OMNI NETWORK

FeeOracleV2 Contract Review

Version: 1.0

December, 2024

Contents

Introduction

Sigma Prime was commercially engaged to perform a time-boxed security review of the Omni Network smart contracts. The review focused solely on the security aspects of the Solidity implementation of the contract, though general recommendations and informational comments are also provided.

Disclaimer

Sigma Prime makes all effort but holds no responsibility for the findings of this security review. Sigma Prime does not provide any guarantees relating to the function of the smart contract in scope. Sigma Prime makes no judgements on, or provides any security review, regarding the underlying business model or the individuals involved in the project.

Document Structure

The first section provides an overview of the functionality of the Omni Network smart contracts contained within the scope of the security review. A summary followed by a detailed review of the discovered vulnerabilities is then given which assigns each vulnerability a severity rating (see Vulnerability Severity Classification), an *open/closed/resolved* status and a recommendation. Additionally, findings which do not have direct security implications (but are potentially of interest) are marked as *informational*.

The appendix provides additional documentation, including the severity matrix used to classify vulnerabilities within the Omni Network smart contracts in scope.

Overview

Omni is a chain abstraction protocol that enables developers to create applications that are accessible across multiple rollups. Apps can interact with the `OmniPortal` contract to send cross-chain messages.

This review focused on the `FeeOracleV2` updates. The fee oracle is in charge of calculating gas costs for cross chain messages, including logic to charge different fees for different destination chains. A number of gas and storage optimisations have been implemented in the second version since the first version.

Security Assessment Summary

Scope

The review was conducted on the files hosted on the omni-network/omni repository.

The scope of this time-boxed review was strictly limited to files at commit 1e69efc.

Note: third party libraries and dependencies were excluded from the scope of this assessment.

Approach

The manual review focused on identifying issues associated with the business logic implementation of the contracts. This includes their internal interactions, intended functionality and correct implementation with respect to the underlying functionality of the Ethereum Virtual Machine (for example, verifying correct storage/memory layout).

Additionally, the manual review process focused on identifying vulnerabilities related to known Solidity antipatterns and attack vectors, such as re-entrancy, front-running, integer overflow/underflow and correct visibility specifiers.

For a more detailed, but non-exhaustive list of examined vectors, see [1, 2].

To support this review, the testing team also utilised the following automated testing tools:

- Mythril: `https://github.com/ConsenSys/mythril`
- Slither: `https://github.com/trailofbits/slither`
- Surya: `https://github.com/ConsenSys/surya`
- Aderyn: `https://github.com/Cyfrin/aderyn`

Output for these automated tools is available upon request.

Coverage Limitations

Due to the time-boxed nature of this review, all documented vulnerabilities reflect best effort within the allotted, limited engagement time. As such, Sigma Prime recommends to further investigate areas of the code, and any related functionality, where majority of critical and high risk vulnerabilities were identified.

Findings Summary

The testing team identified a total of 1 issues during this assessment. Categorised by their severity:

- Informational: 1 issue.

Detailed Findings

This section provides a detailed description of the vulnerabilities identified within the Omni Network smart contracts in scope. Each vulnerability has a severity classification which is determined from the likelihood and impact of each issue by the matrix given in the Appendix: Vulnerability Severity Classification.

A number of additional properties of the contracts, including gas optimisations, are also described in this section and are labelled as "informational".

Each vulnerability is also assigned a **status**:

- ***Open:*** the issue has not been addressed by the project team.
- ***Resolved:*** the issue was acknowledged by the project team and updates to the affected contract(s) have been made to mitigate the related risk.
- ***Closed:*** the issue was acknowledged by the project team but no further actions have been taken.

Summary of Findings

ID	Description	Severity	Status
PREF-01	Slot Ordering Not Preserved From V1	Informational	**Resolved**

PREF-01	Slot Ordering Not Preserved From V1
Asset	`FeeOracleV2.sol`
Status	**Resolved:** See Resolution
Rating	Informational

Description

Upgrading from `FeeOracleV1` does not preserve the storage slot ordering.

Additional storage variables have been added to `FeeOracleV2` and some variables in the first version have been removed. Furthermore, the underlying struct for `_feeParams` has been modified, changing the size and number of fields. As a result the if the `FeeOracleV1` is upgraded to a `FeeOracleV2` implementation the storage variables would not be aligned. Therefore, the proxy contract would not function correctly.

Recommendations

To mediate this issue two solutions exist.

The first solution is to reorganise the state variables of the version 2 contract to match the original.

An alternate solution is to deploy a new proxy and implementation for the `FeeOracleV2` and

Resolution

The development team were aware of this issue and intend to deploy a new proxy and implementation for `FeeOracleV2`. The function `setFeeOracle(address)` can be used on `OmniPortal` to migrate to the version two fee oracle.

Appendix A Vulnerability Severity Classification

This security review classifies vulnerabilities based on their potential impact and likelihood of occurance. The total severity of a vulnerability is derived from these two metrics based on the following matrix.

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

Table 1: Severity Matrix - How the severity of a vulnerability is given based on the *impact* and the *likelihood* of a vulnerability.

References

[1] Sigma Prime. Solidity Security. Blog, 2018, Available: https://blog.sigmaprime.io/solidity-security.html. [Accessed 2018].

[2] NCC Group. DASP - Top 10. Website, 2018, Available: http://www.dasp.co/. [Accessed 2018].