Security Audit Report

Stellar Core Protocol 24 Changes

v1.0

November 13, 2025

Table of Contents

License

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

THIS AUDIT REPORT WAS PREPARED EXCLUSIVELY FOR AND IN THE INTEREST OF THE CLIENT AND SHALL NOT CONSTRUE ANY LEGAL RELATIONSHIP TOWARDS THIRD PARTIES. IN PARTICULAR, THE AUTHOR AND HIS EMPLOYER UNDERTAKE NO LIABILITY OR RESPONSIBILITY TOWARDS THIRD PARTIES AND PROVIDE NO WARRANTIES REGARDING THE FACTUAL ACCURACY OR COMPLETENESS OF THE AUDIT REPORT.

FOR THE AVOIDANCE OF DOUBT, NOTHING CONTAINED IN THIS AUDIT REPORT SHALL BE CONSTRUED TO IMPOSE ADDITIONAL OBLIGATIONS ON COMPANY, INCLUDING WITHOUT LIMITATION WARRANTIES OR LIABILITIES.

This audit has been performed by

Oak Security GmbH

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security GmbH has performed a security audit of Stellar Core Protocol 24 Changes.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following target:

Repository	https://github.com/stellar/stellar-core
Commit	`0d7b4345de396ad4e8d7dcc4460ddc6feeb27b11`
Scope	Changes to the repository since the previous review commit `9819d1b3bcfc13c60dcf88a948811fd2f96e7eef` were reviewed, excluding `src/ledger/P23HotArchiveFix.cpp`.
Fixes verified at commit	`46fb6e469bd44bd7195c43fa921fa8e5084d9a6c` Note that only fixes to the issues described in this report have been reviewed at this commit. Any further changes such as additional features have not been reviewed.

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line-by-line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

Stellar is a decentralized, open-source network designed for fast, low-cost payments and asset issuance. The network's core is the Stellar Consensus Protocol (SCP), a Federated Byzantine Agreement (FBA) system that enables consensus without relying on mining. The network is supported by the non-profit Stellar Development Foundation (SDF).

Stellar has been extended with Soroban, a smart contracts platform designed for scalability and a familiar developer experience using Rust and WebAssembly (Wasm). This allows developers and users to perform cross-border payments, trade digital assets, and build sophisticated DeFi applications on a single, integrated network.

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending**, **Acknowledged**, **Partially Resolved**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Code Quality Criteria

The auditor team assesses the codebase's code quality criteria as follows:

Criteria	Status	Comment
Code complexity	High	-
Code readability and clarity	Low-Medium	Due to the code's complexity and the stratification of the codebase caused by supporting multiple protocol versions over the years, the code is difficult to read.
Level of documentation	Medium	The documentation is extensive; however, it is not always accurate and often refers to older versions.
Test coverage	-	Since the audit was conducted on the diff between two versions, it was not possible to determine the test coverage of the code in scope.

Summary of Findings

No	Description	Severity	Status
1	Incorrect fee pool adjustment triggered on all future protocol upgrades	Minor	Resolved
3	Redundant code	Informational	Acknowledged

Detailed Findings

1. Incorrect fee pool adjustment triggered on all future protocol upgrades

Severity: Minor

In `src/herder/Upgrades.cpp:1238-1245`, the function `applyVersionUpgrade` unconditionally increases the `feePool` by `31,879,035` when upgrading starting from protocol version 24.

The logic uses the condition `protocolVersionStartsFrom(newVersion, ProtocolVersion::V_24)`, which evaluates to true not only for version 24 but for all subsequent versions (e.g., v25, v26, etc.).

Consequently, each future upgrade erroneously re-applies the same balance adjustment.

This behavior leads to incorrect inflation of the network's fee pool balance.

Recommendation

We recommend replacing the `protocolVersionStartsFrom` condition with `needUpgradeToVersion` to ensure that the `31,879,035` adjustment is applied only when upgrading specifically to protocol version 24.

Status: Resolved

2. Redundant code

Severity: Informational

In `src/bucket/BucketSnapshot.cpp:166`, the variable `addEntry` is declared as a constant value equal to `true`.

This variable is used only once in the `if` statement on line `167`. Since its only possible value is `true`, the `if` statement can be safely removed. The call to the `push_back` function on line `169` can be lifted and used as-is.

Recommendation

We recommend removing lines `166-168` and `170`.

Status: Acknowledged