DefiTuna Solana Program Code Review and Security Analysis

Client: DefiTuna

Date: 27th January 2025

Version: 1.0

Table of Contents

Security Review Information

Repository	tuna-programs
Initial commit	ddd372868f1282d720ca4d7b7450eecc2af9cdbc
Final commit	cc2138baaa4c28ddc1e26863a6668d604047106c
Scope	programs/tuna/*
Version	Final Report (v1.0)
Date	27th January 2025

Executive summary

As of January 27th, 2025, our comprehensive security review of the DefiTuna protocol has been concluded. Initially, the assessment identified **4** medium-severity and **1** low-severity vulnerabilities. Following our review and the implementation of recommended fixes, **4** medium-severity issues have been successfully resolved and **1** low-severity issue was acknowledged.

Introduction

Torii Security has been commissioned by DefiTuna to conduct a comprehensive security review of their Solana Program, focusing on the robustness, security, and efficiency of the program's implementation. The review aims to critically evaluate the program's architecture and codebase with the following specific objectives:

- **Verify Protocol Integrity**: Assess the program's operation against its design specifications to ensure it functions correctly and efficiently within the Solana ecosystem. This includes evaluating its interaction with other protocols and services on the Solana network.
- **Identify Security Vulnerabilities**: Uncover potential security weaknesses that could be exploited by malicious actors, including but not limited to, flaws in program logic, transaction handling, and external dependencies.
- **Detect Program Bugs**: Identify bugs and glitches in the code that may result in unintended or erratic program behavior, potentially compromising its performance or security.
- **Provide Improvement Recommendations**: Offer actionable advice to enhance the program's security posture, efficiency, and code clarity, aiming to fortify it against current and future security threats while improving maintainability and scalability.

Exclusions from This Review

While the security review conducted by Torii Security on behalf of DefiTuna provides a comprehensive analysis of the Solana Program's architecture, codebase, and security posture, certain aspects are beyond the scope of this audit. These exclusions are critical for stakeholders to understand, as they may require separate consideration and evaluation. The following areas have not been verified as part of this security review:

- **Deployment and Program Upgrade Process:** The procedures and mechanisms for deploying the Solana Program to the network and subsequent upgrades or modifications to the program are not covered. This includes the validation of deployment scripts, migration strategies, and the security of upgradeable contract mechanisms.
- **Keys Management**: The management, storage, and security practices for keys, including administrator keys and those used for program interactions, are outside the scope of this review. This encompasses both the technical and procedural safeguards in place to protect keys from unauthorized access or misuse.
- **Economic Vulnerabilities**: The review does not delve into the economic aspects or incentive structures of the DefiTuna protocol. Potential vulnerabilities arising from economic models, tokenomics, or financial incentives that could impact the program's security or integrity are not evaluated.

Protocol Overview

DeFiTuna is a decentralized finance (DeFi) platform on Solana designed to offer leveraged, concentrated liquidity provision on the Orca DEX. By allowing users to open positions within specific price ranges and borrow assets for additional exposure, DeFiTuna aims to enhance capital efficiency for liquidity providers (LPs) and yield opportunities for lenders.

Key Features

- **Concentrated Liquidity:** Users can deploy liquidity within a narrowly defined price range on Orca's CLMM, helping optimize capital usage and potentially increase trading fee earnings.
- **Leverage:** Open positions with up to 5x leverage to amplify potential returns. In the future we will be increasing it as we grow our TVL.
- **Lending:** Lenders can supply capital to earn interest, as borrowers pay variable rates that adjust with pool utilization. In times of high utilization, lending APY may spike, rewarding lenders for supplying liquidity.
- **Directional Bias and Hedging:** Liquidity providers can have a directional bias or a hedge by selecting which tokens to borrow as leverage. This allows users to optimize their positions based on market expectations.

Risks

The security review of the DefiTuna protocol has identified several risks.

DEX Dependency (Orca)

- DeFiTuna relies on Orca for its concentrated liquidity market making functionality. Any disruption, downtime, or exploit affecting Orca's operations could lead to partial or total loss of user funds, inability to close positions, or liquidity imbalances.

Network Congestion

- During high network usage or potential Solana slowdowns/halts, user transactions (e.g., adding/removing liquidity, loan management, or liquidation calls) could be delayed or fail. This can result in missed liquidation windows or suboptimal position adjustments, leading to larger-than-expected losses.

Liquidation Risk

- Leveraged positions can be partially or fully liquidated if the collateral's value falls or borrowed tokens appreciate, creating under-collateralized positions. Although DeFiTuna runs a liquidation bot, there is no guarantee of a timely liquidation under extreme market volatility or technical failures.

Lending Pool 100% Utilization Risk

- If borrower demand is extremely high, utilization of the lending pool can reach 100%. Lenders might be temporarily unable to withdraw their funds, though they would earn significantly higher APYs. This scenario persists until borrowers close or reduce positions.

Deployment & Upgradability Risk

- Errors in deployment or protocol updates (e.g., parameter changes, address configurations) can introduce operational issues, potentially causing incorrect calculations, locked positions, or disruptions to liquidity flows.

Configuration & Setup Risk

- DeFiTuna depends on the correct setup of essential parameters (e.g., fee schedules, markets, maximum leftovers from tokens swap). Any misconfiguration during deployment or updates can lead to inaccurate pricing, erroneous liquidations, or unintended financial outcomes.

Methodology

Issue Severity Classification

This report differentiates identified issues into distinct severity levels, each reflecting the potential impact on the system's security and overall functionality.

Severity	Description
Critical	Issues that present an immediate and severe threat, such as significant financial loss, irreversible locking of funds, or catastrophic system failure. These vulnerabilities require urgent remediation.
High	Bugs or vulnerabilities that could disrupt the correct operation of the system, potentially leading to incorrect states or temporary denial of service. Prompt attention and corrective action are necessary.
Medium	Issues that indicate deviations from best practices or suboptimal use of system primitives. While they may not pose immediate security threats, these issues could lead to vulnerabilities or inefficiencies if unaddressed.
Low	Minor concerns that have anegligible impact on system security or functionality. These may include inefficiencies or minor deviations from best practices that are unlikely to affect the system's operation significantly.
Informational	Suggestions related to design decisions, potential enhancements, or optimizations that do not have adirect impact on security. Implementing these recommendations may improve aspects such as usability or code readability but is not essential for system security.

Issue Status Definitions

Each issue is assigned a status reflecting its current resolution stage.

Status	Description
Pending	The issue has been identified but not yet reviewed or addressed by the development team.
Acknowledged	The development team has recognized the issue but has not completed its resolution.
Resolved	The issue has been fully addressed, with implemented changes verified for effectiveness.

Findings Summary

ID	Title	Severity	Status
TS-M1	Invalid Computation in `get_liquidity_for_amount_a` Impedes Adding Liquidity	Medium	Resolved
TS-M2	Loss of Interest on High Call Rate of `accrue_interest`	Medium	Resolved
TS-M3	Potential Leverage Exceedance in `remove_liquidity_orca` Due to Insufficient Validation	Medium	Resolved
TS-M4	Lack of ATA Enforcement for `TunaPosition` Accounts	Medium	Resolved
TS-L1	Lack of Validation for Liquidation Threshold During Market Creation	Low	Acknowledged
TS-I1	The `tuna_config` Creation Can be Front-Run to Set Authorities	Informational	Acknowledged
TS-I2	Risk of Program Lock Due to Single Ownership Pattern in `set_admin_authority` and `set_owner_authority`	Informational	Acknowledged
TS-I3	Missing Events for Critical Admin Actions	Informational	Acknowledged

Detailed Findings

TS-M1 - Invalid Computation in `get_liquidity_for_amount_a` Impedes Adding Liquidity

Classification

Severity: Medium

Status: Resolved

Description

In the `get_liquidity_for_amount_a` function, which is utilized when adding liquidity to a position, a multiplication before division pattern is implemented.
In case this overflows, there is an alternative code path that performs division before multiplication instead.
However, the multiplicator used in the alternative code path is wrong, leading to a miscalculated liquidity amount which in turn can cause failure of the `add_liquidity_orca` instruction.

Recommendation

We recommend applying the following changes:

```
let liquidity: U256 = match intermediate.checked_mul(wide_amount) {
     // If the previous equation overflows, try another one that
does a division first
     None => wide_amount
     .div(delta_sqrt_price)
-    .checked_mul(U256::from(sqrt_price_lower))
+    .checked_mul(U256::from(intermediate))
     .ok_or(ErrorCode::MathOverflow)?,
     Some(r) => r.div(delta_sqrt_price),
};
```

Remediation

Issue was fixed in cc2138baaa4c28ddc1e26863a6668d604047106c by implementing the recommended patch.

TS-M2 - Loss of Interest on High Call Rate of `accrue_interest`

Classification

Severity: Medium

Status: Resolved

Description

The `accrue_interest` function skips interest accrual when the elapsed time since the last call is less than `INTEREST_ACCRUE_MIN_INTERVAL`.

However, the last update time is still updated on each call, even if no interest was accrued. This leads to permanent loss of interest in case of a high call rate.

Recommendation

We recommend only updating the last update time when interest was effectively accrued.

Remediation

The issue was fixed in 77460e344edb1aecf88a25b79de7cab154d24d55 and c3b346e5694f5b8cabeb5f2ca21f23af42112f73 commits.

TS-M3 - Potential Leverage Exceedance in `remove_liquidity_orca` Due to Insufficient Validation

Classification

Severity: Medium

Status: Resolved

Description

The `remove_liquidity_orca` function currently uses the `is_healthy` check to validate the position's health when withdrawing liquidity. However, this approach may not adequately ensure that the position remains within the defined leverage limits. Specifically, it may allow the maximum leverage (`max_leverage`) to be exceeded during liquidity withdrawal, potentially due to leftover tokens. This is because the `is_healthy` check focuses on position health rather than enforcing leverage constraints.

Recommendation

Consider verifying max leverage on liquidity withdrawal.

Remediation

The issue was fixed in a22015d776a33f90739d842dc77c89d6f97d35cb by checking the max leverage on liquidity withdrawal.

TS-M4 - Lack of ATA Enforcement for `TunaPosition` Accounts

Classification

Severity: Medium

Status: Resolved

Description

The TunaPosition account assumes that it must be an Associated Token Account (ATA). However, there is no validation to ensure this requirement (e.g., by checking whether the Program Derived Address (PDA) is the owner and the mint is correct). As a result, it is possible for users to provide a standard token account instead of an ATA, which can lead to the following consequences:

1. Data Desynchronization

When users create a position and add liquidity, they can specify a `TunaPosition` account that is not an ATA. Any leftover tokens from the liquidity addition process are stored in this account, but the `TunaPosition` still accounts for these leftovers. This mismatch can cause data desynchronization, potentially affecting processes such as health checks, as the leftovers are used to verify health.

2. Impact on Liquidation Liquidators rely on the standard ATA for liquidations (as per current script behavior). If the `TunaPosition` is not an ATA, the liquidator will not consider the leftover tokens stored in the non-ATA account. This results in fewer funds allocated to the liquidator and the vault, leaving old leftovers untouched.

3. Post-Liquidation Exploitation

After liquidation, the user can still close the position using the non-ATA account specified during creation. This allows the user to retain any leftover funds, exploiting the discrepancy.

4. Strategic Exploitation

While the amounts involved may not be large (e.g., up to 1% of the total), a user could intentionally exploit this behavior to hedge their strategy by ensuring leftover funds are retained.

Recommendation

Verify if Tuna Position Token Accounts are Associated Token accounts.

Remediation

The issue was fixed in 3b9dc25b63dce1ba1f349ea5022d890c26d7dbd5 by validating Associated Token accounts.

TS-L1 - Lack of Validation for Liquidation Threshold During Market Creation

Classification

Severity: Low

Status: Acknowledged

Description

Although the `create_market` function is permissioned, there is currently no validation to ensure that the `liquidation_threshold` parameter is configured appropriately relative to the `max_leverage`. This oversight could allow markets to be created where positions are leveraged dangerously close to the liquidation threshold, making them highly vulnerable to minor price swings.

Recommendation

Implement a validation check in the `create_market` function to ensure that `liquidation_threshold` is at least `1.05 * (max_leverage - 1) / max_leverage` (or another safety factor of your choice) to prevent positions from being overly exposed to liquidation due to minor price swings.

TS-I1 - The `tuna_config` Creation Can be Front-Run to Set Authorities

Classification

Severity: Informational

Status: Acknowledged

Description

Due to the lack of authority/signer account validation in the `create_tuna_config` instruction, the first one to call it after program deployment can set the `owner_authority`, `admin_authority` and `fee_recipient`.

In case this happens, the program needs to be redeployed.

Recommendation

We recommend to require co-signing of the `create_tuna_config` instruction with the private key of the program, which should only be known to the deployer. This requires the following addition to the `CreateTunaConfig` context:

```
#[account(address = crate::ID)]
pub program: Signer<'info>
```

TS-I2 - Risk of Program Lock Due to Single Ownership Pattern in `set_admin_authority` and `set_owner_authority`

Classification

Severity: Informational

Status: Acknowledged

Description

The set_admin_authority and set_owner_authority functions currently use a single ownership transfer pattern, which poses a risk of the program becoming locked if an incorrect owner or admin is set. Specifically:

- If an incorrect owner is set, the fee receiver cannot be modified.
- If an incorrect admin is set, operational configurations cannot be updated.

Recommendation

Implement a two-step ownership transfer process to mitigate the risks:

1. Require the current owner or admin to initiate the transfer by setting a pending new authority.
2. Require the pending new authority to confirm the transfer to finalize the process.

Alternatively, use a cosigned transaction model where both the current and new owner/admin must approve the transfer during the rotation. This ensures that authority changes are intentional and secure, reducing the risk of program lock.

TS-I3 - Missing Events for Critical Admin Actions

Classification

Severity: Informational

Status: Acknowledged

Description

The system currently lacks event emission for critical administrative actions, such as:

- create_market
- create_vault.
- create_tuna_config
- update_market
- set_fee_authority
- set_admin_authority
- set_suspended_state
- set_owner_authority
- set_max_percentage_of_leftovers
- set_max_swap_slippage

These actions represent significant changes to the system's state and should be logged for transparency, auditing, and debugging purposes.

Recommendation

Emit events for all critical administrative actions - use Anchor events.

Disclaimer

This report has been prepared in accordance with the current best practices and standards applicable at the time of its preparation. It is intended to provide an analysis and evaluation of the subject matter based on the information available, including any potential vulnerabilities, issues, or risks identified during the assessment process. The scope of this analysis is limited to the data, documents, and materials provided for review, and the conclusions drawn are based on the status of the information at the time of the report.

No representation or warranty, express or implied, is made as to the absolute completeness or accuracy of the information contained in this report. It is important to acknowledge that the findings, conclusions, and recommendations presented are subject to change should there be any modifications to the subject matter. This report should not be viewed as aconclusive or exhaustive evaluation of the subject matter's safety, functionality, or reliability. Stakeholders are encouraged to conduct their own independent reviews, assessments, and validations to ensure the integrity and security of the evaluated subject.

The authors and auditors of this report disclaim any liability for any direct, indirect, incidental, or consequential damages or losses that may result from reliance on this report or its contents. It is the responsibility of the stakeholders to ensure the ongoing monitoring and evaluation of the subject matter to mitigate potential risks or vulnerabilities.

The nature of technology and digital systems means that they are inherently subject to risks, including but not limited to vulnerabilities, bugs, and security threats that may not be foreseeable at the time of this report. The dynamic and evolving nature of technological standards, security practices, and threat landscapes means that absolute security cannot be guaranteed. Despite thorough analysis and evaluation, unforeseen vulnerabilities may exist, and new threats may emerge subsequent to the issuance of this report.

The authors and auditors make no guarantee regarding the impenetrability or infallibility of the subject matter under evaluation. Stakeholders are advised to implement comprehensive security measures, including but not limited to regular updates, patches, and monitoring, to safeguard against potential threats and vulnerabilities.