OJA Token Smart Contract Audit

Author: Blockalize
September, 2021

Introduction

The Ojamu project intends to deploy the `OJA` token contract in October 2021. This report analyzes the smart contracts provided by the team.

The smart contract OjamuToken.sol (Token: OJA) is an ERC20 compatible token used by the Ojamu Ecosystem.

This report is a description of the audit performed by Blockalize in September, 2021. Blockalize is a company specialized in smart contract development and audits.

Token name: Ojamu Contract
Token Symbol: OJA

Solidity version: 0.8

Executive summary

No major potential vulnerabilities have been identified in the token contract.

The token is implementing the ERC20 standard and has the following additional functionality:

- Create snapshots: An administrator can create new snapshots. When a snapshot is created the token holdings at that point in time are stored for later use. This can be used when implementing governance. For example one use case is to protect against flashloan attacks during voting.
- Burn tokens: An administrator can burn their own tokens. This will decrease the total supply and the tokens will be removed out of circulation. The Administrator must already own the tokens to be able to burn them.

The token has a maximum total supply of 100 000 000 OJA tokens.

The token is based on a known open standard - the OpenZeppelin contracts, that are a set of contracts intended to be a safe building block for a variety of uses (reference: https://github.com/OpenZeppelin/openzeppelin-contracts). Overall the codebase is of high quality. It is clean, modular and follows best practices. The version 4.2.0 of Open zeppelin is used by the token.

There were no severe anomalies found in the review.

Verification of Smart contract files

Code files reviewed

'OjamuToken.sol'

- The main code file of the Ojamu token encapsulating the properties of the token, and extensions

'openzeppelin-solidity/contracts/token/ERC20/extensions/ERC20Snapshot.sol'

- Includes both the ERC20 standard contract functionality, and the functionality for creating snapshots wrapped around it.

‘openzeppelin-solidity/contracts/access/AccessControl.sol’
- Access control contract provides role based access control functionality

Scope of audit

- This audit is reviewing that the smart contract is executed as expected according to the given requirements, and that the coded algorithms work as expected. It does not guarantee that the code is bugfree, but intends to highlight any areas of weaknesses.

- This audit makes no statements or warranties about the viability of the tokens business proposition, the individuals involved in this business or the regulatory regime for the business model.

Potential Vulnerabilities

- No potential vulnerabilities found

Recommendations

- Literals with many digits are difficult to read and review. The total supply is hard coded using literals with many digits. It is recommended that if the intention is to hardcode ether values the ether unit is used instead of the full wei value. It is easier to see the value in the form of 100 ether than in the form of 100000000000000000000.

- Consider declaring external functions as external for the following functions.
  - burn
  - snapshot

Code review

- The code has been reviewed manually line by line to ensure the code is working as intended.

- The code has been tested using automated test scripts using truffle.

- In addition, automated smart contract analysis tools have been used to analyze the smart contracts for vulnerabilities.

- The tests have resulted in a list of warnings that can be reviewed by the team.

References

ERC-20 Token Standard:

https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md

Detailed result from scanning the contract using automatic tools

No errors found, only warnings with low severity found

Warning: Literals with many digits are difficult to read and review.

- Found in: OjamuToken.sol#10-48 uses literals with too many digits: initialSupply = 100000000 * (10 ** 18)
- Reference:https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Conclusion:

- Non required suggestion regarding the code-style

Warning:

- Functions can be declared external:
  - grantRole(bytes32,address)
  - revokeRole(bytes32,address
  - renounceRole(bytes32,address)
  - ERC20.name()
  - ERC20.symbol()
  - ERC20.decimals()
  - ERC20.transfer(address,uint256)
  - ERC20.allowance(address,address)
  - ERC20.approve(address,uint256)
  - ERC20.transferFrom(address,address,uint256)
  - ERC20.increaseAllowance(address,uint256)
  - ERC20.decreaseAllowance(address,uint256)
  - ERC20Snapshot.setCompleted(uint256)
  - ERC20Snapshot.balanceOfAt(address,uint256)
  - ERC20Snapshot.totalSupplyAt(uint256)
  - ERC20Snapshot.totalSupplyAt(uint256)
  - burn(uint256)
  - snapshot()
- Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

Conclusion:
- Non required suggestion

Tool scan analyzed 16 contracts with 75 detectors, and 39 result(s) found