QuillAudits

# Audit Report
August, 2024

For

RITESTREAM

# Table of Content

Executive Summary ........ 02

Number of Security Issues per Severity ........ 03

Checked Vulnerabilities ........ 04

Techniques and Methods ........ 06

Types of Severity ........ 07

Types of Issues ........ 07

**Informational Issues** 08

1. Avoid using floating pragma 08

Closing Summary ........ 09

Disclaimer ........ 09

## Executive Summary

| | |
|---|---|
| **Project Name** | ritestream |
| **Overview** | ritestream is a Web3-based platform that empowers creators to fund, produce, and distribute original video content through decentralized finance and blockchain technology. It aims to democratize the content creation industry by enabling creators to retain ownership and monetize their work globally. |
| **Timeline** | 18th July 24 - 24th July 2024 |
| **Method** | Manual Review, Functional Testing, Automated Testing, etc. All the raised flags were manually reviewed and re-tested to identify any false positives. |
| **Audit Scope** | The scope of this audit was to analyze the RiteAdapter & RiteBase Contract for quality, security, and correctness. |
| **Blockchain** | EVM |
| **Source Code** | https://github.com/ritestream/ritestream-bridge/blob/main/contracts/RiteAdapter.sol<br>https://github.com/ritestream/ritestream-bridge/blob/main/contracts/RiteBase.sol<br>**8c8af8271d27f1cfd3cc3fc305baff0bdbb53640** |
| **Contracts In-Scope** | contracts/RiteBase.sol<br>contracts/RiteAdapter.sol |
| **Branch** | Main |
| **Fixed In** | NA |

## Number of Security Issues per Severity

1
Issue Found

🟥 High 🟨 Medium

🟩 Low 🟪 Informational

| | High | Medium | Low | Informational |
|---|---|---|---|---|
| Open Issues | 0 | 0 | 0 | 0 |
| Acknowledged Issues | 0 | 0 | 0 | 1 |
| Partially Resolved Issues | 0 | 0 | 0 | 0 |
| Resolved Issues | 0 | 0 | 0 | 0 |

## Checked Vulnerabilities

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

- ✓ Access Management
- ✓ Arbitrary write to storage
- ✓ Centralization of control
- ✓ Ether theft
- ✓ Improper or missing events
- ✓ Logical issues and flaws
- ✓ Arithmetic Correctness
- ✓ Race conditions/front running
- ✓ SWC Registry
- ✓ Re-entrancy
- ✓ Timestamp Dependence
- ✓ Gas Limit and Loops
- ✓ Exception Disorder
- ✓ Gasless Send
- ✓ Compiler version not fixed
- ✓ Address hardcoded
- ✓ Divide before multiply
- ✓ Integer overflow/underflow
- ✓ ERC's conformance
- ✓ Dangerous strict equalities
- ✓ Tautology or contradiction
- ✓ Return values of low-level calls
- ✓ Missing Zero Address Validation
- ✓ Private modifier
- ✓ Revert/require functions
- ✓ Multiple Sends
- ✓ Using suicide
- ✓ Using delegatecall

## Checked Vulnerabilities

- ✓ Use of tx.origin
- ✓ Malicious libraries
- ✓ Using inline assembly
- ✓ Style guide violation
- ✓ Upgradeable safety
- ✓ Using throw
- ✓ Unsafe type inference
- ✓ Implicit visibility level

## Techniques and Methods

Throughout the audit of ritestream, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviorbehaviour.
- Token distribution and calculations are as per the intended behaviorbehaviour mentioned in the whitepaper.
- Implementation of various token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods, and tools were used to review all the Solana programs.

### Structural Analysis

In this step, we have analysed the design patterns and structure of Solana programs. A thorough check was done to ensure the Solana program is structured in a way that will not result in future problems.

### Static Analysis

Static analysis of Solana programs was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of Solana programs.

### Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, and their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

### Gas Consumption

In this step, we have checked the behaviour of Solana programs in production. Checks were done to know how much gas gets consumed and the possibilities of optimising code to reduce gas consumption.

## Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

### High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

### Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

### Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

### Informational

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

## Types of Issues

### Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

### Resolved

These are the issues identified in the initial audit and have been successfully fixed.

### Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

### Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

## Informational Issues

### 1. Avoid using floating pragma

**Path**

Riteadapter, RiteBase

**Description**

Floating pragma should only be used when a contract is intended for consumption by other developers.

Locking the pragma helps ensure that contracts are not accidentally deployed using, for example, the latest compiler, which may have higher risks of undiscovered bugs.

**Recommendation**

Use locked pragma instead.

**Status**

**Acknowledged**

## Closing Summary

In this report, we have considered the security of the ritestream Contracts. We performed our audit according to the procedure described above.

Code quality is good. No issues were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture.

## Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in ritestream smart contract. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of ritestream smart contracts. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of the ritestream to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

## About QuillAudits

QuillAudits is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With six years of expertise, we've secured over 1000 projects globally, averting over $30 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.

| | | |
|---|---|---|
| **1000+** | **$30B** | **1M+** |
| Audits Completed | Secured | Lines of Code Audited |

## Follow Our Journey

# Audit Report
August, 2024

For

RITESTREAM

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com