Audit Report

Helix Bridge

v1.0

July 4, 2024

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 has been engaged by ITERING TECH PTE. LTD. to perform a security audit of Helix 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/helix-bridge/contracts
Commit	`77d89f5ea107213ca8f44f01f5545ec7fb9e6561`
Scope	Helix Bridge contracts in `helix-contract/contracts/ln/lnv3`
Fixes verified at commit	`fa6272dc3845aff814b9f2e9f6d57b5c4b9336e2` 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

Helix Bridge is a decentralized cross-chain asset bridge built on top of common messaging bridges.

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 system contains relatively complex call flows with multiple chains involved.
Code readability and clarity	Medium	-
Level of documentation	Medium-High	Most functions are documented, external documentation was provided.
Test coverage	Medium-High	`80.75%` test coverage

Summary of Findings

No	Description	Severity	Status
1	Transfers can still be initiated when a token is removed	Minor	Resolved
2	Centralization risks	Minor	Partially Resolved
3	Irrecoverable Blast points	Minor	Resolved
4	Anybody can request to withdraw providers liquidity	Minor	Acknowledged
5	Input validation could be improved	Minor	Partially Resolved
6	Incorrect accounting on fee-on-transfer/deflationary tokens	Minor	Acknowledged
7	Withdrawing all protocol fees is not possible	Informational	Resolved
8	The contract locks up ETH received	Informational	Resolved
9	Miscellaneous	Informational	Resolved

Detailed Findings

1. Transfers can still be initiated when a token is removed

Severity: Minor

The DAO can use the function `deleteTokenInfo` within `helix-contract/contracts/ln/lnv3/base/LnBridgeSourceV3.sol` to disallow the usage of a token. However, under some circumstances, users can still initiate transfers. This can happen because `lockAndRemoteRelease` does not check if a token is still registered. It only validates if the `LnProvider` was registered for this token (which checks that the token was not removed). But if the token is removed after the registration, the `LnProvider` will still be registered and users will still be able to initiate transfers for the provider.

Recommendation

We recommend checking that the token is still registered (i.e. checking `tokenInfo.index == tokenIndex`) like it is done in `withdrawLiquidity`.

Status: Resolved

2. Centralization risks

Severity: Minor

The contracts contain a few centralization concerns that users should be aware of:

- In `helix-contracts/contracts/ln/lnv3/base/LnBridgeSourceV3.sol`, the function `deleteTokenInfo` could be used to delete tokens that are in use, in which case funds would be stuck, as `withdrawLiquidity` would fail.

- The current `LnBridgeSourceV3` contract logic allows setting the token penalty to zero. This would enable an `LnProvider` to operate without depositing any penalty reserves.

We classify this issue as minor since only admins can trigger the modifications listed above.

Recommendation

We recommend performing the following changes:

- Removing the `deleteTokenInfo` function.
- Setting a minimum penalty amount and base fee amount.

Status: Partially Resolved

The first bullet point has been resolved and the second one acknowledged.

3. Irrecoverable Blast points

Severity: Minor

`helix-contracts/contracts/ln/lnv3/HelixLnBridgeV3ForBlast.sol` is a version of the bridge with some modifications for deploying on Blast. In line `18`, the points operator is set to the DAO address. However, as can be seen in the Blast FAQs, having a smart contract address as the points operator is not supported because this address needs to sign data when claiming. While the DAO was out of scope for this audit, it will presumably be a smart contract, leading to lost points.

We classify this issue as minor since Blast points are an optional feature of Blast and there is no loss for the protocol user if they are not recoverable.

Recommendation

We recommend introducing an EOA that is only used for claiming Blast points.

Status: Resolved

4. Anybody can request to withdraw providers liquidity

Severity: Minor

The `requestWithdrawLiquidity` function of the `helix-contract/contracts/ln/lnv3/base/LnBridgeTargetV3.sol` contract allows any EOA account with the correct parameters to withdraw the liquidity to its providers. While we did not identify a direct security risk of this implementation, it should be explicitly stated whether this is intentional.

Recommendation

We recommend restricting the `requestWithdrawLiquidity` function to only the `LnProvider` or addresses set by the `LnProvider`. Alternatively, explicitly mention in the documentation that anybody can initiate the withdrawal functionality.

Status: Acknowledged

5. Input validation could be improved

Severity: Minor

In a couple of functions, the input is not properly validated:

- In `helix-contract/contracts/ln/lnv3/base/LnBridgeTargetV3.sol`, it is mentioned that the size of `_transferIds` that is passed to `requestWithdrawLiquidity` should not be too large. However, this is not validated, which could lead to out of gas errors on the source chain.
- The `LnBridgeSourceV3` contract allows setting token penalty to zero. This would allow an `LnProvider` to operate without depositing any penalty reserves.
- The `withdrawProtocolFee` function allows sending all protocol fees to `address(0)`, which equals burning the protocol fees.

- In other functions of the Helix Bridge contracts, it is not checked whether the submitted addresses and/or amounts are not equal to zero. This can lead to unintended behavior in out of scope contracts.

Recommendation

We recommend adding additional input validations.

Status: Partially Resolved

The first and third bullet points have been acknowledged, and the second and last bullet points have been resolved.

6. Incorrect accounting on fee-on-transfer/deflationary tokens

Severity: Minor

Some tokens have a transfer fee (e.g. `STA`, `PAXG`), while others do not currently have fees, but might do so in the future (e.g. `USDT`, `USDC`). Moreover, there are deflationary tokens where the balance of users can decrease without transfers. The documentation does not explicitly mention whether the bridge supports fee-on-transfer/deflationary tokens. Therefore, we will assume it does.

The `depositPenaltyReserve` and `lockAndRemoteRelease` functions in `helix-contract/contracts/ln/lnv3/base/LnBridgeSourceV3.sol` transfer the `_amount` of source token from the senders using the `transferFrom` function. If the transferred token is a fee-on-transfer/deflationary token, the actually received penalty reserves/lock amount could be less than the currently expected amount.

Also, since `amountWithFeeAndPenalty` is calculated based on the passed parameter `_params.amount`, the `lockInfos[transferId]` will hold the information that more tokens are locked and will emit a `TokenLocked` event with the wrong amount locked.

Recommendation

We recommend explicitly mentioning whether the protocol supports fee-on-transfer and deflationary tokens. If so, consider getting the received amount by calculating the difference in token balance (using `balanceOf`) before and after the `transferFrom` call.

Status: Acknowledged

7. Withdrawing all protocol fees is not possible

Severity: Informational

The function `claimProtocolFeeIncome` in `helix-contract/contracts/ln/lnv3/base/LnBridgeSourceV3.sol` checks that the requested amount is smaller than the accrued fees. Because of this, passing in an amount that is equal to the accrued fees (i.e., withdrawing everything) will fail.

Recommendation

We recommend changing > to >=.

Status: Resolved

8. The contract locks up ETH received

Severity: Informational

The `HelixLnBridgeV3` contract is able to receive ETH. However, there is currently no way to retrieve ETH from the contract. The funds will be locked up.

Recommendation

We recommend removing the `receive` function if the contract is not supposed to handle ETH. Otherwise, we recommend adding functionality to withdraw/use ETH from the contract.

Status: Resolved

9. Miscellaneous

Severity: Informational

- There is an orphaned comment in `helix-contract/contracts/ln/lnv3/base/LnBridgeTargetV3.sol:16-20`, which describes the non-existing variable `lockTimestamp`.
- In the `LnBridgeSourceV3` contract, the `TokenInfoUpdated` event has to have `sourceDecimals` and `targetDecimals` parameters set to `uint8` to match the `uint8` type for decimals that is used in the `updateTokenInfo` function instead of the current `uint112` type.

- There is an orphaned comment in `helix-contract/contracts/ln/lnv3/base/LnBridgeTargetV3.sol:16-20`, which describes the non-existing variable `lockTimestamp`.

- One variable has a typo. We recommend to replace `updatedPanaltyReserve` with `updatedPenaltyReserve`.

Recommendation

We recommend fixing these items, and also spell-checking all `String` constants, comments, variable names and error messages. Ensuring clarity of any human-readable text is a preemptive measure for establishing better readability and hence maintainability of the codebase.

Status: Resolved