Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.02.07, the SlowMist security team received the Symbiosis team's security audit application for Symbiosis, 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:

- Reentrancy Vulnerability
- Replay Vulnerability
- Reordering Vulnerability
- Short Address Vulnerability
- Denial of Service Vulnerability
- Transaction Ordering Dependence Vulnerability
- Race Conditions Vulnerability
- Authority Control Vulnerability
- Integer Overflow and Underflow Vulnerability
- TimeStamp Dependence Vulnerability
- Uninitialized Storage Pointers Vulnerability
- Arithmetic Accuracy Deviation Vulnerability
- tx.origin Authentication Vulnerability

- "False top-up" Vulnerability
- Variable Coverage Vulnerability
- Gas Optimization Audit
- Malicious Event Log Audit
- Redundant Fallback Function Audit
- Unsafe External Call Audit
- Explicit Visibility of Functions State Variables Audit
- Design Logic Audit
- Scoping and Declarations Audit

3 Project Overview

3.1 Project Introduction

Symbiosis is a decentralized multi-chain liquidity protocol.

Project official website:

https://algorithmx.vercel.app/

Audit version:

https://github.com/ AlgorithmXlabs/AuditAlgoritmxX

review version:

50dda9f9d2e205c2804599dcc148eea0878f1c23

Audit scope:

contracts/synth-contracts

contracts/metarouter

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Event replay	Replay Vulnerability	Low	Fixed
N2	External call parameters are not verified	Unsafe External Call Audit	Low	Fixed
N3	Event log missing	Malicious Event Log Audit	Suggestion	Ignored
N4	Event log missing	Malicious Event Log Audit	Suggestion	Ignored
N5	Risk of excessive authority	Authority Control Vulnerability	Suggestion	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:

MetaRouterV2			
Function Name	Visibility	Mutability	Modifiers

MetaRouterV2			
<Constructor>	Public	Can Modify State	-
metaRouteV2	External	Payable	-
swap	External	Can Modify State	-
metaMintSwap	External	Can Modify State	-
_swap	Internal	Can Modify State	-
_lazyApprove	Internal	Can Modify State	-

MetaRouterV2Solana			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
metaRouteV2	External	Payable	-
swap	External	Can Modify State	-
metaMintSwap	External	Can Modify State	-
_swap	Internal	Can Modify State	-
_lazyApprove	Internal	Can Modify State	-

BridgeV2			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
mpc	Public	-	-
currentChainId	Public	-	-

BridgeV2			
receiveRequestV2	External	Can Modify State	onlyMPC
receiveRequestV2Signed	External	Can Modify State	onlySignedByMPC
transmitRequestV2	Public	Can Modify State	onlyTransmitter
setTransmitterStatus	External	Can Modify State	onlyOwner
changeMPC	External	Can Modify State	onlyOwnerOrMPC
withdrawFee	External	Can Modify State	onlyOwnerOrAdmin
_processRequest	Private	Can Modify State	-

BridgeV2Solana			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
mpc	Public	-	-
currentChainId	Public	-	-
receiveRequestV2	External	Can Modify State	onlyMPC
receiveRequestV2Signed	External	Can Modify State	onlySignedByMPC
transmitRequestV2	Public	Can Modify State	onlyTransmitter
setTransmitterStatus	External	Can Modify State	onlyOwner
changeMPC	External	Can Modify State	onlyOwnerOrMPC
withdrawFee	External	Can Modify State	onlyOwnerOrAdmin
_prepareRequestId	Internal	Can Modify State	-

BridgeV2Solana			
_processRequest	Private	Can Modify State	-

AdminableUpgradeable			
Function Name	Visibility	Mutability	Modifiers
__Adminable_init	Internal	Can Modify State	initializer
setAdminPermission	External	Can Modify State	onlyOwner

Wrapper			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20Permit ERC20
deposit	External	Payable	-
withdraw	External	Can Modify State	-
isTrustedForwarder	Public	-	-
_msgSender	Internal	-	-
_msgData	Internal	-	-

Portal			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
versionRecipient	External	-	-
synthesize	External	Can Modify State	whenNotPaused

Portal			
metaSynthesize	External	Can Modify State	whenNotPaused
synthesizeNative	External	Payable	whenNotPaused
synthesizeWithPermit	External	Can Modify State	whenNotPaused
revertSynthesize	External	Can Modify State	onlyBridge whenNotPaused
unsynthesize	External	Can Modify State	onlyBridge whenNotPaused
metaUnsynthesize	External	Can Modify State	onlyBridge whenNotPaused
revertBurnRequest	External	Can Modify State	whenNotPaused
pause	External	Can Modify State	onlyOwner
unpause	External	Can Modify State	onlyOwner
setWhitelistToken	External	Can Modify State	onlyOwner
setTokenThreshold	External	Can Modify State	onlyOwner
setMetaRouter	External	Can Modify State	onlyOwner
sendSynthesizeRequest	Internal	Can Modify State	-

RelayRecipientUpgradeable			
Function Name	Visibility	Mutability	Modifiers
__RelayRecipient_init	Internal	Can Modify State	initializer
isTrustedForwarder	Public	-	-
_msgSender	Internal	-	-
_msgData	Internal	-	-

SyntERC20			
Function Name	Visibility	Mutability	Modifiers
mint	External	Can Modify State	onlyOwner
burn	External	Can Modify State	onlyOwner
decimals	Public	-	-
<Constructor>	Public	Can Modify State	ERC20Permit ERC20

SyntFabric			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
getSyntRepresentationByKey	Public	-	-
getSyntRepresentation	Public	-	-
getRealRepresentation	Public	-	-
unsynthesize	External	Can Modify State	onlySynthesis
synthesize	External	Can Modify State	onlySynthesis
createRepresentationByAdmin	External	Can Modify State	onlyOwner
setRepresentation	Internal	Can Modify State	-

SyntFabricSolana			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
getSyntRepresentationByKey	Public	-	-

SyntFabricSolana			
getSyntRepresentation	Public	-	-
getRealRepresentation	Public	-	-
unsynthesize	External	Can Modify State	onlySynthesis
synthesize	External	Can Modify State	onlySynthesis
createRepresentationByAdmin	External	Can Modify State	onlyOwner
setRepresentation	Internal	Can Modify State	-

Synthesis			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
versionRecipient	External	-	-
mintSyntheticToken	External	Can Modify State	onlyBridge whenNotPaused
metaMintSyntheticToken	External	Can Modify State	onlyBridge whenNotPaused
revertSynthesizeRequest	External	Can Modify State	whenNotPaused
burnSyntheticToken	External	Can Modify State	whenNotPaused
metaBurnSyntheticToken	External	Can Modify State	whenNotPaused
revertBurn	External	Can Modify State	onlyBridge whenNotPaused
pause	External	Can Modify State	onlyOwner
unpause	External	Can Modify State	onlyOwner
setTokenThreshold	External	Can Modify State	onlyOwner

Synthesis			
setMetaRouter	External	Can Modify State	onlyOwner
setFabric	External	Can Modify State	onlyOwner

SynthesisSolana			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
versionRecipient	External	-	-
mintSyntheticToken	External	Can Modify State	onlyBridge whenNotPaused
metaMintSyntheticToken	External	Can Modify State	onlyBridge whenNotPaused
revertSynthesizeRequest	External	Can Modify State	whenNotPaused
burnSyntheticToken	External	Can Modify State	whenNotPaused
metaBurnSyntheticToken	External	Can Modify State	whenNotPaused
revertBurn	External	Can Modify State	onlyBridge whenNotPaused
pause	External	Can Modify State	onlyOwner
unpause	External	Can Modify State	onlyOwner
setTokenThreshold	External	Can Modify State	onlyOwner
setMetaRouter	External	Can Modify State	onlyOwner
setFabric	External	Can Modify State	onlyOwner

Timelock			
Function Name	Visibility	Mutability	Modifiers

Timelock			
<Constructor>	Public	Can Modify State	-
getQueuedTx	Public	-	-
<Receive Ether>	External	Payable	-
setDelay	Public	Can Modify State	-
acceptAdmin	Public	Can Modify State	-
setPendingAdmin	Public	Can Modify State	-
queueTransaction	Public	Can Modify State	-
cancelTransaction	Public	Can Modify State	-
executeTransaction	Public	Payable	-
getBlockTimestamp	Internal	-	-

4.3 Vulnerability Summary

[N1] [Low] Event replay

Category: Replay Vulnerability

Content

-
-

```
        address _oppositeBridge,
        uint256 _chainID
    ) external whenNotPaused {
        bytes32 externalID = keccak256(abi.encodePacked(_internalID, address(this),
_msgSender(), block.chainid));
        //SlowMist// if SynthesizeState == SynthesizeState.RevertRequest can still be
judged
        require(
            synthesizeSt
ates[externalID] != SynthesizeState.Synthesized,
            "Symb: synthetic tokens already minted"
        );
        synthesizeStates[externalID] = SynthesizeState.RevertRequest; // close

        {
            bytes memory out = abi.encodeWithSelector(
                bytes4(keccak256(bytes("revertSynthesize(uint256,bytes32)"))),
                _stableBridgingFee,
                externalID
            );
            IBridge(bridge).transmitRequestV2(
                out,
                _receiveSide,
                _oppositeBridge,
                _chainID
            );
        }

        emit RevertSynthesizeRequest(_internalID, _msgSender());
    }
```

Solution

Accurately judge the state of SynthesizeState

Status

Fixed; Communicated with the project party:As it is by design, the user can send a request to revert a transaction until it passes on another network

[N2] [Low] External call parameters are not verified

Category: Unsafe External Call Audit

Content

- contracts/synth-contracts/Synthesis.sol
- contracts/synth-contracts/SynthesisSolana.sol

If the parameter `finalDexRouter` in `_metaBurnTransaction` is a malicious contract address, this cross-chain transaction can always fail. `_metaBurnTransaction.stableBridgingFee` can also be constructed to 0 by itself, resulting in platform losses.

```
function metaBurnSyntheticToken(
        MetaRouteStructs.MetaBurnTransaction memory _metaBurnTransaction
    ) external whenNotPaused returns (bytes32 internalID) {
        require(_metaBurnTransaction.amount >=
tokenThreshold[_metaBurnTransaction.sToken], "Symb: amount under threshold");

        ISyntFabric(fabric).unsynthesize(
            _msgSender(),
            _metaBurnTransaction.amount,
            _metaBurnTransaction.sToken
        );

        if (_metaBurnTransaction.revertableAddress == address(0)) {
            _metaBurnTransaction.revertableAddress =
_metaBurnTransaction.chain2address;
        }
        {
            address rtoken = ISyntFabric(fabric).getRealRepresentation(
                _metaBurnTransaction.sToken
            );

            internalID = keccak256(
                abi.encodePacked(this, requestCount, block.chainid)
            );
            bytes32 externalID = keccak256(abi.encodePacked(internalID,
_metaBurnTransaction.receiveSide, _metaBurnTransaction.revertableAddress,
_metaBurnTransaction.chainID)); // external ID

            bytes memory out = abi.encodeWithSelector(
                bytes4(
                    keccak256(
                        bytes(
"metaUnsynthesize(uint256,bytes32,address,uint256,address,address,bytes)"
                        )
                    )
                ),
                _metaBurnTransaction.stableBridgingFee, //SlowMist// It can be passed
in by the user, and 0 can be passed in for free use
                externalID,
                _metaBurnTransaction.chain2address, //SlowMist//  chain2address
needCheck
                _metaBurnTransaction.amount,
                rtoken,
                _metaBurnTransaction.finalDexRouter,//SlowMist//finalDexRouter
needCheck
                _metaBurnTransaction.swapCallData
            );

            requests[externalID] = TxState({
                recipient: _metaBurnTransaction.syntCaller,
                chain2address: _metaBurnTransaction.chain2address,
                token: rtoken,
                stoken: _metaBurnTransaction.sToken,
                amount: _metaBurnTransaction.amount,
                state: RequestState.Sent
            });

            requestCount++;

            IBridge(bridge).transmitRequestV2(
                out,
                _metaBurnTransaction.receiveSide,
                _metaBurnTransaction.oppositeBridge,
                _metaBurnTransaction.chainID
            );
        }

        emit BurnRequest(
            internalID,
            _metaBurnTransaction.syntCaller,
            _metaBurnTransaction.chainID,
            _metaBurnTransaction.revertableAddress,
            _metaBurnTransaction.chain2address,
            _metaBurnTransaction.amount,
            _metaBurnTransaction.sToken
        );
    }
```

Solution

`_metaBurnTransaction` verifies the important parameters in the structure in detail.

Status

Fixed; Communicated with the project party:If the user constructs malicious parameters, relayers will not even send this transaction.

[N3] [Suggestion] Event log missing

Category: Malicious Event Log Audit

Content

- contracts/synth-contracts/Synthesis.sol
- contracts/synth-contracts/SynthesisSolana.sol

Modifying important parameters in the contract requires corresponding event records.

```
/// ** ONLYOWNER functions **

/**
 * @notice Set paused flag to true
 */
function pause() external onlyOwner {
    paused = true;
}

/**
 * @notice Set paused flag to false
 */
function unpause() external onlyOwner {
    paused = false;
}
```

```
    /**
     * @notice Sets minimal price for token
     * @param _token Address of token to set threshold
     * @param _threshold threshold to set
     */
    function setTokenThreshold(address _token, uint256 _threshold) external onlyOwner
{
        tokenThreshold[_token] = _threshold;
    }

    /**
     * @notice Sets MetaRouter address
     * @param _metaRouter Address of metaRouter
     */
    function setMetaRouter(IMetaRouterV2 _metaRouter) external onlyOwner {
        require(address(_metaRouter) != address(0), "Symb: metaRouter cannot be zero
address");
        metaRouter = _metaRouter;
    }

    /**
     * @notice Sets Fabric address
     * @param _fabric Address of fabric
     */
    function setFabric(address _fabric) external onlyOwner {
        require(fabric == address(0x0), "Symb: Fabric already set");
        fabric = _fabric;
    }
```

Solution

Record key events

Status

Ignored; Communicated with the project party:they decided that these events do not need to be recorded.

[N4] [Suggestion] Event log missing

Category: Malicious Event Log Audit

Content

- contracts/synth-contracts/Portal.sol

Modifying important variables in the contract requires corresponding event records.

```
/**
    * @notice Set paused flag to true
    */
   function pause() external onlyOwner {
       paused = true;
   }

   /**
    * @notice Set paused flag to false
    */
   function unpause() external onlyOwner {
       paused = false;
   }

 /**
    * @notice Sets token to tokenWhitelist
    * @param _token Address of token to add to whitelist
    * @param _activate true - add to whitelist, false - remove from whitelist
    */
   function setWhitelistToken(address _token, bool _activate) external onlyOwner {
       tokenWhitelist[_token] = _activate;
   }

   /**
    * @notice Sets minimal price for token
    * @param _token Address of token to set threshold
    * @param _threshold threshold to set
    */
   function setTokenThreshold(address _token, uint256 _threshold) external onlyOwner
{
       tokenThreshold[_token] = _threshold;
   }

   /**
    * @notice Sets MetaRouter address
    * @param _metaRouter Address of metaRouter
    */
   function setMetaRouter(IMetaRouterV2 _metaRouter) external onlyOwner {
       require(address(_metaRouter) != address(0), "Symb: metaRouter cannot be zero
address");
       metaRouter = _metaRouter;
   }
```

Solution

Record key events

Status

Ignored; Communicated with the project party, they decided that these events do not need to be recorded.

[N5] [Suggestion] Risk of excessive authority

Category: Authority Control Vulnerability

Content

-

If the private key of the MPC role is leaked, a large amount of `stoken` can be obtained by constructing a fake `mintSyntheticToken` calldata. Or construct a fake `revertSynthesize` to transfer `rtoken` away

line 99-104 : `function receiveRequestV2(bytes memory _callData, address _receiveSide) external onlyMPC`

line109-114 : `function receiveRequestV2Signed(bytes memory _callData, address _receiveSide, bytes memory signature) external onlySignedByMPC(keccak256(bytes.concat("receiveRequestV2", _callData, bytes20(_receiveSide))), signature)`

If the owner's private key is leaked, the address of the MPC role can be changed through `changeMPC` to launch an attack.

line152-164 : `changeMPC(address _newMPC) external onlyOwnerOrMPC returns (bool)`

Solution

Need to verify MPC and owner role security.

Status

Ignored; Communicated with the project party:The owner is a multi-signature address and the MPC address is MPC-based (Multi-Party Computation) Threshold Signature Scheme.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002202220002	SlowMist Security Team	2022.02.07 - 2022.02.22	Passed

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 low risk, 4 suggestion vulnerabilities. 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.