T-SWAP Audit Report

Version 1.0

Ramprasad

March 26, 2024

T-SWAP Audit Report

Ramprasad

march 26, 2024

Prepared by: Ramprasad Lead Auditors:

- Ramprasad

Table of Contents

- Medium
  - [M-1] `TSwapPool::deposit` is missing deadline check causing transactions to complete even after the deadline
- Low
  - [L-1] `TSwapPool::LiquidityAdded` event has parameters out of order
  - [L-2] Default value returned by `TSwapPool::swapExactInput` results in incorrect return value given.
- Informationals
  - [I-1] `PoolFactory::PoolFactory_PoolDoesNotExist` is not used and should be removed
  - [I-2] Lacking zero address check
  - [I-3] `PoolFactorr::createPool` should use `.symbol()` instead of `.name()`
- [I-4]: Event is missing `indexed` fields

Disclaimer

I 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
Likelihood	High	H	H/M	M
	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

The findings described in this document corresponded the following commmit hash:**

```
e643a8d4c2c802490976b538dd009b351b1c8dda
```

Scope

```
./src/
#-- PoolFactory.sol
#-- TSwapPool.sol
```

Roles

- Liquidity Providers: Users who have liquidity deposited into the pools. Their shares are represented by the LP ERC20 tokens. They gain a 0.3% fee every time a swap is made.
- Users: Users who want to swap tokens.

Executive Summary

Issues found

Severity	Number of issues found
High	3
Medium	1
Low	2
Information	4
Total	10

Findings

High

[H-1] Incorrect Fee calculation in `TSwapPool::getInputAmmountBasedOnOutput` causes protocall to take too many tokens from users, resulting in lost fees.

Description: The `getInputAmmountBasedOnOutput` function is intended to calsulate the ammount of tokens a user should deposit given an ammount of tokens of output tokens. However the function is currently miscalculating the resulting ammount. When calculating the fee it scals the ammount by 10_000 insted of 1_000.

Impact: Protocall takes more fee than expected from users .

Recommended Mitigation:

```
function getInputAmountBasedOnOutput(
        uint256 outputAmount,
        uint256 inputReserves,
        uint256 outputReserves
    )
        public
        pure
        revertIfZero(outputAmount)
        revertIfZero(outputReserves)
        returns (uint256 inputAmount)
    {
-       return ((inputReserves * outputAmount) * 10_000) / ((outputReserves - outputAmount) * 997);
+       return ((inputReserves * outputAmount) * 1_000) / ((outputReserves - outputAmount) * 997);
    }
```

[H-2] Lacck of slippage protection in `TSwapPoll::swapExactOutput` causes users to potentially receive way fewer tokens

Description: The `swapExactAmmount` function does not include any slippage protection. this function is exactly similar to what we have done in `TSwapPool::swapExactInput`, where the function specifies a `minOutputAmmount`, the `swapExactOutput` function should specify a `maxInputAmmount`

Impact: If the market conditions changes before the transaction process, the user could get a worst swap

Proof of Concept: 1. The price of 1 weth right now is 1,000 usdc. 2. user input a `swapExactOutput` looking for 1weth. 1. inputToken = usdc 2. outputToken = weth 3. outputAmmount = 1 4. deadline = whatever 3. The function does not offer a maxInput ammount. 4. As the transaction is pending in the mempool, the market changes! And the price moves huge -> 1weth is now 10_000 usdc. 10x more than the expected price. 5. The transaction completes, but the user send the protocall is 10,0000 USDC insted of expected 1000 USDC

Recommended Mitigation: We should include `maxInputAmmount` so the user so the user as only as to spend up to a specific ammount, and predict how much they will spend on protocol.

```
1
2  function swapExactOutput(
3          IERC20 inputToken,
4          IERC20 outputToken,
5  +   uint256 maxInputAmmount,
6  .
7  .
8  .
9   inputAmount = getInputAmountBasedOnOutput(outputAmount,inputReserves,
       outputReserves);
10 +    if(inputAmmpunt > maxInputAmmount) {
11 +     revert();
12 +     }
13
14           _swap(inputToken, inputAmount, outputToken, outputAmount);
15     }
```

[H-3] `TSwapPool::sellPoolTokens` mismatches input and output tokens causing users to receive the incorrect ammount of tokens

Description: The `sellTokens` function is intended to allow users to sell pool tokens and receive weth in exchange. Users indicate howmany pool tokens they are wiiling to sell in the `poolTokenAmmount` paramete. However the fnction is currently miscalculating the swapped ammount.

This is due to the fact that the `exactOutput` function is called, wheares the `swapExactInput` function is the one that should be called.Because users specify the exact ammount of input tokens not output.

Impact: Users will swap the wrong ammount of tokens, which is a sever disruption of protocol functionality

Recommended Mitigation: Consider changing the implimentation to use `swapExactInput` instead of `swapExactOutput`. Note that this would also require changing the `sellPoolTokens` functions to accept a new parameter (ie. `minWetToReceive` to be passed to `swapExactInput`)

```
    function sellPoolTokens(
        uint256 poolTokenAmount,
+        uint256 minWethReceive,
    ) external returns (uint256 wethAmount) {
-        return swapExactOutput(  i_poolToken, i_wethToken, poolTokenAmount, uint64(block.timestamp)
+           return swapExactInput(  i_poolToken,poolTokenAmmount, i_wethToken,minWethToReceive   poolTokenAmount, uint64(block.timestamp)

            );
    }
```

Additionally, it might be wise to add deadline to this function, as there is no deadline currently.

Medium

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

Description: The`deposit` function accepts a deadline parameter , which according to the documentation is “The dead line for the transaction is completed by”. However this parameter is never used. As a consiqence, operations that add liquidity might be executedt unexpected times, in market conditions where the deposit rate is unfavrable.

Impact: Transactions should be sent when the marcket conditions is unfavrable to deposit, even when adding a deadline parameter.

Proof of Concept: The `deadline` parameter is unused.

Recommended Mitigation: Consider making the following changes to the function

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

Low

[L-1] `TSwapPool::LiquidityAdded` event has parameters out of order

Description: When the `LiquidityAdded` event is emmited in the `TSwapPool::_addLiquidityMintAndTran` function, it logs values in an incorrect order. The `PoolTokenDeposit` value should go in the third position. wheares the `wethDeposit` value should go to the second position.

Impact: Event emmision is incorrect, leading to off-chain functions potentially mallfunctioning.

Recommended Mitigation:

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

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

Description: The `swapExactInput` function is expected to return the acctual ammount of tokens bought by the caller. However, it is never declares the named return value `output` it is never assigned a value, nor uses an exclipt return statement.

Impact: The return value is always be zero, giving incorrect information to the caller.

Recommended Mitigation:

```
    uint256 inputReserves = inputToken.balanceOf(address(this));
         uint256 outputReserves = outputToken.balanceOf(address(this));

-        uint256 outputAmount = getOutputAmountBasedOnInput(\
+        output = getOutputAmountBasedOnInput(
            inputAmount,
            inputReserves,
            outputReserves
        );

-        if (output < minOutputAmount) {
+         if (output < minOutput) {
-            revert TSwapPool__OutputTooLow(outputAmount, minOutputAmount);
+            revert TSwapPool__OutputTooLow(output, minOutput);
        }

-        _swap(inputToken, inputAmount, outputToken, outputAmount);
+        _swap(inputToken, inputAmount, outputToken, output);
```

Informationals

[I-1] `PoolFactory::PoolFactory_PoolDoesNotExist` is not used and should be removed

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

[I-2] Lacking zero address check

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

[I-3] `PoolFactorr::createPool` should use `.symbol()` instead of `.name()`

```
- 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

Index event fields make the field more quickly accessible to off-chain tools that parse events. However, note that each index field costs extra gas during emission, so it's not necessarily best to index the maximum allowed per event (three fields). Each event should use three indexed fields if there are three or more fields, and gas usage is not particularly of concern for the events in question. If there are fewer than three fields, all of the fields should be indexed.

- Found in src/PoolFactory.sol Line: 35

```
    event PoolCreated(address tokenAddress, address poolAddress);
```

- Found in src/TSwapPool.sol Line: 52

```
    event LiquidityAdded(
```

- Found in src/TSwapPool.sol Line: 57

```
1       event LiquidityRemoved(
```

- Found in src/TSwapPool.sol Line: 62

```
1       event Swap(
```