High

[H-1] Incorrect fee calculation in `TSwapPool::getInputAmountBasedOnOutput` cause protocol to take too many tokens from users, resulting in loss of funds.

Description: The fee calculation logic in the `getInputAmountBasedOnOutput` function does not accurately account for the current state of the pool, leading to situations where users are charged excessive fees during swaps.

Impact: This can result in users losing funds unexpectedly, damaging the protocol's reputation and user trust.

```
function getOutputAmountBasedOnInput(
        uint256 inputAmount,
        uint256 inputReserves,
        uint256 outputReserves
    )
        public
        pure
        revertIfZero(inputAmount)
        revertIfZero(outputReserves)
        returns (uint256 outputAmount)
    {

        uint256 inputAmountMinusFee = inputAmount * 997;
        uint256 numerator = inputAmountMinusFee * outputReserves;
        uint256 denominator = (inputReserves * 1000) + inputAmountMinusFee;
        return numerator / denominator;
    }

    function getInputAmountBasedOnOutput(
        uint256 outputAmount,
        uint256 inputReserves,
        uint256 outputReserves
    )
        public
        pure
        revertIfZero(outputAmount)
        revertIfZero(outputReserves)
        returns (uint256 inputAmount)
    {
        return
            ((inputReserves * outputAmount) * 10000) /
            ((outputReserves - outputAmount) * 997);
    }
```

[H-2] Lack of Slippage Protection in `TSwapPool::swapExactOutput` causes users to potentially receive less output than expected.

Description: The `swapExactOutput` function does not include any slippage protection mechanisms, which means users may receive less output than they anticipated if the market conditions change unfavorably

during the transaction.

Impact: Users may be exposed to unexpected losses if the output amount falls below their expectations.

```
function swapExactOutput(
       IERC20 inputToken,
      IERC20 outputToken,
       uint256 outputAmount,
        uint64 deadline
    )
        public
        revertIfZero(outputAmount)
        revertIfDeadlinePassed(deadline)
        returns (uint256 inputAmount)
    {
        uint256 inputReserves = inputToken.balanceOf(address(this));
        uint256 outputReserves = outputToken.balanceOf(address(this));

        inputAmount = getInputAmountBasedOnOutput(
            outputAmount,
            inputReserves,
            outputReserves
        );

        _swap(inputToken, inputAmount, outputToken, outputAmount);
    }

    function sellPoolTokens(
        uint256 poolTokenAmount
    ) external returns (uint256 wethAmount) {
        return
            swapExactOutput(
             i_poolToken,
             i_wethToken,
             poolTokenAmount,
                uint64(block.timestamp)
            );
    }
```

[High-3] `TSwapPool::sellPoolTokens` mismatches input and output tokens causing users to receive incorrect amounts of tokens.

Description: The sellPoolTokens function calls swapExactOutput with the wrong input and output tokens, leading to users receiving incorrect amounts of tokens.

```
    function sellPoolTokens(
        uint256 poolTokenAmount
    ) external returns (uint256 wethAmount) {
        return
```

```
            swapExactOutput(
                i_poolToken,
                i_wethToken,
                poolTokenAmount,
                uint64(block.timestamp)
            );
```

Medium

[M-1] `TSwapPool::deposit` Deadline not being enforced

Description: The deposit function accepts a deadline parameter, but it is not being enforced, allowing users to deposit funds even after the deadline has passed.

Impact: A user who expects a deposit to fail will go through, leading to server disruption of functionality.

POC: A user can call the deposit function with a past deadline and still have their deposit processed.

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

Mitigation: Implement a check to revert the transaction if the deadline has passed.

Low

[L-1] `TSwapPool::LiquidityAdded` Incorrect Event Parameter Order in LiquidityAdded Event

Description: The LiquidityAdded event emits parameters in the wrong order, reporting poolTokensDeposited as wethDeposited and vice versa, which can mislead off-chain consumers and analytics.

Impact: This can lead to incorrect assumptions and calculations by off-chain services relying on the event data.

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

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

Description: The swapExactInput function returns 0 by default if the input token is neither the pool token nor WETH, which can lead to confusion and incorrect handling by users and integrators.

Impact: Users and integrators may not handle the 0 return value `output` correctly, leading to potential loss of funds or failed transactions.

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

        uint256 outputAmount = getOutputAmountBasedOnInput(
            inputAmount,
            inputReserves,
            outputReserves
        );

        if (outputAmount < minOutputAmount) {
            revert TSwapPool__OutputTooLow(outputAmount, minOutputAmount);
        }

        _swap(inputToken, inputAmount, outputToken, outputAmount);
```

Informational

[I-1] `TSwapPool.sol:swapExactInput` Missing external Visibility on swapExactInput Function

Description: The swapExactInput function is declared as public instead of external, potentially exposing it to unintended internal calls and increasing the attack surface.

```
function swapExactInput(
        IERC20 inputToken,
        uint256 inputAmount,
        IERC20 outputToken,
        uint256 minOutputAmount,
        uint64 deadline
    )
```

[I-2] `TSwapPool.sol:poolTokenReserves` Remove unnecessary poolTokenReserves assignment in deposit function

Description: Eliminates redundant variable assignment to optimize gas usage in the deposit logic.

```
uint256 poolTokenReserves = i_poolToken.balanceOf(address(this));
```

[I-3] `PoolFactory:PoolFactory__PoolDoesNotExist` Unused Custom Error Declaration

Description: The custom error PoolFactory__PoolDoesNotExist is declared but never used, leading to unnecessary code bloat and potential confusion for maintainers.

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

[I-4] `PoolFactory:constructor` Lack of 0 address check

```
 constructor(address wethToken) {
        i_wethToken = wethToken;
    }
```