PUBLIC

Code Assessment

of the Sky Wormhole NTT Migration Updates

October 20, 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 Sky Wormhole NTT according to Scope to support you in forming an opinion on their security risks.

Sky implements an upgrade for the USDS Wormhole bridge for Ethereum and Solana. The upgrade disables the initiation of transfers to allow for an eventual migration of the token bridge to the new LayerZero infrastructure.

The most critical subjects covered in our audit are access control, completeness of removed functionality and suitability for the described migration process. The general subjects covered are upgradeability and documentation. Security regarding all the aforementioned subjects is high.

In summary, we find that the changes introduced provide a high level of security. However, note that the security heavily relies on the non-audited code, too.

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 scope of the assessment was limited to review the functionality changes of the existing Sky Wormhole bridge in following files:

```
evm/src/
    NttManager.sol
    INttManager.sol
solana/programs/native-token-transfers/src/
    lib.rs
    instructions/
        mod.rs
        transfer.rs
        transfer_mint_authority.rs
```

The assessment was performed on the source code files inside the Sky Wormhole NTT 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	11 Sep 2025	832781e228c078a757c1e3c26bc1714ca053b440	Initial Version
2	17 Oct 2025	e296f1ef1b6ff2bae7186c6106fc8c93fc925dad	Second Version

Note that as part of this review only the diff with commit `a38b0f69e7b4ca3f7dbebe79c525f12e70ca9302` is in scope.

For the solidity smart contracts, the compiler version `0.8.19` was chosen and `evm_version` was set to `london`.

For the solana smart contracts, the anchor version `v0.29` and the solana version `v1.18.10` were chosen.

2.1.1 Excluded from scope

This review is performed under the assumption that the existing Sky Wormhole NTT bridge is correct and secure as documented and focused only on the functionality changes as follows:

- The removal of transfer functionality of both EVM and SVM implementation.
- The new instruction `transfer_mint_authority` in the SVM implementation.

The goal of the review is to ensure that the initiation of transfers is disabled and that functionality is provided to allow for the planned migration to the newer LZ bridging infrastructure.

Any other files are not in scope of this review. However, other files have been considered for analyzing the completeness of changes.

Wormhole itself is out of scope and assumed to function correctly as per its documentation.

The migration process from the Wormhole bridge to LayerZero bridge is out of scope of this review.

Further, the Solana programs make use of the Anchor framework and integrate with SPL / SPL2022 programs, which are assumed to function properly and work as documented.

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 an update of Sky Wormhole NTT, the infrastructure for cross-chain token transfers and governance. In this update, functionalities to initiate new cross-chain token transfers have been removed and an instruction to transfer SPL mint authority has been introduced.

2.2.1 Overview

The integration and interaction flow for a Wormhole NTT transfer is outlined below while the detailed execution may vary on different chains:

1. Source Chain User: Initiates a transfer by calling the source chain NTT Manager contract, specifying the token amount, destination chain, and recipient address.
2. Source Chain NTT Manager & Transceiver: The NTT Manager validates the request and secures the tokens from the user by either locking them in the contract or burning them. It then constructs a standardized NTT message payload and instructs its associated Transceiver contract to send this payload through the Wormhole Core protocol. The user pays the necessary fees during this step.
3. Wormhole Network (Guardians & Relayers). The Wormhole Guardian network observes the message emitted on the source chain, verifies it, and produces a signed Verifiable Action Approval (VAA). An off-chain Relayer then picks up this VAA and delivers it to the destination chain.
4. Destination Chain Transceiver & NTT Manager: The Relayer submits the VAA to the destination chain Transceiver contract. The Transceiver verifies the VAA's authenticity against the on-chain Wormhole Core contract. Upon successful verification, it decodes the NTT message and invokes the destination NTT Manager, which then completes the transfer by either minting new tokens or unlocking existing ones for the recipient.

2.2.2 Migration Necessary Changes

To facilitate the migration from Wormhole NTT to LayerZero OFT, it is necessary to pause the initiation of cross-chain transfer on all peers to prevent in-flight messages. Additionally, it is required that locked tokens and privileges can be migrated. Hence, the following changes were introduced:

- In the **EVM** contract: Both `transfer()` functions (with different input parameters) have been removed, hence no new outbound transfers can be initiated in the new implementation. However, a queued outbound transfer can still be finalized with `completeOutboundQueuedTransfer()`. Note that inbound transfers are not influenced. Further, `migrateLockedTokens()` has been introduced to transfer the locked tokens to a recipient (i.e. new token bridge) by the owner.
- In the **SVM** contract: The instruction `transfer_burn` has been removed. Assuming the bridge mode is burn instead of lock, `transfer_lock` cannot be used hence this prevents initiating new outbound transfers. The pending outbound transfers in the outbox and inbound transfers are not influenced. Futher, a new instruction `transfer_mint_authority` has been introduced to set the SPL token's mint authority to a new address.

The deployed EVM and SVM contracts will be upgraded to the new implementations before the final migration to prevent unintended in flight transfers being stuck.

2.3 Trust Model

For the **EVM** contracts:

- The owner of the NTT Manager is fully trusted (expected to be the Sky Pause Proxy). It has the privileges to ugprade the contract implementation, set configurations, and migrate the locked tokens. In the worst case it can retrieve all locked tokens.

For the **SVM** contracts:

- The `upgrade_authority` of the Program is trusted, assumed to be a PDA of the SVM governance program that is fully controlled by the L1 Sky governance. Otherwise, it can upgrade the program to a malicious implementation.
- The `mint_authority` of the SPL token is fully trusted, assumed to be a PDA of the native-token-transfers program. Otherwise, it can freely mint the underlying tokens. It is expected to be transferred to the PDA controlled by the LayerZero OFT during the migration.
- The `freeze_authority` of the SPL token is fully trusted, assumed to be a PDA of the SVM governance program that is fully controlled by the L1 Sky governance. Otherwise, it can freely freeze token accounts.
- The `update_authority` of the SPL token is fully trusted, assumed to be a PDA of the SVM governance program that is fully controlled by the L1 Sky governance. Otherwise, it can change the token metadata at will.

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 Informational

We utilize this section to point out informational findings that are less severe than issues. These informational issues allow us to point out more theoretical findings. Their explanation hopefully improves the overall understanding of the project's security. Furthermore, we point out findings which are unrelated to security.

6.1 Unused Functionality

Informational **Version 1** **Acknowledged**

CS-NTT-MIG-001

Ethereum. The `transfer()` functions have been removed to disable the initiation of cross-chain transfers. Additionally, the internal function `_transferEntryPoint` has been removed. Other now unused internal functionality still exists. Below is a list of such:

1. Various internal functions: e.g. `_getTokenBalanceOf()`, `_enqueueOutboundTransfer()` and various other functions.
2. Various errors: e.g. `BurnAmountDifferentThanBalanceDiff`, `NotEnoughCapacity` and various other errors.
3. Various events: e.g. `OutboundTransferRateLimited`, `OutboundTransferQueued` and various other events.

Note that removing the functions could make the diff harder to compare. However, removing internal functionality related to the initiation of outbound transfers could further help in following why no outbound transfer are possible anymore.

Solana. The `transfer_burn` entry point has been removed to disable the initiation of cross-chain transfers. Note that the mode on Solana is expected to be "burning".

However, the entry point `transfer_lock` still exists. While not being used it could be meaningful to remove it.

Further, other internal functionality is still in code similar to the Ethereum side (e.g. `insert_into_outbox`).

7 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.

7.1 Migration Considerations

Note Version 1

Timeline Considerations

- **Pause transfer**: On Ethereum the transfer functionality is paused immediately after the implementation upgrade, however, the upgrade of SVM program is delayed by the cross-chain message delivery and execution. Hence, there may be a period when outbound transfer is paused on Ethereum but not on Solana.
- **Inflight transfers**: Sufficient time is required between transfer pausing and bridge migration to ensure all inflight transfers are finalized on the destination chain, otherwise these transfers will be stuck due to insufficient locked balance or missing minting privilege. Governance should ideally ensure that queued transfers are handled.
- **Migration delay**: Once the migration happens, locked funds on Ethereum will be transferred and the new bridge will be setup for use immediately, however, the cross-chain message to migrate the Solana bridge incurs a delay (at least the Ethereum functionality period). During this period, outbound transfers can be initiated on the new Ethereum bridge, however, they cannot be finalized on Solana.

Roles Considerations

- **Ethereum Roles**: Since the bridges use a lock-and-release approach, the only privileged roles are the NTT Manager's owner and pauser, which are both set to Sky Pause Proxy. These roles do not need to be transferred.
- **Solana Roles**: Since the bridges use a mint-and-burn approach, the token related roles need to be transferred. In particular, the freeze authority and update authority need to be transferred to the new governance bridge (The Sky LayerZero Governance OApp CPI Authority). And the mint authority need to be transferred to the Sky OFT bridge.