SLOWMIST

Smart Contract Security Audit Report

[2021]

Table Of Contents

1 Executive Summary

2 Audit Methodology

3 Project Overview

3.1 Project Introduction

3.2 Vulnerability Information

4 Code Overview

4.1 Contracts Description

4.2 Visibility Description

4.3 Vulnerability Summary

5 Audit Result

6 Statement

1 Executive Summary

On 2021.09.01, the SlowMist security team received the EdgeSwap team's security audit application for EdgeSwap, 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 Aduit

Design Logic Audit

Scoping and Declarations Audit

3 Project Overview

3.1 Project Introduction

Audit scope:

https://github.com/EdgeSwap/EdgeSwap

Commit:

f3f191a072abb1ae60e44a3b4a0f04a5588c89f0

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Gas optimization	Gas Optimization Audit	Suggestion	Confirming
N2	Risk of upgrade DoS	Denial of Service Vulnerability	Low	Confirming

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:

ZkSync			
Function Name	Visibility	Mutability	Modifiers
createPair	External	Can Modify State	requireActive requireGovernor
createETHPair	External	Can Modify State	requireActive requireGovernor
registerCreatePair	Internal	Can Modify State	-
getNoticePeriod	External	-	-
upgradeNoticePeriodStarted	External	Can Modify State	-
upgradePreparationStarted	External	Can Modify State	-
upgradeCanceled	External	Can Modify State	-
upgradeFinishes	External	Can Modify State	-
isReadyForUpgrade	External	-	-
initialize	External	Can Modify State	initializeReentrancyGuard

ZkSync			
addPriorityRequest	Internal	Can Modify State	-
upgrade	External	Can Modify State	nonReentrant
_transferERC20	External	Can Modify State	-
requireActive	Internal	-	-
cancelOutstandingDepositsForExodusMode	External	Can Modify State	nonReentrant exodusMode
depositETH	External	Payable	requireActive
depositERC20	External	Can Modify State	nonReentrant requireActive
getPendingBalance	Public	-	-
getBalanceToWithdraw	Public	-	-
requestFullExit	Public	Can Modify State	nonReentrant requireActive
activateExodusMode	Public	Can Modify State	-
registerDeposit	Internal	Can Modify State	-
<Fallback>	External	Payable	-

ZkSyncCommitBlock			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	-
commitOneBlock	Internal	-	-

ZkSyncCommitBlock			
commitBlocks	External	Can Modify State	nonReentrant requireActive
proveBlocks	External	Can Modify State	nonReentrant requireActiveValidator
revertBlocks	External	Can Modify State	nonReentrant requireActiveValidator
registerWithdrawal	Internal	Can Modify State	-
increaseBalanceToWithdraw	Internal	Can Modify State	-
_transferERC20	External	Can Modify State	-
withdrawPendingBalance	External	Can Modify State	nonReentrant
withdrawOrStore	Internal	Can Modify State	-
executeOneBlock	Internal	Can Modify State	-
emitDepositCommitEvent	Internal	Can Modify State	-
emitFullExitCommitEvent	Internal	Can Modify State	-
emitCreatePairCommitEvent	Internal	Can Modify State	-
collectOnchainOps	Internal	-	-
verifyChangePubkey	Internal	-	-
verifyChangePubkeyECRECOVER	Internal	-	-
verifyChangePubkeyCREATE2	Internal	-	-
createBlockCommitment	Internal	-	-

ZkSyncCommitBlock			
checkPriorityOperation	Internal	-	-
checkPriorityOperation	Internal	-	-
checkPriorityOperation	Internal	-	-
executeBlocks	External	Can Modify State	nonReentrant requireActive requireActiveValidator
requireActive	Internal	-	-
sendETHNoRevert	Internal	Can Modify State	-
performExodus	External	Can Modify State	nonReentrant
updateBalance	Internal	Can Modify State	-
checkLpL1Balance	Internal	Can Modify State	-
checkPairAccount	Internal	-	-
lpExit	External	Can Modify State	nonReentrant

Proxy			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
initialize	External	-	-
upgrade	External	-	-
getTarget	Public	-	-
setTarget	Internal	Can Modify State	-

Proxy			
upgradeTarget	External	Can Modify State	requireMaster
_fallback	Internal	Can Modify State	-
<Fallback>	External	Payable	-
<Receive Ether>	External	Payable	-
getNoticePeriod	External	Can Modify State	-
upgradeNoticePeriodStarted	External	Can Modify State	requireMaster
upgradePreparationStarted	External	Can Modify State	requireMaster
upgradeCanceled	External	Can Modify State	requireMaster
upgradeFinishes	External	Can Modify State	requireMaster
isReadyForUpgrade	External	Can Modify State	-

UpgradeGatekeeper			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
addUpgradeable	External	Can Modify State	requireMaster
startUpgrade	External	Can Modify State	requireMaster
cancelUpgrade	External	Can Modify State	requireMaster
startPreparation	External	Can Modify State	requireMaster
finishUpgrade	External	Can Modify State	requireMaster

UniswapV2Factory

UniswapV2Factory			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	-
setZkSyncAddress	External	Can Modify State	zkSyncAddress
upgrade	External	Can Modify State	-
allPairsLength	External	-	-
createPair	External	Can Modify State	zkSyncAddress
mint	External	Can Modify State	zkSyncAddress
burn	External	Can Modify State	zkSyncAddress

Governance			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	-
upgrade	External	Can Modify State	-
changeGovernor	External	Can Modify State	requireGovernor
addToken	External	Can Modify State	requireGovernor
setTokenPaused	External	Can Modify State	requireGovernor
setValidator	External	Can Modify State	requireGovernor
requireGovernor	Public	-	-
requireActiveValidator	External	-	-

Governance			
isValidTokenId	External	-	-
validateTokenAddress	External	-	-

4.3 Vulnerability Summary

[N1] [Suggestion] Gas optimization

Category: Gas Optimization Audit

Content

The execution of `setTokenPaused` will return success regardless of whether the modification is successful or not. It is recommended to use `require` to return if the execution fails.

```
function setTokenPaused(address _tokenAddr, bool _tokenPaused) external {
    requireGovernor(msg.sender);

    uint16 tokenId = this.validateTokenAddress(_tokenAddr);
    if (pausedTokens[tokenId] != _tokenPaused) {
        pausedTokens[tokenId] = _tokenPaused;
        emit TokenPausedUpdate(_tokenAddr, _tokenPaused);
    }
}
```

The execution of `changeGovernor` will return success regardless of whether the modification is successful or not. It is recommended to use `require` to return if the execution fails.

```
function changeGovernor(address _newGovernor) external {
        requireGovernor(msg.sender);
        if (networkGovernor != _newGovernor) {
            networkGovernor = _newGovernor;
            emit NewGovernor(_newGovernor);
        }
    }
```

The execution of `setValidator` will return success regardless of whether the modification is successful or not. It is recommended to use `require` to return if the execution fails.

```
function setValidator(address _validator, bool _active) external {
        requireGovernor(msg.sender);
        if (validators[_validator] != _active) {
            validators[_validator] = _active;
            emit ValidatorStatusUpdate(_validator, _active);
        }
    }
```

Solution

1. Change `if (pausedTokens[tokenId] != _tokenPaused)` to `require(pausedTokens[tokenId] == _tokenPaused, "");`
2. Change `if (networkGovernor != _newGovernor)` to `require(networkGovernor == _newGovernor, "");`
3. Change `if (validators[_validator] != _active)` to `require(validators[_validator] == _active, "");`

Status

Confirming

[N2] [Low] Risk of upgrade DoS

Category: Denial of Service Vulnerability

Content

There are more and more contracts managed by `UpgradeGatekeeper`, and the list of `managedContracts` will continue to grow. If the recursion depth exceeds its recursion depth, it will cause DoS problems.

```
    function finishUpgrade(bytes[] calldata targetsUpgradeParameters) external {
        requireMaster(msg.sender);
        require(upgradeStatus == UpgradeStatus.Preparation, "fpu11"); // fpu11 -
unable to finish upgrade without preparation status active
```

```
        require(targetsUpgradeParameters.length == managedContracts.length, "fpu12");
// fpu12 - number of new targets upgrade parameters must be equal to the number of
managed contracts
        require(mainContract.isReadyForUpgrade(), "fpu13"); // fpu13 - main contract
is not ready for upgrade
        mainContract.upgradeFinishes();

        for (uint64 i = 0; i < managedContracts.length; i++) {
            address newTarget = nextTargets[i];
            if (newTarget != address(0)) {
                managedContracts[i].upgradeTarget(newTarget,
targetsUpgradeParameters[i]);
            }
        }
        versionId++;
        emit UpgradeComplete(versionId, nextTargets);

        upgradeStatus = UpgradeStatus.Idle;
        noticePeriodFinishTimestamp = 0;
        delete nextTargets;
    }
```

Solution

Control the maximum limit, or add and delete logic

Status

Confirming

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002109100002	SlowMist Security Team	2021.09.01 - 2021.09.10	Low 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 medium 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.