hexens x RISC ZERO

Security Review Report for RISC Zero

December 2024

Table of Contents

1. About Hexens

Hexens is a pioneering cybersecurity firm dedicated to establishing robust security standards for Web3 infrastructure, driving secure mass adoption through innovative protection technology and frameworks. As an industry elite experts in blockchain security, we deliver comprehensive audit solutions across specialized domains, including infrastructure security, Zero Knowledge Proof, novel cryptography, DeFi protocols, and NFTs.

Our methodology combines industry-standard security practices combined with unique methodology of two teams per audit, continuously advancing the field of Web3 security. This innovative approach has earned us recognition from industry leaders.

Since our founding in 2021, we have built an exceptional portfolio of enterprise clients, including major blockchain ecosystems and Web3 platforms.

2. Security Review Details

- **Review Led by**

Hayk Andriasyan, Lead Security Researcher

- **Scope**

The analyzed resources are located on:

https://github.com/risc0/risc0-ethereum

🔗
- /aggregation
- /contracts/src/IRiscZeroSetVerifier.sol
- /contracts/src/RiscZeroSetVerifier.sol

📌 **Commit:** `3c1fd2a859e40ea009a580aac294191196968c60`

The issues described in this report were fixed in the following commit:

🔗 https://github.com/risc0/risc0-ethereum/pull/406

📌 **Commit:** `6c0a11d789442372aaa2f802da9d0f5b7dcbb135`

- **Changelog**

16 December 2024	Audit Start
16 January 2025	Initial Report
22 January 2025	Revision Received
23 May 2025	Final Report

3. Severity Structure

The vulnerability severity is calculated based on two components:

1. Impact of the vulnerability
2. Probability of the vulnerability

Impact	Probability			
	Rare	Unlikely	Likely	Very likely
Low	Low	Low	Medium	Medium
Medium	Low	Medium	Medium	High
High	Medium	Medium	High	Critical
Critical	Medium	High	Critical	Critical

- **Severity Characteristics**

Smart contract vulnerabilities can range in severity and impact, and it's important to understand their level of severity in order to prioritize their resolution. Here are the different types of severity levels of smart contract vulnerabilities:

Critical

Vulnerabilities that are highly likely to be exploited and can lead to catastrophic outcomes, such as total loss of protocol funds, unauthorized governance control, or permanent disruption of contract functionality.

High

Vulnerabilities that are likely to be exploited and can cause significant financial losses or severe operational disruptions, such as partial fund theft or temporary asset freezing.

Medium

Vulnerabilities that may be exploited under specific conditions and result in moderate harm, such as operational disruptions or limited financial impact without direct profit to the attacker.

Low

Vulnerabilities with low exploitation likelihood or minimal impact, affecting usability or efficiency but posing no significant security risk.

Informational

Issues that do not pose an immediate security risk but are relevant to best practices, code quality, or potential optimizations.

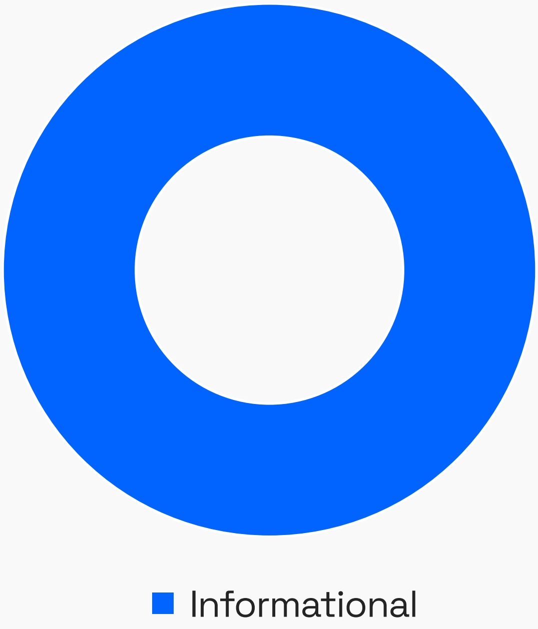
Issue Symbolic Codes

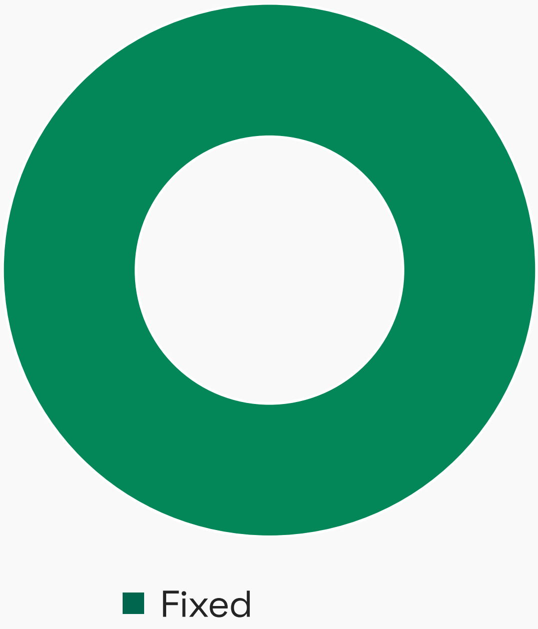
Each identified and validated issue is assigned a unique symbolic code during the security research stage.

Due to the structure of the vulnerability reporting flow, some rejected issues may be missing.

4. Findings Summary

Severity	Number of Findings
🟥 Critical	0
🟥 High	0
🟧 Medium	0
🟩 Low	0
🟦 Informational	1
Total:	**1**





5. Weaknesses

This section contains the list of discovered weaknesses.

RSCZD-1 | Discrepancy between the SELECTOR implementation and documentation

Fixed ✓

Severity:	Informational	Probability:	Very likely	Impact:	Informational

Path:

contracts/src/RiscZeroSetVerifier.sol

Description:

Risc0 verifiers have a SELECTOR parameter which differentiates verifier types. **RiscZeroSetVerifier** has a **bytes4 public immutable SELECTOR**; which is implemented using the image id as a parameter:

```
constructor(IRiscZeroVerifier verifier, bytes32 imageId, string memory
_imageUrl) {
    VERIFIER = verifier;
    IMAGE_ID = imageId;
    imageUrl = _imageUrl;

    SELECTOR = RiscZeroSetVerifierLib.selector(imageId);
}
```

```
library RiscZeroSetVerifierLib {
    function selector(bytes32 imageId) internal pure returns (bytes4) {
        return bytes4(
            sha256(
                abi.encodePacked(
                    // tag
                    sha256("risc0.SetInclusionReceiptVerifierParameters"),
                    // down
                    imageId,
                    // down length
                    uint16(1) << 8
                )
            )
        );
    }
}
```

Documentation for the immutable **SELECTOR** variable:

```
/// @dev The selector is taken from the hash of the verifier parameters
including the Groth16
///      verification key and the control IDs that commit to the RISC Zero
circuits.
```

isn’t aligned with the implementation.

Remediation:

Fix the comment to be aligned with the implementation.

hexens x RISC
ZERO