Prepared by: Michael Tang

Table of Contents

- Table of Contents
- Protocol Summary
- Issues found
- Findings

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:

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

[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:

```diff
    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-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:

```diff
+   error Swap__SlippageError(uint256 maxInputAmount, uint256
    inputAmount);
    function swapExactOutput(
        IERC20 inputToken,
+       uint256 maxInputAmount,
.
.
.
        inputAmount = getInputAmountBasedOnOutput(outputAmount,
            inputReserves, outputReserves);
+       if(inputAmount > maxInputAmount){
+           revert(Swap__SlippageError(maxInputAmount, inputAmount));
+       }
        _swap(inputToken, inputAmount, outputToken, outputAmount);
```

Low

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

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 explict return statement.

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

Recommended Mitigation:

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

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

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

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

Informationals

[I-1] Lacking zero address checks

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

[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.