Prepared by: Michael Tang

Table of Contents

Protocol Summary

MemeSwap a decentralized exchange protocol that allows users to swap their memes and weth. The protocol is based on the Uniswap v2 architecture and uses a constant product formula to determine the price of each meme.

Issues found

Severtity	Number of issues found
High	3
Low	1
Info	3

Severtity	Number of issues found
Total	7

Findings

High

[H-1] Swap::deposit is missing deadline check, causing transaction to complete after a long time.

Description: According to the documentation of the protocol, `deadline` parameter should be taken into account when execute the `deposit` function. However, the current implementation does not check the deadline and allows the transaction to complete after a long time.

Impact: Transaction could be pending in the mempool for a long time and get processed at an unfavoourable time later, causing economic loss for operators.

Proof of Concept: The `deadline` parameter is not used.

Recommended Mitigation: Consider making following changes:

```
 1  function deposit(
 2        uint256 wethToDeposit,
 3        uint256 minimumLiquidityTokensToMint,
 4        uint256 maximumPoolTokensToDeposit,
 5        uint64 deadline
 6     )
 7        external
 8  +     revertIfDeadlinePassed(deadline)
 9        revertIfZero(wethToDeposit)
10        returns (uint256 liquidityTokensToMint)
11     {
```

[H-2] Miscalculation in Swap::getInputAmountBasedOnOutput, which leads to user spending more tokens than they should to deposit.

Description: The function is intended to calculate how much tokens user should spend to swap a specific amount of weth. However, the calculation is wrong by scaling the result by 10,000 instead of 1,000.

Impact: Swapper could spend more tokens than they should to swap certain amount of weth, which means they sell tokens at a lower price.

Recommended Mitigation:

```
1      function getInputAmountBasedOnOutput(
2          uint256 outputAmount,
3          uint256 inputReserves,
4          uint256 outputReserves
5      )
6          public
7          pure
8          revertIfZero(outputAmount)
9          revertIfZero(outputReserves)
10         returns (uint256 inputAmount)
11     {
12  -        return ((inputReserves * outputAmount) * 10_000) / ((
        outputReserves - outputAmount) * 997);
13  +        return ((inputReserves * outputAmount) * 1_000) / ((
        outputReserves - outputAmount) * 997);
14     }
```

[H-3] Missing slippage check in `Swap::swapExactOutput`, which leads to user falling victim to potential front-running attack.

Description: The function lacks slippage check and protection, which may expose transactions vulnerable to MEV attacks.

Recommended Mitigation:

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

Low

[L-1] Default value returned by `TSwapPool::swapExactInput` results in incorrect return value given

Description: The `swapExactInput` function is expected to return the actual amount of tokens bought by the caller. However, while it declares the named return value `ouput` it is never assigned a value, nor uses an explicit return statement.

Impact: The return value will always be 0, giving incorrect information to the caller.

Recommended Mitigation:

```
1        {
2            uint256 inputReserves = inputToken.balanceOf(address(this));
3            uint256 outputReserves = outputToken.balanceOf(address(this));
4
5 -          uint256 outputAmount = getOutputAmountBasedOnInput(inputAmount
      , inputReserves, outputReserves);
6 +          output = getOutputAmountBasedOnInput(inputAmount,
      inputReserves, outputReserves);
7
8 -          if (output < minOutputAmount) {
9 -              revert TSwapPool__OutputTooLow(outputAmount,
      minOutputAmount);
10 +          if (output < minOutputAmount) {
11 +              revert TSwapPool__OutputTooLow(outputAmount,
      minOutputAmount);
12          }
13
14 -          _swap(inputToken, inputAmount, outputToken, outputAmount);
15 +          _swap(inputToken, inputAmount, outputToken, output);
16        }
```

Informationals

[I-1] Lacking zero address checks

```
1        constructor(address wethToken) {
2 +          if(wethToken == address(0)) {
3 +              revert();
4 +          }
5          i_wethToken = wethToken;
6        }
```

[I-2] Event is missing `indexed` fields

Indexed fields in events can be quickly accessed by off-chain services. However, extra gas is required to store the indexed fields, so more context should be taken into account when deciding whether to use them.

[I-3] Function Swap::swapExactInput visibility should be restricted to external for gas-saving purpose, since it is not intended to be called by itself.