POOLS

SMART CONTRACTS REVIEW

zokyo

May 12th 2025 | v. 1.0

Security Audit Score

PASS

Zokyo Security has concluded that these smart contracts passed a security audit.

ZOKYO AUDIT SCORING POOLS FINANCE

1. Severity of Issues:
    - Critical: Direct, immediate risks to funds or the integrity of the contract. Typically, these would have a very high weight.
    - High: Important issues that can compromise the contract in certain scenarios.
    - Medium: Issues that might not pose immediate threats but represent significant deviations from best practices.
    - Low: Smaller issues that might not pose security risks but are still noteworthy.
    - Informational: Generally, observations or suggestions that don't point to vulnerabilities but can be improvements or best practices.
2. Test Coverage: The percentage of the codebase that's covered by tests. High test coverage often suggests thorough testing practices and can increase the score.
3. Code Quality: This is more subjective, but contracts that follow best practices, are well-commented, and show good organization might receive higher scores.
4. Documentation: Comprehensive and clear documentation might improve the score, as it shows thoroughness.
5. Consistency: Consistency in coding patterns, naming, etc., can also factor into the score.
6. Response to Identified Issues: Some audits might consider how quickly and effectively the team responds to identified issues.

SCORING CALCULATION:

Let's assume each issue has a weight:
- Critical: -30 points
- High: -20 points
- Medium: -10 points
- Low: -5 points
- Informational: 0 points

Starting with a perfect score of 100:
- 0 Critical issues: 0 points deducted
- 0 High issues: 0 points deducted
- 1 Medium issue: 1 resolved = 0 points deducted
- 2 Low issues: 2 resolved = 0 points deducted
- 0 Informational issues: 0 points deducted

Thus, the score is 100

TECHNICAL SUMMARY

This document outlines the overall security of the Pools Finance smart contract/s evaluated by the Zokyo Security team.

The scope of this audit was to analyze and document the Pools Finance smart contract/s codebase for quality, security, and correctness.

Contract Status

There were 0 critical issues found during the review. (See Complete Analysis)

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract/s but rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that can withstand the Ton network's fast-paced and rapidly changing environment, we recommend that the Pools Finance team put in place a bug bounty program to encourage further active analysis of the smart contract/s.

Table of Contents

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the Pools Finance repository:
Repo: https://github.com/Pools-Finance/pools-protocol/commit/13349bc17fc2f93e3f56e86b6f8b6b07865f9b9e

Last commit - 6c114e192e4a7ec0876d80aa16ccdefce3622bc0

Within the scope of this audit, the team of auditors reviewed the following contract(s):

- amm_config.move
- amm_entries.move
- amm_math.move
- amm_router.move
- amm_stable_utils.move
- amm_swap.move
- amm_utils.move
- stake_config.move
- stake_entries.move
- stake.move

During the audit, Zokyo Security ensured that the contract:

- Implements and adheres to the existing standards appropriately and effectively;
- The documentation and code comments match the logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices, efficiently using resources without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the most recent vulnerabilities;
- Meets best practices in code readability, etc.

Zokyo Security has followed best practices and industry-standard techniques to verify the implementation of Pools Finance smart contract/s. To do so, the code was reviewed line by line by our smart contract developers, who documented even minor issues as they were discovered. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

01 Due diligence in assessing the overall code quality of the codebase.

02 Cross-comparison with other, similar smart contract/s by industry leaders.

03 Thorough manual review of the codebase line by line.

Executive Summary

The audited codebase demonstrates a solid architectural foundation with well-structured modules for AMM and staking functionality, borrowing heavily from a proven, battle-tested protocol. Our audit identified a small number of medium- and low-severity issues, all of which have been acknowledged and successfully remediated by the development team. With the latest fixes in place, the protocol now reflects a strong security posture and readiness for deployment.

STRUCTURE AND ORGANIZATION OF THE DOCUMENT

For the ease of navigation, the following sections are arranged from the most to the least critical ones. Issues are tagged as “Resolved” or “Unresolved” or “Acknowledged” depending on whether they have been fixed or addressed. Acknowledged means that the issue was sent to the Pools Finance team and the Pools Finance team is aware of it, but they have chosen to not solve it. The issues that are tagged as “Verified” contain unclear or suspicious functionality that either needs explanation from the Client or remains disregarded by the Client. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

FINDINGS SUMMARY

#	Title	Risk	Status
1	Unchecked Overflow in accum_reward Calculations	Medium	Resolved
2	Potential Integer Truncation in Fee Calculations	Low	Resolved
3	Lack of Input Validation on Coin Metadata in Some Paths	Low	Resolved

MEDIUM-1 | RESOLVED

Unchecked Overflow in accum_reward Calculations

File: stake.move

Description:
In update_accum_reward() and accum_rewards_since_last_updated(), large values for reward_per_sec, scale, or long durations can result in a multiplication overflow (u64 × u64 × u64 → u128), especially if the pool runs for an extended period or has large-scale usage.

Recommendation:
Use u128::checked_mul() and checked_add() or a helper function that validates these large operations and reverts gracefully on overflow.

LOW-1 | RESOLVED

Potential Integer Truncation in Fee Calculations

File: amm_utils.move

Description:
Fee-related math safely casts from u64 → u128 during intermediate computation, then back to u64. However, precision may be lost for extreme values or boundary edge cases.
Design seems intentional, with u128 providing intermediate safety. No actual vulnerability found unless truncation leads to a logic error in fee application.

Recommendation:
Add test coverage for large amount_in/amount_out and edge-case fee_numerator/fee_denumerator values to confirm stable behavior across input bounds.

LOW-2 | RESOLVED

Lack of Input Validation on Coin Metadata in Some Paths

File: stake_entries.move

Description:
While not a vuln, there's an assumption that decimalS and decimalR passed into register_pool are meaningful and valid. There's a validation inside stake.move, but no enforcement on external callsites (i.e., clients could misconfigure a pool with incorrect decimals unless tightly integrated).

Recommendation:
Document expected range or enforce limits (e.g., decimal <= 18) closer to the entry point, not just deep inside logic.

	amm_config.move amm_entries.move amm_math.move amm_router.move amm_stable_utils.move
Re-entrancy	Pass
Access Management Hierarchy	Pass
Arithmetic Over/Under Flows	Pass
Unexpected Ether	Pass
Delegatecall	Pass
Default Public Visibility	Pass
Hidden Malicious Code	Pass
Entropy Illusion (Lack of Randomness)	Pass
External Contract Referencing	Pass
Short Address/ Parameter Attack	Pass
Unchecked CALL Return Values	Pass
Race Conditions / Front Running	Pass
General Denial Of Service (DOS)	Pass
Uninitialized Storage Pointers	Pass
Floating Points and Precision	Pass
Tx.Origin Authentication	Pass
Signatures Replay	Pass
Pool Asset Security	Pass

	amm_swap.move amm_utils.move stake_config.move stake_entries.move stake.move
Re-entrancy	Pass
Access Management Hierarchy	Pass
Arithmetic Over/Under Flows	Pass
Unexpected Ether	Pass
Delegatecall	Pass
Default Public Visibility	Pass
Hidden Malicious Code	Pass
Entropy Illusion (Lack of Randomness)	Pass
External Contract Referencing	Pass
Short Address/ Parameter Attack	Pass
Unchecked CALL Return Values	Pass
Race Conditions / Front Running	Pass
General Denial Of Service (DOS)	Pass
Uninitialized Storage Pointers	Pass
Floating Points and Precision	Pass
Tx.Origin Authentication	Pass
Signatures Replay	Pass
Pool Asset Security	Pass

We are grateful for the opportunity to work with the Pools Finance team.

The statements made in this document should not be interpreted as an investment or legal advice, nor should its authors be held accountable for the decisions made based on them.

Zokyo Security recommends the Pools Finance team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo