SMART CONTRACT AUDIT
Report 2 of 6: CLMM Periphery

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

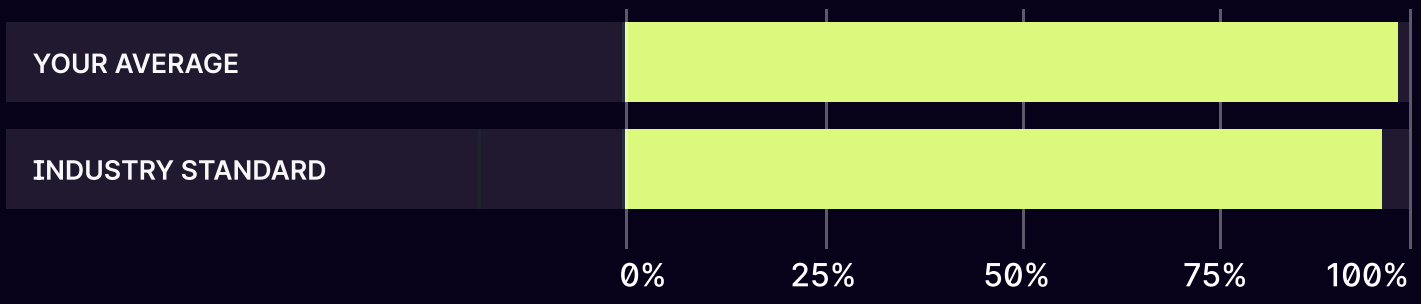
This document outlines the overall security of the TangleSwap smart contracts evaluated by the Zokyo Security team.

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

Contract Status

There were 0 critical issues found during the audit. (See in the Complete Analysis, started from 6 page)

Testable Code



96.25% 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-periphery

Last commit: 154f7e93143f92d72acd2a568e2c029dd46a395c

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

- NonfungiblePositionManager.sol
- SwapRouter.sol

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 Hardhat 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.	03	Testing contract logic against common and uncommon attack vectors.
02	Cross-comparison with other, similar smart contracts by industry leaders.	04	Thorough manual review of the codebase line by line.

Executive Summary

The Zokyo auditing team has run a deep investigation of the given code. The contracts are in good condition, well written and structured.

During the auditing process, there were one issue with medium severity and one issue with low severity found. The TangleSwap team resolved all issues. 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.

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	Users cannot mint positions due to incorrect variable value	Medium	Resolved
2	Improper setup can lead to unpredictable behavior	Low	Resolved

CRITICAL | RESOLVED

Users cannot mint positions due to incorrect variable value

The contract **NonfungiblePositionManager**, allows users to create pools of token pair and fees if it does not exist from the parent contract **PoolInitializer** using the function createAndInitializePoolIfNecessary(...). This function call is then passed to the TangleSwapFactory contract, which deploys a new pool with a deterministic address(see this code).

NonfungiblePositionManager contract also allows users to create positions in the pool and mint nfts (see mint function). This function call is pass on to the addLiquidity(...) function, which is inherited from the **LiquidityManagement** parent contract. Link

At line 6 in **PoolAddress** library, the variable **POOL_INIT_CODE_HASH** is set at compile time, which is the incorrect hash value of the byte code of the **TangleswapPool** contract. Due to the incorrect value, the mint function cannot compute the correct value of the deployed pool, and the transaction reverts because the pool does not exist at the calculated address. So, no user can create a mint NFT position using the periphery contract.

```
Created pool at address:  0x055682a29a140acd49de9dfaae0cf1358694e848
Expecting pool address:  0x85d0a94fdf174f4ca0110419cc55e67b75952eeb
    1) Test - Mint NFT position


  1 passing (1s)
  1 failing

  1) NonfungiblePositionManager
       Test - Mint NFT position:
     Error: Transaction reverted: function call to a non-contract account
```

Recommendation:
Update the value of **POOL_INIT_CODE_HASH** so that **NonfungiblePositionManager** can mint the pool positions correctly.

Existing value:

```
// SPDX-License-Identifier: GPL-2.0-or-later
pragma solidity >=0.5.0;

/// @title Provides functions for deriving a pool address from the factory, tokens, and the fee
library PoolAddress {
    bytes32 internal constant POOL_INIT_CODE_HASH = 0xe34f199b19b2b4f47f68442619d555527d244f78a3297ea89325f843f87b8b54;
```

Suggested value as per the Trangleswap git commit:

```
// SPDX-License-Identifier: GPL-2.0-or-later
pragma solidity >=0.5.0;

/// @title Provides functions for deriving a pool address from the factory, tokens, and the fee
UnitTest stub | dependencies | uml | draw.io
library PoolAddress {
    bytes32 internal constant POOL_INIT_CODE_HASH = 0x323a7b7d7a623fe4c6d9e3009f7c40fd7b77b9859f67b6c59d1c7241f6e89ff3;
```

LOW | RESOLVED

Improper setup can lead to unpredictable behavior.

In SwapRotuer, the swap fails if nonFungiblePositionManager is not set, the swap transactions revert if the nonfungiblePositionManager is TangleswapFactory not initialized by the owner of the contract. All swap transactions through SwapRotuer depend on TangleswapPool contract.

Recommendation:
Revert the contract calls with exact error message whenever **nonfungiblePositionManager** is expected to be initialized but is actually not set by the contract owner.

	NonfungiblePositionManager.sol	SwapRouter.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	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.

Test nonfungiblePositionManager

mint()
✓ should mint a new position as nft (448ms)
increaseLiquidity()
✓ Should increase liquidity (288ms)
collect()
✓ should collect token (228ms)
✓ collect tokens if reciepient is address zero (357ms)
decreaseLiquidity()
✓ successfully decrease liquidity (344ms)
✓ Should not decrease liquidity (299ms)
burn()
✓ Should burn token (401ms)
getApproved()
approved: 0x5FC8d32690cc91D4c39d9d3abcBD16989F875707
✓ gets approved address for token (172ms)
BaseURI()
baseURI:
✓ Should return baseURI
tokenURI()
✓ should return tokenURI
getIncrememntNonce
✓ Should returns next getIncrememntNonce (801ms)

SWAPROUTER

exactInputSingle()
✓ exactInputSingle (605ms)
exactOutputSingle()
✓ Should calculate correct exact output (239ms)

```
  exactInput()
  ✓ should calculate exact input (230ms)
  tangleswapSwapCallback()
  ✓ Should execute tangleswapSwapCallback successfully (271ms)
  tangleswapSwapCallback()
  ✓ Successfully execute tangleswapCallBack (248ms)
  ✓ Should revert if has multiple pools with but empty token address (290ms)
  exactOutput()
  ✓ should get exact Output (202ms)


18 passing (8s)
```

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

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	% UNCOVERED LINES
NonfungiblePositionManager.sol	95.83	89.47	100	96	
SwapRouter.sol	96.88	81.58	100	88.89	
All files	**96.25**	**85.53**	100	93.33	

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