verichains

SECURITY AUDIT OF

RONIN BRIDGE SMART CONTRACTS

Private Report

Aug 21, 2024

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
Ether (ETH)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Solc	A compiler for Solidity.
ERC20	ERC20 (BEP20 in Binance Smart Chain or *x*RP20 in other chains) tokens are blockchain-based assets that have value and can be sent and received. The primary difference with the primary coin is that instead of running on their own blockchain, ERC20 tokens are issued on a network that supports smart contracts such as Ethereum or Binance Smart Chain.

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on Aug 21, 2024. We would like to thank the Sky Mavis for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the Ronin Bridge Smart Contracts. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified a vulnerable issue in the source code. Sky Mavis team has acknowledged and resolved the issue.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Ronin Bridge Smart Contracts

The Ronin Bridge allows funds to be transferred from Ethereum to the Ronin Network and vice versa. The main features of the bridge are:

- Fast & seamless transactions with almost instant confirmation.
- Drastically reduced gas fees.
- The ability to withdraw Axie assets back to Ethereum Mainnet.
- Simplified on-boarding for new users, through a customized wallet solution.

1.2. Audit Scope

This audit focused on identifying security flaws in code and the design of Ronin Bridge Smart Contracts. It was conducted on commit `df1d80e48fa1e0854133775eb94aa5bdb64a2a47` from git repository *https://github.com/ronin-chain/bridge-contract*.

The detailed audit scope is LIMITED ONLY to the following files and directories:

- *https://github.com/ronin-chain/bridge-contract/tree/df1d80e48fa1e0854133775eb94aa5bdb64a2a47/src/mainchain*
- *https://github.com/ronin-chain/bridge-contract/tree/df1d80e48fa1e0854133775eb94aa5bdb64a2a47/src/ronin/gateway*

1.3. 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 RK87, our in-house smart contract security analysis tool.
- 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 were considered during the audit of the smart contract:

- Integer Overflow and Underflow
- Timestamp Dependence
- Race Conditions
- Transaction-Ordering Dependence
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Usage, Gas Limit and Loops

- Redundant fallback function
- Unsafe type Inference
- Reentrancy
- Explicit visibility of functions state variables (external, internal, private and public)
- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Sky Mavis acknowledges that the security services provided by Verichains, are conducted to the best of their professional abilities but cannot guarantee 100% coverage of all security vulnerabilities. Sky Mavis understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, Sky Mavis agrees that Verichains shall not be held responsible or liable, and shall not be charged for any hacking incidents that occur due to security vulnerabilities not identified during the audit process.

1.5. Acceptance Minute

This final report served by Verichains to the Sky Mavis will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the Sky Mavis, the final report will be considered fully accepted by the Sky Mavis without the signature.

2. AUDIT RESULT

2.1. Overview

The Ronin Bridge Smart Contracts was written in `Solidity` language, based on OpenZeppelin and PRBMath libraries.

This audit is conducted following the incident on August 6, 2024, caused by a faulty upgrade deployment script. In this review, the audit team will focus on the following aspects:

- Ensuring that no other mistakes were made in the deployment script beyond the one that caused the incident.
- Reviewing the proposal for the bug fix in PR *https://github.com/ronin-chain/bridge-contract/pull/65*.
- Reviewing the hardening logic of the withdrawal function in PR *https://github.com/ronin-chain/bridge-contract/pull/58*.
- Assessing the overall security of major contracts such as `MainchainBridgeManager.sol`, `MainchainGatewayV3.sol`, `RoninBridgeManager.sol`, and `RoninGatewayV3.sol`, along with their corresponding dependencies.

2.2. Findings

During the audit process, the audit team identified some vulnerabilities in the given version of Ronin Bridge Smart Contracts.

Sky Mavis team has updated the code, according to Verichains's draft report.

#	Issue	Severity	Status
1	Incorrect handling of `msg.value` in `requestDepositForBatch` function	CRITICAL	Open

2.2.1. Incorrect handling of `msg.value` in `requestDepositForBatch` function CRITICAL

Affected files:

- `src/mainchain/MainchainGatewayV3.sol`#requestDepositForBatch()

The `requestDepositForBatch` function in the `MainchainGatewayV3` contract allows users to submit multiple deposit requests within a single transaction, including both native tokens and ERC20 tokens. However, the function fails to correctly handle `msg.value` when processing native token deposits. Specifically, it does not verify that `msg.value` matches the total sum of the requested native token deposits. This oversight allows users to pay only once while making

multiple native token deposits, potentially leading to underpayment and incorrect deposit processing.

```
function requestDepositForBatch(Transfer.Request[] calldata _requests) external payable
virtual whenNotPaused {
    uint length = _requests.length;
    for (uint256 i; i < length; ++i) {
      _requestDepositFor(_requests[i], msg.sender);
    }
}

function _requestDepositFor(Transfer.Request memory _request, address _requester) internal
virtual {
    MappedToken memory _token;
    address _roninWeth = address(wrappedNativeToken);

    _request.info.validate();
    if (_request.tokenAddr == address(0)) {
        if (_request.info.quantity != msg.value) revert ErrInvalidRequest(); //@auditor:
The function only check msg.value for a single deposit.

        _token = getRoninToken(_roninWeth);
        if (_token.erc != _request.info.erc) revert ErrInvalidTokenStandard();

        _request.tokenAddr = _roninWeth;
    } else {
        // ...
    }

    uint256 _depositId = depositCount++;
    Transfer.Receipt memory _receipt = _request.into_deposit_receipt(_requester,
_depositId, _token.tokenAddr, roninChainId);

    emit DepositRequested(_receipt.hash(), _receipt);
}
```

RECOMMENDATION

The code should be updated to ensure that `msg.value` matches the total sum of the requested deposits. If the payment is insufficient, the function should revert the transaction.

Note: This issue was identified and resolved in the previous audit, but the vulnerable code has been reintroduced in the current version.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Aug 21, 2024*	Private Report	Verichains Lab

Table 2. Report versions history