Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2025.02.12, the SlowMist security team received the MYX team's security audit application for MYX HyperVault & Airdrop, 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 audit mainly focuses on the Hyper Vault module and Airdrop module of MYX Finance. In the Hyper Vault module, the HyperToken corresponding to the specified asset token is created through the HyperFactory contract. Users can deposit the asset token in the HyperVault contract and mint the corresponding HyperToken, and then can retrieve the asset token by burning the HyperToken.

In the Airdrop module, users can claim the corresponding MYX tokens by verifying the Merkle proof through the Airdrop contract, and claim the native tokens by verifying the Merkle proof in the DistributeETH contract.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Missing scope limit	Others	Suggestion	Fixed
N2	Lack of checks on the number of tokens claimed	Design Logic Audit	Medium	Fixed
N3	call() should be used instead of transfer()	Others	Low	Fixed
N4	Missing zero address check	Others	Suggestion	Fixed
N5	Missing the event records	Others	Suggestion	Fixed
N6	Risks of excessive privilege	Authority Control Vulnerability Audit	Medium	Acknowledged

4 Code Overview

4.1 Contracts Description

Audit Version:

https://github.com/d11-myx/myx-contracts/tree/main/contracts/hyper

commit: 77abeb1704b06947a618adab8600da4ed3e7965a

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

commit: d5d994002c79dca040bbd51d40cc09266da81ccf

Fixed Version:

https://github.com/d11-myx/myx-contracts/tree/main/contracts/hyper

commit: 420bfd6a687d0160193f63cee1538e016587d02b

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

commit: e8a53d3c0a67a6230f2e633e06df34252d054363

Audit scope:

```
myx-contracts/contracts/hyper
├── HyperFactory.sol
├── HyperToken.sol
└── HyperVault.sol

myx-token/contracts
├── Airdrop.sol
├── DistributeETH.sol
├── GradualRelease.sol
├── MYX.sol
├── interfaces
└── libraries
```

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:

GradualRelease			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
isDelay	Public	-	-
canClaimAmount	Public	-	-
setGov	External	Can Modify State	onlyOwner
claimToken	Public	Can Modify State	nonReentrant onlyGov
urgentRelease	Public	Can Modify State	onlyOwner
cancelUrgentRelease	Public	Can Modify State	onlyOwner
executeUrgentRelease	Public	Can Modify State	onlyOwner
_urgentClaim	Internal	Can Modify State	-

Airdrop			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
initialize	Public	Can Modify State	onlyOwner initializer
togglePauseState	External	Can Modify State	onlyOwner
batchClaimToken	External	Can Modify State	nonReentrant whenNotPaused
_claimToken	Internal	Can Modify State	-
refundToken	External	Can Modify State	nonReentrant onlyOwner
_refundToken	Internal	Can Modify State	-
rescueTokens	External	Can Modify State	onlyOwner
_verify	Private	-	-
_isClaimed	Private	-	-
isValid	Public	-	-
isExpired	Public	-	-
unClaimedAmount	Public	-	-
getPeriodInfo	Public	-	-

DistributeETH			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
initialize	Public	Can Modify State	onlyOwner initializer
togglePauseState	External	Can Modify State	onlyOwner
claimETH	External	Can Modify State	nonReentrant whenNotPaused
rescueETH	External	Can Modify State	onlyOwner

DistributeETH			
_verify	Private	-	-
isValid	Public	-	-
_isClaimed	Private	-	-
<Receive Ether>	External	Payable	-

MYX			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20 ERC20Permit

HyperFactory			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
createHyperToken	External	Can Modify State	-

HyperToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20 ERC20Permit Roleable
decimals	Public	-	-
mint	External	Can Modify State	onlyMiner
burn	External	Can Modify State	onlyMiner
setMiner	External	Can Modify State	-

HyperVault			
Function Name	Visibility	Mutability	Modifiers

HyperVault			
initialize	Public	Can Modify State	initializer
updateHyperFactory	External	Can Modify State	onlyPoolAdmin
getAcceptableAssets	Public	-	-
getHyperPair	External	-	-
addPair	External	Can Modify State	onlyPoolAdmin
updatePairStatus	External	Can Modify State	onlyPoolAdmin
deposit	External	Can Modify State	whenNotPaused nonReentrant
withdraw	External	Can Modify State	whenNotPaused nonReentrant

4.3 Vulnerability Summary

[N1] [Suggestion] Missing scope limit

Category: Others

Content

1.In the Airdrop contract, the owner role sets the information of each period for claiming the airdrop tokens by calling the initialize function. However, here there is a lack of range checks for the _periodInfos parameter. If the set expireTime is less than the releaseTime, it may cause the claiming operation to not proceed normally.

Code Location:

myx-token/contracts/Airdrop.sol#L45

```
function initialize(
    uint256[] memory _phases,
    PeriodInfo[] memory _periodInfos,
    bytes32[][] calldata _baseProofs
) public onlyOwner initializer {
    ...

    for (uint256 i = 0; i < _periodInfos.length; i++) {
        ...
```

```
            periodInfos[uint8(_phases[i])] = periodInfo;
        }
    }
```

2.In the DistributeETH contract, the owner role sets the data of each period for claiming ETH by calling the initialize function. However, here there is a lack of range checks for the data parameter. If the set expireTime is less than the startTime, it may cause the claiming operation to not proceed normally.

Code Location:

myx-token/contracts/DistributeETH.sol#L40

```
    function initialize(
        Data memory _data,
        bytes32[] calldata _baseProofs
    ) public onlyOwner initializer {
        ...

        data = _data;
        _unpause();
    }
```

Solution

It is recommended to perform the range checks when setting key parameters.

Status

Fixed

[N2] [Medium] Lack of checks on the number of tokens claimed

Category: Design Logic Audit

Content

In the Airdrop contract, the _claimToken function is used for claiming tokens, in which the claiming time and Merkle proof are checked to see if they meet expectations. However, it does not check whether the amount parameter passed in by the user is not greater than the remaining claiming quota. If the passed-in amount at this time is greater than the remaining claiming quota in the current period (that is, periodInfo.totalAmount - periodInfo.claimedAmount), then it is possible that tokens that should have been allocated to other periods are claimed in the current period.

Code Location:

myx-token/contracts/Airdrop.sol#L71-87

```
    function _claimToken(
         uint8 phase,
         uint256 amount,
         bytes32[] calldata merkleProof
     ) internal {
         ...

         periodInfo.claimedAmount += amount;
         claimState[phase][msg.sender] = true;

         IERC20(MYX).safeTransfer(msg.sender, amount);

         emit Airdropped(phase, MYX, amount, msg.sender);
     }
```

Solution

It is suggested to check during the claiming that the amount parameter passed in by the user needs to be no greater than the remaining claiming quota in the current period.

Status

Fixed

[N3] [Low] call() should be used instead of transfer()

Category: Others

Content

The transfer() and send() functions forward a fixed amount of 2300 gas. Historically, it has often been recommended to use these functions for value transfers to guard against reentrancy attacks. However, the gas cost of EVM instructions may change significantly during hard forks which may break already deployed contract systems that make fixed assumptions about gas costs. For example. EIP 1884 broke several existing smart contracts due to a cost increase of the SLOAD instruction.

References:

https://consensys.io/diligence/blog/2019/09/stop-using-soliditys-transfer-now/

Code Location:

myx-token/contracts/DistributeETH.sol#L62&L75

```
function claimETH(
    uint256 amount,
    bytes32[] calldata merkleProof
) external nonReentrant whenNotPaused {
    ...

    payable(msg.sender).transfer(amount);

    emit Distribute(amount, msg.sender);
}


function rescueETH(
    address receiver,
    uint256 amount
) external onlyOwner {
    ...

    payable(receiver).transfer(amount);
}
```

Solution

It is recommended to use call() instead of transfer(), but be sure to respect the CEI pattern or add re-entrancy guards, and the return value should be checked.

Status

Fixed

[N4] [Suggestion] Missing zero address check

Category: Others

Content

In the Airdrop contract, the Owner role can call the refundToken function after the claiming period expires to transfer the remaining unclaimed tokens within the period. However, here there is a lack of a non-zero address check for the receiver parameter. If a zero address is passed in by mistake, it may cause the tokens to be lost.

Code Location:

myx-token/contracts/Airdrop.sol#L89-96

```
function refundToken(
     uint8[] memory phases,
     address receiver
 ) external nonReentrant onlyOwner{
     for (uint256 i = 0; i < phases.length; i++) {
         _refundToken(phases[i], receiver);
     }
 }
```

Solution

It is recommended to add a zero address check.

Status

Fixed

[N5] [Suggestion] Missing the event records

Category: Others

Content

In the following contracts, the owner role can modify some sensitive parameters, but there are no event logs in these functions.

Code Location:

myx-contracts/contracts/hyper/HyperToken.sol#L43-46

```
function setMiner(address account, bool enable) external {
     ...
 }
```

myx-token/contracts/Airdrop.sol#L89-96

```
function refundToken(
     uint8[] memory phases,
     address receiver
 ) external nonReentrant onlyOwner{
```

```
    ...
}
```

Solution

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

Status

Fixed

[N6] [Medium] Risks of excessive privilege

Category: Authority Control Vulnerability Audit

Content

1.In the HyperVault contract, the owner role can call the updateHyperFactory and updatePairStatus functions to update the address of HyperFactory and the opening status of the pair. If the privilege is lost or misused, this may have an impact on the user's assets.

Code Location:

myx-contracts/contracts/hyper/HyperVault.sol

```
function updateHyperFactory(address _hyperFactory) external onlyPoolAdmin  {
    ...
}

function updatePairStatus(address asset, bool enable) external onlyPoolAdmin {
    ...
}
```

2.In the HyperToken contract, the timelock role can set the miner role for minting HyperToken, and the miner role can mint HyperToken tokens to the specified address arbitrarily. If the privilege is lost or misused, this may have an impact on the user's assets.

Code Location:

myx-contracts/contracts/hyper/HyperToken.sol

```
function mint(address to, uint256 amount) external onlyMiner {
    ...
}
```

```
function burn(uint256 amount) external onlyMiner {
    ...
}

function setMiner(address account, bool enable) external {
    ...
}
```

3.In the Airdrop contract, the owner role can withdraw the tokens in the contract by calling the refundToken and rescueTokens functions. If the privilege is lost or misused, this may lead to the unexpected theft of tokens.

Code Location:

myx-token/contracts/Airdrop.sol

```
function refundToken(
    uint8[] memory phases,
    address receiver
) external nonReentrant onlyOwner{
    ...
}

function rescueTokens(
    address token,
    address receiver,
    uint256 amount
) external onlyOwner {
    ...
}
```

4.In the DistributeETH contract, the owner role can withdraw the ETH in the contract by calling the rescueETH function. If the privilege is lost or misused, this may lead to the unexpected theft of tokens.

Code Location:

myx-token/contracts/DistributeETH.sol

```
function rescueETH(
    address receiver,
    uint256 amount
) external onlyOwner {
    ...
}
```

5.In the GradualRelease contract, the owner role can withdraw all the MYX tokens in the contract by calling the executeUrgentRelease function. In addition, he can also call the setGov function to set the gov for claiming the myx tokens. If the privilege is lost or misused, this may lead to the unexpected theft of tokens.

Code Location:

myx-token/contracts/GradualRelease.sol

```
function setGov(address _gov) external onlyOwner {
     ...
 }

function executeUrgentRelease(
     address receiver,
     uint256 eta
 ) public onlyOwner {
     ...
 }
```

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 the owner role 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 and executing through a timelock can effectively mitigate the excessive privilege risk and increase the community users' trust in the protocol.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002502140002	SlowMist Security Team	2025.02.12 - 2025.02.14	Medium 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 2 medium risk, 1 low risk and 3 suggestion. All the findings were fixed and acknowledged. Since the project has not yet been deployed to the mainnet and the permissions of the core roles have not yet been transferred, the risk level reported is temporarily medium.

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