Smart Contract Security Audit Report

Contents

1. Executive Summary

On Mar. 01, 2021, the SlowMist security team received the Hot Cross team's security audit application for Hot Cross, 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.

SlowMist Smart Contract DeFi project test method:

Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code module 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.

SlowMist Smart Contract DeFi project risk level:

Critical vulnerabilities	Critical vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High-risk vulnerabilities	High-risk vulnerabilities will affect the normal operation of DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium-risk vulnerabilities	Medium vulnerability will affect the operation of DeFi project. It is recommended to fix medium-risk vulnerabilities.

Low-risk vulnerabilities	Low-risk vulnerabilities may affect the operation of DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weaknesses	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Enhancement Suggestions	There are better practices for coding or architecture.

2. Audit Methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and in-house 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 attack and other Race Conditions
- Replay attack
- Reordering attack
- Short address attack
- Denial of service attack
- Transaction Ordering Dependence attack
- Conditional Completion attack
- Authority Control attack
- Integer Overflow and Underflow attack

- TimeStamp Dependence attack
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Explicit visibility of functions state variables
- Logic Flaws
- Uninitialized Storage Pointers
- Floating Points and Numerical Precision
- tx.origin Authentication
- "False top-up" Vulnerability
- Scoping and Declarations

3. Project Background

3.1 Project Introduction

Audit version file information

Initial audit file

https://github.com/hotcrosscom/proj-bsc-bridge-v1-solidity

commit: 5c0e5c03c868f1f3d0e4eb0ef1349599e6d64fef

Fixed file:

https://github.com/hotcrosscom/proj-bsc-bridge-v1-solidity

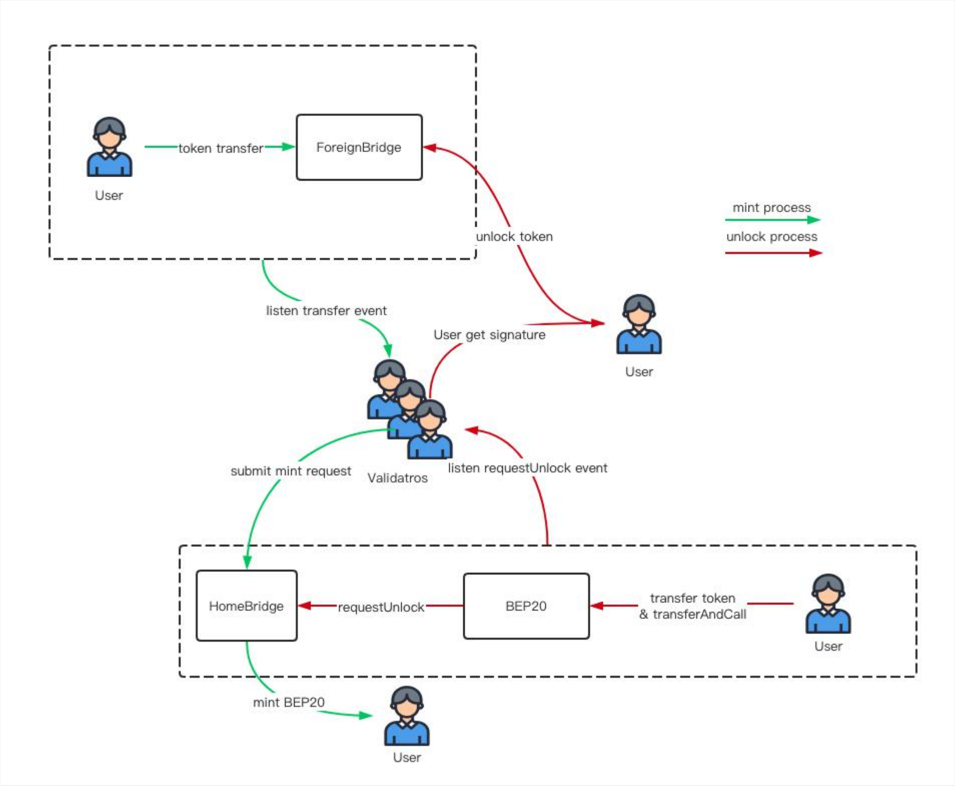
commit: 85d9b09f2c7f1b0c830e6982667a07336fd1d1b2

3.2 Project Structure

```
contracts
├── BaseBridge.sol
├── Migrations.sol
├── ResponseCodes.sol
├── foreign
│   ├── ForeignBridge.sol
│   ├── ForeignBridgeStore.sol
│   ├── ForeignProcessor.sol
│   └── UnlockProcessor.sol
├── home
│   ├── BEP20.sol
│   ├── HomeBridge.sol
│   ├── HomeBridgeStore.sol
│   ├── HomeRequest.sol
│   ├── MintRequest.sol
│   └── UnlockRequest.sol
├── libs
│   ├── ECDSA.sol
│   └── Misc.sol
├── mocks
│   ├── foreign
│   │   ├── BatchLock.sol
│   │   ├── ERC20Mock.sol
│   │   └── ForeignBridgeMock.sol
│   ├── home
│   │   ├── HomeBridgeMock.sol
│   │   └── UnlockMock.sol
│   └── validator
│       └── ValidatorRegistryMock.sol
├── utils
│   ├── Guard.sol
│   └── Store.sol
└── validator
    └── ValidatorRegistry.sol
```

3.3 Contract Structure

Hot Cross Binance Smart Chain Bridge V1 is mainly divided into two parts, namely the HomeBridge contract deployed on the Binance Smart Chain and the ForeignBridge contract deployed on the Ethereum chain. Users can deposit tokens into the ForeignBridge contract, and validators will mint the corresponding BEP tokens through Binance Smart Chain's HomeBridge contract after listening to the corresponding transfer event. Conversely, the user can transfer tokens into the BEP20 contract of Binance Smart Chain, and then call the corresponding transferAndCall function to initiate an unlock request. After the validators listen the corresponding unlock request, they will first submit the corresponding signatures in the HomeBridge contract of Binance Smart Chain. After the mult-sig process is completed, the user can unlock the corresponding token in the ForeignBridge contract on the Ethereum chain by providing the signatures. The overall system architecture diagram is as follows:



4. Code Overview

4.1 Contract Visibility Analysis

ForeignBridge			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	can modify state	initializer
withdrawTokens	External	can modify state	onlyOwner

ForeignBridge			
Function Name	Visibility	Mutability	Modifiers
__BaseBridge_init	Internal	can modify state	initializer
receive	External	payable	-

ValidatorRegistry			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	can modify state	initializer
isValidator	Public	-	-
addValidator	Public	can modify state	onlyOwner
removeValidator	External	can modify state	onlyOwner
setQuorum	Public	can modify state	onlyOwner

ValidatorRegistry			
Function Name	Visibility	Mutability	Modifiers
__UnlockProcessor_init	Internal	can modify state	initializer
canUnlock	External	-	-
unlock	External	can modify state	-
executeRequest	Private	can modify state	-

ECDSA			
Function Name	Visibility	Mutability	Modifiers
isMessageValid	Public	-	-
arrayContains	Public	-	-
recoverAddress	External	-	-
verifyValidatorSigs	External	-	-
hashMessage	Internal	-	-
parseMessage	Internal	-	-

ForeignProcessor			
Function Name	Visibility	Mutability	Modifiers
__ForeignProcessor_init	Internal	can modify state	initializer

ForeignBridgeStore

Function Name	Visibility	Mutability	Modifiers
isUnlockRequestProcessed	Public	-	-
setUnlockRequestProcessed	Internal	can modify state	-

BEP20

Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	ERC20
mint	Public	can modify state	onlyOwner
_beforeTokenTransfer	Internal	can modify state	-
transferAndCall	External	can modify state	-

HomeBridge

Function Name	Visibility	Mutability	Modifiers
initialize	Public	can modify state	initializer
withdrawTokens	External	can modify state	onlyOwner

MintRequest

Function Name	Visibility	Mutability	Modifiers
__MintRequest_init	Internal	can modify state	initializer
canProcessMintRequest	External	-	-
processRequest	External	can modify state	onlyValidator

executeRequest	Private	can modify state	-

HomeRequest			
Function Name	Visibility	Mutability	Modifiers
__HomeRequest_init	Internal	can modify state	initializer

HomeBridgeStore			
Function Name	Visibility	Mutability	Modifiers
requestExists	Public	-	-
storeRequest	Internal	can modify state	-
getMsgSignatures	Public	-	-
setMsgSignatures	Internal	can modify state	-
isMsgProcessed	Public	-	-
setMsgProcessed	Internal	can modify state	-
storeSignature	Internal	can modify state	-
getSignature	External	-	-
storeMessage	Internal	can modify state	-
readMessage	Public	-	-

UnlockRequest			
Function Name	Visibility	Mutability	Modifiers
__UnlockRequest_init	Internal	can modify state	initializer

requestUnlock	External	can modify state	-
canSubmitUnlockSignature	External	-	-
submitUnlockSignature	External	can modify state	onlyValidator

Guard			
Function Name	Visibility	Mutability	Modifiers
__Guard_init	Internal	can modify state	initializer
pause	External	can modify state	onlyOwner
unpause	External	can modify state	onlyOwner

4.2 Code Audit

4.2.1 Enhancement Suggestions

4.2.1.1 __Guard_init function was not call when initialize

ForeignProcessor contract and HomeRequest contract are inherit the Guard contract but dost not call the __Gurad_init function

```
function __ForeignProcessor_init(
      IERC20 token,
      ValidatorRegistry validatorRegistry_
   ) internal initializer {
      //SlowMist// Missing call of __Guard_init function


      require(
         address(token) != address(0) && Misc.isContract(address(token)),
         "ForeignProcessor:Invalid erc20 address provided"
```

```
    );

    require(
        address(validatorRegistry_) != address(0) && Misc.isContract(address(validatorRegistry_)),
        "ForeignProcessor:Invalid validator registry address"
    );

    erc20 = token;
    validatorRegistry = validatorRegistry_;
}

function __HomeRequest_init(
    BEP20 token,
    ValidatorRegistry validatorRegistry_
) internal initializer {
    //SlowMist// Missing call of __Guard_init function


    require(
        address(token) != address(0) && Misc.isContract(address(token)),
        "HomeRequest:Invalid BEP20 address"
    );
    require(
        address(validatorRegistry_) != address(0) && Misc.isContract(address(validatorRegistry_)),
        "HomeRequest:Invalid validator registry address"
    );

    bep20 = token;
    validatorRegistry = validatorRegistry_;
}
```

Fix status: fixed.

5. Audit Result

5.1 Conclusion

Audit Result : Passed

Audit Number : 0X002103020004

Audit Date : Mar. 02, 2021

Audit Team : SlowMist Security Team

Summary conclusion: The SlowMist security team use a manual and SlowMist Team in-house analysis tool audit of the codes for security issues. There is one enhancement suggestions. After communication and feedback with the Hot Cross team, it was confirmed that the risks found in the audit process were repaired or within a tolerable range.

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 base 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 this 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.

SLOWMIST

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist