BlockchainLabs.NZ

Blockchain Solution Design & Development

Audit report for POA Parity Bridge Contracts

May 2018

=Contents

Preamble

This audit report was undertaken by BlockchainLabs.nz for the purpose of providing feedback regarding POA Bridge Smart Contracts.

It has subsequently been shared publicly without any express or implied warranty.

Solidity contracts were sourced from the public Github repo poanetwork/poa-parity-bridge-contracts and the most recent commit we have audited is this 23d45d2d0a10d8e38704e7610c302aa3ebbe5dd6 - we would encourage all community members and token holders to make their own assessment of the contracts.

Scope

The following contracts were subject for static, dynamic and functional analyses:

- Libraries
  - Helpers.sol
  - Message.sol
  - SafeMath.sol
- Upgradeability
  - EternalStorage.sol
  - EternalStorageProxy.sol
  - OwnedUpgradeabilityProxy.sol
  - Proxy.sol
  - UpgradeabilityOwnerStorage.sol
  - UpgradeabilityProxy.sol
  - UpgradeabilityStorage.sol
- Upgradeable Contracts
  - Ownable.sol
  - U_BridgeValidators.sol
  - U_ForeignBridge.sol
  - U_Validatable.sol

Focus areas

The audit report is focused on the following key areas - though this is not an exhaustive list.

Correctness

- No correctness defects uncovered during static analysis?
- No implemented contract violations uncovered during execution?
- No other generic incorrect behaviour detected during execution?

- Adherence to adopted standards such as ERC20?

Testability

- Test coverage across all functions and events?
- Test cases for both expected behaviour and failure modes?
- Settings for easy testing of a range of parameters?
- No reliance on nested callback functions or console logs?
- Avoidance of test scenarios calling other test scenarios?

Security

- No presence of known security weaknesses?
- No funds at risk of malicious attempts to withdraw/transfer?
- No funds at risk of control fraud?
- Prevention of Integer Overflow or Underflow?

Best Practice

- Explicit labeling for the visibility of functions and state variables?
- Proper management of gas limits and nested execution?
- Latest version of the Solidity compiler?

Analysis Reports

- Functional Tests
- Dynamic Tests
- Gas Usage
- Test Coverage

Issues

Severity Description

Minor	A defect that does not have a material impact on the contract execution and is likely to be subjective.
Moderate	A defect that could impact the desired outcome of the contract execution in a specific scenario.
Major	A defect that impacts the desired outcome of the contract execution or introduces a weakness that may be exploited.

Critical	A defect that presents a significant security vulnerability or failure of the contract across a range of scenarios.

Minor

- None found

Moderate

- If the two bridge contracts have different settings (i.e `maxPerTx`, `minPerTx`) then it is possible to send funds/tokens to a contract, have it accepted, then have the transaction fail when it transfers through the bridge. This will result in the funds/tokens being stuck in the contract.

Major

- None found

Critical

- None found

Observations

- It is possible to set the minimum number of required signatures from validators to zero: Foreign: 0x116ab7
- It is possible to add validators that don't have any funds in their accounts: Home: 0x6b0123 / Foreign: 0x532290
- You can set the MAX transaction limit to less than the MIN transaction limit Foreign: 0xe78264
- You can set the MAX transaction limit to 0. This could be useful for locking transfers, but a dedicated `lock` function would be better. Foreign: 0x18bbfa
- When you upgrade the implementation of contracts:
    - you can set it to a previous implementation Foreign: 0x01d4449
    - and to a non-contract address Foreign: 0x3f8fe1

Conclusion

We are satisfied that these Smart Contracts do not exhibit any known security vulnerabilities. Overall the code is well written and the developers have been responsive and active throughout the audit process. The contracts show care taken by the developers to follow best practices and a strong knowledge of Solidity. There is very high test coverage which should increase

confidence in the security of these contracts, and their maintainability in the future. As part of our auditing process we added some new tests to improve the coverage.

Disclaimer

Our team uses our current understanding of the best practises for Solidity and Smart Contracts. Development in Solidity and for Blockchain is an emerging area of software engineering which still has a lot of room to grow, hence our current understanding of best practice may not find all of the issues in this code and design.

We have not analysed any of the assembly code generated by the Solidity compiler. We have not verified the deployment process and configurations of the contracts. We have only analysed the code outlined in the scope. We have not verified any of the claims made by any of the organisations behind this code.

Security audits do not warrant bug-free code. We encourage all users interacting with smart contract code to continue to analyse and inform themselves of any risks before interacting with any smart contracts.