BEOSIN
Blockchain Security

Ronin Bridge

Smart Contract Security Audit

No. 202309081700

Sep 8th, 2023

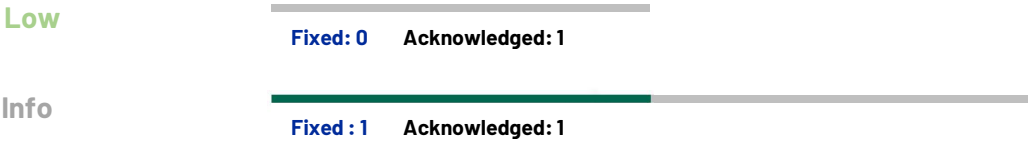
SECURING BLOCKCHAIN ECOSYSTEM

WWW.BEOSIN.COM

Contents

Summary of Audit Results

After auditing, 1 Low-risk and 2 Info items were identified in the Ronin Bridge project. Specific audit details will be presented in the Findings section. Users should pay attention to the following aspects when interacting with this project:



- **Risk Description:**

1. The Ronin Bridge project relies on multiple bridge operator to vote on users' cross-chain transactions. If the private keys of multiple bridge operator are leaked, it will lead to the loss of the whole cross-chain bridge project's funds, so bridge operator are requested to keep their private keys properly and prevent them from being leaked.

2. Bridge operator rewards may suffer when BRIDGE_MANAGER updates PERIOD rewards, read RB-01 vulnerability for more details.

- **Project Description:**

Business overview

Following the completion of the entire Ronin Network project audit last time, the main audit content of this time is the Bridge-related part of the Ronin Network project. Compared with the previous architecture, the Ronin development team has separated the cross-chain Bridge function from the DPOS module, which simplifies the entire Ronin Network project.

According to the code, the Ronin Bridge project is deployed on Ethereum and the Ronin chain. The Ronin Bridge project allows users to stake assets to the Ethereum Bridge contract, and then the Bridge Operator forwards the user's cross-chain request events to the Bridge contract of the Ronin chain, and finally the bridge operator votes. If the bridge operator votes, the user will get the corresponding value of the peg token on the Ronin chain. When users withdraw their funds (withdraw funds from the Ronin chain to the Ethereum chain), the withdrawal of the Ronin Bridge project is divided into three tiers, which limits the user's withdrawal requests, thereby ensuring the security of the entire cross-chain Bridge project. Specific withdrawal limits can be found at this link: https://docs.roninchain.com/docs/basics/dapps/ronin-bridge.

The MainchainBridgeManager and RoninBridgeManager contracts are used to initiate proposals and manage bridge operators. The BridgeReward contract is used to distribute rewards to bridge operators and keep track of the reward distribution throughout the cross-chain project. The BridgeTracking contract is used to keep track of all bridge operator votes. The BridgeSlash contract is used to punish bridge operators for not voting on time by depriving them of their rewards.

1 Overview

1.1 Project Overview

Project Name	Ronin Bridge
Project language	Solidity
Platform	Ethereum/Ronin chain
Github	https://github.com/axieinfinity/Ronin-dpos-contracts/tree/v0.6.1-testnet
Audit scope	contracts/mainchain/MainchainBridgeManager.sol contracts/mainchain/MainchainGatewayV2.sol contracts/Ronin/gateway/RoninBridgeManager.sol contracts/Ronin/gateway/BridgeReward.sol contracts/Ronin/gateway/BridgeSlash.sol contracts/Ronin/gateway/BridgeTracking.sol contracts/Ronin/gateway/PauseEnforcer.sol contracts/Ronin/gateway/RoninGatewayV2.sol contracts/extensions/GatewayV2.sol contracts/extensions/Bridge-operator-governance/BridgeManager.sol contracts/extensions/Bridge-operator-governance/BridgeManagerCallbackRegister.sol contracts/extensions/Bridge-operator-governance/BridgeTrackingHelper.sol contracts/extensions/sequential-governance/CoreGovernance.sol contracts/extensions/sequential-governance/GlobalCoreGovernance.sol contracts/Ronin/validator/CoinbaseExecution.sol contracts/extensions/sequential-governance/governance-proposal/CommonGovernanceProposal.sol contracts/extensions/sequential-governance/governance-proposal/GlobalGovernanceProposal.sol contracts/extensions/sequential-governance/governance-proposal/GovernanceProposal.sol contracts/extensions/sequential-governance/governance-relay/CommonGovernanceRelay.sol contracts/extensions/sequential-governance/governance-relay/GlobalGovernanceRelay.sol contracts/extensions/sequential-governance/governance-relay/GovernanceRelay.solcontracts/extensions/MinimumWithdrawal.sol contracts/extensions/RONTransferHelper.sol contracts/extensions/WithdrawalLimitation.sol

	contracts/extensions/collections/HasContracts.sol contracts/extensions/collections/HasProxyAdmin.sol contracts/utils/IdentityGuard.sol contracts/libraries/AddressArrayUtils.sol contracts/libraries/Ballot.sol contracts/libraries/EmergencyExitBallot.sol contracts/libraries/EnumFlags.sol contracts/libraries/ErrorHandler.sol contracts/libraries/GlobalProposal.sol contracts/libraries/IsolatedGovernance.sol contracts/libraries/Math.sol contracts/libraries/Proposal.sol contracts/libraries/Token.sol contracts/libraries/Transfer.sol https://github.com/axieinfinity/ronin-dpos-contracts/pull/253/files#diff-319e0aba4e8c9facb6e94bdabb55dc58b0d585381348b4f92cebc2f1c9d1b56d
Commit Hash	f4deb89ae8cfe97bed7bd429ac6ed9069ca20215

1.2 Audit Overview

Audit work duration: Aug 22, 2023 – Sep 8, 2023

Audit team: Beosin Security Team

1.3 Audit Method

The audit methods are as follows:

1. Manual Review

Using manual auditing methods, the code is read line by line to identify potential security issues. This ensures that the contract's execution logic aligns with the client's specifications and intentions, thereby safeguarding the accuracy of the contract's business logic.

The manual audit is divided into three groups to cover the entire auditing process:

The Basic Testing Group is primarily responsible for interpreting the project's code and conducting comprehensive functional testing.

The Simulated Attack Group is responsible for analyzing the audited project based on the collected historical audit vulnerability database and security incident attack models. They identify potential attack vectors and collaborate with the Basic Testing Group to conduct simulated attack tests.

The Expert Analysis Group is responsible for analyzing the overall project design, interactions with third parties, and security risks in the on-chain operational environment. They also conduct a review of the entire audit findings.

2. Static Analysis

Static analysis is a method of examining code during compilation or static analysis to detect issues. Beosin-VaaS can detect more than 100 common smart contract vulnerabilities through static analysis, such as reentrancy and block parameter dependency. It allows early and efficient discovery of problems to improve code quality and security.

2 Findings

Index	Risk description	Severity level	Status
RB-01	Wrong bridge operator Rewards	Low	Acknowledged
RB-02	Excessive range of fees that can be set	Info	Acknowledged
RB-03	Wrong length	Info	Fixed

Finding Details:

[RB-01] Wrong bridge operator Rewards

Severity Level	Low
Type	Business Security
Lines	BridgeReward.sol #L320-331
Description	In the `setRewardPerPeriod` function of the BridgeReward contract, when BRIDGE_MANAGER uses the `setRewardPerPeriod` function to update the period reward, because the rewardPerPeriod before the modification will not be used to settle the bridge operator's reward in the `setRewardPerPeriod` function,If the new rewardPerPeriod is less than the previous rewardPerPeriod, the bridge operator's reward will be less. Conversely, if the new rewardPerPeriod is greater than the previous rewardPerPeriod, the bridge operator's reward will be more.
Recommendation	It is recommended to settle bridge operator rewards before changing the rewardPerPeriod.
Status	**Acknowledged.** The project has confirmed the issue, and after their internal discussion, decided not to modify the issue.

```
  function setRewardPerPeriod(uint256 rewardPerPeriod) external
onlyContract(ContractType.BRIDGE_MANAGER) {
    _setRewardPerPeriod(rewardPerPeriod);
  }
  /**
   * @dev Internal function for setting the total reward per period.
   * Emit an {UpdatedRewardPerPeriod} event after set.
   */
  function _setRewardPerPeriod(uint256 rewardPerPeriod) internal {
    REWARD_PER_PERIOD_SLOT.store(rewardPerPeriod);
    emit UpdatedRewardPerPeriod(rewardPerPeriod);
  }
```

[RB-02] Excessive range of fees that can be set

Severity Level	Info
Type	Business Security
Lines	WithdrawalLimitation.sol #L248-261
Description	In the `_setUnlockFeePercentages` function of the WithdrawalLimitation contract, the `_setUnlockFeePercentages` function allows the admin to set the withdrawal fee for a specified token to be close to 100%. If the admin sets the fee too high, it will cause the user to be charged a high fee by `WITHDRAWAL_UNLOCKER_ROLE` when unlocking funds stored in the cross-link bridge.
Recommendation	It is recommended that in the `_setUnlockFeePercentages` function, the handling fee is judged to be a reasonable range.
Status	**Acknowledged.** The project team has confirmed the problem and plans to modify it in the next stage.

```
  function _setUnlockFeePercentages(address[] calldata _tokens,
uint256[] calldata _percentages) internal virtual {

    if (_tokens.length != _percentages.length) revert
ErrLengthMismatch(msg.sig);

    for (uint256 _i; _i < _tokens.length; ) {

      if (_percentages[_i] > _MAX_PERCENTAGE) revert
ErrInvalidPercentage();

      unlockFeePercentages[_tokens[_i]] = _percentages[_i];

      unchecked {

        ++_i;

      }

    }

    emit UnlockFeePercentagesUpdated(_tokens, _percentages);

  }
```

[RB-03] Wrong length

Severity Level	Info
Type	Business Security
Lines	BridgeManager.sol #L488-498
Description	In the `_sumgovernorweight` function of the BridgeManager contract, the `_sumgovernorweight` function incorrectly uses the length of the bridge operator obtained through the `_getBridgeOperatorSet` function. As can be seen from the code, the length of the bridge operator is greater than or equal to the length of the governors. If the length of the governors is less than the length of the bridge operator, it will cause gas waste.
Recommendation	It is recommended to traverse the length of the governors array.
Status	**Fixed.** Project has been in the https://github.com/axieinfinity/ronin-dpos-contracts/pull/273/commits/2fb8af4b172aebbaaf8ec393ed7e5ab078802d7e The branch fixes this problem

```
  function _sumGovernorsWeight(address[] memory governors)
internal view nonDuplicate(governors) returns (uint256 sum) {

    uint256 length = _getBridgeOperatorSet().length();

    mapping(address => BridgeOperatorInfo) storage
_governorToBridgeOperatorInfo =
_getGovernorToBridgeOperatorInfo();

    for (uint256 i; i < length; ) {

      sum +=
_governorToBridgeOperatorInfo[governors[i]].voteWeight;

      unchecked {

        ++i;

      }

    }

  }
```

3 Appendix

3.1 Vulnerability Assessment Metrics and Status in Smart Contracts

3.1.1 Metrics

In order to objectively assess the severity level of vulnerabilities in blockchain systems, this report provides detailed assessment metrics for security vulnerabilities in smart contracts with reference to CVSS 3.1(Common Vulnerability Scoring System Ver 3.1).

According to the severity level of vulnerability, the vulnerabilities are classified into four levels: "critical", "high", "medium" and "low". It mainly relies on the degree of impact and likelihood of exploitation of the vulnerability, supplemented by other comprehensive factors to determine of the severity level.

Likelihood \ Impact	Severe	High	Medium	Low
Probable	Critical	High	Medium	Low
Possible	High	Medium	Medium	Low
Unlikely	Medium	Medium	Low	Info
Rare	Low	Low	Info	Info

3.1.2 Degree of impact

- **Severe**

Severe impact generally refers to the vulnerability can have a serious impact on the confidentiality, integrity, availability of smart contracts or their economic model, which can cause substantial economic losses to the contract business system, large-scale data disruption, loss of authority management, failure of key functions, loss of credibility, or indirectly affect the operation of other smart contracts associated with it and cause substantial losses, as well as other severe and mostly irreversible harm.

- **High**

High impact generally refers to the vulnerability can have a relatively serious impact on the confidentiality, integrity, availability of the smart contract or its economic model, which can cause a greater economic loss, local functional unavailability, loss of credibility and other impact to the contract business system.

- **Medium**

Medium impact generally refers to the vulnerability can have a relatively minor impact on the confidentiality, integrity, availability of the smart contract or its economic model, which can cause a small amount of economic loss to the contract business system, individual business unavailability and other impact.

- **Low**

Low impact generally refers to the vulnerability can have a minor impact on the smart contract, which can pose certain security threat to the contract business system and needs to be improved.

3.1.4 Likelihood of Exploitation

- **Probable**

Probable likelihood generally means that the cost required to exploit the vulnerability is low, with no special exploitation threshold, and the vulnerability can be triggered consistently.

- **Possible**

Possible likelihood generally means that exploiting such vulnerability requires a certain cost, or there are certain conditions for exploitation, and the vulnerability is not easily and consistently triggered.

- **Unlikely**

Unlikely likelihood generally means that the vulnerability requires a high cost, or the exploitation conditions are very demanding and the vulnerability is highly difficult to trigger.

- **Rare**

Rare likelihood generally means that the vulnerability requires an extremely high cost or the conditions for exploitation are extremely difficult to achieve.

3.1.5 Fix Results Status

Status	Description
Fixed	The project party fully fixes a vulnerability.
Partially Fixed	The project party did not fully fix the issue, but only mitigated the issue.
Acknowledged	The project party confirms and chooses to ignore the issue.

3.2 Audit Categories

No.	Categories	Subitems
1	Coding Conventions	Compiler Version Security
		Deprecated Items
		Redundant Code
		require/assert Usage
		Gas Consumption
2	General Vulnerability	Integer Overflow/Underflow
		Reentrancy
		Pseudo-random Number Generator (PRNG)
		Transaction-Ordering Dependence
		DoS (Denial of Service)
		Function Call Permissions
		call/delegatecall Security
		Returned Value Security
		s.ContractRef.MsgSender Usage
		Replay Attack
		Overriding Variables
		Third-party Protocol Interface Consistency
3	Business Security	Business Logics
		Business Implementations
		Manipulable Token Price
		Centralized Asset Control
		Asset Tradability
		Arbitrage Attack

Beosin classified the security issues of smart contracts into three categories: Coding Conventions, General Vulnerability, Business Security. Their specific definitions are as follows:

- **Coding Conventions**

Audit whether smart contracts follow recommended language security coding practices. For example, smart contracts developed in Solidity language should fix the compiler version and do not use deprecated keywords.

- **General Vulnerability**

General Vulnerability include some common vulnerabilities that may appear in smart contract projects. These vulnerabilities are mainly related to the characteristics of the smart contract itself, such as integer overflow/underflow and denial of service attacks.

- **Business Security**

Business security is mainly related to some issues related to the business realized by each project, and has a relatively strong pertinence. For example, whether the lock-up plan in the code match the white paper, or the flash loan attack caused by the incorrect setting of the price acquisition oracle.

*Note that the project may suffer stake losses due to the integrated third-party protocol. This is not something Beosin can control. Business security requires the participation of the project party. The project party and users need to stay vigilant at all times.

3.3 Disclaimer

The Audit Report issued by Beosin is related to the services agreed in the relevant service agreement. The Project Party or the Served Party (hereinafter referred to as the "Served Party") can only be used within the conditions and scope agreed in the service agreement. Other third parties shall not transmit, disclose, quote, rely on or tamper with the Audit Report issued for any purpose.

The Audit Report issued by Beosin is made solely for the code, and any description, expression or wording contained therein shall not be interpreted as affirmation or confirmation of the project, nor shall any warranty or guarantee be given as to the absolute flawlessness of the code analyzed, the code team, the business model or legal compliance.

The Audit Report issued by Beosin is only based on the code provided by the Served Party and the technology currently available to Beosin. However, due to the technical limitations of any organization, and in the event that the code provided by the Served Party is missing information, tampered with, deleted, hidden or subsequently altered, the audit report may still fail to fully enumerate all the risks.

The Audit Report issued by Beosin in no way provides investment advice on any project, nor should it be utilized as investment suggestions of any type. This report represents an extensive evaluation process designed to help our customers improve code quality while mitigating the high risks in blockchain.

3.4 About Beosin

Beosin is the first institution in the world specializing in the construction of blockchain security ecosystem. The core team members are all professors, postdocs, PhDs, and Internet elites from world-renowned academic institutions. Beosin has more than 20 years of research in formal verification technology, trusted computing, mobile security and kernel security, with overseas experience in studying and collaborating in project research at well-known universities. Through the security audit and defense deployment of more than 2,000 smart contracts, over 50 public blockchains and wallets, and nearly 100 exchanges worldwide, Beosin has accumulated rich experience in security attack and defense of the blockchain field, and has developed several security products specifically for blockchain.

BEOSIN
Blockchain Security

Official Website
https://www.beosin.com

Telegram
https://t.me/beosin

Twitter
https://twitter.com/Beosin_com

Email
service@beosin.com