Quantstamp Contract Security Certificate

October 29, 2018 | Quantstamp Verified

Omisego Plasma MVP

This smart contract audit was prepared by Quantstamp, the protocol for securing smart contracts.

Executive Summary

Type	Protocol Proof of Concept
Auditors	Kacper Bąk, Senior Research Engineer Jan Gorzny, Blockchain Research Lead Martin Derka, Senior Research Engineer Ivdal Ciko, Research Engineer Jon Stornetta, Blockchain Researcher
Timeline	2018-09-24 through 2018-10-11
Languages	Solidity, Python
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Tesuji Plasma Blockchain Design
Source Code	Repository Commit Current
	plasma-contracts d0ebaaf 3ccb007

Total Issues	19 (0 Fixed)
High Risk Issues	2
Medium Risk Issues	0
Low Risk Issues	2
Informational Risk Issues	10
Undetermined Risk Issues	5

Severity Categories	
High	The issue puts a large number of users' sensitive information at risk, or is likely to lead to a catastrophic impact for client's reputation or serious financial implications for client and users.
Medium	The issue puts a subset of users' sensitive information at risk, would be exploitable by a knowledgeable insider (malicious or negligent), or is reasonably likely to lead to moderate financial impacts.
Low	The risk is relatively small and could not be exploited on a recurring basis, or is a vulnerability that is unlikely to have an impact in view of the circumstances of the project.
Informational	The issue does not pose an immediate threat to contract operation or usage, but is deemed to be of relevance to the project, enhance engineering best practices, or deliver robustness.
Undetermined	The impact of the issue is uncertain.

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the omisego plasma mvp repository for security related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for include (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number quirks (any)
- Compiler compatibility issues, recommendations
- Denial of service / logic errors
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clarity, redundancy, duplication
- Gas usage
- Arbitrary token minting

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following:
   1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contracts.
   2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Goals

This report focused on evaluating security of smart contracts, as requested by the omisego plasma mvp team. Specific questions to answer:

- can users' funds get locked up in the Plasma child chain?
- can users successfully exit their funds from the root chain?
- can the operator steal users' funds?
- are hash protected signatures used?

Changelog

This report focused on evaluating security of smart contracts, as requested by the omisego plasma mvp team. Specific questions to answer:

- Date: 2018-10-11 - Initial report
- Date: 2018-10-16 - Added recommendations and updated test section
- Date: 2018-10-29 - Updated the relevant part of the JSE/Geneva contracts discussion and results

Overall Assessment

The contracts provide a prototype implementation of Plasma. Quantstamp has commented on numerous ways in which the code could be improved, including manuals that are meant to protect against except (fraud in current SolidEX), and a possibility of carrying out a denial of service attack on exits. Furthermore, we point out a number of recommendations to ensure that the code conforms to the best practices.

Toolset

The below notes outline the setup and steps performed in the process of this audit.

Setup

Testing setup:

- Truffle v4.1.5
- MyEtherWallet v3.7
- truffle-flattener v0.0.5
- MAAS contracts: d0ebfbf
- Security

Testing

```
truffle compile
truffle migrate --reset --compile-all
truffle exec ./scripts/deposit.js 10
truffle exec ./scripts/submitBlock.js
truffle exec ./scripts/startExit.js 0 0 0
truffle exec ./scripts/challenge.js
truffle exec ./scripts/challenge.js
truffle exec ./scripts/challenge.js
```