C-BLOCK

Rock'n'Block

SMART CONTRACTS EXPERTISE —
RIGHT WAY TO SUCCESS
C-BLOCK audit report

Content

1. Overview

C-BLOCK team asked us to perform a review of their contracts. We performed a review of their code from and published this document as a write-up of our findings.

1.1 Terms of Reference for the creation of a smart contract

1.1.1 Contract details:

- Token factory contract provide possibility to create following types of contract: freezable, nonfreezable, burnable, nonburnable, mintable, nonmintable;
- Crowdsale factory contract provide possibility to create following types of contract: NonSoftCappable, SoftCappable (NSC и SC) Bonusable and DatesChangeable;
- Will contract - this contract provides the ability to create contracts for inheritance user funds, tokens and native blockchain coins.
- Lost key for tokens and lost key for native coins - this contract provides the ability to create contracts for backup user funds, tokens and native blockchain coins.
- Wedding - this contract provides the ability to create contracts to manage tokens and native blockchain coins for few users.

1.1.2 Token contract factory:

deployERC20...Token	A function that accepts a number of parameters to later deploy the contract
getToken	A function that allows you to withdraw any tokens from a smart contract
renounceOwnership	The contract allows the owner to renounce ownership. After calling renounceOwnership no one can be owner of the contract
getToken	A function that allows you to specify the current contract price using this factory
transferOwnership	The contract allows the owner to transfer ownership to another address

1.1.3 TCrowdsale contract factory:

deploy...Crowdsale	A function that accepts a number of parameters to later deploy the contract
getToken	A function that allows you to withdraw any tokens from a smart contract
renounceOwnership	The contract allows the owner to renounce ownership. After calling renounceOwnership no one can be owner of the contract
setPrice	A function that allows you to specify the current contract price using this factory
transferOwnership	The contract allows the owner to transfer ownership to another address

1.1.3 Lostkey contract factory:

deployLostKey	A function that accepts a number of parameters to later deploy the contract
getToken	A function that allows you to withdraw any tokens from a smart contract
renounceOwnership	The contract allows the owner to renounce ownership. After calling renounceOwnership no one can be owner of the contract
setPrice	A function that allows you to specify the current contract price using this factory
transferOwnership	The contract allows the owner to transfer ownership to another address

1.1.4 Will contract factory:

deploy...Crowdsale	A function that accepts a number of parameters to later deploy the contract
getToken	A function that allows you to withdraw any tokens from a smart contract
renounceOwnership	The contract allows the owner to renounce ownership. After calling renounceOwnership no one can be owner of the contract
setPrice	A function that allows you to specify the current contract price using this factory
transferOwnership	The contract allows the owner to transfer ownership to another address

1.1.5 Will contract factory:

deployLostKey	A function that accepts a number of parameters to later deploy the contract
getToken	A function that allows you to withdraw any tokens from a smart contract
renounceOwnership	The contract allows the owner to renounce ownership. After calling renounceOwnership no one can be owner of the contract
setPrice	A function that allows you to specify the current contract price using this factory
transferOwnership	The contract allows the owner to transfer ownership to another address

1.1.6 Wedding contract factory:

deploy...Crowdsale	A function that accepts a number of parameters to later deploy the contract
getToken	A function that allows you to withdraw any tokens from a smart contract
renounceOwnership	The contract allows the owner to renounce ownership. After calling renounceOwnership no one can be owner of the contract
setPrice	A function that allows you to specify the current contract price using this factory
transferOwnership	The contract allows the owner to transfer ownership to another address

2. Introduction

2.1. Authenticity

The contracts audited are a subset of the contracts compiled and deployed in testnet Celo blockchain.

Factory of token contracts:

Non-Mintable Non-Freezable -
Contract 0x568EE75009950B15e9e91a9A99DedF749f3AcBBf - Alfajores Explorer

Non-Mintable Freezable -
Contract 0x9D61A75467BF17ea3947cc52fCdF5285e8A202f3 - Alfajores Explorer

Mintable Non-Freezable -
Contract 0x9B8797085E0c916E25a860Ad3015F6A8a5ff5f37 - Alfajores Explorer

Mintable Freezable -
Contract 0x5A3A9c31151A5A125F6baBaDc1e997017cAC1eeC - Alfajores Explorer

Factory of crowdsale contracts:

Non-SoftCappable Non-Bonusable - FactoryNSCNBCrowdsale
(0x8129A40EA8fA34C342b04BE1a9Ba379148F99D7F) - Alfajores Explorer

NonSoftCappable Bonusable - FactoryNSCBCrowdsale
(0x4f96423a3aB01F821c98E9a6D72ca6fB6c9ED49D) - Alfajores Explorer

SoftCappable NonBonusable - FactorySCNBCrowdsale
(0x9d1D9c4E4F622708210Ce4C7de7b76d6fC087733) - Alfajores Explorer

SoftCappable Bonusable - FactorySCBCrowdsale
(0x7C5d7986259354a80bB83c101754587Bfc3bBCAc) - Alfajores Explorer

Factory of Lost key:

Contract
0xd0fF8b5a7723752309ab2222A40b0485aA53C558 - Alfajores

Factory of Last will contract:

Contract
0x2b1cd0ADFec6Ed778238Cc31315a7b96E877f48a - Alfajores

Factory of wedding contracts:

Contract
0x1122F13B0666Ec7146Bd77f47040c94450ccbACf - Alfajores

2.2. Scope

The audit reviewed contract source code from alfajores-blockscout.celo-testnet.org. Contracts were reviewed in the context of the flattened file, which included solidity files. The review performed did not assess any scripts, tests, or other non-Solidity files.

2.3. Methodology

This audit was performed as a comprehensive review of the codebase and takes into consideration both the Solidity code, as well as the target platform: Celo network. The Solidity was reviewed not just for common vulnerabilities and antipatterns, but also for its parity with the intent of the deployer, for its efficiency, and for the practices used during development.

2.4. Description of the complex of procedures for reviewing the smart contract

2.4.1 Primary architecture review

- Checking the architecture of the contract.
- The correctness of the code.
- Check for linearity, shortness, and self-documentation.
- Static verification and code analysis for validity and the presence of syntactic errors.

2.4.2 Comparison of requirements and implementation

- Checking the code of the smart contract for compliance with the requirements of the customer code logic, writing algorithms, matching the initial constant values.
- Identification of potential vulnerabilities

2.5. Risk Assessment

Findings were categorized using a risk rating model based on the OWASP method. Each vulnerability takes into consideration the impact and likelihood of exploitation, as well as the relative ease with which the vulnerability is resolved; findings that permeate throughout the codebase will require much more review and work to solve and are rated higher as a result.

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology:

• Likelihood represents how likely a particular vulnerability is to be uncovered and exploited in the wild;

• Impact measures the technical loss and business damage of a successful attack;

• Severity demonstrates the overall criticality of the risk;

Likelihood and impact are categorized into three ratings: H, M and L, i.e., high, medium and low respectively. Severity is determined by likelihood and impact and can be classified into four categories accordingly, i.e., Critical, High, Medium, Low shown in following Table

Table: Vulnerability Severity Classification

impact			
High	**Critical**	**High**	**Medium**
Medium	**High**	**Medium**	**Low**
Low	**Medium**	**Low**	**Low**
	High	*Medium*	*Low*

Likelihood

2.6. Disclaimer

This document reflects the understanding of security flaws and vulnerabilities as they are known to Rock`n`Block, and as they relate to the reviewed project. This document makes no statements on the viability of the project or the safety of its code. This audit does not represent investment advice and should not be interpreted as such.

3. Findings

3.1. Critical Severity

No critical-severity vulnerabilities were found.

3.2. High Severity

No high-severity vulnerabilities were found.

3.3. Medium Severity

No medium-severity vulnerabilities were found.

3.4. Low Severity

3.4.1. File(s): All factory crowdsale files (contracts/crowdsale/Factory*), LostKey.sol, Wedding.sol

Lines: -
Issue description:

Crowdsale contracts: Method addPaymentMethod()has external calls inside a loop, calling crowdsale.addPaymentMethod;

LostKey contract: Method distribute() has external calls inside a loop, calling token transfer;

Wedding contract: Method getFundsAfterDivorce() has external calls inside a loop, calling token transfer;

Recommendations: Such logic is essential to add multiple payment methods in one transaction, but it is necessary to say that calling an external contract in a loop might lead to denial-of-service attacks if the called contract is malfunctioning.

3.4.2. Files(s): All soft-cappable crowdsale files (contracts/crowdsale/SCCrowdsale/*)

Lines: -
Issue description: Method refund() has costly operations inside a loop

Recommendations: Costly operations inside a loop might waste gas, so optimizations are justified.

Informational issues

3.4.3. File(s): LostKey.sol
Lines: 90-93

Issue description: Reentrancy in method terminateContract. State variable terminated written after the token transfer call.

Recommendations: Change code lines in order to fix reentrancy, or apply a nonReentrant modifier.

Code style issues:

3.4.4. File(s): CrowdsaleBase.sol
Lines: 34-37

Issue description: Method setLimits() should emit an event for setting upper and lower limits.

Recommendations: Add event for this method.

3.4.5. File(s): DatesChangeable.sol
Lines: 8-10

Issue description: Method endCrowdsale() should emit an event for crowdsale ending

Recommendations: Add event for this method.

3.4.6. File(s): Wedding.sol
Lines: 44-47

Issue description: Modifier divorceNotProposed uses a dangerous strict equality (divorceTimestamp == 0).

Recommendations: When using strict equality, pay additional attention for access to such variables in other contract methods, because such checks can be easily manipulated by an attacker.

4. Documents and Resources

4.1. References

OWASP. Risk Rating Methodology. https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

5. Conclusion

The information in this review is a list of recommendations on what needs to be done to ensure the quality and security of the smart contract. The Rock`n`block experts conducted the verification of the smart contract. Based on the results of the reviewing and testing, it is established that the token smart contract complies with the specifications specified in the terms of reference.

During the reviewing and testing of the contracts, critical errors and possible vulnerabilities were not detected. Outside of the included notes, the code reviewed was simple and clean. The formatting, naming, and other conventions used were fairly regular, and the inheritance structure was well-organized, resulting in a codebase that was easier to review.

For all questions regarding the review and testing of the smart contract, we recommend contacting audit@rocknblock.io