Staykov Security

TSwap Audit Report

Version 1.0

Cyfrin.io

March 21, 2025

Staykov Audit Report

Staykov

March 21, 2025

Prepared by: Staykov Lead Auditors: - Staykov

Table of Contents

input and output, causing the user s to recieve incorrect amount of tokens … # MEDIUM ### [M-1] ’TSwapPool:deposit” is missing deadline check, causing transactions to complete even afther the deadline .. # LOWS ### [L-1] ’TSwapPool:LiquidityAdded’ event has parameted out of order causing event to emit incorrect information … ### [L-2] Default value returned by TSwapPool::SwapExactInput’ results in incorect return value given # INFORMATIONALS ### [I-11 error PoolFactory:: PoolFactory PoolDoesNotExist’ does not used and should be removed ### [I-2] Lacking zero checks … ### [I-3] {PoolFactory::createPool’ should use (symbol()), instead of (name() 2 times ### [I-4] Event is missing indexed fields

Protocol Summary

This project is to enter a raffle to win a cute dof NFT. 1. Call the `enterRaffle` function with the followin parameters : 1. `address[] participants`: A list of addresses that enter. You can use thus ti ebter yourself multiple times, or you and group of your friends. can use this to enter yourself multiple times, or yourself and a group of your friends. 1. Duplicate addresses are not allowed 1. Users are allowed to get a refund of their ticket & (value” if they call the ’refund function 2. Every X seconds, the raffle will be able to draw a winner and be minted a random puppy 3. The owner of the protocol will set a feeAddress to take a cut of the (value, and the rest of the funds will be sent to the winner of the puppy.

Disclaimer

The YOUR_NAME_HERE team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
	High	H	H/M	M

		Impact		
Likelihood	Medium	H/M	M	M/L
	Low	M	M/L	L

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Audit Details

- Commit Hash: e30d199697bbc822b646d76533b66b7d529b8ef5

Scope

```
.src.
---- PoolFactory.sol
---- TSwapPool.sol
```

Roles

-Owner -Player # Executive Summary Spend x hours for auditing this protocol. ## Issues found |Severity|Numbers of issues found| |——–|——————–| |High| | 3 |Medium| | 2 |Low| | 2 |Gas| | 2 |Info| | 9 |Total| | 16 ## Findings

HIGH

[H-1] Incorrect fee calculation in `TSwapPool::getInputAmountBasedOnOutput` causes protocol to take too many tokens from users

Description: `getInputAmountBasedOnOutput` function is intented to calculate the amount of tokens a user should deposit given an amount of output tokens . However the func miscalculates the resulting amount. Calculates with 10_000 instead of 1_000

Impact: A lot of losed fees

Proof of Concept:

Recommended Mitigation:

```
  return
-             ((inputReserves * outputAmount) * 10000) /
+             ((inputReserves * outputAmount) * 1000) /
             ((outputReserves - outputAmount) * 997);
    }
```

### [H-2] Lack of slippage protection in `TSwapPool::swapExcactOutput` causes users to potentialy recieve way fewer tokens

Description: The `swapExcactOutput` function does not include any sort of slippage protection. This func is simular to what is done in `TSwapPool::swapExcactInput` , where the function specifies a `minOutputAmount`, the `swapExactOutput` function should specify a `maxInputAmount`

Impact: If marked condition change before the transaction precesses the user could get a much worse swap.

Proof of Concept: 1. The price of WETH is 1,000 USDC 2. User inputs a `swapExcactOutput` looking for 1 WETH 3. The function does not offer max input amount 4. As the transaction is pending in the mempool, the market changes and the prices moves HUGE -> 1WETH is now 10_000USDC 5. The tx completes , but the user sent the protocol 10_000USDC instead of the expected 1_000 USDC

Recommended Mitigation: We should include `MaxInputAmount` to so user only has to spent up to specifix amount and can predict how much they will spent on the protocol.

```
+ uint256 MaxInputAmount;

+ if(inputAmount > MaxInputAmount) {
    revert();
}
```

[H-3] `TSwap::sellPoolTokens` missmatches input and output , causing the user s to recieve incorrect amount of tokens

Description: `sellPoolTokens` func is intended to allow users to easily sell pool tokens and recieve WETH in exchange. This is due to the fact that the `swapExactOutput` function is called, whereas the `swapExactInput` function is the one that should be called. Because users specify the exact amount of input tokens, not output.

Impact: Users will swap wrong amount of tokens, which is a severe dissruption of protocol functionality

Proof of Concept:

Recommended Mitigation: Consider changing the implementation to use `swapExactInput` instead of `swapExactOutput`. Will requie changing the `sellPoolTokens` func to accept a new parameter.

MEDIUM

[M-1] `TSwapPool:deposit` is missing deadline check, causing transactions to complete even afther the deadline

Description: The `deposit` functions accepts the deadline parameter, which , according to the documentation is `The deadline for the transaction to be completed by`. However this parameter is never used. Liquidity might be added to the pool at unexpected times , in marked conditions where deposite rate is unfavorable

Impact: Transactions could be sent when the market conditions are unfavorable , even when adding a deedline parameter

Proof of Concept: The `deadline` parameter is unused

Recommended Mitigation: Consider macking the following changes to the function

```
function deposit(
        uint256 wethToDeposit,
        uint256 minimumLiquidityTokensToMint,
        uint256 maximumPoolTokensToDeposit,
        uint64 deadline
    )
        external
+        revertIfDeadlinePassed(deadline)
        revertIfZero(wethToDeposit)
        returns (uint256 liquidityTokensToMint)
    {
```

LOWS

[L-1] `TSwapPool:LiquidityAdded` event has parameted out of order causing event to emit incorrect information

Description: When the `liquidityAdded` event is emitted in the `TSwapPool::_addLiquidityMintAndTran` function, it logs value in incorect order

Impact: offchain functions potentially can malfunction **Proof of Concept:**

Recommended Mitigation:

```
-  emit LiquidityAdded(msg.sender, poolTokensToDeposit, wethToDeposit);
+  emit LiquidityAdded(msg.sender, wethToDeposit, poolTokensToDeposit )
    ;
```

[L-2] Default value returned by `TSwapPool::swapExactInput` results in incorect return value given

Description: The `swapExactInput` function is expected to return the actual amount of tokens bought by the caller . While it deckares the named return value `output` it is never assigned a vakue nor uses explicit return statement

Impact: The return value will alwaus be zero , incorect info to caller

INFORMATIONALS

[I-1] error `PoolFactory::PoolFactory__PoolDoesNotExist` does not used and should be removed

```
- error PoolFactory__PoolDoesNotExist(address tokenAddress);
```

[I-2] Lacking zero checks

```
 constructor(address wethToken) {
+         if(wethToken == address(0)){
            revert()
            }
        i_wethToken = wethToken;
    }
```

[I-3] `PoolFactory::createPool` should use `.symbol()`, instead of `.name()` 2 times

```
- string memory liquidityTokenSymbol = string.concat("ts", IERC20(
   tokenAddress).name());
+ string memory liquidityTokenSymbol = string.concat("ts", IERC20(
   tokenAddress).symbol());
```

[I-4] Event is missing `indexed` fields