SOLIDIFIED

Audit Report for CultosCoin - May 26, 2022

Summary

Audit Report prepared by Solidified covering the CultosCoin smart contracts.

Process and Delivery

Independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on May 9, 2022, and the results are presented here.

Audited Files

The source code has been supplied in a private source code repository:

https://github.com/Cultos-App/token

Commit number: `28c0a18571f2804ff42ed76fc0ec8749341dfc00`

The contract has been deployed on the Ethereum main network at address `0x7fB73882E55931631875E94d9F3C3Cb6aec6414c`

Intended Behavior

The audited codebase implements a standard ERC-20 token contract.

Findings

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than a security audit and vice versa.

Criteria	Status	Comment
Code complexity	Low	-
Code readability and clarity	High	-
Level of Documentation	N/A	-
Test Coverage	N/A	-

Issues Found

Solidified found that the CultosCoin contract contains no critical issues, no major issues, no minor issues, and 2 informational notes.

We recommend issues are amended, while informational notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	Wide version pragma allows for versions with know bugs	Note	-
2	Minting of total supply is performed in a non-standard way	Note	-

Critical Issues

No critical issues have been found.

Major Issues

No major issues have been found.

Minor Issues

No minor issues have been found.

Informational Notes

1. Wide version pragma allows for versions with know bugs

The contract's version pragma allows versions greater than or equal to `0.8.0`. However, there have been several critical bug fixes since version `0.8.0`.

Recommendation
We recommend using a Solidity version greater than or equal to `0.8.4` or fixing the compiler pragma to the latest version supported by common smart contract frameworks, such as Hardhat.

4. Minting of total supply is performed in a non-standard way

The codebase uses OpenZeppelin's ERC-20 implementation. However, the minting of the token supply during contract deployment does not follow the common pattern outlined in the OpenZeppelin documentation (https://docs.openzeppelin.com/contracts/4.x/erc20-supply):

- The amount to be minted is calculated by multiplying the `supply_` parameter by the number of decimal places used by the token. However, it is recommended best practice

to not use the `decimals` value internally and leave any decimal conversions to off-chain code. The documentation states: “This information is only used for display purposes: it in no way affects any of the arithmetic of the contract, including IERC20.balanceOf and IERC20.Transfer.”

- The minting is performed directly by the `_mint` function. Commonly, this function is called from the constructor with the amount to be minted supplied as a parameter.

Recommendation

We recommend only using base units within the smart contract and avoiding multiplying the external supply parameter. This calculation can be performed by the off-chain deployment code. In addition, we recommend using the internal `_mint` function with the amount to be minted as a parameter from the constructor.

Audit Report for CultosCoin - May 26, 2022

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of CultosCoin or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Oak Security GmbH