SOLIDIFIED

Audit Report for Cybercomm - July 17, 2021

Summary

Audit Report prepared by Solidified covering the AluCoin smart contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code. The debrief was on 28 June 2021.

Audited Files

The source code has been supplied in the form of a GitHub repository:

https://github.com/alucoin-token/smart-contracts/

Commit number: `a2b46ff53c174bfff6838203a53896a6e368590c`

The scope of the audit was limited to the following files:

```
contracts
├── BeforeTokenTransferHook.sol
├── BlockedUserRegistry.sol
├── BurnerRegistry.sol
├── Migrations.sol
├── Registry.sol
└── Token.sol
```

Intended Behavior

The smart contracts implement an ERC-20 token with a blacklist functionality.

Code Complexity and Test Coverage

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 equate to a higher risk. Certain bugs are more easily detected in unit testing than a security audit and vice versa. It is, therefore, more likely that undetected issues remain if the test coverage is low or non-existent.

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

SOLIDIFIED

Audit Report for Cybercomm - July 17, 2021

Issues Found

Solidified found that the AluCoin smart contract contained no critical, major or minor issues and 1 warning, in addition to 4 informational notes.

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

Issue #	Description	Severity	Status
1	BeforeTokenTransferHook.sol: Settings controller can mint tokens and cause DoS	Warning	Acknowledged
2	Token.sol: Hard-coded number of decimals does not match documentation / comments	Note	Resolved
3	Pragma allows for a wide range of compiler versions	Note	Resolved
4	BeforeTokenTransfer.sol Blocked addresses can still receive tokens	Note	Resolved
5	Code cleanup	Note	Resolved

SOLIDIFIED

Audit Report for Cybercomm - July 17, 2021

Critical Issues

No critical issues have been found.

Major Issues

No major issues have been found.

Minor Issues

No minor issues have been found.

Warnings

1. BeforeTokenTransferHook.sol: Settings controller can mint tokens and cause DoS

The address `_settingsController` has a lot of power. It can set the minter (thus increasing token supply) and it can halt the entire `transfer()` functionality by updating the `_blockedUsers` contract to a contract where `_blockedUsers.get(from)` would always return true.

Recommendation
Consider highlighting this warning as part of the documentation to educate users.

Team Reply
"*This is in line with our process and it's security will be backed up by a multi-sig authorization of all related transactions.*"

Informational Notes

2. Token.sol: Hard-coded number of decimals does not match documentation / comments

The constructor sets up the token contract to have 7 decimals. However, the comments suggest a value of 3.

Recommendation
Resolve inconsistent commenting.

3. Pragma allows for a wide range of compiler versions

The `pragma` statement allows for a wide range of compiler versions, including some versions with known bugs. In addition, the language syntax has changed since the earlier versions that are allowed.

Recommendation
Consider limiting the compiler to at least a single major version number.

4. BeforeTokenTransfer.sol Blocked addresses can still receive tokens

The function `execute()` checks only the from address whether it's blacklisted. This opens up a scenario where the target address can receive funds even though it's blacklisted.

Recommendation
Consider validating the target address as well if this is not intended.

5. Code cleanup

Consider addressing the following notes.

1. `Token.sol`: The transfer method is already implemented as part of OpenZeppelin ERC20 contract. There is no need to override the method in `Token.sol` if there are no changes.
2. `Token.sol`: No need to use `super` to access the methods in super class since the function names are different.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of AluCoin 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.

Solidified Technologies Inc.