Wormhole Foundation EVM-NTT Diff Audit Report

Prepared by Cyfrin

Version 1.0

Lead Auditors

0kage

Giovanni Di Siena

Assisting Auditors

Hans

July 23, 2024

Contents

1 About Cyfrin

Cyfrin is a Web3 security company dedicated to bringing industry-leading protection and education to our partners and their projects. Our goal is to create a safe, reliable, and transparent environment for everyone in Web3 and DeFi. Learn more about us at cyfrin.io.

2 Disclaimer

The Cyfrin team makes every effort to find as many vulnerabilities in the code as possible in the given time but holds no responsibility for the findings in this document. A security audit by the team does not endorse the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the solidity implementation of the contracts.

3 Risk Classification

	Impact: High	**Impact: Medium**	**Impact: Low**
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4 Protocol Summary

Wormhole Native Token Transfers (NTT) framework is designed to facilitate the transfer of tokens across different blockchain networks without relying on liquidity pools, offering an open, flexibile and composable solution. The framework grants integrators complete control over the behavior of NTTs on each chain, including the token standard and meta data.

5 Audit Scope

Cyfrin conducted a diff audit of the EVM-related Solidity contracts of the Wormhole Native Token Transfers. The previous audit was performed on commit f4e2277, and the current audit focuses on commit 0d37b0f. The scope of the audit was determined by the differences between these two commits, as outlined by the following git diff command:

```
git diff f4e2277b358349dbfb8a654d19a925628d48a8af 0d37b0f4975084492c72ca881c1218d6e1aae9e3
evm/src
```

The changes reviewed in this diff included modifications to the Solidity contracts located in the directories:

- evm/src/interfaces/*
- evm/src/NttManager/*
- evm/src/Transceiver/*
- evm/src/libraries/*
- evm/src/wormhole/*.

6 Executive Summary

Over the course of 4 days, the Cyfrin team conducted an audit on the Wormhole Foundation EVM-NTT Diff smart contracts provided by Wormhole Foundation. In this period, a total of 4 issues were found.

The current diff audit reviewed all changes made to the EVM Solidity contracts from the previously audited commit f4e2277 to the current commit 0d37b0f. The audit found no issues with a security impact. There were four informational findings related to how events and errors were managed in the codebase. These findings are intended to improve code quality and maintainability but do not pose any security risks.

Summary

Project Name	Wormhole Foundation EVM-NTT Diff
Repository	example-native-token-transfers
Commit	0d37b0f49750...
Audit Timeline	Jul 18th - Jul 23rd
Methods	Manual Review, Stateful Fuzzing

Issues Found

Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	0
Informational	4
Gas Optimizations	0
Total Issues	4

Summary of Findings

[I-1] Incorrectly documented error selector	Open
[I-2] Inconsistent inline documentation for errors and events	Open
[I-3] Lack of events for setting inbound and outbound limits	Open
[I-4] Lack of indexing in `TransferSent` event	Open

7 Findings

7.1 Informational

7.1.1 Incorrectly documented error selector

Description: The `bytes4` error selector for the `IWormholeTransceiver::TransferAlreadyCompletedError` is incorrectly documented as `0x406e719e`. The correct selector is `0xb4c3b00c`.

```
    /// @notice Error when the VAA has already been consumed.
    /// @dev Selector: 0x406e719e.
    /// @param vaaHash The hash of the VAA.
    error TransferAlreadyCompleted(bytes32 vaaHash);
```

Recommended Mitigation: Consider updating the selector to `0xb4c3b00c`.

7.1.2 Inconsistent inline documentation for errors and events

Description: The current codebase follows an inline documentation standard for events and errors, including parameter descriptions and `topic[0]` for events and `bytes4` selectors for errors. However, some events and errors lack either parameter descriptions, selectors, or both. This inconsistency can reduce code readability and maintainability.

Here are a few examples from `INttManager.sol`

```
    /// @notice The caller is not the deployer.
    error UnexpectedDeployer(address expectedOwner, address owner);

    /// @notice Peer for the chain does not match the configuration.
    /// @param chainId ChainId of the source chain.
    /// @param peerAddress Address of the peer nttManager contract.
    error InvalidPeer(uint16 chainId, bytes32 peerAddress);

   /// @notice Peer chain ID cannot be zero.
    error InvalidPeerChainIdZero();

    /// @notice Peer cannot be the zero address.
    error InvalidPeerZeroAddress();

    /// @notice Peer cannot have zero decimals.
    error InvalidPeerDecimals();
```

Recommended Mitigation: Ensure consistent documentation across all event and error definitions by including parameter descriptions, `topic[0]` and `bytes4` selectors where applicable.

7.1.3 Lack of events for setting inbound and outbound limits

Description: The `NttManager::setPeer` function sets a peer `NttManager` contract address on a foreign chain. The `inboundLimit` is now passed as an input when setting a peer contract. In the earlier implementation, inboundLimit was set to `type(uint64).max`. However, this input is missing from the `PeerUpdated` event, which does not reflect the change in the `setPeer` input parameters.

Recommended Mitigation: Consider including the `inboundLimit` as part of the `PeerUpdated` event to accurately reflect the parameters set by the `setPeer` function. Additionally, in the context of third party integrations, since the inbound and outbound limits might be updated multiple times for different destination chains, it is recommended to add an event emission whenever the `NttManager` owner sets the inbound or outbound limit. This will improve transparency and traceability of these parameter changes.

7.1.4 Lack of indexing in `TransferSent` event

Description: The `INttManager::TransferSent` event is emitted when a message is sent from the `NttManager` of the source chain. The current event signature does not index the `recipient` and `refundAddress` parameters. When transfers are performed at scale, this lack of indexing might impede the searchability of transfers across chains.

Recommended Mitigation: Consider indexing the `recipient` and `refundAddress` parameters in the `TransferSent` event for improved searchability.