cannot Transfer Ownership
Code Snippet	N/A
Recommendation	Owner address is currently hardcoded and ownership cannot be transferred. If the address is compromised, this could lead to a loss of control or funds.
Status	

Severity	**Low**
Program	lib.rs
Description	Empty RewardPoolAccount: ln148
Code Snippet	`#[account]` `pub struct RewardPoolAccount {}`
Recommendation	There is a lack of useful metadata for the reward pool.By adding meaningful fields to RewardPoolAccount, you make your program more transparent, easier to debug, and adaptable to future needs.
Status	

Severity	**Low**
Program	lib.rs
Description	No Rent Exemption Enforcement: ln103

Code Snippet	`#[derive(Accounts)]`
	`pub struct Initialize<'info> {`
	`#[account(init, seeds = [b"state"], bump, payer = user, space = 8 + 8)]`
	`pub state: Account<'info, State>,`
	`#[account(init, seeds = [b"reward_pool"], bump, payer = user, space = 8 + 8)]`
	`pub reward_pool: Account<'info, RewardPoolAccount>,`
	`#[account(mut)]`
	`pub user: Signer<'info>,`
	`pub system_program: Program<'info, System>,`
	`}`

Recommendation	There is a risk of account data loss due to insufficient lamports. Consider updating the Initialize Context to enforce a rent exemption.
Status	

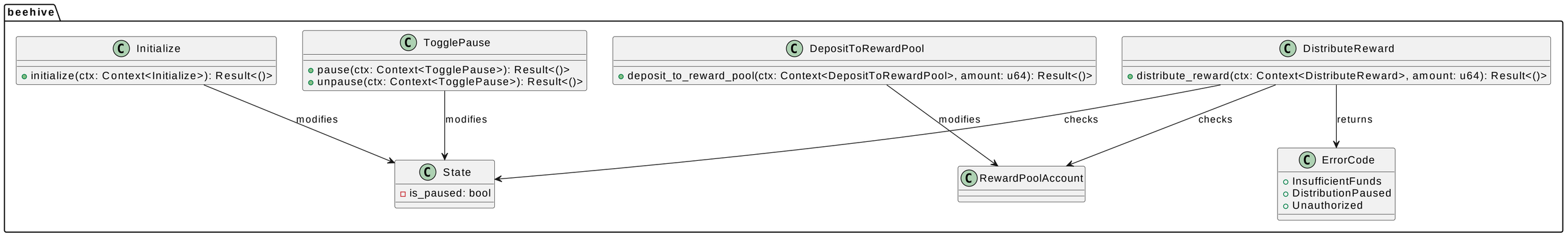
Severity	Informational
Program	lib.rs
Description	Hardcoded owner address: ln 6
Code Snippet	`// Will be a owner address` `const OWNER: &str = "6sGFmkLvs9FiyUszh7T6yT4soonUzB12tcwBLhtrEp7B";`
Recommendation	Hardcoded values, in general, can make the program inflexible and limit its usage.
Status	

Severity	**Informational**
Program	lib.rs
Description	No Events for State Changes
Code Snippet	N/A
Recommendation	The lack of events reduces transparency and on-chain traceability. These logs can be indexed off-chain to provide transparency, facilitate debugging, and create user-friendly interfaces for interacting with the program. Your program currently lacks events for critical state changes, such as when: the program is paused or unpaused, or when rewards are deposited or distributed. This absence of events reduces transparency and on-chain traceability, making it difficult for users or developers to monitor program behavior
Status	

Omitted Results

Note: Any issues that have been omitted from this report have been deemed by the reviewing team as irrelevant, inapplicable, and/or negligible to the proper functioning of this program. Thus, any omitted issues can be safely ignored.

Automated Review


beehive
Initialize
+ initialize(ctx: Context<Initialize>): Result<()>
TogglePause
+ pause(ctx: Context<TogglePause>): Result<()>
+ unpause(ctx: Context<TogglePause>): Result<()>
DepositToRewardPool
+ deposit_to_reward_pool(ctx: Context<DepositToRewardPool>, amount: u64): Result<()>
DistributeReward
+ distribute_reward(ctx: Context<DistributeReward>, amount: u64): Result<()>
State
- is_paused: bool
RewardPoolAccount
ErrorCode
+ InsufficientFunds
+ DistributionPaused
+ Unauthorized
modifies
modifies
modifies
checks
checks
returns

Conclusion

The program reviewed in this audit contain no critical severity issues and all Medium to Low issues have either been corrected or acknowledged.

Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only program mentioned in the report and does not include any other potential programs deployed by Owner.

Ascendant

Ascendant

@ascendantfi

www.ascendant.finance

Ascendant