Protocol Audit Report

2024-12-LAMBOWIN

Veselin Vachkov

December 11, 2024

Issues Found

Severity	**Number of Issues Found**
Medium	1
Total	**1**

Medium

Impact: Due to the lack of slippage parameter an MEV attacker could sandwich attack the swap to return fewer output tokens to the protocol than would otherwise be returned. Also, this can lead to significant losses due to price impact or slippage in volatile markets.

Proof of Concept

```
function _executeBuy(uint256 amountIn, uint256[] memory pools) internal
     {
        uint256 initialBalance = address(this).balance;

        // Execute buy
        require(IERC20(weth).approve(address(OKXTokenApprove), amountIn
            ), "Approve failed");
        uint256 uniswapV3AmountOut = IDexRouter(OKXRouter).
            uniswapV3SwapTo(
             uint256(uint160(address(this))),
             amountIn,
             //@audit `minReturn` in `uniswapV3SwapTo` should not be
                 hard coded as 0
             0,
             pools
        );
        VirtualToken(veth).cashOut(uniswapV3AmountOut);

        // SlowMist [N11]
        uint256 newBalance = address(this).balance - initialBalance;
        if (newBalance > 0) {
             IWETH(weth).deposit{value: newBalance}();
        }
    }
```

```
function _executeSell(uint256 amountIn, uint256[] memory pools)
    internal {
        IWETH(weth).withdraw(amountIn);
        VirtualToken(veth).cashIn{value: amountIn}(amountIn);
        require(IERC20(veth).approve(address(OKXTokenApprove), amountIn
            ), "Approve failed");
        //@audit `minReturn` in `uniswapV3SwapTo` should not be hard
            coded as 0
        IDexRouter(OKXRouter).uniswapV3SwapTo(uint256(uint160(address(
            this))), amountIn, 0, pools);
    }
```

```
function _getQuoteAndDirection(
        address tokenIn,
        address tokenOut,
        uint256 amountIn
    ) internal view returns (uint256 amountOut, uint256 directionMask)
        {
        (amountOut, , , ) = IQuoter(quoter).
            quoteExactInputSingleWithPool(
            IQuoter.QuoteExactInputSingleWithPoolParams({
                tokenIn: tokenIn,
                tokenOut: tokenOut,
                amountIn: amountIn,
                fee: fee,
                pool: uniswapPool,
                //@audit No Slippage Control: By hardcoding this value,
                     the function cannot prevent trades
                // that might execute at significantly worse prices
                    than intended.
                // Reduced Flexibility: Users cannot set a custom price
                     limit for trades,
                // which might be crucial in volatile markets or when
                    precise price boundaries are required.
                sqrtPriceLimitX96: 0
            })
            );
            directionMask = (tokenIn == weth) ? _BUY_MASK : _SELL_MASK;
        }
```

This code tells the swap that the user will accept a minimum amount of 0 output tokens from the swap, opening up the user to a catastrophic loss of funds via MEV bot sandwich attacks.

- All the issues are in the src/rebalance/LamboRebalanceOnUniwap.sol
- https://github.com/code-423n4/2024-12-lambowin/blob/main/src/rebalance/LamboRebalanceOnUniwap.sol
- lines 97, 113, 152

Hardcoding minReturn as 0 also opens the door for a front-running attack. If there is an attacker with higher gas prices, they might manipulate the transaction by executing their trade first, resulting in a reduced return for your transaction, leading to a loss. With a dynamic minReturn, you'd be ensuring that only trades within an acceptable price range are executed.

Typically, minReturn should be set dynamically based on the expected price range or the slippage tolerance that the USER is willing to accept. For example, if the USER expect the trade to return 100 tokens, setting a minReturn of 95 tokens (or 95%) ensures that the trade will only execute if you receive at least 95 tokens, protecting you from severe price changes.

Recommended mitigation steps

1. Set a Dynamic minReturn Value Based on Slippage Tolerance Instead of hardcoding 0, you should set minReturn based on a slippage tolerance that is determined dynamically. This means that you would calculate the acceptable minimum return based on your expected price and the percentage of slippage you're willing to tolerate.
2. Implement a Safety Check Before the Swap Implement a function that checks whether the expected output meets certain conditions before proceeding with the swap. For example, you could simulate the swap (or call a function like getAmountOut if available) to estimate the expected return and verify it: