Security Audit Report

Snowbridge Updates 4

v1.0

May 6, 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 been engaged by Snowfork to perform a security audit of updates to Snowbridge.

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 targets:

Repository	https://github.com/Snowfork/snowbridge
Label	Paths referencing this target are prefixed below with `snowbridge:`
Commit	`cb05e1f8bf7018dfca42a8d93d73826a97c99100`

Scope	The scope was restricted to the changes compared to commit `53baf6a3a40aa38bd3bf962d88e3ba2d8f6e476e` for the following directories and files:
	
	`.`
	`└── contracts`
	`    └── src`

Repository	https://github.com/paritytech/polkadot-sdk
Label	Paths referencing this target are prefixed below with `polkadot-sdk:`
Scope	The scope is restricted to the changes applied in the following pull requests: • https://github.com/paritytech/polkadot-sdk/pull/6855 reviewed at commit `4059282fc7b6ec965cc22a9a0df5920a4f3a4101`, base branch at `645878a27115db52e5d63115699b4bbb89034067`. • https://github.com/paritytech/polkadot-sdk/pull/7075 reviewed at commit `d7cb548a819b30796db6ed622080f8f0c6c387f4`, base branch at `135b7183626e4ac675fe368cf11496050b9e7821`.
Fixes verified at commit	`cf4944ec800b3914dab60e7946c64600e0048147` 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

Snowbridge is a general-purpose, trustless, and decentralized bridge between Polkadot and Ethereum. This is achieved by using light clients.

The protocol uses a BEEFY light client implemented in Solidity smart contracts to track the Polkadot chain, and an Altair-compliant light client to keep track of the Ethereum Beacon Chain implemented in a Substrate pallet.

This audit covers changes as described above.

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**	The code implements complex operations and makes use of the latest features coming from Substrate and Cumulus. It also uses the latest XCM specification. The bridge uses/integrates with low-level functionality from different ecosystems.
Code readability and clarity	**Medium**	-
Level of documentation	**High**	The protocol and the modifications applied are well documented.
Test coverage	**Medium-High**	Test coverage for Solidity contracts reported by `forge coverage` is `75.73%`. Test coverage for `pallets`, `primitives`, and `runtime` reported by `cargo tarpaulin` is `74.60%`.

Summary of Findings

No	Description	Severity	Status
1	Incorrect fork version defined for Electra hard fork	Minor	Resolved
2	Lack of segregation for relayer incentives in Gateway contract increases security and fairness risks	Minor	Acknowledged
3	Incomplete reward funds migration in Gateway upgrade	Minor	Acknowledged
4	Redundant `agent` field in `Channel` structure	Informational	Acknowledged
5	Redundant hard fork check for `EXECUTION_HEADER_INDEX`	Informational	Resolved
6	Lack of observability for deprecated `TransferNativeFromAgent` command execution	Informational	Acknowledged
7	Outdated weight for the `submit` extrinsic	Informational	Acknowledged

Detailed Findings

1. Incorrect fork version defined for Electra hard fork

Severity: Minor

In `polkadot-sdk:cumulus/parachains/runtimes/bridge-hubs/bridge-hub-rococo/src/bridge_to_ethereum_config.rs:173` and `polkadot-sdk:cumulus/parachains/runtimes/bridge-hubs/bridge-hub-westend/src/bridge_to_ethereum_config.rs:175`, the `ForkVersions` struct is responsible for defining the parameters of supported Ethereum forks.

However, an incorrect version has been assigned to the Electra hard fork, which has been mistakenly set as identical to the preceding Deneb fork.

Specifically, the Electra fork version is currently set to `0x90000073`, whereas it should be `0x90000074`.

Recommendation

We recommend updating the `ForkVersions` struct to correctly reflect the Electra hard fork version identifier.

Status: Resolved

2. Lack of segregation for relayer incentives in Gateway contract increases security and fairness risks

Severity: Minor

In `snowbridge:contracts/src/Gateway.sol:259`, the `submitV1` function determines the amount of native tokens to transfer to the relayer submitting the message.

However, in the current code version, incentives are no longer separated by channels since refunds for all channels are now issued directly from the Gateway contract.

This change forces channel owners to deposit incentives into a shared contract, potentially resulting in disproportionate relaying costs across channels, depending on message volume and available incentives.

Additionally, storing rewards in the Gateway contract introduces security risks, as it is a complex and upgradeable contract. Future updates may inadvertently introduce vulnerabilities, creating potential attack vectors for fund theft.

Recommendation

We recommend segregating refund funds into a dedicated treasury for each channel instead of storing them within the Gateway contract.

This would enhance security by isolating funds from potential upgrade-related vulnerabilities and improve fairness by ensuring incentives remain specific to each channel, preventing imbalances in relayer costs.

Status: Acknowledged

3. Incomplete reward funds migration in Gateway upgrade

Severity: Minor

In `snowbridge:contracts/src/upgrades/Gateway202502.sol`, the `initialize` function overrides the corresponding function in the Gateway contract to register the ether's `TokenInfo`, which is required for enabling ether bridging.

However, it does not migrate the remaining reward funds that were previously held in the agent contract and are now managed directly by the Gateway contract. This oversight results in mixed funds remaining in the agent contract.

Recommendation

We recommend modifying the initialize function to include a migration process for the remaining reward funds. This should involve transferring all residual funds from the agent contract to the Gateway contract before finalizing the upgrade.

Status: Acknowledged

4. Redundant `agent` field in `Channel` structure

Severity: Informational

In `snowbridge:contracts/src/Types.sol:49`, the `Channel` structure declares an `agent` field, but its usage is minimal and functionally redundant.

The field appears only three times in `snowbridge:contracts/src/Gateway.sol`:

- In lines `356` and `586`, `ch.agent` is compared to zero to determine if the channel is initialized.
- In line `361`, it is assigned the address of the actual channel agent, but this value is never read afterward.

While this field may have had a functional role in earlier versions of the codebase, it currently serves only as an initialization check.

Though the performance impact is negligible, keeping an unused field reduces code clarity and maintainability.

Recommendation

We recommend replacing the agent field with a simple boolean flag to explicitly track channel initialization. Adjust lines `356`, `361`, and `586` accordingly to improve readability and maintainability.

Status: Acknowledged

5. Redundant hard fork check for `EXECUTION_HEADER_INDEX`

Severity: Informational

In `polkadot-sdk:bridges/snowbridge/pallets/ethereum-client/src/lib.rs:762-770`, the function `execution_header_gindex_at_slot` is intended to return different indexes based on the Ethereum hardfork version: Altair or Electra.

However, the constant `EXECUTION_HEADER_INDEX` is defined with the same value for both hardforks, making the conditional check unnecessary.

This redundancy results in minor inefficiencies and unnecessary complexity in the code. While the performance impact is minimal, keeping an ineffective conditional check reduces code clarity and maintainability.

Recommendation

We recommend removing the redundant hardfork check if `EXECUTION_HEADER_INDEX` remains the same for both versions.

Status: Resolved

6. Lack of observability for deprecated `TransferNativeFromAgent` command execution

Severity: Informational

In `snowbridge:contracts/src/Gateway.sol:219`, the `Command.TransferNativeFromAgent` functionality has been disabled due to recent changes introduced in the scoped PRs.

However, the function does not emit an event or explicitly set `success` to false, making it harder for developers and external systems to track deprecated message execution attempts.

This lack of visibility could lead to confusion, debugging difficulties, and potential misinterpretation of transaction outcomes.

Recommendation

We recommend modifying the function to explicitly emit an event when `Command.TransferNativeFromAgent` is invoked, indicating that the operation is disabled.

Additionally, ensure that `success` is explicitly set to `false` to prevent any unintended assumptions about the execution result.

Status: Acknowledged

7. Outdated weight for the `submit` extrinsic

Severity: Informational

In `polkadot-sdk:bridges/snowbridge/pallets/inbound-queue/src/lib.rs:232`, the weight assigned to the `submit` extrinsic is outdated due to the introduction of a new code branch in `polkadot-sdk:bridges/snowbridge/primitives/router/src/inbound/mod.rs:398-409`.

This change introduces a new branch that marginally increases the computational workload. While the impact is small, the extrinsic weight has not been updated accordingly, leading to a minor discrepancy where slightly more work is performed than the allocated gas covers.

Recommendation

We recommend re-benchmarking the `submit` extrinsic to account for the newly introduced branch. The adjustment is expected to be minimal, but updating the weight ensures accuracy in execution cost estimation and maintains optimal network efficiency.

Status: Acknowledged