Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.08.24, the SlowMist security team received the Mixin team's security audit application for trusted-group, 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.

Level	Description
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

Serial Number	Audit Class	Audit Subclass
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
		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

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Return value not checked	Design Logic Audit	Medium	Ignored
N2	Missing zero-address validation	Design Logic Audit	Suggestion	Ignored
N3	Reentrancy vulnerabilities	Reentrancy Vulnerability	Medium	Ignored
N4	`count` always >= 0	Integer Overflow and Underflow Vulnerability	Low	Ignored
N5	Transaction reordering may cause the transaction to fail	Reordering Vulnerability	Medium	Ignored
N6	Fee can be bypassed	Design Logic Audit	High	Fixed
N7	Loss of precision due to division	Arithmetic Accuracy Deviation Vulnerability	Suggestion	Fixed
N8	`receiver` can be zero-address	Design Logic Audit	Low	Ignored

4 Code Overview

4.1 Contracts Description

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:

Bridge			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
<Receive Ether>	External	Payable	-
release	Public	Payable	-
vault	Public	Can Modify State	-
bind	Public	Can Modify State	-
pass	Public	Can Modify State	-
passXIN	Internal	Can Modify State	-
canonical	Internal	-	-

Withdrawal			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
submit	Public	Payable	-
transferXIN	Internal	Can Modify State	-
transferERC20	Internal	Can Modify State	-

StandardToken

Function Name	Visibility	Mutability	Modifiers
balanceOf	Public	-	-
_transfer	Internal	Can Modify State	-
_transferFrom	Internal	Can Modify State	-
approve	Public	Can Modify State	-
allowance	Public	-	-

Asset

Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
transferWithExtra	Public	Can Modify State	-
transfer	Public	Can Modify State	-
transferFrom	Public	Can Modify State	-
mint	External	Can Modify State	onlyRegistry
burn	External	Can Modify State	onlyRegistry

Storage			
Function Name	Visibility	Mutability	Modifiers
read	Public	-	-
write	Public	Can Modify State	-

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

User			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
run	External	Can Modify State	onlyRegistry
handle	Internal	Can Modify State	-

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

Factory			
Function Name	Visibility	Mutability	Modifiers
getOrCreateAssetContract	Internal	Can Modify State	-
getOrCreateUserContract	Internal	Can Modify State	-

Factory			
getUserContractCode	Internal	-	-
getAssetContractCode	Internal	-	-
getContractAddress	Internal	-	-
deploy	Internal	Can Modify State	-

Registry			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
iterate	Public	Can Modify State	-
halt	Public	Can Modify State	-
claim	External	Can Modify State	-
burn	External	Can Modify State	-
sendMixinTransaction	Internal	Can Modify State	-
buildMixinTransaction	Internal	Can Modify State	-
mixin	Public	Can Modify State	-
parseEventExtra	Internal	-	-
parseEventAsset	Internal	-	-
parseEventUser	Internal	Can Modify State	-
parseEventInput	Internal	Can Modify State	-

4.3 Vulnerability Summary

[N1] [Medium] Return value not checked

Category: Design Logic Audit

Content

trusted-group/mvm/quorum/bridge/contracts/Bridge.sol

```
//#L62
IERC20(XIN).transferWithExtra(receiver, amount, input);
//#L71
IERC20(asset).transferFrom(msg.sender, address(this), amount);
//#L90
IERC20(asset).transferFrom(msg.sender, receiver, amount);
//#L97
IERC20(XIN).transferFrom(msg.sender, address(this), amount);
```

trusted-group/mvm/quorum/bridge/contracts/Withdrawal.sol

```
//#L66-L67
IERC20(asset).transferFrom(msg.sender, address(this), amount);
IERC20(asset).transferWithExtra(receiver, amount, input);
```

trusted-group/mvm/quorum/registry/contracts/Asset.sol

```
//#L117
IRegistry(registry).burn(to, value, extra);
//#L135
IRegistry(registry).burn(to, value, new bytes(0));
```

trusted-group/mvm/quorum/registry/contracts/User.sol

```
//#L29-L37
if (extra.length < 28) {
    IRegistry(registry).claim(asset, amount);
    return true;
```

```
}
uint16 count = extra.toUint16(0);
if (count < 1 || count > 16) {
    IRegistry(registry).claim(asset, amount);
    return true;
}
//#L48
try IRegistry(registry).claim(asset, amount) {} catch {}
//#L65-L66
IERC20(asset).approve(process, 0);
IERC20(asset).approve(process, amount);
```

Solution

Check return value.

Status

Ignored

[N2] [Suggestion] Missing zero-address validation

Category: Design Logic Audit

Content

trusted-group/mvm/quorum/bridge/contracts/Bridge.sol

```
constructor(address factory, address xin) {
    FACTORY = factory;
    XIN = xin;
    OWNER = msg.sender;
}
```

trusted-group/mvm/quorum/bridge/contracts/Withdrawal.sol

```
constructor(address bridge, address xin) {
    BRIDGE = bridge;
```

```
        XIN = xin;
    }
```

Solution

Check input addresses.

Status

Ignored

[N3] [Medium] Reentrancy vulnerabilities

Category: Reentrancy Vulnerability

Content

trusted-group/mvm/quorum/registry/contracts/Registry.sol

```
    function iterate(bytes memory raw) public {
        require(HALTED, "invalid state");
        require(raw.length == 256, "invalid input size");
        uint256[4] memory group = [
            raw.toUint256(0),
            raw.toUint256(32),
            raw.toUint256(64),
            raw.toUint256(96)
        ];
        uint256[2] memory sig1 = [raw.toUint256(128), raw.toUint256(160)];
        uint256[2] memory sig2 = [raw.toUint256(192), raw.toUint256(224)];
        uint256[2] memory message = raw.slice(0, 128).hashToPoint();
        require(sig1.verifySingle(GROUP, message), "invalid signature");
        require(sig2.verifySingle(group, message), "invalid signature");
        emit Iterated(GROUP, group);
        GROUP = group; //SlowMist//
    }

    function halt(bytes memory raw) public {
        bytes memory input = bytes("HALT").concat(
            Integer.uint64ToFixedBytes(INBOUND)
        );
        uint256[2] memory sig = [raw.toUint256(0), raw.toUint256(32)];
        uint256[2] memory message = input.hashToPoint();
```

```
    require(sig.verifySingle(GROUP, message), "invalid signature");
    HALTED = !HALTED; //SlowMist//
    emit Halted(HALTED);
}

function sendMixinTransaction(
    address user,
    address asset,
    uint256 amount,
    bytes memory extra
) internal {
    uint256 balance = balances[assets[asset]];
    bytes memory log = buildMixinTransaction(
        OUTBOUND,
        users[user],
        assets[asset],
        amount,
        extra
    );
    emit MixinTransaction(log);
    balances[assets[asset]] = balance - amount;
    OUTBOUND = OUTBOUND + 1; //SlowMist//
}

function mixin(bytes memory raw) public returns (bool) {
    require(!HALTED, "invalid state");
    require(raw.length >= 141, "event data too small");

    Event memory evt;
    uint256 offset = 0;

    uint128 id = raw.toUint128(offset);
    require(id == PID, "invalid process");
    offset = offset + 16;

    evt.nonce = raw.toUint64(offset);
    require(evt.nonce == INBOUND, "invalid nonce");
    INBOUND = INBOUND + 1;
    offset = offset + 8;

    (offset, id, evt.amount) = parseEventAsset(raw, offset);
    (offset, evt.extra, evt.timestamp) = parseEventExtra(raw, offset);
    (offset, evt.user) = parseEventUser(raw, offset);
    (evt.asset, evt.extra) = parseEventInput(id, evt.extra);
```

```
        offset = offset + 2;
        evt.sig = [raw.toUint256(offset), raw.toUint256(offset + 32)];
        uint256[2] memory message = raw
            .slice(0, offset - 2)
            .concat(new bytes(2))
            .hashToPoint();
        require(evt.sig.verifySingle(GROUP, message), "invalid signature");

        offset = offset + 64;
        require(raw.length == offset, "malformed event encoding");

        uint256 balance = balances[assets[evt.asset]];
        balances[assets[evt.asset]] = balance + evt.amount;//SlowMist//

        emit MixinEvent(
            evt.nonce,
            evt.user,
            evt.asset,
            evt.amount,
            evt.extra,
            evt.timestamp
        );
        Asset(evt.asset).mint(evt.user, evt.amount);
        return User(evt.user).run(evt.asset, evt.amount, evt.extra);
    }
```

Solution

Change state before invoke contract.

Status

Ignored

[N4] [Low] `count` always >= 0

Category: Integer Overflow and Underflow Vulnerability

Content

trusted-group/mvm/quorum/registry/contracts/User.sol

```
uint16 count = extra.toUint16(0);
if (count < 1 || count > 16) {
    IRegistry(registry).claim(asset, amount);
    return true;
}

for (uint256 offset = 2; count >= 0 && offset < extra.length; count--) { //SlowMist//
    bool primary = offset == 2;
    bytes memory data = extra.slice(offset, extra.length - offset);
    (uint256 size, bool success) = handle(data, asset, amount, primary);
    if (!success) {
        break;
    }
    offset = offset + size;
}
```

`count` type is `uint16`, `count--` will overflow or revert if `count==0`, so `count` always >= 0, the logical here is useless.

Solution

Status

Ignored

[N5] [Medium] Transaction reordering may cause the transaction to fail

Category: Reordering Vulnerability

Content

trusted-group/mvm/quorum/registry/contracts/Registry.sol

```
function mixin(bytes memory raw) public returns (bool) {
    require(!HALTED, "invalid state");
    require(raw.length >= 141, "event data too small");

    Event memory evt;
    uint256 offset = 0;

    uint128 id = raw.toUint128(offset);
```

```
    require(id == PID, "invalid process");
    offset = offset + 16;

    evt.nonce = raw.toUint64(offset);
    require(evt.nonce == INBOUND, "invalid nonce"); //SlowMist//
    INBOUND = INBOUND + 1;
```

Solution

Status

Ignored

[N6] [High] Fee can be bypassed

Category: Design Logic Audit

Content

trusted-group/mvm/quorum/bridge/contracts/Withdrawal.sol

```
function submit(
    address receiver,
    address asset,
    uint256 amount,
    address feeAsset,
    uint256 feeAmount,
    bytes memory ma,
    bytes memory mb
) public payable {
    require(feeAsset != XIN, "invalid fee asset");
    if (asset == XIN) {
        require(msg.value / BASE == amount, "invalid withdrawal amount");
        transferXIN(receiver, ma);
    } else {
        transferERC20(receiver, asset, amount, ma);
    }
    transferERC20(receiver, feeAsset, feeAmount, mb);
}
```

There are two problem:

a). `feeAsset` no limit, users can use any asset to pay for fee;

b). Users do not need to call `submit`, they can call asset contract and bridge contract immediately.

Solution

Status

Fixed; Removed `Withdrawal` contract.

[N7] [Suggestion] Loss of precision due to division

Category: Arithmetic Accuracy Deviation Vulnerability

Content

trusted-group/mvm/quorum/bridge/contracts/Withdrawal.sol

```
require(msg.value / BASE == amount, "invalid withdrawal amount");
```

Loss of precision due to division.

Solution

Status

Fixed; Removed `Withdrawal` contract.

[N8] [Low] `receiver` can be zero-address

Category: Design Logic Audit

Content

trusted-group/mvm/quorum/bridge/contracts/Bridge.sol

```
function release(address receiver, bytes memory input) public payable {
    uint256 amount = msg.value / BASE;
    require(amount > 0, "value too small");
```

```
        address bound = bridges[msg.sender];
        require(bound == address(0) || receiver == bound, "bound not match");
    //SlowMist//

        IERC20(XIN).transferWithExtra(receiver, amount, input);
        emit Through(XIN, msg.sender, receiver, amount);
    }
```

`receiver` can be zero-address.

Solution

Check `receiver` address.

Status

Ignored

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002208310001	SlowMist Security Team	2022.08.24 - 2022.08.31	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 1 high risk, 3 medium risk, 2 low risk, 2 suggestion vulnerabilities. And 3 medium risk, 2 low risk, 1 suggestion vulnerabilities were ignored.

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