Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.05.25, the SlowMist security team received the Razor Network team's security audit application for Razor Network, 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

Project address:

https://github.com/razor-network/contracts

Module:

Core module + Token + Random module

Commit:

8e9324759f95fa41c2a7e8b95311267fed0aa970

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Risk of excessive authority	Authority Control Vulnerability	Low	Ignored
N2	Event log missing	Malicious Event Log Audit	Suggestion	Confirming
N3	Event log missing	Malicious Event Log 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:

BlockManagerParams			
Function Name	Visibility	Mutability	Modifiers
setMaxAltBlocks	External	Can Modify State	onlyRole
setBufferLength	External	Can Modify State	onlyRole
setBlockReward	External	Can Modify State	onlyRole
setMinStake	External	Can Modify State	onlyRole

CollectionManagerParams			
Function Name	Visibility	Mutability	Modifiers
setMaxTolerance	External	Can Modify State	onlyRole
setBufferLength	External	Can Modify State	onlyRole

RandomNoManagerParams			
Function Name	Visibility	Mutability	Modifiers
setBufferLength	External	Can Modify State	onlyRole

RewardManagerParams			
Function Name	Visibility	Mutability	Modifiers
setPenaltyNotRevealNum	External	Can Modify State	onlyRole
setBlockReward	External	Can Modify State	onlyRole
setGracePeriod	External	Can Modify State	onlyRole

RewardManagerParams			
setMaxAge	External	Can Modify State	onlyRole
setMaxTolerance	External	Can Modify State	onlyRole
setMaxCommission	External	Can Modify State	onlyRole

StakeManagerParams			
Function Name	Visibility	Mutability	Modifiers
setSlashParams	External	Can Modify State	onlyRole
setDeltaCommission	External	Can Modify State	onlyRole
setEpochLimitForUpdateCommission	External	Can Modify State	onlyRole
setUnstakeLockPeriod	External	Can Modify State	onlyRole
setWithdrawLockPeriod	External	Can Modify State	onlyRole
setWithdrawInitiationPeriod	External	Can Modify State	onlyRole
setResetUnstakeLockPenalty	External	Can Modify State	onlyRole
setMinStake	External	Can Modify State	onlyRole
setMinSafeRazor	External	Can Modify State	onlyRole
setGracePeriod	External	Can Modify State	onlyRole
setMaxCommission	External	Can Modify State	onlyRole
disableEscapeHatch	External	Can Modify State	onlyRole
setBufferLength	External	Can Modify State	onlyRole

VoteManagerParams

VoteManagerParams			
Function Name	Visibility	Mutability	Modifiers
setMinStake	External	Can Modify State	onlyRole
setToAssign	External	Can Modify State	onlyRole
setBufferLength	External	Can Modify State	onlyRole

ACL			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-

Governance			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer onlyRole
setPenaltyNotRevealNum	External	Can Modify State	initialized onlyRole
setSlashParams	External	Can Modify State	initialized onlyRole
setUnstakeLockPeriod	External	Can Modify State	initialized onlyRole
setWithdrawLockPeriod	External	Can Modify State	initialized onlyRole
setWithdrawInitiationPeriod	External	Can Modify State	initialized onlyRole
setResetUnstakeLockPenalty	External	Can Modify State	initialized onlyRole
setMaxAltBlocks	External	Can Modify State	initialized onlyRole
setMinStake	External	Can Modify State	initialized onlyRole
setMinSafeRazor	External	Can Modify State	initialized onlyRole

Governance			
setBlockReward	External	Can Modify State	initialized onlyRole
setGracePeriod	External	Can Modify State	initialized onlyRole
setMaxAge	External	Can Modify State	initialized onlyRole
setMaxCommission	External	Can Modify State	initialized onlyRole
disableEscapeHatch	External	Can Modify State	initialized onlyRole
setDeltaCommission	External	Can Modify State	onlyRole
setEpochLimitForUpdateCommission	External	Can Modify State	onlyRole
setMaxTolerance	External	Can Modify State	onlyRole
setToAssign	External	Can Modify State	onlyRole
setBufferLength	External	Can Modify State	onlyRole

BlockManager			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer onlyRole
propose	External	Can Modify State	initialized checkEpochAndState
giveSorted	External	Can Modify State	initialized checkEpochAndState
resetDispute	External	Can Modify State	initialized checkEpochAndState
claimBlockReward	External	Can Modify State	initialized checkState
confirmPreviousEpochBlock	External	Can Modify State	initialized onlyRole

BlockManager			
disputeBiggestStakeProposed	External	Can Modify State	initialized checkEpochAndState
disputeCollectionIdShouldBeAbsent	External	Can Modify State	initialized checkEpochAndState
disputeCollectionIdShouldBePresent	External	Can Modify State	initialized checkEpochAndState
disputeOnOrderOfIds	External	Can Modify State	initialized checkEpochAndState
finalizeDispute	External	Can Modify State	initialized checkEpochAndState
getBlock	External	-	-
getProposedBlock	External	-	-
getNumProposedBlocks	External	-	-
isBlockConfirmed	External	-	-
getLatestResults	External	-	-
_confirmBlock	Internal	Can Modify State	-
_insertAppropriately	Internal	Can Modify State	-
_executeDispute	Internal	Can Modify State	-
_isElectedProposer	Internal	-	initialized

StakeManager			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer onlyRole

StakeManager			
stake	External	Can Modify State	initialized checkEpoch whenNotPaused
delegate	External	Can Modify State	initialized whenNotPaused
unstake	External	Can Modify State	initialized whenNotPaused
initiateWithdraw	External	Can Modify State	initialized whenNotPaused
unlockWithdraw	External	Can Modify State	initialized whenNotPaused
claimStakerReward	External	Can Modify State	initialized whenNotPaused
escape	External	Can Modify State	initialized onlyRole whenPaused
srzrTransfer	External	Can Modify State	onlyRole
setDelegationAcceptance	External	Can Modify State	-
updateCommission	External	Can Modify State	-
resetUnstakeLock	External	Can Modify State	initialized whenNotPaused
setStakerStake	External	Can Modify State	onlyRole
setStakerReward	External	Can Modify State	onlyRole
slash	External	Can Modify State	onlyRole
redeemBounty	External	Can Modify State	-
setStakerEpochFirstStakedOrLastPenalized	External	Can Modify State	onlyRole

StakeManager			
setStakerAge	External	Can Modify State	onlyRole
getStakerId	External	-	-
getStaker	External	-	-
getNumStakers	External	-	-
getAge	External	-	-
getInfluence	External	-	-
getStake	External	-	-
getEpochFirstStakedOrLastPenalized	External	-	-
maturitiesLength	External	-	-
_setStakerStake	Internal	Can Modify State	-
_setStakerReward	Internal	Can Modify State	-
_isStakerActive	Internal	-	-
_getMaturity	Internal	-	-
_convertSRZRToRZR	Internal	-	-
_convertRZRtoSRZR	Internal	-	-
_resetLock	Private	Can Modify State	-

StateManager			
Function Name	Visibility	Mutability	Modifiers

StateManager			
_getEpoch	Internal	-	-
_getState	Internal	-	-

VoteManager			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer onlyRole
commit	External	Can Modify State	initialized checkEpochAndState
reveal	External	Can Modify State	initialized checkEpochAndState
snitch	External	Can Modify State	initialized checkEpochAndState
storeSalt	External	Can Modify State	onlyRole
storeDepth	External	Can Modify State	onlyRole
getCommitment	External	-	-
getVoteValue	External	-	-
getVoteWeight	External	-	-
getInfluenceSnapshot	External	-	-
getStakeSnapshot	External	-	-
getTotalInfluenceRevealed	External	-	-
getEpochLastCommitted	External	-	-
getEpochLastRevealed	External	-	-
getSalt	External	-	-

VoteManager			
_isAssetAllotedToStaker	Internal	-	initialized
_prng	Internal	-	-

RandomNoManager			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer onlyRole
register	External	Can Modify State	initialized
provideSecret	External	Can Modify State	onlyRole
getRandomNumber	External	-	-
getGenericRandomNumberOfLastEpoch	External	-	-
getGenericRandomNumber	External	-	-
_generateRandomNumber	Internal	-	-

RewardManager			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer onlyRole
givePenalties	External	Can Modify State	initialized onlyRole
giveBlockReward	External	Can Modify State	onlyRole
giveInactivityPenalties	External	Can Modify State	onlyRole
_giveInactivityPenalties	Internal	Can Modify State	-
_givePenalties	Internal	Can Modify State	-

RewardManager			
_calculateInactivityPenalties	Internal	-	-

CollectionManager			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer onlyRole
createJob	External	Can Modify State	onlyRole
updateJob	External	Can Modify State	onlyRole notState
setCollectionStatus	External	Can Modify State	onlyRole checkState
createCollection	External	Can Modify State	onlyRole checkState
updateCollection	External	Can Modify State	onlyRole notState
updateDelayedRegistry	External	Can Modify State	onlyRole
getJob	External	-	-
getCollection	External	-	-
getResult	External	-	-
getCollectionStatus	External	-	-
getCollectionTolerance	External	-	-
getCollectionPower	External	-	-
getCollectionID	External	-	-
getNumJobs	External	-	-
getNumCollections	External	-	-

CollectionManager			
getNumActiveCollections	External	-	-
getUpdateRegistryEpoch	External	-	-
getLeafIdOfCollection	External	-	-
getLeafIdOfCollectionForLastEpoch	External	-	-
getCollectionIdFromLeafId	External	-	-
getActiveCollections	External	-	-
getResultFromID	Public	-	-
_updateRegistry	Internal	Can Modify State	-
_updateDelayedRegistry	Internal	Can Modify State	-
_setIDName	Internal	Can Modify State	-
_getDepth	Internal	-	-

RAZOR			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20

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

StakedToken			
getRZRDeposited	Public	-	-
_beforeTokenTransfer	Internal	Can Modify State	-

StakedTokenFactory			
Function Name	Visibility	Mutability	Modifiers
createStakedToken	External	Can Modify State	-

4.3 Vulnerability Summary

[N1] [Low] Risk of excessive authority

Category: Authority Control Vulnerability

Content

- contracts/Core/StakeManager.sol

If the `DEFAULT_ADMIN_ROLE` permission is controlled by the attacker, then the attacker can set an `ESCAPE_HATCH_ROLE` role and `PAUSE_ROLE` to transfer the contract tokens through the escape function.

```
function escape(address _address) external override initialized
onlyRole(ESCAPE_HATCH_ROLE) whenPaused {
        if (escapeHatchEnabled) {
            // Ignoring below line for testing as this is standard erc20 function
            require(razor.transfer(_address, razor.balanceOf(address(this))), "razor
transfer failed");
        } else {
            revert("escape hatch is disabled");
        }
    }
```

If the `DEFAULT_ADMIN_ROLE` permission is controlled by the attacker, the attacker can set the `STAKE_MODIFIER_ROLE` role and call the `setStakerStake` function to modify the amount of stake in the stake pool.

```
function setStakerStake(
    uint32 _epoch,
    uint32 _id,
    Constants.StakeChanged reason,
    uint256 prevStake,
    uint256 _stake
) external override onlyRole(STAKE_MODIFIER_ROLE) {
    _setStakerStake(_epoch, _id, reason, prevStake, _stake);
}
```

If the `DEFAULT_ADMIN_ROLE` permission is controlled by the attacker, the attacker can set the `STAKE_MODIFIER_ROLE` role, and then call `setStakerStakerReward` to set the reward, and then the token of the contract can be taken away.

```
/// @inheritdoc IStakeManager
function setStakerStakerReward(
    uint32 _epoch,
    uint32 _id,
    Constants.StakerRewardChanged reason,
    uint256 prevStakerReward,
    uint256 _stakerReward
) external override onlyRole(STAKE_MODIFIER_ROLE) {
    _setStakerStakerReward(_epoch, _id, reason, prevStakerReward, _stakerReward);
}
```

Solution

It is recommended to transfer the permissions of `DEFAULT_ADMIN_ROLE` to the governance contract or use multi-signature for management.

Status

Ignored; Role management will be a multi-signature account.

[N2] [Suggestion] Event log missing

Category: Malicious Event Log Audit

Content

We recommend that all calls to key functions need to record events to facilitate subsequent self-examination and community review.

- contracts/Core/StakeManager.sol

```
function redeemBounty(uint32 bountyId) external {
    uint32 epoch = _getEpoch();
    uint256 bounty = bountyLocks[bountyId].amount;

    require(msg.sender == bountyLocks[bountyId].bountyHunter, "Incorrect Caller");
    // slither-disable-next-line timestamp
    require(bountyLocks[bountyId].redeemAfter <= epoch, "Redeem epoch not reached");
    delete bountyLocks[bountyId];
    // Ignoring below line for testing as this is standard erc20 function
    require(razor.transfer(msg.sender, bounty), "couldnt transfer");
}
```

- contracts/Core/VoteManager.sol

```
function snitch(
    uint32 epoch,
    bytes32 root,
    bytes32 secret,
    address stakerAddress
) external initialized checkEpochAndState(State.Commit, epoch, buffer) {
    require(msg.sender != stakerAddress, "cant snitch on yourself");
    uint32 thisStakerId = stakeManager.getStakerId(stakerAddress);
    require(thisStakerId > 0, "Staker does not exist");
    require(commitments[thisStakerId].epoch == epoch, "not committed in this epoch");
    // avoid innocent staker getting slashed due to empty secret
    require(secret != 0x0, "secret cannot be empty");

    bytes32 seed = keccak256(abi.encode(salt, secret));
    require(keccak256(abi.encode(root, seed)) ==
commitments[thisStakerId].commitmentHash, "incorrect secret/value");
```

```
    //below line also avoid double reveal attack since once revealed, commitment has
will be set to 0x0
    commitments[thisStakerId].commitmentHash = 0x0;
    stakeManager.slash(epoch, thisStakerId, msg.sender);
}
```

Solution

Record key events.

Status

Confirming; Fix the issue in this commit 2c66d319719dd6a01ffae998fdac0b7c41b8e749.

[N3] [Suggestion] Event log missing

Category: Malicious Event Log Audit

Content

- contracts/Core/BlockManager.sol

The functions `claimBlockReward`, `giveSorted`, `disputeBiggestStakeProposed`, `disputeCollectionIdShouldBeAbsent`, `disputeCollectionIdShouldBePresent`, `disputeOnOrderOfIds`, `finalizeDispute` do not record the call event.We recommend that all calls to key functions need to record events to facilitate subsequent self-examination and community review.

Solution

Record key events.

Status

Fixed; Fix the issue in this commit 2c66d319719dd6a01ffae998fdac0b7c41b8e749.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002206080001	SlowMist Security Team	2022.05.25 - 2022.06.08	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, 2 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.

SLOWMIST

Official Website

www.slowmist.com

✉

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist