SMART CONTRACT AUDIT
Report 1 of 6: CLMM Core

March 14th 2023 | v. 1.0

Security Audit Score

PASS

Zokyo Security has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

TECHNICAL SUMMARY

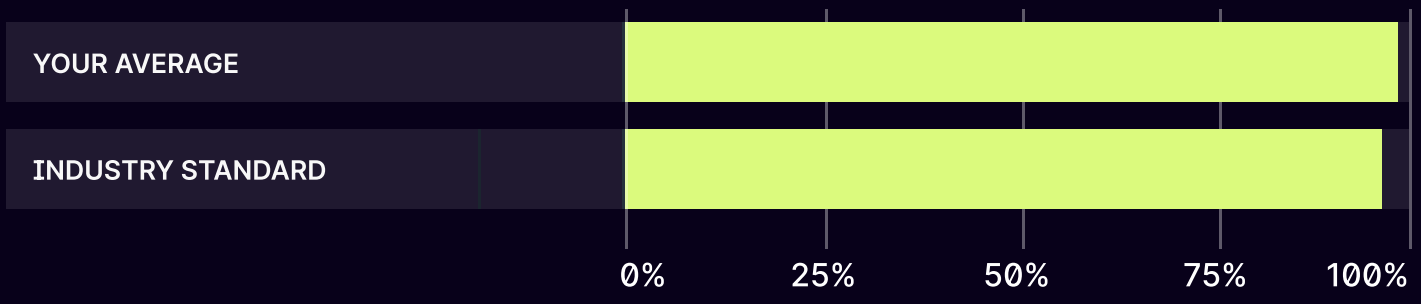
This document outlines the overall security of the TangleSwap smart contracts evaluated by the Zokyo Security team.

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

Contract Status

There was 1 critical issue found during the audit. (See in the Complete Analysis, started from 6 page)

Testable Code



100% of the code is testable, which is above the industry standard of 95%.

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

Table of Contents

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the TangleSwap repository: https://github.com/TangleSwap/tangleswap-core

Last commit: 793cdcbfca3d9660804e595336abd4bbd12b54ac

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

- TangleswapFactory
- TangleswapPool
- IUniswapV3Factory
- IUniswapV3Pool
- IPriceOracle
- IUniswapV3PoolState
- IUniswapV3PoolActions
- IUniswapV3PoolOwnerActions
- INonfungiblePositionManager

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 reentrancy 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 TangleSwap smart contracts. To do so, the code was reviewed line by line by our smart contract developers, who documented even minor issues as they were discovered. Part of this work includes writing a test suite using the Truffle testing framework. 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 contracts by industry leaders.
03	Testing contract logic against common and uncommon attack vectors.
04	Thorough manual review of the codebase line by line.

Executive Summary

There was one critical issue found during the audit, alongside one with medium severity and some of low severity. All the mentioned findings may have an effect only in case of specific conditions performed by the contract owner and the investors interacting with it. They are described in detail in the “Complete Analysis” section.

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” depending on whether they have been fixed or addressed. 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	No initialization	Critical	Resolved
2	Abuse of the owner power	Medium	Resolved
3	No confirmation support interface	Low	Resolved
4	Unnecessary operation	Low	Resolved

COMPLETE ANALYSIS

CRITICAL | RESOLVED

No initialization.

In contract **TangleswapFactory**, in the ***initNonfungiblePositionManager,*** `nonfungiblePositionManager` must be initialized when NonfungiblePositionManager contract is deployed, and it should be be initialized only once, but it isn't initialized when NonfungiblePositionManager contract deployed and the owner of the factory contract can update the it anytime.

Recommendation:
Refactor the code to eliminate the centralization risk of being able to set the value multiple time, one possible way to achieve it's to refactor it so that nonFungiblePositionManager is initalized only one time in the TangleSwapFactory at the moment of deployment of the NonFungiblePositionManager.

MEDIUM | RESOLVED

No initialization.

The Factory owner can change state variables that affect the behavior of the contracts. These include calls to several functions like updating priceOracle, burn wallet and burn fee percent.

Recommendation:
As this is known to be called a Centralization Risk. The recommended way to deal with this is to consider using multisig wallets or having a governance module.

Note:
The client have stated the following: "Owner itself is a multisig wallet. When the TangleswapFactory contract is successfully deployed, it will be set to a multisig wallet by calling the setOwner method."

LOW | RESOLVED

No confirmation support interface

In contract **TangleswapPool**, in the ***setPriceOracle*** there are no confirmations support interface for address. If the provided address doesn't support the ***IPriceOracle*** interface,some method call will be reverted or the fallback function will be called which can lead to unpredictable behaviour.

```
priceOracle = newPriceOracle;
```

Recommendation:

```
priceOracle = IPriceOracle(newPriceOracle);
```

LOW | RESOLVED

Unnecessary operation

In contract **TangleswapPool**, in the ***increaseObservationCardinalityNext***, if the new ***observationCardinality*** is same with old ***observationCardinality***, ***observationCardinalityNext*** of ***slot0*** is not needed to update

```
slot0.observationCardinalityNext = observationCardinalityNextNew;
if (observationCardinalityNextOld != observationCardinalityNextNew)
    emit IncreaseObservationCardinalityNext(observationCardinalityNextOld, observationCardinalityNextNew);
```

Recommendation:

```
if (observationCardinalityNextOld != observationCardinalityNextNew)
    slot0.observationCardinalityNext = observationCardinalityNextNew;
    emit IncreaseObservationCardinalityNext(observationCardinalityNextOld, observationCardinalityNextNew);
```

	TangleswapFactory **TangleswapPool** **IUniswapV3Factory** **IUniswapV3Pool** **INonfungiblePositionManager** **IPriceOracle** **IUniswapV3PoolActions** **IUniswapV3PoolOwnerActions** **IUniswapV3PoolState**
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 (backdoors in the underlying ERC-20)	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Security

As a part of our work assisting TangleSwap in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Hardhat testing framework.

The tests were based on the functionality of the code, as well as a review of the TangleSwap contract requirements for details about issuance amounts and how the system handles these.

TangleswapFactory

- ✓ Should be deployed correctly (382ms)
- ✓ Should create a pool for the given two tokens and fee
- ✓ Should update the owner of the factory contract
- ✓ Should enable a fee amount with the given tickSpacing
- ✓ Should update burn wallet address
- ✓ Should update NonfungiblePositionManager address

TangleswapPool

- ✓ Should be deployed correctly (53ms)
- ✓ Should set the initial price for the pool
- ✓ Should increase the maximum number of price and liquidity observations that this pool will store
- ✓ Should add liquidity for the given recipient/tickLower/tickUpper position
- ✓ Should collect tokens owed to a position
- ✓ Should burn liquidity from the sender and account tokens owed for the liquidity to the position
- ✓ Should swap token0/token1, token1/token0
- ✓ Should set the denominator of the protocol's % share of the fees
- ✓ Should collect the protocol fee accrued to the pool
- ✓ Should change the priceOracle
- ✓ Should update maximunThreshold
- ✓ Should update burnFeePercent
- ✓ Should collect burn fee

19 passing (40s)

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	% UNCOVERED LINES
TangleswapFactory.sol	100	100	100	100	
TangleswapPool.sol	100	92,19	100	100	
FILE	**100**	**94.19**	**100**	**100**	

We are grateful for the opportunity to work with the TangleSwap 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 TangleSwap team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo