Zenon Network Smart Contracts Review

By: ChainSafe Systems

May 2023

Zenon Network Smart Contracts Review

Auditors: Anderson Lee, Oleksii Matiiasevych

WARRANTY

Introduction

Zenon Network requested ChainSafe Systems to perform a review of the bridge smart contracts. The contract can be identified by the following git commit hash:

`15bbf40ef7cba95dba797133b0da5ab3792a6b9e`

There are 3 smart contracts in scope, with the main one being the Bridge.
After the initial review, Zenon Network team applied a number of updates which can be identified by the following git commit hash:

`5204c0df4e0a2a1bcaa69e5fa22c9131c09e76e9`

Additional verification was performed after that.

Disclaimer

The review makes no statements or warranties about the utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about the fitness of the contracts for any specific purpose, or their bug free status.

Executive Summary

There are no known compiler bugs for the specified compiler version (0.8.19), that might affect the contracts' logic.

There were 0 critical, 0 major, 0 minor, 16 informational/optimizational issues identified in the initial version of the contracts. They are described below for historical purposes. During the engagement we had numerous fruitful discussions of the security risk, emergency recoveries and system resilience. We are looking forward to future collaborations.

Critical Bugs and Vulnerabilities

No critical issues were identified.

Line by Line Review. Fixed Issues

1. Bridge, line 43: Optimization, `uint256max` could be made constant.

2. Bridge, line 44: Optimization, `networkClass` could be made constant.

3. Bridge, line 45: Optimization, `minNominatedGuardians` could be made constant.

4. Bridge, line, 77. Note, the `isNotHalted()` modifier could utilize the `isHalted()` function in order to avoid code duplication.

5. Bridge, line, 100. Note, the `constructor()` function could init the guardians list with duplicate entries.

6. Bridge, line, 104. Note, the `constructor()` function could init the `estimatedBlockTime` with 0 value, which is less than allowed by the `setEstimatedBlockTime()` function.

7. Bridge, line, 105. Note, the `constructor()` function could init the `confirmationsToFinality` with 0 or 1 value, which is less than allowed by the `setConfirmationsToFinality()` function.

8. Bridge, line, 117. Optimization, the `redeem()` function reads `redeemsInfo[nonce]` value from storage multiple times.

9. Bridge, line, 140. Note, the `redeem()` function uses a `low-level address.call()` instead of an interface to do minting.

10. Bridge, line, 163. Note, the `unwrap()` function uses a low-level `address.call()` instead of an interface to do burning.

11. Bridge, line, 289. Note, the `nominateGuardians()` function doesn't have the upper limit on the number of new guardians which could result in unexpected behavior and gas consumption. Consider introducing an upper limit.

12. Bridge, line, 333. Optimization, the `proposeAdministrator()` function reads `guardiansVotes[i]` value from storage multiple times. Should use the local variable `newAdministrator` instead.

13. Bridge, line, 341. Optimization, the `proposeAdministrator()` function reads the administrator value from storage twice.

14. Bridge, line, 348. Note, the set* functions could emit an event for easier accounting.

Line by Line Review. Acknowledged Findings.

1. Bridge, line, 222. Optimization, the `halt()` function reads the `actionsNonce` value from storage twice.

2. Bridge, line, 266. Optimization, the `setTss()` function reads the `actionsNonce` value from storage twice.

Anderson Lee

Oleksii Matiiasevych