Shaolin Security Audit

Report Version 1.0

January 15, 2025

Conducted by **Hunter Security**

Table of Contents

1 About Hunter Security

Hunter Security is an industry-leading smart contract security auditing firm. Having conducted over 100 security audits protecting over $1B of TVL, our team always strives to deliver top-notch security services to the best DeFi protocols. For security audit inquiries, you can reach out on Telegram or Twitter at *@georgehntr*.

2 Disclaimer

Audits are a time-, resource-, and expertise-bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can reveal the presence of vulnerabilities, but cannot guarantee their absence.

3 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	High	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

3.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - involves a small loss of funds or affects a core functionality of the protocol.
- **Low** - encompasses any unexpected behavior that is non-critical.

3.2 Likelihood

- **High** - a direct attack vector; the cost is relatively low compared to the potential loss of funds.
- **Medium** - only a conditionally incentivized attack vector, with a moderate likelihood.
- **Low** - involves too many or unlikely assumptions; offers little to no incentive.

3.3 Actions required by severity level

- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

4 Executive summary

Overview

Project Name	Shaolin
Repository	https://github.com/De-centraX/shaolin-contracts
Commit hash	8e3d55d0bf0c51f3d0c5ff99da94b6d1b335af2d
Resolution	5856224071628d881ddb2f4e914fb0a5ea232453
Methods	Manual review & testing

Scope

src/pools/LamboPool.sol
src/pools/WBTCPool.sol
src/BuyAndBurn.sol
src/LotusEdenBnBFeeder.sol
src/Mining.sol
src/Shaolin.sol
src/Staking.sol
src/WBTCPoolFeeder.sol

Issues Found

High risk	0
Medium risk	0
Low risk	6
Informational	8

5 Findings

5.1 Low

5.1.1 Incorrect pool ownership setup

Severity: Low

Files: src/pools/LamboPool.sol

Description: Since the LamboPool is deployed by the Mining contract, the latter would automatically claim ownership despite having an *admin* passed in the constructor. This happens because LamboPool inherits from VRFConsumerBaseV2Plus which is inherits ConfirmedOwnerWithProposal. Therefore, there will be no access to the *setCoordinator* which may be needed at some point.

Recommendation: Consider setting the passed *admin* as the owner upon construction.

Resolution: Resolved.

5.1.2 Hard-coding gas limit may cause a Denial-of-Service

Severity: Low

Files: src/pools/LamboPool.sol

Description: When requesting randomness, 400_000 gas units is set as the callback gas limit. Hard-coding such parameters may lead to critical issues in the future.

Recommendation: Consider implementing a setter method which can increase the gas limit if the current one is not sufficient. Carefully test what the maximum gas consumption by the callback function could be and use it as a minimum.

Resolution: Resolved.

5.1.3 Uneven rewards distribution if less than 5 total pool participants

Severity: Low

Files: src/pools/LamboPool.sol

Description: The pool requests randomness and based on the result picks 5 winners out of the participants list. However, if there are less than 5 participants in the pool, the callback will revert as it would try to access a non-existent index in the set of participants. The severity of this issue is low as unblocking mechanism is already implemented and the likelihood of having less than 5 participants in extremely low.

Recommendation: Consider splitting the rewards equally among the 1-4 participants. Additionally, consider a mechanism which does not allow a single user to be picked as a winner twice in a single round.

Resolution: Resolved.

5.1.4 Denial-of-Service of the BnB contracts due to unsafe integer downcast.

Severity: Low

Files: src/BuyAndBurn.sol, src/LotusEdenBnBFeeder.sol, src/WBTCPoolFeeder.sol

Description: All BnB contracts share the same *_calculateMissedIntervals* function which downcasts the result of *timeElapsedSince / 5 minutes* to type *uint16*. The problem is that if there has been no update for about 8 months, the downcast will cause a silent integer overflow which would permanently DoS the according BnB contract.

Recommendation: Consider using *uint32* instead of *uint16*.

Resolution: Resolved.

5.1.5 Not accounting for mining bonus

Severity: Low

Files: src/Mining.sol

Description: The *_mine* function returns the tokens that would be mined without accounting for the lRank bonus which is added upon claim. Therefore, the *toMine* value returned by *startMining* and *startMiningBatch* which is also stored in *totalMined* and used for participation in the WBTCPool, will be lower than what will actually be mined.

Recommendation: Consider adding the lRank bonus to the *toMine* amount.

Resolution: Acknowledged.

5.1.6 Dust amount of tokens could be lost due to potential slippage

Severity: Low

Files: src/Mining.sol

Description: If any slippage occurs upon adding the initial liquidity to the UniswapV3 pool, some tokens will remain locked in the Mining contract.

Recommendation: Consider sending any dust tokens to the owner.

Resolution: Resolved.

5.2 Informational

5.2.1 Typographical mistakes, non-critical issues or centralization vulnerabilities

Severity: Informational

Files: src/*

Description: The contracts contain one or more typographical mistakes, non-critical issues or centralization vulnerabilities. In an effort to keep the report size reasonable, we enumerate these below:

1. The *admin* storage variable in LamboPool could be marked *immutable*.

2. Redundant check - notAmount0(_newRequestConfirmations).
3. *_startOfMining* is always *block.timestamp*.
4. *startTs* and *cost* properties stored in miners, but never used (read).
5. Typo - *distribeDay16*.
6. *startTimestamp* could be marked *immutable*.
7. Check against *address(0)* for *receiver* in *claim*.
8. The *minerCostDailyIncrease* property is never used.

Recommendation: Consider fixing the above typographical mistakes, non-critical issues or centralization vulnerabilities.

Resolution: Resolved.