PUBLIC

Code Assessment

of the Wormhole LayerZero Migration Library

October 24, 2025

Produced for

Sky

by

CHAINSECURITY

Contents

1 Executive Summary

Dear all,

Thank you for trusting us to help Sky with this security audit. Our executive summary provides an overview of subjects covered in our audit of the latest reviewed contracts of Wormhole LayerZero according to Scope to support you in forming an opinion on their security risks.

Sky offers a library for migrating the Ethereum-Solana governance and token bridges from Wormhole stack to LayerZero V2 stack. This review covers the governance spells facilitating the migration. The underlying contracts involved in the migration have been reviewed in separate reports, as detailed in the Assessment Overview.

The most critical subjects covered in our audit are functional correctness of the migration spells and migration procedure integrity. Security regarding all the aforementioned subjects is high.

The general subjects covered are code complexity, documentation, specification, and operational procedures. Security regarding all the aforementioned subjects is high.

In summary, we find that the codebase provides a high level of security.

It is important to note that the successful and secure execution of the migration depends on following the documented procedures in the README. Specifically, newly deployed contracts must be manually inspected (see Deployment Verification), parameters must be verified, results of spell 0 must be verified on both chains prior to executing spell 1, and for the token bridge migration it must be ensured that no funds are in flight before proceeding.

It is important to note that security audits are time-boxed and cannot uncover all vulnerabilities. They complement but don't replace other vital measures to secure a project.

The following sections will give an overview of the system, our methodology, the issues uncovered, and how they have been addressed. We are happy to receive questions and feedback to improve our service.

Sincerely yours,

ChainSecurity

1.1 Overview of the Findings

Below we provide a brief numerical overview of the findings and how they have been addressed.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	0

2 Assessment Overview

In this section, we briefly describe the overall structure and scope of the engagement, including the code commit which is referenced throughout this report.

2.1 Scope

The following Solidity libraries are in scope:

```
deploy/
    MigrationDeploy.sol
    MigrationInit.sol
```

The assessment was performed on the source code files inside the Wormhole LayerZero repository based on the documentation files. The table below indicates the code versions relevant to this report and when they were received.

V	Date	Commit Hash	Note
1	17 Oct 2025	17397879385d42521b0fe9783046b3cf25a9fec6	Initial Version

For the solidity libraries, the compiler version `0.8.22` was chosen and the `evm_version` was set to `shanghai`.

2.1.1 *Excluded from scope*

Any other files are out of scope such as tests, dependencies and third-party protocols. In particular:

- LayerZero V2 itself is out of scope and assumed to function correctly as per its documentation. The configurations of OApp / OFT and their settings (e.g. choice of libraries) on LayerZero V2 are out of scope. The DVN, executor, send and receive libraries are assumed to be properly configured.
- Wormhole itself is out of scope and assumed to function correctly as per its documentation. The majority of Wormhole guardians are assumed to be honest.

For the LayerZero and Wormhole NTT contracts deployed and the governance payload used, please refer to separate ChainSecurity reviews:

- Sky LZ OFT
- Sky LZ Governance OApp
- Sky LZ Governance Relay
- Sky Wormhole NTT Migration Update
- Sky Solana Crosschain Governance Payload Scripts

2.2 System Overview

This system overview describes the initially received version (Version 1) of the contracts as defined in the Assessment Overview.

Furthermore, in the findings section, we have added a version icon to each of the findings to increase the readability of the report.

Sky offers a library for migrating the Ethereum-Solana governance and token bridges from Wormhole stack to LayerZero V2 stack.

2.2.1 Migration Overview

Wormhole stack has been used by Sky to bridge USDS and to relay governance actions to Solana. NTT programs are deployed on both sides to handle the tokens transfer and a governance program is deployed on Solana that performs all the privileged actions on behalf of Sky governance. The USDS token on Solana is created with the SPL token program and uses the metaplex Metadata program.

Currently the following privileged roles exist on Solana:

- A NTT PDA holds the mint authority of the USDS program for token minting.
- A Governance PDA holds the metadata update authority, the USDS freeze authority, and the NTT program upgrade authority.

LayerZero V2 stack contracts are implemented (covered in ChainSecurity OFT Review and Governance OApp Review respectively.) to replace the Wormhole stack contracts:

- OFT contracts are implemented to facilitate token transfers with rate limits.
- Governance OApp contracts are implemented to relay and execute Sky governance spells on Solana.

2.2.2 Migration Steps

The migration is in general separated into two spells.

2.2.2.1 Preparations Before Spells

Before the spells execution, the following steps need to be accomplished by the deployers:

- The EVM contracts need to be deployed, including the OFT, Governance OApp and Governance Relay contracts. Their owners and delegates need to be switched and the configurations (for instance send receive libs, peers, enforced options) need to be set.
- The Solana programs need to be deployed, including the OFT and Governance OApp. Similarly they need to be configured with for instance send receive libs, peers, and enforced options. Further, the program upgrade authority (if exist) should be set to the authority PDA of the Governance OApp. The rate limit of OFT is also expected to be set.
- The new implementation of the EVM NTT contract should be deployed and the SVM NTT bytecode should be stored in a buffer contract. Note the changes in the new implementation are covered in another ChainSecurity review.

2.2.2.2 Spell 0

Spell 0 will eventually block the initiation of new outbound transfers on both the EVM and SVM NTT Managers:

- It immediately upgrades the EVM NTT Manager after some sanity checks.
- It sends out a governance message (`publishMessage()`) via Wormhole Core to upgrade the SVM NTT Manager.

It is expected the upgrade message to Solana will eventually be delivered after the transaction is finalized on Ethereum.

2.2.2.3 Spell 1

After verifying there is no outbound transfers in flight for NTT Manager on both sides, the migration can proceed to spell 1:

- Sanity checks of owner, endpoint, Solana peer, and endpoint delegate are performed on OApp and OFT.
- Other OFT related sanity checks are performed on OFT, in particular, the token, `defaultFeeBps`, `feeBps`, pausing status, rate limits, and accounting type are checked.
- The locked USDS tokens are migrated from the NTT Manager to the OFT.
- The rate limit of the OFT will be set.
- Three governance messages will be sent to Solana to transfer the mint authority, the freeze authority, and the metadata update authority.
- The SUSDS bridge will also be turned on in a similar way by setting up its rate limit. Note this requires another deployment of EVM OFT adapter and initialization of another SVM OFT instance with `init_oft` instruction.

Prior to spell 1 the deployed EVM and SVM contracts and their configurations need to be verified offchain.

Further note, once the SVM OFT is configured, outbound transfers can be initiated, which can be delivered immediately after spell 1. However, outbound transfers initiated from EVM OFT can only be finalized on Solana after the governance message to transfer mint authority is executed on Solana.

2.3 Trust Model

Contract / Program Deployer: not trusted; could deploy malicious bytecode or manipulate the initial states.

Sky Governance: fully trusted; assumed to inspect all the deployed bytecode, states, their configurations, and payload / parameters passed to the spell. Otherwise a misconfigured contract may lead to loss of locked USDS or Solana governance take over.

Wormhole: fully trusted; the Wormhole governance can DoS the cross-chain message by raising the fee in a front-running transaction.

Wormhole guardians: majority is assumed to be honest; otherwise a malicious message can be delivered and take over the Solana governance.

LayerZero: is out of scope for this review and is trusted to behave correctly and deliver messages to the correct destination. The LayerZero executor is expected to deliver messages with the specified message value and gas limit, but it is generally untrusted. In the worst case the attached native tokens may be lost, and if execution depends on native tokens or the gas limit the result may be unexpected. Depending on the exact configuration, LayerZero might be fully trusted to not censor messages or similar. However, the configuration is out-of-scope for this review.

The cross-chain Solana governance payload generation library was reviewed by ChainSecurity. It is expected the governance generates and simulates the payload with the correct mainnet addresses prior to the spell.

3 Limitations and use of report

Security assessments cannot uncover all existing vulnerabilities; even an assessment in which no vulnerabilities are found is not a guarantee of a secure system. However, code assessments enable the discovery of vulnerabilities that were overlooked during development and areas where additional security measures are necessary. In most cases, applications are either fully protected against a certain type of attack, or they are completely unprotected against it. Some of the issues may affect the entire application, while some lack protection only in certain areas. This is why we carry out a source code assessment aimed at determining all locations that need to be fixed. Within the customer-determined time frame, ChainSecurity has performed an assessment in order to discover as many vulnerabilities as possible.

The focus of our assessment was limited to the code parts defined in the engagement letter. We assessed whether the project follows the provided specifications. These assessments are based on the provided threat model and trust assumptions. We draw attention to the fact that due to inherent limitations in any software development process and software product, an inherent risk exists that even major failures or malfunctions can remain undetected. Further uncertainties exist in any software product or application used during the development, which itself cannot be free from any error or failures. These preconditions can have an impact on the system's code and/or functions and/or operation. We did not assess the underlying third-party infrastructure which adds further inherent risks as we rely on the correct execution of the included third-party technology stack itself. Report readers should also take into account that over the life cycle of any software, changes to the product itself or to the environment in which it is operated can have an impact leading to operational behaviors other than those initially determined in the business specification.

4 Terminology

For the purpose of this assessment, we adopt the following terminology. To classify the severity of our findings, we determine the likelihood and impact (according to the CVSS risk rating methodology).

- *Likelihood* represents the likelihood of a finding to be triggered or exploited in practice
- *Impact* specifies the technical and business-related consequences of a finding
- *Severity* is derived based on the likelihood and the impact

We categorize the findings into four distinct categories, depending on their severity. These severities are derived from the likelihood and the impact using the following table, following a standard risk assessment procedure.

Likelihood	Impact		
	High	Medium	Low
High	Critical	High	Medium
Medium	High	Medium	Low
Low	Medium	Low	Low

As seen in the table above, findings that have both a high likelihood and a high impact are classified as critical. Intuitively, such findings are likely to be triggered and cause significant disruption. Overall, the severity correlates with the associated risk. However, every finding's risk should always be closely checked, regardless of severity.

5 Open Findings

In this section, we describe our findings. The findings are split into these different categories:

Below we provide a numerical overview of the identified findings, split up by their severity.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	0

6 Notes

We leverage this section to highlight further findings that are not necessarily issues. The mentioned topics serve to clarify or support the report, but do not require an immediate modification inside the project. Instead, they should raise awareness in order to improve the overall understanding.

6.1 Deployment Verification

Note **Version 1**

Since deployment of the involved contracts is not performed by the governance directly, special care has to be taken that all contracts have been deployed correctly.

We therefore assume that the initcode, bytecode, traces and storage (e.g. mappings) are checked for unintended entries, calls or similar.

While the spell performs some limited sanity checks, these checks are nevertheless essential. The sanity checks done in the spells are intentionally covering critical parameters only, designed to catch honest deployer mistakes (e.g., mistakenly setting non-zero rate limit values on Ethereum).

The following checks and validations are expected to be performed off-chain:

1. Mappings and non critical parameters (such as amount in flight, last updated, window, oftVersion, oAppVersion, pauser, sharedDecimals, msgInspector) which are excluded from verification in the spell.
2. LayerZero configurations (such as send / receive library, configs) and Wormhole related configurations.
3. The deployed SVM programs and its configurations.
4. The governance payloads for SVM instructions and the accounts involved.

6.2 Execution Confirmation

Note **Version 1**

The migration process involves the execution of several cross-chain governance messages. Since the operations on EVM and SVM are asynchronous, it should be ensured that the migration only proceeds to the next step when these asynchronous operations are accomplished, namely by confirming off-chain that both the EVM and SVM actions are finalized.

6.3 `canCallTarget` of Governance OApp Is Not Set

Note **Version 1**

In `initMigrationStep1`, the deployed EVM Governance OApp is sanity checked, however its `canCallTarget` is not configured. Namely, sending messages cross-chain `sendTx()` is still prohibited.

Governance still needs to configure the `canCallTarget` before relaying a governance message to a target on the destination chain. In the context of Sky governance, the governance relay contract should be whitelisted to send messages.