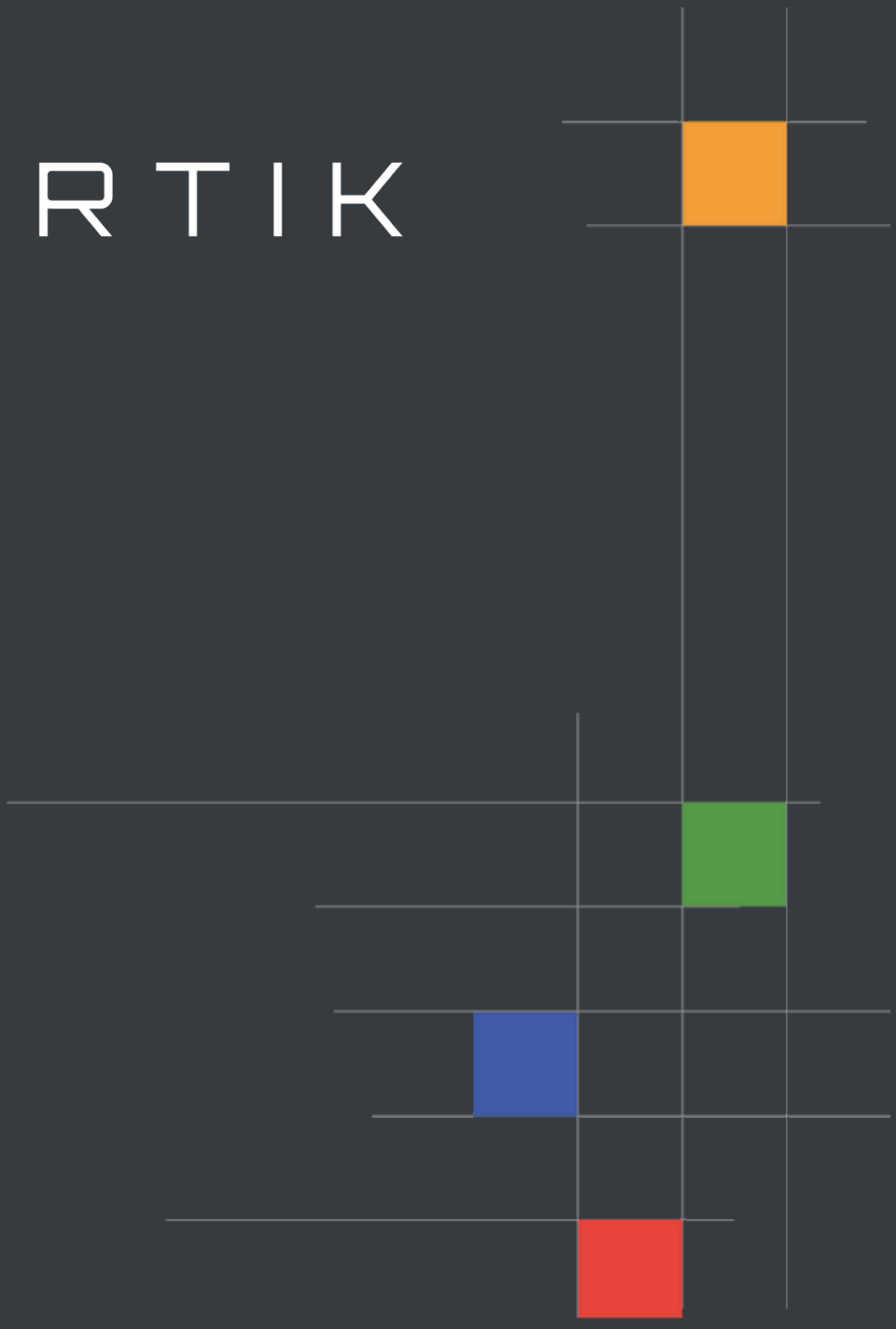
Furucombo

rCOMBO

Security Assessment

March 20th, 2021

Audited By:
Alex Papageorgiou @ CertiK
alex.papageorgiou@certik.org
Reviewed By:
Camden Smallwood @ CertiK
camden.smallwood@certik.org



Disclaimer

CertiK reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	Furucombo - rCOMBO
Description	A gradual-release ERC20 token.
Platform	Ethereum; Solidity, Yul
Codebase	dinngodev/RCOMBO
Commits	6d3d04f8a3a833ff60edab274202de2c88659ca0

Audit Summary

Delivery Date	March 20th, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	1
Timeline	March 18th, 2021 - March 20th, 2021

Vulnerability Summary

Total Issues	3
Total Critical	0
Total Major	0
Total Medium	0
Total Minor	2
Total Informational	1

Executive Summary

We were tasked with auditing the codebase of two deployed contracts as well as a contract repository of Furucombo encompassing their COMBO token, rCOMBO token meant to represent an IOU and finally a token vesting contract.

We were not able to pinpoint any severe vulnerabilities to the system, however, we did detect certain points where better security practices can be applied as well as a single point where the design can be optimized better towards the ideals of the project.

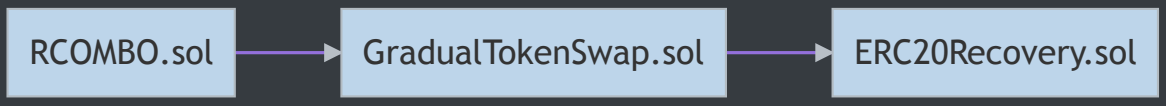
All outward and inward transfers of the system conform to the Checks-Effects-Interactions pattern and no common vulnerabilities such as re-entrancies were identified.

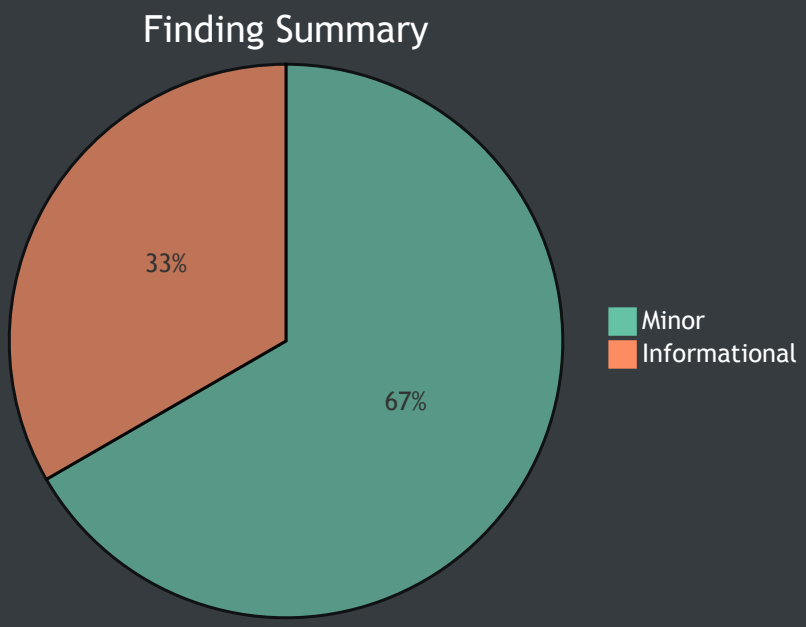
The `rCOMBO` token mints its total supply directly to its deployer and the gradual release program contains a function whereby the owner is able to rescue ERC20 tokens at will, including the gradually released as well as locked tokens.

Files In Scope

ID	Contract	Location
ERC	ERC20Recovery.sol	ERC20Recovery.sol
GTS	GradualTokenSwap.sol	GradualTokenSwap.sol
RCO	RCOMBO.sol	RCOMBO.sol

File Dependency Graph





Manual Review Findings

ID	Title	Type	Severity	Resolved
ERC-01M	Potentially Ill-Perceived Functionality	Logical Fault	Minor	
RCO-01M	Redundant Getter Invocation	Gas Optimization	Informational	

Static Analysis Findings

ID	Title	Type	Severity	Resolved
GTS-01S	Inexistent Address Sanitization	Logical Fault	Minor	

ERC-01M: Potentially Ill-Perceived Functionality

Type	Severity	Location
Logical Issue	Minor	ERC20Recovery.sol L9-L11

Description:

The `ERC20Recovery` contract is meant to allow outward transfers towards its `owner` of potentially locked funds, however, this contract is inherited from the `GradualTokenSwap` contract which is meant to hold on tokens for a time period which should not be retrievable by the `owner` .

Recommendation:

We advise this functionality to be revised in a more decentralized manner, potentially by ensuring the function can be invoked beyond the "staking" period.

Alleviation:

The Furucombo team has stated that there may be instances where the team decides to pause or stop the redemption process such as when lost funds are being recovered.

RCO-01M: Redundant Getter Invocation

Type	Severity	Location
Gas Optimization	Informational	RCOMBO.sol L20

Description:

The `constructor` of the `rCOMBO` token utilizes the `decimals` getter variable redundantly so as the `decimals` is equal to `18` when not manually set within the OpenZeppelin library.

Recommendation:

We advise it to be removed from the codebase and swapped by the `18` value literal.

Alleviation:

The Furucombo team has stated that this finding doesn't affect the functionality of the contract and as such, will not be updated to the codebase of `rCOMBO`.

GTS-01S: Inexistent Address Sanitization

Type	Severity	Location
Logical Issue	Minor	GradualTokenSwap.sol L34, L35

Description:

The `constructor` of the contract does not sanitize its two input address arguments representing the rCOMBO and COMBO tokens.

Recommendation:

We advise that the appropriate `require` checks are imposed at this point.

Alleviation:

The Furucombo team responded by stating that the two addresses passed to the `GradualTokenSwap` contract are hardcoded in the `rCOMBO` token and as such do not warrant an additional `require` check.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.