Smart Contract Security Audit Report

[2021]

Table Of Contents

1 Executive Summary

On 2021.12.22, the SlowMist security team received the Celer Network team's security audit application for SGNv2 iterative audit, 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

Audit Version:

https://github.com/celer-network/sgn-v2-contracts/tree/main/contracts/pegged

commit: 6c1fef04cbcd6dda054bddb0e46b87038b06ff36

Fixed Version:

https://github.com/celer-network/sgn-v2-contracts/tree/main/contracts/pegged

commit: 435da444dc294898bd2f5430d107be9016f5b15f

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status

NO	Title	Category	Level	Status
N1	Logic defect	Design Logic Audit	Medium	Fixed
N2	Token transfer issue	Others	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:

FraxBridgeToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
mint	External	Can Modify State	onlyBridge
burn	External	Can Modify State	onlyBridge
updateBridge	External	Can Modify State	onlyOwner
getOwner	External	-	-

MintSwapCanonicalTokenPermit			
Function Name	Visibility	Mutability	Modifiers

MintSwapCanonicalTokenPermit			
<Constructor>	Public	Can Modify State	MintSwapCanonicalToken ERC20Permit
decimals	Public	-	-

MintSwapCanonicalTokenPermit			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	MintSwapCanonicalToken ERC20Permit
decimals	Public	-	-

SingleBridgeTokenPermit			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	SingleBridgeToken ERC20Permit
decimals	Public	-	-

MintSwapCanonicalToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	MultiBridgeToken
swapBridgeForCanonical	External	Can Modify State	-
swapCanonicalForBridge	External	Can Modify State	-
setBridgeTokenSwapCap	External	Can Modify State	onlyOwner

MultiBridgeToken

MultiBridgeToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
mint	External	Can Modify State	-
burn	External	Can Modify State	-
decimals	Public	-	-
updateBridgeSupplyCap	External	Can Modify State	onlyOwner
getOwner	External	-	-

SingleBridgeToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
mint	External	Can Modify State	onlyBridge
burn	External	Can Modify State	onlyBridge
decimals	Public	-	-
updateBridge	External	Can Modify State	onlyOwner
getOwner	External	-	-

SwapBridgeToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
mint	External	Can Modify State	onlyBridge

SwapBridgeToken			
burn	External	Can Modify State	onlyBridge
updateBridge	External	Can Modify State	onlyOwner
approveCanonical	External	Can Modify State	onlyOwner
revokeCanonical	External	Can Modify State	onlyOwner
getOwner	External	-	-
decimals	Public	-	-

OriginalTokenVault			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit	External	Can Modify State	nonReentrant whenNotPaused
withdraw	External	Can Modify State	whenNotPaused
executeDelayedTransfer	External	Can Modify State	whenNotPaused
setMinDeposit	External	Can Modify State	onlyGovernor
setMaxDeposit	External	Can Modify State	onlyGovernor

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

PeggedTokenBridge			
executeDelayedTransfer	External	Can Modify State	whenNotPaused
setMinBurn	External	Can Modify State	onlyGovernor
setMaxBurn	External	Can Modify State	onlyGovernor

4.3 Vulnerability Summary

[N1] [Medium] Logic defect

Category: Design Logic Audit

Content

In the SwapBridgeToken contract, the bridge contract will first internally call the `_burn` function to burn the bridgeToken during the burn operation, but at this time the MintSwapCanonicalToken contract has not transferred the bridgeToken to the SwapBridgeToken contract, which will cause the contract to have no tokens to burn.

Code location: contracts/pegged/tokens/SwapBridgeToken.sol

```
    function burn(address _from, uint256 _amount) external onlyBridge returns (bool)
{
        _burn(address(this), _amount);
        IERC20(canonical).safeTransferFrom(_from, address(this), _amount);
        ISwapCanoToken(canonical).swapCanonicalForBridge(address(this), _amount);
        return true;
    }
```

Solution

It is recommended to perform the `swapCanonicalForBridge` operation first, and then the `_burn` operation.

Status

Fixed

[N2] [Suggestion] Token transfer issue

Category: Others

Content

The transfer function and transferFrom function are used in the FraxBridgeToken contract to transfer tokens, but the return value is checked.

Code location: contracts/pegged/tokens/FraxBridgeToken.sol

```
    function mint(address _to, uint256 _amount) external onlyBridge returns (bool) {
        _mint(address(this), _amount); // add amount to myself so
exchangeOldForCanonical can transfer amount
        _approve(address(this), canonical, _amount);
        uint256 got =
IFraxCanoToken(canonical).exchangeOldForCanonical(address(this), _amount);
        // now this has canonical token, next step is to transfer to user
        return IERC20(canonical).transfer(_to, got);
    }

    function burn(address _from, uint256 _amount) external onlyBridge returns (bool)
{
        IERC20(canonical).transferFrom(_from, address(this), _amount);
        uint256 got =
IFraxCanoToken(canonical).exchangeCanonicalForOld(address(this), _amount);
        _burn(address(this), got);
        return true;
    }
```

Solution

It is recommended to use SafeERC20 for token transfer.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002112230001	SlowMist Security Team	2021.12.22 - 2021.12.23	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, 1 suggestion vulnerabilities. All the 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