OMG Network

Rapid Code Review

November 3, 2020

Prepared For:
Gerhard Wagner | *OMG Network*
gerhard@omg.network

Prepared By:
Alexander Remie | *Trail of Bits*
alexander.remie@trailofbits.com

Review Summary

Code Maturity Evaluation

Project Dashboard

Appendix A. Code Maturity Classifications

Review Summary

From October 28 to 30, 2020, Trail of Bits performed an assessment of the `plasma-network` repository with one engineer, and reported no findings. We reviewed the v2.0.0 branch, which contains a non-final version of the OMG Network v2 contracts. The goal of this assessment was to give OMG Network additional feedback on the security posture of the contracts, identify areas where they can improve, and identify any issues.

Trail of Bits found one issue of informational severity. Our review suggests that the following components have significant complexity and risks, and require additional review:

- The processing of In Flight Exits (especially those including piggybacks) and the different scenarios that could play out.
- Edge cases in the `RLPReader`.
- `MoreVP` Finalization protocol implementation.

Additionally, due to time restrictions, some areas were not reviewed during this rapid review:

- Spending of Fee Claim `utxos`.
- Liquidity contract.
- Multisig contract.

On the following page, we review the maturity of the codebase and the likelihood of future issues. In each area of control, we rate the maturity from strong to weak, or missing, and give a brief explanation of our reasoning. OMG Network should consider these steps to improve their security maturity:

- Add (more elaborate) comments to all contracts, libraries, and functions.
- Fuzz the `RLPReader` library to find edge case bugs.
- Update the documentation if it's not up to date, and try to centralize it, i.e., reduce the number of locations in which documentation is stored (Docs Website + Github doc files).
- Simplify the add exit queue mechanism (as explained in the issue).
- Perform a proper audit on the codebase to verify the security of In Flight Exits.

Code Maturity Evaluation

Category Name	Description
Access Controls	**Strong.** There are a limited number of privileged operations/roles and they are clearly documented.
Arithmetic	**Further Investigation Required.** No issues were discovered related to arithmetic. However, due to the amount and size of contracts, not all contracts/arithmetic could be checked.
Assembly Use	**Further Investigation Required.** The RLPReader contains a significant amount of assembly code. Although this library seems to be used by numerous projects, it might still contain bugs that could be uncovered through fuzzing.
Centralization	**Strong.** The Authority represents a single point of failure. However, this is clearly cited in the documentation, and users can exit at any time. Another centralized role is the Maintainer, but it's controlled through a multi-sig. Also, changes made by the Maintainer only go into effect after a two-week delay, giving users time to exit if they disagree.
Contract Upgradeability	**Not Applicable.** The contracts are not upgradeable.
Function Composition	**Strong.** The functions are grouped and named consistently, and each serves a single purpose.
Front-Running	**Strong.** The `senderData` function argument provides basic protection against front-running.
Monitoring	**Strong.** Events are consistently emitted. Failed bond/bounty ETH transfers don't cause a revert (and DoS), but continue execution and emit an event to log the failed transfer.
Specification	**Moderate.** The high-level documentation was comprehensive, although scattered in multiple places, and it's unclear if everything is up to date. Also, many functions, contracts, and libraries in the codebase were lacking (sufficient) comments.
Testing & Verification	**Satisfactory.** OMG Network provided many tests, and tested both the happy path and failure cases. OMG Network stated that they used MythX. However, fuzzing could be used on the `RLPReader`.

Project Dashboard

Commit hashes of the reviewed repositories:

- `plasma-framework: 5ced05a49f4bb141d15d921c40432f214573c8e8`

Appendix A. Code Maturity Classifications

Code Maturity Classes	
Category Name	**Description**
Access Controls	Related to the authentication and authorization of components.
Arithmetic	Related to the proper use of mathematical operations and semantics.
Assembly Use	Related to the use of inline assembly.
Centralization	Related to the existence of a single point of failure.
Upgradeability	Related to contract upgradeability.
Function Composition	Related to separation of the logic into functions with clear purpose.
Front-Running	Related to resilience against front-running.
Key Management	Related to the existence of proper procedures for key generation, distribution, and access.
Monitoring	Related to use of events and monitoring procedures.
Specification	Related to the expected codebase documentation.
Testing & Verification	Related to the use of testing techniques (unit tests, fuzzing, symbolic execution, etc.).

Rating Criteria	
Rating	**Description**
Strong	The component was reviewed and no concerns were found.
Satisfactory	The component had only minor issues.
Moderate	The component had some issues.
Weak	The component led to multiple issues; more issues might be present.
Missing	The component was missing.

Not Applicable	The component is not applicable.
Not Considered	The component was not reviewed.
Further Investigation Required	The component requires further investigation.