Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.04.28, the SlowMist security team received the Helio team's security audit application for Helio, 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

Audit Version:

File name: smart-contracts.zip

SHA256: 794ac9c70aa01aa2ceb19186ea0f60438eaa35ac135d73596e8ce2586ec215c1

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Token Approval Issues	Design Logic Audit	Low	Fixed
N2	Borrow dart issue	Design Logic Audit	Medium	Fixed
N3	Recycling Allowance issue	Design Logic Audit	Low	Fixed
N4	Collect stability fees issue	Design Logic Audit	Medium	Fixed
N5	Safe Token Transfer Issue	Others	Suggestion	Fixed
N6	Potential Compatibility Issues	Design Logic Audit	Suggestion	Fixed

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:

Clipper			
Function Name	Visibility	Mutability	Modifiers
rely	External	Can Modify State	auth
deny	External	Can Modify State	auth
<Constructor>	Public	Can Modify State	-
file	External	Can Modify State	auth lock
file	External	Can Modify State	auth lock
min	Internal	-	-
add	Internal	-	-
sub	Internal	-	-
mul	Internal	-	-
wmul	Internal	-	-
rmul	Internal	-	-
rdiv	Internal	-	-
getFeedPrice	Internal	Can Modify State	-
kick	External	Can Modify State	auth lock isStopped
redo	External	Can Modify State	lock isStopped
take	External	Can Modify State	lock isStopped

Clipper			
_remove	Internal	Can Modify State	-
count	External	-	-
list	External	-	-
getStatus	External	-	-
status	Internal	-	-
upchost	External	Can Modify State	-
yank	External	Can Modify State	auth lock

DAOInteraction			
Function Name	Visibility	Mutability	Modifiers
rely	External	Can Modify State	auth
deny	External	Can Modify State	auth
initialize	Public	Can Modify State	initializer
_authorizeUpgrade	Internal	Can Modify State	onlyOwner
setCollateralType	External	Can Modify State	auth
removeCollateralType	External	Can Modify State	auth
stringToBytes32	Public	-	-
deposit	External	Can Modify State	-
borrow	External	Can Modify State	-
payback	External	Can Modify State	-

DAOInteraction			
withdraw	External	Can Modify State	-
drip	Public	Can Modify State	-
collateralPrice	Public	-	-
usbPrice	External	-	-
collateralRate	External	-	-
depositTVL	External	-	-
collateralTVL	External	-	-
free	Public	-	-
locked	External	-	-
borrowed	External	-	-
availableToBorrow	External	-	-
willBorrow	External	-	-
currentLiquidationPrice	External	-	-
estimatedLiquidationPrice	External	-	-
rpow	Internal	-	-
borrowApr	Public	-	-

Dog			
Function Name	Visibility	Mutability	Modifiers
rely	External	Can Modify State	auth

Dog			
deny	External	Can Modify State	auth
<Constructor>	Public	Can Modify State	-
min	Internal	-	-
add	Internal	-	-
sub	Internal	-	-
mul	Internal	-	-
file	External	Can Modify State	auth
file	External	Can Modify State	auth
file	External	Can Modify State	auth
file	External	Can Modify State	auth
chop	External	-	-
bark	External	Can Modify State	-
digs	External	Can Modify State	auth
cage	External	Can Modify State	auth

DssCdpManager			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
add	Internal	-	-
sub	Internal	-	-

DssCdpManager			
toInt	Internal	-	-
cdpAllow	Public	Can Modify State	cdpAllowed
urnAllow	Public	Can Modify State	-
open	Public	Can Modify State	-
give	Public	Can Modify State	cdpAllowed
frob	Public	Can Modify State	cdpAllowed
flux	Public	Can Modify State	cdpAllowed
flux	Public	Can Modify State	cdpAllowed
move	Public	Can Modify State	cdpAllowed
quit	Public	Can Modify State	cdpAllowed urnAllowed
enter	Public	Can Modify State	urnAllowed cdpAllowed
shift	Public	Can Modify State	cdpAllowed cdpAllowed

GemJoin			
Function Name	Visibility	Mutability	Modifiers
rely	External	Can Modify State	auth
deny	External	Can Modify State	auth
<Constructor>	Public	Can Modify State	-
cage	External	Can Modify State	auth
join	External	Can Modify State	-

GemJoin			
exit	External	Can Modify State	-

UsbJoin			
Function Name	Visibility	Mutability	Modifiers
rely	External	Can Modify State	auth
deny	External	Can Modify State	auth
<Constructor>	Public	Can Modify State	-
cage	External	Can Modify State	auth
mul	Internal	-	-
join	External	Can Modify State	-
exit	External	Can Modify State	-

Jug			
Function Name	Visibility	Mutability	Modifiers
rely	External	Can Modify State	auth
deny	External	Can Modify State	auth
<Constructor>	Public	Can Modify State	-
rpow	Internal	-	-
add	Internal	-	-
diff	Internal	-	-
rmul	Internal	-	-

Jug			
init	External	Can Modify State	auth
file	External	Can Modify State	auth
file	External	Can Modify State	auth
file	External	Can Modify State	auth
drip	External	Can Modify State	-

mBNB			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
mint	External	Can Modify State	-
burn	External	Can Modify State	-

Pot			
Function Name	Visibility	Mutability	Modifiers
rely	External	Can Modify State	auth
deny	External	Can Modify State	auth
<Constructor>	Public	Can Modify State	-
rpow	Internal	-	-
rmul	Internal	-	-
add	Internal	-	-
sub	Internal	-	-

Pot			
mul	Internal	-	-
file	External	Can Modify State	auth
file	External	Can Modify State	auth
cage	External	Can Modify State	auth
drip	External	Can Modify State	-
join	External	Can Modify State	-
exit	External	Can Modify State	-

Spotter			
Function Name	Visibility	Mutability	Modifiers
rely	External	Can Modify State	auth
deny	External	Can Modify State	auth
<Constructor>	Public	Can Modify State	-
mul	Internal	-	-
rdiv	Internal	-	-
file	External	Can Modify State	auth
file	External	Can Modify State	auth
file	External	Can Modify State	auth
poke	External	Can Modify State	-
cage	External	Can Modify State	auth

Usb			
Function Name	Visibility	Mutability	Modifiers
rely	External	Can Modify State	auth
deny	External	Can Modify State	auth
add	Internal	-	-
sub	Internal	-	-
<Constructor>	Public	Can Modify State	-
transfer	External	Can Modify State	-
transferFrom	Public	Can Modify State	-
mint	External	Can Modify State	auth
burn	External	Can Modify State	-
approve	External	Can Modify State	-
push	External	Can Modify State	-
pull	External	Can Modify State	-
move	External	Can Modify State	-
permit	External	Can Modify State	-

Vat			
Function Name	Visibility	Mutability	Modifiers
rely	External	Can Modify State	auth
deny	External	Can Modify State	auth

Vat			
behalf	External	Can Modify State	auth
regard	External	Can Modify State	auth
hope	External	Can Modify State	-
nope	External	Can Modify State	-
wish	Internal	-	-
<Constructor>	Public	Can Modify State	-
add	Internal	-	-
sub	Internal	-	-
mul	Internal	-	-
add	Internal	-	-
sub	Internal	-	-
mul	Internal	-	-
init	External	Can Modify State	auth
file	External	Can Modify State	auth
file	External	Can Modify State	auth
cage	External	Can Modify State	auth
slip	External	Can Modify State	auth
flux	External	Can Modify State	-
move	External	Can Modify State	-
either	Internal	-	-

Vat			
both	Internal	-	-
frob	External	Can Modify State	-
fork	External	Can Modify State	-
grab	External	Can Modify State	auth
heal	External	Can Modify State	-
suck	External	Can Modify State	auth
fold	External	Can Modify State	auth

Vow			
Function Name	Visibility	Mutability	Modifiers
rely	External	Can Modify State	auth
deny	External	Can Modify State	auth
<Constructor>	Public	Can Modify State	-
add	Internal	-	-
sub	Internal	-	-
min	Internal	-	-
file	External	Can Modify State	auth
file	External	Can Modify State	auth
fess	External	Can Modify State	auth
flog	External	Can Modify State	-

Vow			
heal	External	Can Modify State	-
kiss	External	Can Modify State	-
flop	External	Can Modify State	-
flap	External	Can Modify State	-
cage	External	Can Modify State	auth

4.3 Vulnerability Summary

[N1] [Low] Token Approval Issues

Category: Design Logic Audit

Content

In the DAOInteraction contract, the current contract will do the USB token approval to the usbJoin contract during the initialize operation. However, there is no re-approval interface in the DAOInteraction contract. If the usbJoin contract runs out of allowances, it cannot be re-approved. The same is true for the setCollateralType function.

Code location: contracts/DAOInteraction.sol

```
function initialize(address vat_,
    address spot_,
    address usb_,
    address usbJoin_,
    address jug_) public initializer {
    __Ownable_init();

    wards[msg.sender] = 1;

    vat = Vat(vat_);
    spotter = Spotter(spot_);
    usb = Usb(usb_);
    usbJoin = UsbJoin(usbJoin_);
```

```
        jug = Jug(jug_);

        vat.hope(usbJoin_);

        usb.approve(usbJoin_,

115792089237316195423570985008687907853269984665640564039457584007913129639935);
    }

    function setCollateralType(address token, address gemJoin, bytes32 ilk) external
auth {
        collaterals[token] = CollateralType(GemJoin(gemJoin), ilk, 1);
        IERC20(token).approve(gemJoin,

115792089237316195423570985008687907853269984665640564039457584007913129639935);
        vat.init(ilk);
        vat.rely(gemJoin);
    }
```

Solution

It is recommended to add an interface to increase allowances.

Status

Fixed; Fixed in commit 58147b5c10a09580ef619af9bfd9e7a550c95ac1

[N2] [Medium] Borrow dart issue

Category: Design Logic Audit

Content

In the DAOInteraction contract, users can borrow usb tokens through the borrow function. During this process, the number of frob operations modified should be dart / rate, and the operations of move and exit should be the number of usb.

Code location: contracts/DAOInteraction.sol

```
    function borrow(address token, uint256 dart) external returns(uint256) {
        CollateralType memory collateralType = collaterals[token];
        require(collateralType.live == 1, "Interaction/inactive collateral");
```

```
        vat.frob(collateralType.ilk, msg.sender, msg.sender, msg.sender, 0,
int256(dart));
        vat.move(msg.sender, address(this), dart * 10**27);
        usbJoin.exit(msg.sender, dart);
        emit Borrow(msg.sender, dart);
        return dart;
    }
```

Solution

It is recommended to clarify design expectations

Status

Fixed; Fixed in commit e6cef04ef47081697fcbb829ffa9ff12738d57b4

[N3] [Low] Recycling Allowance issue

Category: Design Logic Audit

Content

In the DAOInteraction contract, when the setCollateralType operation is performed, the contract will tokenize the gemJoin contract. However, when the CollateralType was removed, the approved allowance was not recovered.

Code location: contracts/DAOInteraction.sol

```
    function removeCollateralType(address token, address gemJoin) external auth {
        collaterals[token].live = 0;
    }
```

Solution

It is recommended to set the approved allowance to 0 when doing the removeCollateralType operation.

Status

Fixed; Fixed in commit 58147b5c10a09580ef619af9bfd9e7a550c95ac1

[N4] [Medium] Collect stability fees issue

Category: Design Logic Audit

Content

In DAOInteraction function, stability fees are not collected when borrowing.

Code location: contracts/DAOInteraction.sol

```
    function borrow(address token, uint256 dart) external returns(uint256) {
        CollateralType memory collateralType = collaterals[token];
        require(collateralType.live == 1, "Interaction/inactive collateral");

        vat.frob(collateralType.ilk, msg.sender, msg.sender, msg.sender, 0,
int256(dart));
        vat.move(msg.sender, address(this), dart * 10**27);
        usbJoin.exit(msg.sender, dart);
        emit Borrow(msg.sender, dart);
        return dart;
    }
```

Solution

It is recommended to perform the drip operation when performing the borrow operation.

Status

Fixed

[N5] [Suggestion] Safe Token Transfer Issue

Category: Others

Content

The transfer of tokens is involved in the deposit and payback operations of the DAOInteraction contract, but the return value is not checked, which will have potential compatibility risks.

Code location: contracts/DAOInteraction.sol

```
    function deposit(address token, uint256 dink) external returns (uint256){
        CollateralType memory collateralType = collaterals[token];
        require(collateralType.live == 1, "Interaction/inactive collateral");
```

```
    IERC20(token).transferFrom(msg.sender, address(this), dink);
    ...
}

function payback(address token, uint256 dart) external returns(uint256) {
    CollateralType memory collateralType = collaterals[token];
    require(collateralType.live == 1, "Interaction/inactive collateral");

    usb.transferFrom(msg.sender, address(this), dart);
    ...
}
```

Solution

It is recommended to use OpenZeppelin's SafeERC20 library for transfer operations.

Status

Fixed; Fixed in commit 58147b5c10a09580ef619af9bfd9e7a550c95ac1

[N6] [Suggestion] Potential Compatibility Issues

Category: Design Logic Audit

Content

In the DAOInteraction contract, when the user performs the deposit operation, the user will transfer the tokens into the contract through the transferFrom function and perform the join operation. However, if the transferred tokens are deflationary tokens, the actual number of tokens received by the contract is inconsistent with the dink parameter passed in by the user.

Code location: contracts/DAOInteraction.sol

```
function deposit(address token, uint256 dink) external returns (uint256){
    CollateralType memory collateralType = collaterals[token];
    require(collateralType.live == 1, "Interaction/inactive collateral");

    IERC20(token).transferFrom(msg.sender, address(this), dink);
    collateralType.gem.join(msg.sender, dink);
    vat.behalf(msg.sender, address(this));
```

```
        vat.frob(collateralType.ilk, msg.sender, msg.sender, msg.sender,
int256(dink), 0);

        deposits[token] += dink;

//        drip(token);

        emit Deposit(msg.sender, dink);
        return dink;
    }
```

Solution

It is recommended to obtain the balance of the contract before and after the user's transfer as the actual number of tokens received.

Status

Fixed; Fixed in commit 53cd5c5bcdabe32e5eb54a1a21779a5a4ad1b6cb

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002205100005	SlowMist Security Team	2022.04.28 - 2022.05.10	Passed

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 medium risk, 3 low risks, and 6 suggestions. And 1 medium risk, 2 low risks, and 3 suggestions were confirmed and fixed; All other findings were fixed. 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