BlockchainLabs.NZ

Blockchain Solution Design & Development

Audit report for POA PoPA Smart Contracts

May 2018

Contents

POA Network - Proof of Physical Address (PoPA) Audit Report

Preamble

This audit report was undertaken by BlockchainLabs.nz for the purpose of providing feedback to POA Network.

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

Solidity contracts were sourced from github.com `e259cec1f`, and stored at BlockchainLabsNZ/poa-popa/tree/audit-blnz

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:

- EthereumClaimsRegistry.sol
- EthereumClaimsRegistryInterface.sol
- PhysicalAddressClaim.sol
- ProofOfPhysicalAddress.sol

Focus areas

The audit report focuses on the following key areas, although this list is not exhaustive.

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 Analysis
- Dynamic Analysis
- Gas Consumption
- 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

- Checks for contract owner and user existence are repeated for many times in different functions in `ProofOfPhysicalAddress.sol`. Suggest to use a modifier to replace repeated codes. Check for contract owner : Line61, Line66, Line74, Line83 Check for user existence: Line97, Line111, Line129, Line147, Line161, Line169, Line204, Line221, Line293, Line325
- View on GitHub
- Fixed - https://github.com/poanetwork/poa-popa/pull/127

- The claims registry doesn't implement the claims registry interface - it does implement all the features in the interface - however by not explicitly declaring inheritance of the interface the project is not able to benefit from the additional checks that the Solidity compiler would perform if it was declared. e.g "contract EthereumClaimsRegistry is EthereumClaimsRegistryInterface {"
  View on GitHub
- Fixed - https://github.com/poanetwork/poa-popa/pull/128

Moderate

- None found

Major

- None found

Critical

- None found

Observations

Event log is beneficial to the observation of setting variables

When calling `setSigner()` with a new signer address, it is highly recommended that you emit an event to log the execution. This is not standard behavior, though it helps you track the history and lets you notice the variable has been changed. Similar functions are listed below: `setRegistry()`, `registerAddress()`, `unregisterAddress()`

Fixed - https://github.com/poanetwork/poa-popa/pull/129

Third party Tokens can be sent to the contract with no way of retrieving them

It is possible for someone to transfer tokens to the contract address when they meant to send ETH. It is best practice to implement a function for the owner to retrieve tokens - An example is the `claimToken()` safety function in the latest version of MinimeToken (example)

Fixed - https://github.com/poanetwork/poa-popa/pull/130

Conclusion

Overall the code is well written and we have not identified any potential vulnerabilities. These contracts have a low level risk of being tampering with data from the inspected contracts. There is high test coverage which should increase confidence in the security of these contracts, and their maintainability in the future.

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 practices 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.