Audit Report

Asteroid Bridge

v1.0

June 17, 2024

Table of Contents

License

Disclaimer

This audit has been performed by

Oak Security GmbH

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

Introduction

Purpose of This Report

Oak Security has been engaged by Delphi Labs Ltd. to perform a security audit of Asteroid Bridge.

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/asteroidprotocol/bridge
Commit	`21469eba9b8af0f7c46273a9b8f9184efd090fcb`
Scope	All contracts were in scope.
Fixes verified at commit	`60c533890783e5b82eac28c82388f2ab1fd16a2c` 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

The Asteroid Protocol offers a reference implementation for adding arbitrary data onto a Cosmos SDK blockchain.

It enables permanent storage of content on Cosmos SDK blockchains which lacks support for tokens and smart contracts like the Cosmos Hub.

The protocol defines the Cosmos Fungible Token (CFT-20) specification which is leveraged to inscribe data in arbitrary transactions.

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**, 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	Medium	The protocol leverages new concepts of the Cosmos ecosystem like the CFT-20 specification
Code readability and clarity	Medium-High	-
Level of documentation	Medium	-
Test coverage	Medium-High	`cargo tarpaulin` reports an `86.58%` test coverage.

Summary of Findings

No	Description	Severity	Status
1	Message verification does not require the supermajority of signers	Minor	Resolved
2	Signers can replay transactions	Minor	Acknowledged
3	Unbounded nested iterations in `verify_signature` could run out of gas	Minor	Acknowledged
4	Misconfiguration of the bridge chain ID could result in stuck tokens	Minor	Resolved
5	Disabling tokens can lead to inconsistent contract state	Minor	Resolved
6	IBC fee excess is accumulated and stuck in the contract	Minor	Resolved
7	Missing validation for public key `name` identifier	Minor	Resolved
8	Lack of validation for the IBC channel	Minor	Resolved
9	“Migrate only if newer” pattern is not followed	Informational	Resolved
10	Lack of validation when disabling a token may mislead users	Informational	Resolved
11	Sorting of signatures is inefficient	Informational	Acknowledged
12	The protocol owner can arbitrarily disable tokens from being bridged back	Informational	Acknowledged
13	Miscellaneous comments	Informational	Resolved

Detailed Findings

1. Message verification does not require the supermajority of signers

Severity: Minor

In contracts/neutron-bridge/src/helpers.rs:16-78, the function verify_signatures takes a message and a list of signatures and verifies that at least a number equal to signer_threshold of public keys in the SIGNERS map has signed the message.

The signer_threshold is part of the config and can be set by the protocol owner, with the only requirement being to be more than two, as defined by the MIN_SIGNER_THRESHOLD.

However, since the threshold does not depend on the number of registered signers, and no majority or supermajority is required for the verification of signatures, the protocol is vulnerable to attacks by malicious coalitions of signers which could execute arbitrary messages or front-run honest signers.

Additionally, there is no misbehavior mechanism to allow fishermen and third parties to report and punish signers acting maliciously.

We classify this issue as minor since signers are assumed to be trusted.

Recommendation

We recommend modifying the verify_signatures function to require signatures from at least two-thirds of the registered signers, aligning with common supermajority practices.

Additionally, we recommend implementing economic incentives and a misbehavior reporting mechanism to detect and charge malicious signers.

Status: Resolved

The client introduced a verification process to confirm that the message is signed by a supermajority of signers.

2. Signers can replay transactions

Severity: Minor

In contracts/neutron-bridge/src/execute.rs:316, the bridge_receive function protects against transaction replay by verifying if the provided transaction_hash is already recorded in the HANDLED_TRANSACTIONS map.

However, the `transaction_hash` is not computed during the contract execution but supplied directly by the signers.

Consequently, signers, who are responsible for signing an attestation that includes the hash, could potentially submit any arbitrary string as the hash allowing the same transaction to be executed multiple times.

We classify this issue as minor since signers are assumed to be trusted.

Recommendation

We recommend modifying the `bridge_receive` function to compute the `transaction_hash` based on the transaction data rather than relying on hashes provided as input.

Status: Acknowledged

The client states that the transaction hash in this case refers to the bridge transaction hash generated by the Cosmos Hub itself. They have noted that, within the current contract framework, there appears to be no mechanism to prevent signers who may collude from bridging transactions at their discretion.

3. Unbounded nested iterations in `verify_signature` could run out of gas

Severity: Minor

In `contracts/neutron-bridge/src/helpers.rs:16-78`, the function `verify_signatures` takes a message and a list of signatures and verifies that the message has been signed by at least a threshold of registered signers.

This is achieved by performing a nested iteration: First through all the public keys in the `SIGNERS` map, and then through all the `decoded_signatures`. Inside the loops, each combination of key and signature is verified although only a fraction of these combinations is expected to match. The number of iterations is additionally increased by checking the signatures already verified, against other keys.

Consequently, if the number of signers and signatures is large, the execution will run out of gas since there is no upper limit enforced in the contract.

We are reporting this issue with minor severity since the cardinality of signers is alterable only by the owner and is expected to be small.

Recommendation

We recommend tagging signatures with names and retrieving them in the `SIGNERS` map with constant time asymptotical complexity.

Status: Acknowledged

The client states that they manage the number of signers and will ensure to consider gas limits when changing them.

4. Misconfiguration of the bridge chain ID could result in stuck tokens

Severity: Minor

In the `contracts/neutron-bridge/src/execute.rs:635-642`, the protocol owner can update `config.bridge_chain_id` to an arbitrary chain ID.

While this allows the chain ID to be configured in case of chain upgrades that alter the ID, misconfiguration of the `bridge_chain_id` could lead to tokens that are unable to be bridged back to the Cosmos Hub as well as the necessity to synchronize signers and the contract to establish consistency between the contract config and signers' attestations.

Recommendation

We recommend allowing the modification of the `bridge_chain_id` only during contract migration.

Status: Resolved

5. Disabling tokens can lead to inconsistent contract state

Severity: Minor

In `contracts/neutron-bridge/src/execute.rs:417-418`, the `bridge_send` function checks whether the ticker and corresponding `TokenFactory` denomination are disabled.

However, when disabling a token, the `disable_token` function only disables one of them provided as a `ticker`.

Consequently, the user can pass the `TokenFactory` denomination to the `bridge_send` function to bypass the aforementioned check.

We are reporting this issue as minor because the `disable_token` function can only be executed by the owner, and the owner can execute it twice on both denoms to mitigate the problem.

Recommendation

We recommend adding both the ticker and `TokenFactory` denom into the `DISABLED_TOKENS` map when disabling a token and, conversely, removing both of them when enabling a token. Additionally, to avoid excessive storage queries, functions `bridge_receive` and `bridge_send` should check only one possible entry: ticker upon reception, and `TokenFactory` when sending.

Status: Resolved

6. IBC fee excess is accumulated and stuck in the contract

Severity: Minor

In `contracts/neutron-bridge/src/execute.src:450-457`, the `total_fee` variable is calculated by summing up various standard IBC fees: `ack_fee`, `recv_fee` and `timeout_fee`. The amount of fee tokens passed by the user is validated to be greater than the `total_fee` variable.

However, Neutron's IBC transfer does not typically utilize all the fees – some are refunded when not used. When a relayer submits an `Ack` message for a particular packet, the module sends the specified amount of `ack_fee` to the relayer from the escrow address and returns the specified `timeout_fee` to the contract that issued the original `Transfer` message. In case the relayer submits a `Timeout` message, things go the other way around: the relayer is refunded with `timeout_fee` and the contract gets `ack_fee` back.

Consequently, the IBC fee excess will be accumulated and stuck in the contract.

Recommendation

We recommend tracking all fee refunds and sending them back to users.

Status: Resolved

7. Missing validation for public key `name` identifier

Severity: Minor

In `contracts/neutron-bridge/src/execute.rs:494-541`, the `add_signer` function allows the owner to register a new signer.

The function ensures that the `public_key` provided is unique and not already registered, however, it does not verify whether the provided `name` is already associated with a different registered key.

Consequently, multiple public keys could be associated with the same `name`, leading to confusion or misuse in signer identification.

Recommendation

We recommend checking the uniqueness of the `name` associated with any new `public_key` being registered.

Status: Resolved

8. Lack of validation for the IBC channel

Severity: Minor

In `contracts/neutron-bridge/src/contract.rs:42-49`, the `instantiate` function stores the provided `bridge_ibc_channel`.

However, no validation ensures that the `bridge_ibc_channel` identifier is a registered and active channel.

Consequently, this lack of validation could lead to the acceptance of an invalid or closed channel, which would not allow the forwarding of IBC messages to the Cosmos Hub.

Recommendation

We recommend adding a validation step in the `instantiate` function to ensure that the `bridge_ibc_channel` exists and is open.

Status: Resolved

9. “Migrate only if newer” pattern is not followed

Severity: Informational

The contracts within the scope of this audit are currently migrated without regard to their version. This can be improved by adding validation to ensure that migration is only performed if the supplied version is newer.

Recommendation

We recommend following the “migrate only if newer” pattern defined in the CosmWasm documentation.

Status: Resolved

10. Lack of validation when disabling a token may mislead users

Severity: Informational

During token disabling, in `contracts/neutron-bridge/src/execute.rs:273`, it is verified whether the indicated ticker is in the `TOKEN_MAPPING`. If so, it will be disabled.

However, validation to verify whether it is already disabled is not applied. As a consequence, the caller does not receive reliable information about whether there has been any change in the state. If the token is already in the `DISABLED_TOKENS` mapping, overwriting will still return a positive response, which may be misleading.

Recommendation

We recommend checking whether the `DISABLED_TOKENS` mapping already has the ticker that the caller wants to disable. If so, a meaningful error should be returned.

Status: Resolved

11. Sorting of signatures is inefficient

Severity: Informational

In `contracts/neutron-bridge/src/verifier.src`, the `verify_signatures` function takes a message and a list of signatures and verifies that the message has been signed by at least a threshold of registered signers.

The current implementation sorts and deduplicates the list of signatures, which has an asymptotic complexity of `O(n * log(n))`. However, simple verification whether the list is already sorted and deduplicated would only require a single linear pass through the vector.

The function is called during the handling of `LinkToken` and `Receive` messages.

Recommendation

We recommend requiring a sorted list of signatures by the contract and rejecting `LinkToken` and `Receive` messages if the signatures are not sorted or contain duplicates. The sender should sort all the signatures.

Status: Acknowledged

The client states that they prefer to not sort and manipulate data on the frontend.

12. The protocol owner can arbitrarily disable tokens from being bridged back

Severity: Informational

The protocol owner maintains a registry of disabled tokens and can add or remove tokens from the `DISABLED_TOKENS` map at any time.

When a token is disabled, it cannot be bridged in either direction, as both functions `bridge_receive` and `bridge_send` revert. If a token is first bridged to Neutron and then disabled by the admin, that CFT-20 token cannot be bridged back to the Cosmos Hub.

Similarly, the protocol owner maintains a registry of allowed signers in the `SIGNERS` map. Since the protocol relies on the availability and authorization of signers for token bridging, removing or adding signers in the `SIGNERS` map could affect the liveliness of the protocol.

This poses centralization risks, which makes the protocol more susceptible to censorship and could lead to catastrophic consequences if the owner is compromised.

Recommendation

We recommend implementing a governance model or using a multisignature to handle the protocol owner role.

Status: Acknowledged

The client states that the contract will be managed by a multisig composed of members from the Asteroid team, trusted third parties, and community members.

13. Miscellaneous comments

Severity: Informational

Miscellaneous recommendations can be found below:

- `MigrationError` in `contracts/neutron-bridge/src/error.rs` is never used in the contract logic, so it should be removed, or used.

- The comment in `contracts/neutron-bridge/src/execute.rs:342` covers only four of the seven variables expected by the formatter —- it should be extended, or removed completely.
- During validation in `contracts/neutron-bridge/src/verifier.rs:66`, the `verified_signatures` variable cannot be bigger than `config.signer_threshold`. As long as it is iterated, it will successfully return during equality. Consequently, the the >= expression should be changed to ==.

Recommendation

We suggest implementing the aforementioned recommendations.

Status: Resolved