Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2024.05.21, the SlowMist security team received the MYX team's security audit application for MYX Protocol Phase3, 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 an iterative audit of the MYX Protocol, focusing on changes to the fee collection module, pool module and the addition of the new oracle machine module.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	The overflow error in the givebackTradingFee function	Design Logic Audit	Critical	Fixed

NO	Title	Category	Level	Status
N2	Missing event record	Others	Suggestion	Fixed
N3	Potential bad debt risk	Design Logic Audit	Low	Acknowledged

4 Code Overview

4.1 Contracts Description

Audit Version:

https://github.com/d11-myx/myx-contracts

commit: 59a185fc1736bb58435ab385c7ddc1c92d9366c9

Fixed Version:

https://github.com/d11-myx/myx-contracts

commit: 614d217b419bd51709a0c74fb09506f1b0356d66

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:

FeeCollector			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
getTradingFeeTier	External	-	-
getRegularTradingFeeTier	External	-	-
getKeeperNetworkFee	External	-	-

FeeCollector			
updatePositionManagerAddress	External	Can Modify State	onlyPoolAdmin
updateExecutionLogicAddress	External	Can Modify State	onlyPoolAdmin
updateStakingPoolAddress	External	Can Modify State	onlyPoolAdmin
updateTradingFeeTiers	External	Can Modify State	onlyPoolAdmin
updateTradingFeeTier	External	Can Modify State	onlyPoolAdmin
updateMaxReferralsRatio	External	Can Modify State	onlyPoolAdmin
claimStakingTradingFee	External	Can Modify State	onlyStakingPool
claimTreasuryFee	External	Can Modify State	onlyTreasury
claimReferralFee	External	Can Modify State	nonReentrant
claimUserTradingFee	External	Can Modify State	nonReentrant
claimKeeperTradingFee	External	Can Modify State	nonReentrant
claimKeeperNetworkFee	External	Can Modify State	nonReentrant
distributeTradingFee	External	Can Modify State	onlyPositionManagerOrLogic
distributeNetworkFee	External	Can Modify State	onlyPositionManagerOrLogic
_collectTradingFee	Internal	Can Modify State	-
_updateTradingFeeTier	Internal	Can Modify State	-
rescueKeeperNetworkFee	External	Can Modify State	nonReentrant onlyAdmin

PositionManager			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setRouter	External	Can Modify State	onlyPoolAdmin

PositionManager			
increasePosition	External	Can Modify State	onlyExecutor
decreasePosition	External	Can Modify State	onlyExecutor
adjustCollateral	External	Can Modify State	onlyRouter
updateFundingRate	External	Can Modify State	onlyRouter
_takeFundingFeeAddTraderFee	Internal	Can Modify State	-
_settleLPPosition	Internal	Can Modify State	-
_calLpProfit	Internal	Can Modify State	-
_handleCollateral	Internal	Can Modify State	-
_regularTradingFee	Internal	-	-
_updateFundingRate	Internal	Can Modify State	-
_nextFundingRate	Internal	-	-
getTradingFee	Public	-	-
getFundingFee	Public	-	-
getCurrentFundingRate	External	-	-
getNextFundingRate	External	-	-
getNextFundingRateUpdateTime	External	-	-
needADL	External	-	-
needLiquidation	Public	-	-
getExposedPositions	Public	-	-
getPosition	Public	-	-
getPositionByKey	Public	-	-
getPositionKey	Public	-	-

PositionManager			
_max	Internal	-	-
_min	Internal	-	-

PythOraclePriceFeedV2			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
updateExecutorAddress	External	Can Modify State	onlyPoolAdmin
updatePriceAge	External	Can Modify State	onlyPoolAdmin
updatePythAddress	External	Can Modify State	onlyPoolAdmin
setTokenPriceIds	External	Can Modify State	onlyPoolAdmin
updatePrice	External	Payable	-
_getMinMax	Internal	-	-
updateHistoricalPrice	External	Payable	onlyExecutor onlyBacktracking
removeHistoricalPrice	External	Can Modify State	onlyExecutor onlyBacktracking
getHistoricalPrice	External	-	onlyBacktracking
getPrice	External	-	-
getPriceSafely	External	-	-
_getPriceId	Internal	-	-
_returnPriceWithDecimals	Internal	-	-
_setTokenPriceIds	Internal	Can Modify State	-
decimals	Public	-	-

UiPositionDataProvider			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getPositionsData	Public	-	-
getUserPositionData	External	-	-
getUserPositionDataV2	External	-	-

Pool			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
<Receive Ether>	External	Payable	-
_unwrapWETH	Private	Can Modify State	-
setPoolView	External	Can Modify State	onlyPoolAdmin
setSpotSwap	External	Can Modify State	onlyPoolAdmin
setRiskReserve	External	Can Modify State	onlyPoolAdmin
setFeeCollector	External	Can Modify State	onlyPoolAdmin
setPositionManager	External	Can Modify State	onlyPoolAdmin
setOrderManager	External	Can Modify State	onlyPoolAdmin
setRouter	External	Can Modify State	onlyPoolAdmin
addStableToken	External	Can Modify State	onlyPoolAdmin
removeStableToken	External	Can Modify State	onlyPoolAdmin
addPair	External	Can Modify State	onlyPoolAdmin
updatePair	External	Can Modify State	onlyPoolAdmin
updateTradingConfig	External	Can Modify State	onlyPoolAdmin

Pool			
updateTradingFeeConfig	External	Can Modify State	onlyPoolAdmin
_increaseTotalAmount	Internal	Can Modify State	-
_decreaseTotalAmount	Internal	Can Modify State	-
increaseReserveAmount	External	Can Modify State	onlyPositionManager
decreaseReserveAmount	External	Can Modify State	onlyPositionManager
updateAveragePrice	External	Can Modify State	onlyPositionManager
setLPStableProfit	External	Can Modify State	onlyPositionManagerOrFeeCollector
givebackTradingFee	External	Can Modify State	onlyFeeCollector
getAvailableLiquidity	External	-	-
addLiquidity	External	Can Modify State	onlyRouter
removeLiquidity	External	Can Modify State	onlyRouter
_transferToken	Internal	Can Modify State	-
_swapInUni	Private	Can Modify State	-
_getStableTotalAmount	Internal	-	-
_getIndexTotalAmount	Internal	-	-
_addLiquidity	Private	Can Modify State	-
_removeLiquidity	Private	Can Modify State	-
claimFee	External	Can Modify State	onlyTreasury
transferTokenTo	External	Can Modify State	transferAllowed
transferEthTo	External	Can Modify State	transferAllowed
transferTokenOrSwap	External	Can Modify State	transferAllowed
getLpPnl	Public	-	-

Pool			
lpProfit	Public	-	-
getVault	Public	-	-
getPair	Public	-	-
getTradingConfig	External	-	-
getTradingFeeConfig	External	-	-

PoolView			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setPool	External	Can Modify State	onlyPoolAdmin
setPositionManager	External	Can Modify State	onlyPoolAdmin
getMintLpAmount	External	-	-
lpFairPrice	Public	-	-
getDepositAmount	External	-	-
getReceivedAmount	External	-	-
_getDiscount	Internal	-	-
_getStableTotalAmount	Internal	-	-
_getIndexTotalAmount	Internal	-	-

PositionCaller			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getAddress	Public	-	-

PositionCaller			
setAddress	Private	Can Modify State	-
getFundingFees	Public	-	-
getTradingFees	Public	-	-
getUserTradingFees	Public	-	-
getReferralFees	Public	-	-
getKeeperNetworkFees	Public	-	-
batchErcBalance	Public	-	-

4.3 Vulnerability Summary

[N1] [Critical] The overflow error in the givebackTradingFee function

Category: Design Logic Audit

Content

In the _collectTradingFee function of the FeeCollector contract, after calculating the fees to be collected from the LPs, it calls the givebackTradingFee function of the pool contract to collect the fees. In the givebackTradingFee function, when the value of vault.stableTotalAmount is less than the fees to be collected, it calls the _swapInUni function to exchange the tokens in the contract.

However, the token exchange operation does not increase the value of vault.stableTotalAmount, and after the exchange is completed, it does not directly end the function with a return. As a result, vault.stableTotalAmount will still perform a subtraction of the fee, causing an overflow error and the transaction to be reverted.

Code Location: contracts/pool/Pool.sol#L356-369

```
function givebackTradingFee(
    uint256 pairIndex,
    uint256 amount
) external onlyFeeCollector {
    Vault storage vault = vaults[pairIndex];
    Pair memory pair = pairs[pairIndex];
```

```
        if (vault.stableTotalAmount < amount) {
            _swapInUni(pairIndex, pair.stableToken, amount);
        }
        vault.stableTotalAmount -= amount;

        emit GivebackTradingFee(pairIndex, pair.stableToken, amount);
    }
```

Solution

It is recommended that after calling the _swapInUni function to exchange the tokens, the function should be directly ended with a return; or the `vault.stableTotalAmount -= amount` should be included in the else block of the if statement.

Status

Fixed

[N2] [Suggestion] Missing event record

Category: Others

Content

The following functions in the contract are missing event logs for recording key parameter settings.

Code Location: contracts/oracle/PythOraclePriceFeedV2.sol

```
    function updatePrice(
        address[] calldata tokens,
        bytes[] calldata updateData,
        uint64[] calldata publishTimes
    ) external payable override {
        ...

        for (uint256 i = 0; i < tokens.length; i++) {
            require(tokens[i] != address(0), "zero token address");
            require(tokenPriceIds[tokens[i]] != 0, "unknown price id");

            priceIds[i] = tokenPriceIds[tokens[i]];
        }

       ...

        for (uint256 i = 0; i < priceFeeds.length; i++) {
            PythStructs.PriceFeed memory priceFeed = priceFeeds[i];
```

```
            uint64 publishTime = uint64(priceFeed.price.publishTime);
            require(publishTime + priceAge >= block.timestamp, "too old price");

            address token = priceIdTokens[priceFeed.id];
            tokenPrices[publishTime][token] =
_returnPriceWithDecimals(priceFeed.price);

            if (publishTime > tokenPricePublishTime[token]) {
                tokenPricePublishTime[token] = publishTime;
            }
        }
    }
```

Solution

It is recommended to record events when sensitive parameters are modified for self-inspection or community review.

Status

Fixed

[N3] [Low] Potential bad debt risk

Category: Design Logic Audit

Content

In the ExecutionLogic contract, the function executeDecreaseOrder that executes the decrease order has removed the part of the logic that checks whether the position can be liquidated. If the position is in a state of liquidation, executing the decrease order may be faster than the keeper executing the liquidation, which could potentially lead to bad debt remaining in the protocol.

Code Location: contracts/core/ExecutionLogic.sol#L387-388

```
    function executeDecreaseOrders(
        address keeper,
        ExecuteOrder[] memory orders,
        TradingTypes.TradeType tradeType
    ) external override onlyExecutorOrSelf {
        ...

//        bytes32 positionKey = positionManager.getPositionKey(order.account,
order.pairIndex, order.isLong);
//        require(!positionManager.needLiquidation(positionKey, executionPrice), "need
```

```
liquidation");

        ...
}
```

Solution

It is recommended to restore the check to see if the position is in a liquidation state, but add another check before this check to see if order.needADL is false. This way, when executing ADL, the liquidation check will not conflict and prevent the execution due to this check.

Status

Acknowledged; Project team response: This will be updated in the next version.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002405240003	SlowMist Security Team	2024.05.21 - 2024.05.24	Low Risk

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 critical risk, 1 low risk vulnerabilities and 1 suggestion. All the findings were fixed and acknowledged. The code was not deployed to the mainnet.

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