Milestone Report: On-chain Components for Decentralized Farming (Milestone 2)

Internal Audit and Testing Report

November 10, 2024

Contents

1 Introduction

This milestone report outlines the completion of **Milestone 2**: "On-chain Components" for our Cardano Catalyst project focused on decentralized farming solutions. The deliverables for this milestone include:

- Implementation of on-chain code for staking contracts, top-pools contracts, and governance-based farms using the OpShin programming language.
- An internal security audit to ensure code robustness and security for all components.
- A published blog post detailing the on-chain implementation and audit results.

The internal security audit was conducted by two members of the MuesliSwap team who were not involved in the development of the smart contract code, ensuring an unbiased review.

This document provides a comprehensive internal audit report and summarizes the testing conducted on the smart contracts developed.

2 On-chain Components Overview

Our decentralized farming solution comprises three main on-chain components. For detailed information on system architecture and design choices, please refer to the report provided for **Milestone 1** of the project.

1. **Staking Contracts**: Manage staking positions, track rewards, and enforce correct reward distribution in a trustless manner.
2. **Top-Pools Contracts**: Automatically allocate rewards to the largest liquidity pools based on their total value locked (TVL).
3. **Governance-Based Farms**: Allow the community to allocate rewards to pools through governance proposals, promoting decentralized decision-making.

3 Internal Security Audit

The internal security audit focused on ensuring that each component meets essential security properties and operates as intended without vulnerabilities. We summarize the key aspects that need to be ensured by the smart contracts, which are critical for the safety and integrity of the system. These properties were the main focus during our audit and testing.

3.1 Security Properties

3.1.1 Staking Contracts

The staking contracts are central to the farming solution, handling users' staking positions and reward calculations. The following security properties were identified as critical:

- **Accurate Reward Calculation**: Rewards must be calculated precisely based on staking amount and duration to ensure fairness.
- **Immutable Staking Positions**: Unauthorized modifications to staking positions must be prevented. Only the owner of a staking position is authorized to modify or unstake it.
- **Strict Time Handling**: Time variables must be handled carefully to prevent negative rewards or exploitation through time manipulation.
- **Secure Unstaking Process**: Only the rightful owner or an authorized entity (e.g., a batcher with permission) should be able to unstake positions.

3.1.2 Top-Pools Contracts

The top-pools contracts manage the automatic allocation of rewards to the largest liquidity pools. The following security properties are essential:

- **Valid Pool Ranking**: Pool rankings should accurately reflect the liquidity of pools, ensuring that rewards are allocated to the top pools by TVL.
- **Liquidity Verification**: Pool liquidity must be accurately verified to prevent manipulation. This is achieved by using the respective liquidity pool (LP) smart contract UTxO as a reference input in the transaction, ensuring that the reported liquidity matches the on-chain state.
- **Secure Reward Allocation**: Rewards must be allocated correctly based on the verified pool rankings.
- **Parameter Integrity**: Contract parameters should be protected from unauthorized changes, maintaining system integrity.

3.1.3 Governance-Based Farms

Governance-based farms empower the community to decide on reward allocations through proposals. The critical security properties include:

- **Community Governance**: Farm creation and updates should be controlled via governance proposals, ensuring that changes reflect the community's will.
- **Secure Execution**: Unauthorized creation or modification of farms must be prevented. Only proposals that have been approved through the governance process can enact changes.
- **Transparency**: Governance decisions should be transparent and auditable, maintaining trust within the community.

3.2 Audit Findings and Resolutions

The audit identified several issues of varying severity, which have been addressed in the codebase. Issues are categorized based on their potential impact:

- **Critical Severity**: Issues that could lead to significant security vulnerabilities, financial loss, or system failure.
- **Moderate Severity**: Issues that could affect system performance, reliability, or minor security concerns.
- **Minor Severity**: Issues that impact efficiency, user experience, or code maintainability without posing direct security risks.

Below is a detailed log of the findings, their significance, discussions, resolutions, and references to the specific code commits where fixes were implemented.

3.2.1 Critical Severity Issues

Critical Severity

Finding: Ensuring Exclusive Spending of Staking Positions
During unstaking transactions, it was possible for additional staking positions beyond the intended one to be spent from the contract. This vulnerability could allow malicious actors to unstake positions they do not own.

- ◇ **Potential Risk:** High. Unauthorized unstaking could lead to loss of user funds and undermine trust in the platform.
- ◇ **Discussion:** The team recognized the need to enforce strict checks to ensure only the specified staking position is spent during an unstaking transaction.
- ◇ **Resolution:** Implemented validations to guarantee that no additional staking positions are spent from the contract during unstaking. The contract now verifies that exactly one staking position is consumed per unstaking transaction.
- ◇ **Commit:** 38fcb46f11b9ae5fad851c0a73e118c017d11917

Critical Severity

Finding: Strictly Increasing `last_update_time`
The `last_update_time` in the `FarmState` did not enforce a strictly increasing constraint. If manipulated, this could result in negative rewards or incorrect reward calculations.

- **Potential Risk:** High. Users could receive incorrect rewards, leading to financial discrepancies and potential exploitation.
- **Discussion:** It was crucial to ensure that time moves forward in the contract's logic to maintain reward calculation integrity.
- **Resolution:** Added a check to enforce that `last_update_time` is strictly increasing with each transaction, preventing any backward time manipulation.
- **Commit:** 998281c7ba099b092322a44afd8d8b323957f15e

3.2.2 Moderate Severity Issues

Moderate Severity

Finding: Incorrect Time Unit in Reward Calculation
The computation of the `cumulative_rewards_per_token` variable used days as the time unit, which led to fractions with large denominators and potential precision loss, especially for short staking durations.

- **Potential Risk:** Medium. Inaccurate reward calculations could lead to users receiving less or more than their fair share.
- **Discussion:** Switching to milliseconds improves calculation precision and aligns with the time units used elsewhere in the contract.
- **Resolution:** Updated the reward calculation to use milliseconds instead of days, enhancing precision and consistency.
- **Commit:** 4ea292425b2887db21f2dbb79c431fe376876952

Moderate Severity

Finding: Switching from Slot Numbers to POSIX Time
Initially, the contracts used slot numbers for time handling, which can be inconsistent due to variable slot lengths and network parameters.

- **Potential Risk:** Medium. Time inconsistencies could affect reward calculations and contract behavior.
- **Discussion:** Using POSIX time provides a consistent and reliable time measurement, reducing potential discrepancies.
- **Resolution:** Replaced slot numbers with POSIX time in milliseconds for all time-related computations, ensuring consistency across different network conditions.
- **Commit:** 0f2db4d6d64a191baa540f9afb564cc03010b391

3.2.3 Minor Severity Issues

Minor Severity

Finding: Allowing Batching of Multiple Orders
The initial implementation did not support batching multiple staking orders, leading to inefficiencies in transaction processing and higher costs for users.

- ◇ **Potential Risk:** Low. Affects operational efficiency but not security.
- ◇ **Discussion:** Enabling batching would improve network efficiency and reduce fees for users by processing multiple orders in a single transaction.
- ◇ **Resolution:** Modified the contracts to support batching multiple staking orders, enhancing performance and user experience.
- ◇ **Commit:** 94be12fc8acdcd86a9b862ed3f1038ad0c40db1e

Minor Severity

Finding: Enabling Batching of Multiple Unstaking Orders
Similarly, the contracts did not support batching multiple unstaking orders, limiting efficiency and increasing transaction costs.

- ◇ **Potential Risk:** Low. Impacts transaction throughput and user costs.
- ◇ **Discussion:** Batching unstaking orders reduces network congestion and lowers fees, benefiting both users and the network.
- ◇ **Resolution:** Updated the contracts to allow multiple unstaking orders to be processed in a single transaction.
- ◇ **Commit:** 5545a17d11372a56278e886b730d9388bb9bd321

Minor Severity

Finding: Introduction of Unstake Permission NFT
To facilitate third-party batchers processing unstaking on behalf of users, an unstake permission NFT was introduced. This NFT grants permission to batchers to unstake tokens in the name of the staker.

- ◇ **Potential Risk:** Low. Enhances functionality without compromising security.
- ◇ **Discussion:** The unstake permission NFT improves user experience by allowing batchers to process unstaking requests efficiently.
- ◇ **Resolution:** Implemented the unstake permission NFT within the contract logic, ensuring only authorized batchers can unstake on behalf of users.
- ◇ **Commit:** 0c565d489e6d78d692ad00ef946133892b67716d

3.3 Additional Fixes and Improvements

Minor Severity

Finding: General Code Refactoring and Optimization
Throughout the audit, opportunities for general code improvements were identified, including optimizing functions and improving readability.

- ◇ **Potential Risk:** Low. Enhances maintainability and future development.
- ◇ **Discussion:** Refactoring code improves performance and makes future audits and updates more manageable.
- ◇ **Resolution:** Applied code refactoring where appropriate, improving efficiency and readability without altering functionality.
- ◇ **Commits:** Various commits addressing code optimization and refactoring.

4 Testing Report

Comprehensive testing was conducted to validate the functionality and reliability of the smart contracts. Transactions were constructed using preliminary off-chain scripts written in `pycardano` and submitted to the Cardano pre-production testnet.

The chosen tests mimic typical interactions with the main smart contract functionalities, covering the entire workflow from farm creation to staking, cancellation, and batching. This ensures that critical parts of the smart contract logic are thoroughly tested.

4.1 Test Cases and Results

1. **Creation of Farm**
   - **Objective**: Test the initialization and deployment of a new farm contract.
   - **Transaction**: 73c2ec0
   - **Result**: Successfully deployed the farm contract with correct parameters. The farm UTxO was created with a unique NFT representing this farm, which was minted in the same transaction. This can be verified using the testnet explorer via the provided link.
2. **Placing a Stake Order**
   - **Objective**: Verify that a user can place a stake order.
   - **Transaction**: af5827b
   - **Result**: Stake order was placed successfully, and the order datum was correctly recorded on-chain. Key fields such as the staking amount, owner address, and pool ID were verified using the testnet explorer.
3. **Canceling a Stake Order**
   - **Objective**: Ensure that users can cancel their stake orders before processing.
   - **Transaction**: 7f7be38
   - **Result**: Stake order was canceled successfully, and funds were returned to the user without issues. The cancellation can be verified via the testnet explorer, confirming that the order UTxO was consumed appropriately.
4. **Placing Another Stake Order**
   - **Objective**: Test that multiple stake orders can be placed sequentially without interference.
   - **Transaction**: 5484008

- **Result**: Second stake order was placed successfully, demonstrating the contract's ability to handle multiple orders. Key datum fields were verified to ensure correctness.

5. **Batch Processing Stake Orders**

   - **Objective**: Validate the batching functionality for processing multiple stake orders in a single transaction.
   - **Transaction**: a110b11
   - **Result**: Multiple stake orders were processed successfully in one transaction, confirming the efficiency gains from batching. This batching can be performed by any third-party batcher, as the smart contract verifies correct application, enabling permissionless batching.

4.2 Testing Environment

- **Network**: Cardano Pre-production Testnet
- **Tools Used**: `pycardano` library for constructing and submitting transactions.
- **Wallets**: Test wallets funded with test ADA for transaction fees and staking amounts.

4.3 Observations

- All transactions were successfully accepted by the network and behaved as expected according to the contract logic.
- Batching significantly reduced the number of transactions required, leading to lower fees and faster processing times.
- Respective datums and NFTs were publicly inspected via the testnet explorer to verify that staking positions are correctly encoded. This transparency allows users to independently verify the state of their staking positions.
- No anomalies or unexpected behaviors were observed during testing, indicating robust contract performance.

5 Conclusion

The completion of **Milestone 2** marks a significant step towards providing a fully decentralized and feature-rich farming solution on the Cardano blockchain. The internal audit has ensured that the smart contracts are secure, robust, and function as intended. Testing on the pre-production testnet has validated the operational aspects of the contracts.

A fundamental next step is to develop robust and production-ready off-chain infrastructure for interaction with the smart contracts (Milestone 3). This will involve creating user interfaces and off-chain components that allow users to interact seamlessly with the on-chain contracts. Making everything work together is key to moving towards mainnet deployment in the final step.