Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2024.11.20, the SlowMist security team received the Lambo.Win team's security audit application for Lambo Virtual Liquidity, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

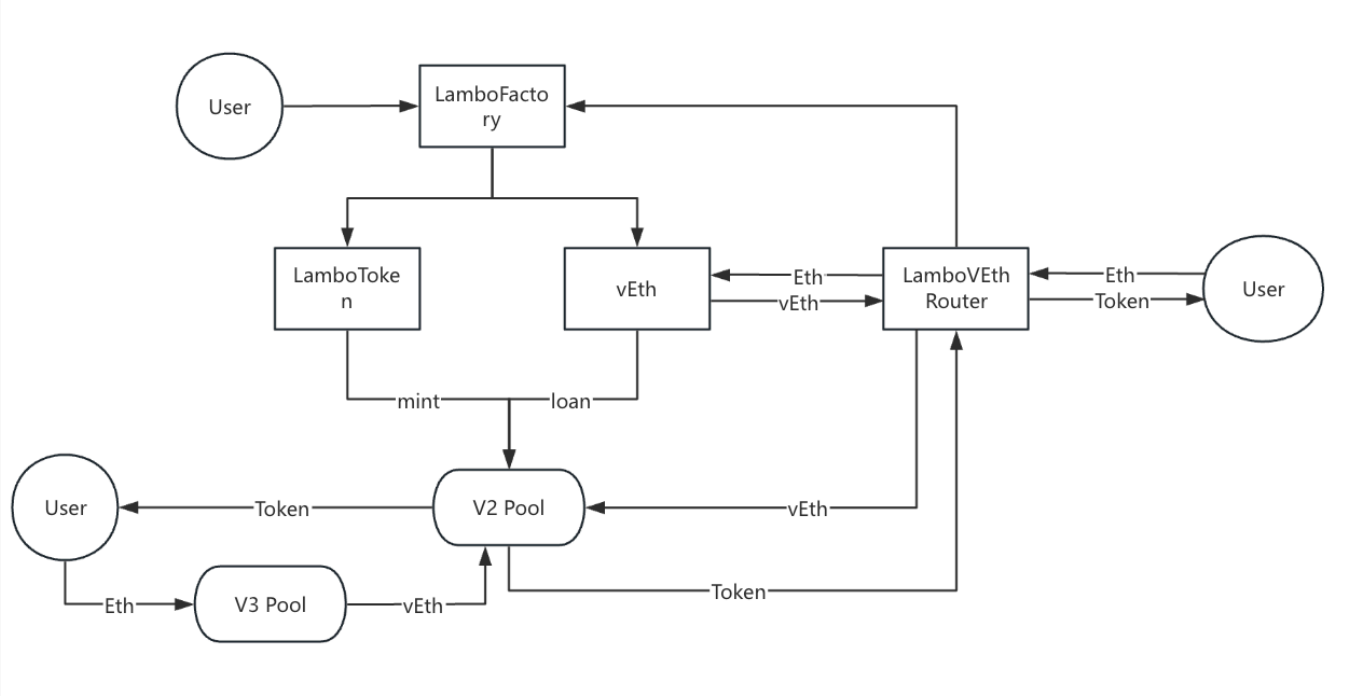
Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit
		Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

This is Lambo.Win's Virtual Liquidity protocol, which mainly includes Rebalance, Factory, Token and Router modules.



3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Redundant code	Others	Suggestion	Fixed
N2	Missing zero address check	Others	Suggestion	Fixed
N3	Missing event log	Others	Suggestion	Fixed
N4	Unchecked return value	Others	Suggestion	Fixed
N5	Risk of excessive authority	Authority Control Vulnerability Audit	Medium	Acknowledged
N6	Incorrect Amount Parameter in ERC20 CashIn Function	Design Logic Audit	High	Fixed
N7	Parameter type is not standardized	Others	Suggestion	Fixed
N8	The contract uniswapV2Factory contract address was not checked	Others	Suggestion	Fixed

NO	Title	Category	Level	Status
N9	Gas Optimization	Gas Optimization Audit	Suggestion	Fixed
N10	Insufficient minimum profit protection	Design Logic Audit	Low	Acknowledged
N11	Unintended ETH Balance Inclusion in WETH Conversion	Design Logic Audit	Low	Fixed
N12	Arbitrage preview results are inaccurate	Design Logic Audit	Low	Acknowledged
N13	Recommendation to add reentrancy protection	Reentrancy Vulnerability	Suggestion	Fixed
N14	Deprecated ETH Transfer Method	Others	Suggestion	Fixed
N15	Function not yet used	Others	Information	Acknowledged

4 Code Overview

4.1 Contracts Description

https://github.com/Lambo-Win/Lambo-Virtual-Liquidity

Initial audit version: f75629046242cd91596e15778a93f3e0d3aac3e3

Final audit version: 01614566c687266780d64158c1964f4df1b7db98

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

LamboVEthRouter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
updateFeeRate	External	Can Modify State	onlyOwner
createLaunchPadAndInitialBuy	Public	Payable	-
getBuyQuote	Public	-	-
getSellQuote	Public	-	-
buyQuote	Public	Payable	-
sellQuote	Public	Can Modify State	-
_sellQuote	Internal	Can Modify State	-
_buyQuote	Internal	Can Modify State	-
<Receive Ether>	External	Payable	-

LamboRebalanceOnUniwap			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
_authorizeUpgrade	Internal	Can Modify State	onlyOwner
extractProfit	External	Can Modify State	onlyOwner
rebalance	External	Can Modify State	-
onMorphoFlashLoan	External	Can Modify State	-
_executeBuy	Internal	Can Modify State	-
_executeSell	Internal	Can Modify State	-
previewRebalance	Public	-	-
_getTokenBalances	Internal	-	-

LamboRebalanceOnUniwap			
_getTokenInOut	Internal	-	-
_getQuoteAndDirection	Internal	-	-
<Receive Ether>	External	Payable	-

VirtualToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20 Ownable
isValidFactory	External	-	-
updateFactory	External	Can Modify State	onlyOwner
addToWhiteList	External	Can Modify State	onlyOwner
removeFromWhiteList	External	Can Modify State	onlyOwner
cashIn	External	Payable	onlyWhiteListed
cashOut	External	Can Modify State	onlyWhiteListed
takeLoan	External	Payable	onlyValidFactory
repayLoan	External	Can Modify State	onlyValidFactory
getLoanDebt	External	-	-
_increaseDebt	Internal	Can Modify State	-
_decreaseDebt	Internal	Can Modify State	-
_transferAssetFromUser	Internal	Can Modify State	-
_transferAssetToUser	Internal	Can Modify State	-
_update	Internal	Can Modify State	-

LamboToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
initialize	Public	Can Modify State	-
name	Public	-	-
symbol	Public	-	-
decimals	Public	-	-
totalSupply	Public	-	-
balanceOf	Public	-	-
transfer	Public	Can Modify State	-
allowance	Public	-	-
approve	Public	Can Modify State	-
transferFrom	Public	Can Modify State	-
_transfer	Internal	Can Modify State	-
_update	Internal	Can Modify State	-
_mint	Internal	Can Modify State	-
_burn	Internal	Can Modify State	-
_approve	Internal	Can Modify State	-
_approve	Internal	Can Modify State	-
_spendAllowance	Internal	Can Modify State	-

LamboFactory			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable

LamboFactory			
addVTokenWhiteList	Public	Can Modify State	onlyOwner
removeVTokenWhiteList	Public	Can Modify State	onlyOwner
_deployLamboToken	Internal	Can Modify State	-
createLaunchPad	Public	Can Modify State	onlyWhiteListed

4.3 Vulnerability Summary

[N1] [Suggestion] Redundant code

Category: Others

Content

1.The LamboToken contract imported the Ownable contract of Openzeppelin, but it was not used in the contract. In addition, the LamboToken contract was created through the Clones contract of Openzeppelin. The contract created by this method cannot execute the `constructor` function, so the Ownable contract cannot be initialized. And the `_burn` function in the contract is not used.

- src/LamboToken.sol#L10, L30, L230-L235

```
    import {Ownable} from
"../node_modules/@openzeppelin/contracts/access/Ownable.sol";

    constructor() Ownable(msg.sender) {}

    function _burn(address account, uint256 value) internal {
        if (account == address(0)) {
            revert ERC20InvalidSender(address(0));
        }
        _update(account, address(0), value);
    }
```

2.In the VirtualToken contract, the `totalCashOutFeesCollected` parameter and the `Withdraw` event are not used.

- src/VirtualToken.sol#L13, L24

```
    uint256 public totalCashOutFeesCollected;
    event Withdraw(address owner, uint256 amount);
```

3.In the LamboVEthRouter contract, the `_sellQuote` function repeatedly checks whether the `amountXOut` parameter is greater than or equal to the `minReturn` parameter. It only needs to check the `amountXOut` after deducting the fee.

- src/LamboVEthRouter.sol#L117-L168

```
    function _sellQuote(
        address quoteToken,
        uint256 amountYIn,
        uint256 minReturn
    )   internal
        // ...
        require(amountXOut >= minReturn, "Insufficient output amount");
        // ...
        amountXOut = amountXOut - fee;

        // handle amountOut
        require(amountXOut >= minReturn, "MinReturn Error");
        // ...
    }
```

4.In the LamboRebalanceOnUniswp contract, the `MIN_SQRT_RATIO` parameter and the `MAX_SQRT_RATIO` parameter are not used.

- src/rebalance/LamboRebalanceOnUniwap.sol#L21-L22

```
    uint160 internal constant MIN_SQRT_RATIO = 4295128739;
    uint160 internal constant MAX_SQRT_RATIO =
1461446703485210103287273052203988822378723970342;
```

Solution

It is recommended to delete the redundant code.

Status

Fixed

[N2] [Suggestion] Missing zero address check

Category: Others

Content

1.In the LamboFactory contract, the `constructor` function lacks a zero address check for the `_lamboTokenImplementation` parameter.

- src/LamboFactory.sol#L24-L26

```
    constructor(address _lamboTokenImplementation) Ownable(msg.sender) {
        lamboTokenImplementation = _lamboTokenImplementation;
    }
```

2.In the VirtualToken contract, the `constructor` function lacks a zero address check for the `_underlyingToken` parameter.

- src/VirtualToken.sol#L38-L44

```
    constructor(
        string memory name,
        string memory symbol,
        address _underlyingToken
    ) ERC20(name, symbol) Ownable(msg.sender) {
        underlyingToken = _underlyingToken;
    }
```

3.In the LamboVEthRouter contract, the `constructor` function lacks zero address checks for the `_vETH` parameter and the `_uniswapV2Factory` parameter.

- src/LamboVEthRouter.sol#L23-L28

```
    constructor(address _vETH, address _uniswapV2Factory, address _multiSign)
Ownable(_multiSign) public {
        feeRate = 100;

        vETH = _vETH;
        uniswapV2Factory = _uniswapV2Factory;
    }
```

4.In the LamboRebalanceOnUniswap contract, the `initialize` function lacks zero address checks for the `_owner`, `_vETH`, and `_uniswap` parameters.

- src/rebalance/LamboRebalanceOnUniwap.sol#L34-L40

```
    function initialize(address _owner, address _vETH, address _uniswap, uint24 _fee)
public initializer {
        __Ownable_init(_owner);

        fee = _fee;
        veth = _vETH;
        uniswapPool = _uniswap;
    }
```

Solution

It is recommended to add a zero address check.

Status

Fixed

[N3] [Suggestion] Missing event log

Category: Others

Content

1.In the LamboFactory contract, the `Owner` role can add or delete the token whitelist whiteList through the `addVTokenWhiteList` function and `removeVTokenWhiteList` function, but the function lacks event records.

- src/LamboFactory.sol#L33-L36, L37-L39

```
    function addVTokenWhiteList
    function removeVTokenWhiteList
```

2.In the VirtualToken contract, the `updateFactory` function, `addToWhiteList` function, and `removeFromWhiteList` function are used to modify important parameters in the contract, but event records are missing in the functions.

- src/VirtualToken.sol#L50-L52, L54-L56, L58-L60

```
    function updateFactory
    function addToWhiteList
    function removeFromWhiteList
```

Solution

It is recommended to add event logging.

Status

Fixed

[N4] [Suggestion] Unchecked return value

Category: Others

Content

1.In the LamboFactory contract, the `createLaunchPad` function did not check the return value when calling the `transfer` function of the `quoteToken` contract and the `pool` contract.

- src/LamboFactory.sol#L51-L69

```
    function createLaunchPad(
        string memory name,
        string memory tickname,
        uint256 virtualLiquidityAmount,
        address virtualLiquidityToken
    ) public onlyWhiteListed(virtualLiquidityToken) returns (address quoteToken,
address pool) {
        quoteToken = _deployLamboToken(name, tickname);
        pool =
IPoolFactory(LaunchPadUtils.UNISWAP_POOL_FACTORY_).createPair(virtualLiquidityToken,
quoteToken);
        // ...
        IERC20(quoteToken).transfer(pool, LaunchPadUtils.TOTAL_AMOUNT_OF_QUOTE_TOKEN);
        // ...
        IERC20(pool).transfer(address(0), IERC20(pool).balanceOf(address(this)));
        // ...
    }
```

2.In the VirtualToken contract, the `_transferAssetFromUser` function and the `_transferAssetToUser` function did not check the return value when calling the `transfer` function and `transferFrom` function of the `underlyingToken` token contract.

- src/VirtualToken.sol#L110-L116, L118-L126

```
    function _transferAssetFromUser(uint256 amount) internal {
        // ...
```

```
            IERC20(underlyingToken).transferFrom(msg.sender, address(this), amount);
        // ...
    }

    function _transferAssetToUser(uint256 amount) internal {
        // ...
            IERC20(underlyingToken).transfer(msg.sender, amount);
        // ...
    }
```

3.In the LamboRebalanceOnUniswp contract, the `extractProfit` function, `onMorphoFlashLoan` function, `_executeBuy` function, and `_executeSell` function did not check the return value of the function when calling the `transfer` function and `approve` function of the ERC20 token contract.

- src/rebalance/LamboRebalanceOnUniwap.sol#L44-L49, L60-L76, L78-L83, L85-L90

```
    function extractProfit(address to, address token) external onlyOwner {
        // ...
        if (balance > 0) {
            IERC20(token).transfer(to, balance);
        }
    }

    function onMorphoFlashLoan(uint256 assets, bytes calldata data) external {
        // ...
        IERC20(weth).approve(address(morphoVault), assets);
    }

    function _executeBuy(uint256 amountIn, uint256[] memory pools) internal {
        IERC20(weth).approve(address(OKXTokenApprove), amountIn);
        // ...
    }

    function _executeSell(uint256 amountIn, uint256[] memory pools) internal {
        // ...
        IERC20(veth).approve(address(OKXTokenApprove), amountIn);
        // ...
    }
```

Solution

It is recommended to check the return value of the function and use the safeTransfer function or safeTransferFrom function instead of the transfer function.

Status

Fixed

[N5] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

1.In the LamboFactory contract, the `Owner` role can add or delete the token whitelist whiteList through the `addVTokenWhiteList` function and `removeVTokenWhiteList` function.

- src/LamboFactory.sol#L33-L35, L37-L39

```
    function addVTokenWhiteList
    function removeVTokenWhiteList
```

2.The VirtualToken contract grants the `Owner` role administrative control over two key mappings: `validFactories` for managing approved factory addresses, and `whiteList` for controlling user access permissions. These can be modified through the `updateFactory`, `addToWhiteList`, and `removeFromWhiteList` functions respectively.

- src/VirtualToken.sol#L50-L52, L54-L56, L58-L60

```
    function updateFactory
    function addToWhiteList
    function removeFromWhiteList
```

3.In the VirtualToken contract, the `WhiteListed` role can exchange `underlyingToken` and VirtualToken through the `cashIn` function and `cashOut` function.

- src/VirtualToken.sol#L62-L66, L68-L72

```
    function cashIn
    function cashOut
```

4.In the VirtualToken contract, the `ValidFactory` role can loan VirtualToken through the `takeLoan` function.

- src/VirtualToken.sol#L74-L86

```
function takeLoan
```

5.In the LamboVEthRouter contract, the `Owner` role can modify the `feeRate` parameter through the `updateFeeRate` function.

- src/LamboVEthRouter.sol

```
function updateFeeRate
```

6.The LamboRebalanceOnUniwap contract uses OpenZeppelin's UUPSUpgradeable upgradeable contract module. The contract `Owner` has the authority to upgrade the contract, which may cause permission issues.

- src/rebalance/LamboRebalanceOnUniwap.sol#L4, L42

```
import "@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol";

function _authorizeUpgrade
```

7.In the LamboRebalanceOnUniswap contract, the `Owner` role can transfer tokens in the contract through the `extractProfit` function, which is mainly used to withdraw profits in the contract.

- src/rebalance/LamboRebalanceOnUniwap.sol#L44-L49

```
function extractProfit
```

Solution

In the short term, during the early stages of the project, the protocol may need to frequently set various parameters to ensure the stable operation of the protocol. Therefore, transferring the ownership of core roles to a multisig management can effectively solve the single-point risk, but it cannot mitigate the excessive privilege risk. In the long run, after the protocol stabilizes, transferring the owner ownership to community governance or executing through a timelock can effectively mitigate the excessive privilege risk and increase the community users' trust in the protocol.

Status

Acknowledged; According to the project team, privileged roles in the contract will be managed through a multi-signature wallet system to enhance security governance.

[N6] [High] Incorrect Amount Parameter in ERC20 CashIn Function

Category: Design Logic Audit

Content

The `cashIn` function in the VirtualToken contract has a vulnerability when processing ERC20 token conversion. When the `underlyingToken` is not ETH but an ERC20 token, the function still uses `msg.value` as the transfer amount to pass to the `_transferAssetFromUser` function, while `msg.value` is only meaningful in ETH transactions, which causes the ERC20 token conversion function to not work properly.

- src/VirtualToken.sol#L62-L66, L110-L116

```
    function cashIn() external payable onlyWhiteListed {
        _transferAssetFromUser(msg.value);
        _mint(msg.sender, msg.value);
        emit CashIn(msg.sender, msg.value);
    }

    function _transferAssetFromUser(uint256 amount) internal {
        if (underlyingToken == LaunchPadUtils.NATIVE_TOKEN) {
            require(msg.value >= amount, "Invalid ETH amount");
        } else {
            IERC20(underlyingToken).transferFrom(msg.sender, address(this), amount);
        }
    }
```

Solution

It is recommended to modify the cashIn function to add an amount parameter for ERC20 transfers, and handle them separately according to the underlying Token type.

Status

Fixed

[N7] [Suggestion] Parameter type is not standardized

Category: Others

Content

In the VirtualToken contract, the `underlyingToken` parameter should be immutable.

- src/VirtualToken.sol#L11

```
    address public underlyingToken;
```

Solution

It is recommended to set the parameter type in a standardized way.

Status

Fixed

[N8] [Suggestion] The contract uniswapV2Factory contract address was not checked

Category: Others

Content

In the LamboVEthRouter contract, the constructor function did not check whether the `_uniswapV2Factory` parameter was equal to the `UNISWAP_POOL_FACTORY_` parameter in the LaunchPadUtils library.

- src/LamboVEthRouter.sol#L23-L28

```
    constructor(address _vETH, address _uniswapV2Factory, address _multiSign)
Ownable(_multiSign) public {
        feeRate = 100;

        vETH = _vETH;
        uniswapV2Factory = _uniswapV2Factory;
    }
```

Solution

It is recommended to directly use the UNISWAP_POOL_FACTORY_ parameter in the LaunchPadUtils library as the address of the uniswapV2Factory parameter.

Status

Fixed

[N9] [Suggestion] Gas Optimization

Category: Gas Optimization Audit

Content

In the LamboVEthRouter contract, the `_sellQuote` and `_buyQuote` functions use `assert` to check the return

value of the transfer function, which is inappropriate because a failed `assert` will consume all remaining gas and roll back the transaction, which will cause unnecessary gas loss for users, especially when there are a large number of transactions or when the gas price is high.

- src/LamboVEthRouter.sol#L117-L168, L171-L215

```
    function _sellQuote(
        address quoteToken,
        uint256 amountYIn,
        uint256 minReturn
    )   internal
        returns (uint256 amountXOut) {
        // ...
        assert(IERC20(quoteToken).transfer(pair, amountYIn));
        // ...
    }


    function _buyQuote(
        address quoteToken,
        uint256 amountXIn,
        uint256 minReturn
    )
        internal
        returns (uint256 amountYOut)
    {
        // ...
        assert(VirtualToken(vETH).transfer(pair, amountXIn));
        // ...
    }
```

Solution

It is recommended to replace the assert statements in the _sellQuote and _buyQuote functions with require statements to check the return value of transfer.

Status

Fixed

[N10] [Low] Insufficient minimum profit protection

Category: Design Logic Audit

Content

In the LamboRebalanceOnUniwap contract, the profit check in `require(profit > 0, "No profit made")` in the `rebalance` function is too simple, and only verifies whether the profit is greater than 0. Because gas fees are required for each transaction, even if a positive profit of 1 wei is obtained, it may be a net loss after considering the gas cost. Especially when the Ethereum network is congested and gas prices are high, a seemingly profitable transaction may actually result in a significant loss.

- src/rebalance/LamboRebalanceOnUniwap.sol#L51-L58

```
    function rebalance(uint256 directionMask, uint256 amountIn, uint256 amountOut)
external {
        uint256 balanceBefore = IERC20(weth).balanceOf(address(this));
        bytes memory data = abi.encode(directionMask, amountIn, amountOut);
        IMorpho(morphoVault).flashLoan(weth, amountIn, data);
        uint256 balanceAfter = IERC20(weth).balanceOf(address(this));
        uint256 profit = balanceAfter - balanceBefore;
        require(profit > 0, "No profit made");
    }
```

Solution

It is recommended to calculate the actual gas cost when executing a transaction and set a reasonable minimum profit threshold to ensure that the net profit after deducting the gas cost reaches the preset standard.

Status

Acknowledged; According to the project team, the RebalanceOnUniswap contract is designed to maintain the VETH/WETH pool ratio at 1:1 rather than for profit. Gas costs are intentionally omitted to increase rebalancing frequency, accepting gas losses as a trade-off for improved price stability.

[N11] [Low] Unintended ETH Balance Inclusion in WETH Conversion

Category: Design Logic Audit

Content

In the `_executeBuy` function of the LamboRebalanceOnUniswp contract, the last step `IWETH(weth).deposit{value: address(this).balance}()` converts all ETH balances in the contract into WETH. However, if the contract already has ETH balances before executing this function, these original ETH will also

be converted, resulting in the final amount of WETH obtained being greater than the amount generated by the actual transaction, which may lead to errors in profit calculation.

- src/rebalance/LamboRebalanceOnUniwap.sol#L78-L83

```
    function _executeBuy(uint256 amountIn, uint256[] memory pools) internal {
        IERC20(weth).approve(address(OKXTokenApprove), amountIn);
        uint256 uniswapV3AmountOut =
IDexRouter(OKXRouter).uniswapV3SwapTo(uint256(uint160(address(this))), amountIn, 0,
pools);
        VirtualToken(veth).cashOut(uniswapV3AmountOut);
        IWETH(weth).deposit{value: address(this).balance}();
    }
```

Solution

It is recommended to record the initial ETH balance at the beginning of the function, and convert only the newly added ETH balance (current balance minus initial balance) to WETH at the end, rather than converting all ETH in the contract.

Status

Fixed

[N12] [Low] Arbitrage preview results are inaccurate

Category: Design Logic Audit

Content

In the LamboRebalanceOnUniswap contract, the `previewRebalance` function only determines arbitrage opportunities by `amountOut > amountIn`, ignoring the gas cost required for transaction execution, which may cause expected profitable transactions to actually lose money after deducting the gas cost. Secondly, due to the characteristics of blockchain transactions, there is a time difference between preview price and actual execution. During this period, market prices may change due to natural fluctuations or the operations of other traders, resulting in a mismatch between the predicted results and the actual execution results.

- src/rebalance/LamboRebalanceOnUniwap.sol#L92-L98

```
    function previewRebalance() public view returns (bool result, uint256
directionMask, uint256 amountIn, uint256 amountOut) {
```

```
        address tokenIn;
        address tokenOut;
        (tokenIn, tokenOut, amountIn) = _getTokenInOut();
        (amountOut, directionMask) = _getQuoteAndDirection(tokenIn, tokenOut,
amountIn);
        result = amountOut > amountIn;
    }
```

Solution

It is recommended to implement gas cost calculation and consider the risk of price deviation between preview and execution time to improve accuracy.

Status

Acknowledged; According to the project team, the RebalanceOnUniswap contract is designed to maintain the VETH/WETH pool ratio at 1:1 rather than for profit. Gas costs are intentionally omitted to increase rebalancing frequency, accepting gas losses as a trade-off for improved price stability.

[N13] [Suggestion] Recommendation to add reentrancy protection

Category: Reentrancy Vulnerability

Content

1.In the createLaunchPad function of the createLaunchPad contract, there is an external contract call, which may lead to reentry risks, but no anti-reentry lock is added.

- src/LamboFactory.sol#L51-L69

```
    function createLaunchPad
```

2.In the `createLaunchPadAndInitialBuy` function, `buyQuote` function, and `sellQuote` function of the LamboVEthRouter contract, there are external contract calls and ETH transfers, which may lead to reentry risks, but no anti-reentry lock is added.

- src/LamboVEthRouter.sol#L36-L56, L91-L101, L103-L112

```
    function createLaunchPadAndInitialBuy
    function buyQuote
    function sellQuote
```

3.In the `rebalance` function of the LamboRebalanceOnUniwap contract, there are external contract calls and ETH transfers, which may lead to reentry risks, but no anti-reentry lock is added.

- src/rebalance/LamboRebalanceOnUniwap.sol#L51-L58

```
    function rebalance
```

Solution

It is recommended to add an anti-reentry lock to avoid reentry risks.

Status

Fixed

[N14] [Suggestion] Deprecated ETH Transfer Method

Category: Others

Content

In the LamboVEthRouter contract, the `_buyQuote` function uses the deprecated `transfer` method for ETH transactions.

- src/LamboVEthRouter.sol#L171-L215

```
    function _buyQuote(
        address quoteToken,
        uint256 amountXIn,
        uint256 minReturn
    )
        internal
        returns (uint256 amountYOut)
    {
        // ...
        if (msg.value > (amountXIn + fee + 1)) {
            payable(msg.sender).transfer(msg.value - amountXIn - fee - 1);
        }
        // ...
    }
```

Solution

It's recommended to replace it with the more secure call method for ETH transfers, implementing proper checks for

the return value.

Status

Fixed

[N15] [Information] Function not yet used

Category: Others

Content

In the VirtualToken contract, the `repayLoan` function and the `_decreaseDebt` function are not used yet.

- src/VirtualToken.sol#L91-L95, L105-L108

```
    function repayLoan(address to, uint256 amount) external onlyValidFactory {
        _decreaseDebt(to, amount);
        _burn(to, amount);
        emit LoanRepaid(to, amount);
    }

    function _decreaseDebt(address user, uint256 amount) internal {
        require(_debt[user] >= amount, "Decrease amount exceeds current debt");
        _debt[user] -= amount;
    }
```

Solution

It is recommended to delete the redundant code.

Status

Acknowledged; According to the project team, the repayLoan function and the _decreaseDebt function will be used in subsequent versions.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002411250002	SlowMist Security Team	2024.11.20 - 2024.11.25	Medium Risk

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 2 high risk, 1 medium risk, 2 low risk, 9 suggestion, 1 Information.

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist