Furucombo

rCOMBO

SMART CONTRACT AUDIT

07.04.2021

Made in Germany by Chainsulting.de

Table of contents

1. Disclaimer

The audit makes no statements or warrantees about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Furucombo by DINNGO Pte. Ltd. . If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (29.03.2021)	Layout
0.2 (30.03.2021)	Test Deployment
0.5 (31.03.2021)	Automated Security Testing Manual Security Testing
0.7 (31.03.2021)	Test Deployment & Verify Claims
0.9 (01.04.2021)	Summary and Recommendation
1.0 (01.04.2021)	Final document
1.2 (07.04.2021)	Added and verified contract address

2. About the Project and Company

Company address:

DINNGO Pte. Ltd.
100 Tras Street #16-01
Singapore 079027

Website: https://furucombo.app

Twitter: https://twitter.com/furucombo

Medium: https://medium.com/furucombo

Telegram: https://t.me/furucombo

YouTube: https://www.youtube.com/channel/UCa1kGD4lvTSrmfKbDjQNOxQ

Discord: https://discord.furucombo.app

2.1 Project Overview

Furucombo is a tool built for end-users to optimize their DeFi strategy simply by drag and drop. It visualizes complex DeFi protocols into cubes. Users setup inputs/outputs and the order of the cubes, then Furucombo bundles all the cubes into one transaction and sends out. Furucombo calls this building-blocks setup a “combo”.

The allocation of COMBO ensures stable development and maintenance of Furucombo while allowing the community to lead Furucombo’s governance. The launch of COMBO Token represents a tremendous milestone for Furucombo and DeFi. It demonstrates the maturity of our product and our determination of becoming a super aggregator. Furucombo believes that a community-driven product would open up a world of infinite possibilities and they’re absolutely excited to explore the next chapter of Furucombo.

Furucombo RCOMBO is an Open source implementation based on Hegic GradualTokenSwap.

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

4.2 Used Code from other Frameworks/Smart Contracts (direct imports)

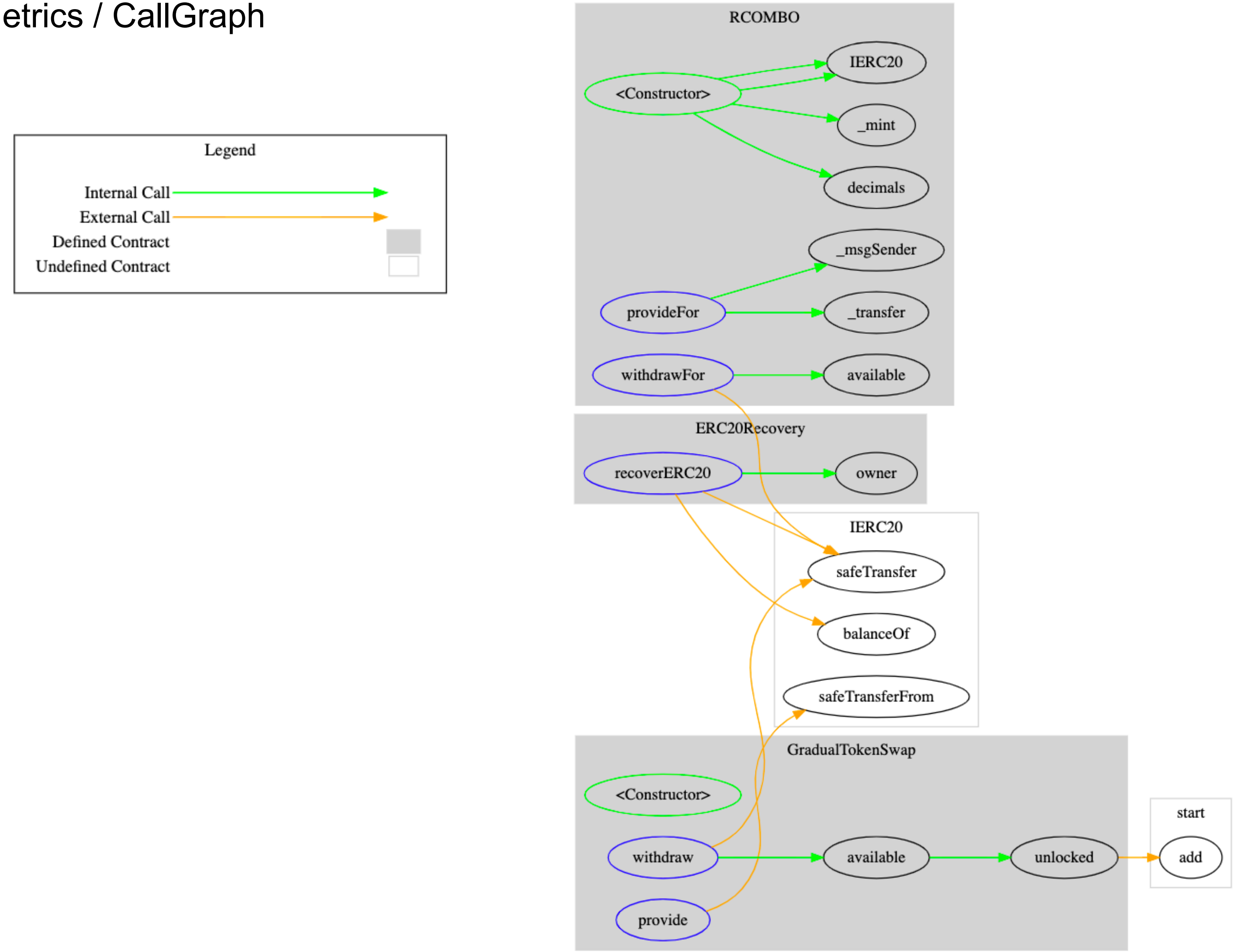
1. SafeMath.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.1/contracts/math/SafeMath.sol
2. IERC20.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.1/contracts/token/ERC20/IERC20.sol
3. SafeERC20.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.1/contracts/token/ERC20/SafeERC20.sol
4. Ownable.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.1/contracts/access/Ownable.sol
5. Context.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.1/contracts/GSN/Context.sol
6. ERC20.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.1/contracts/token/ERC20/ERC20.sol

4.3 Tested Contract Files

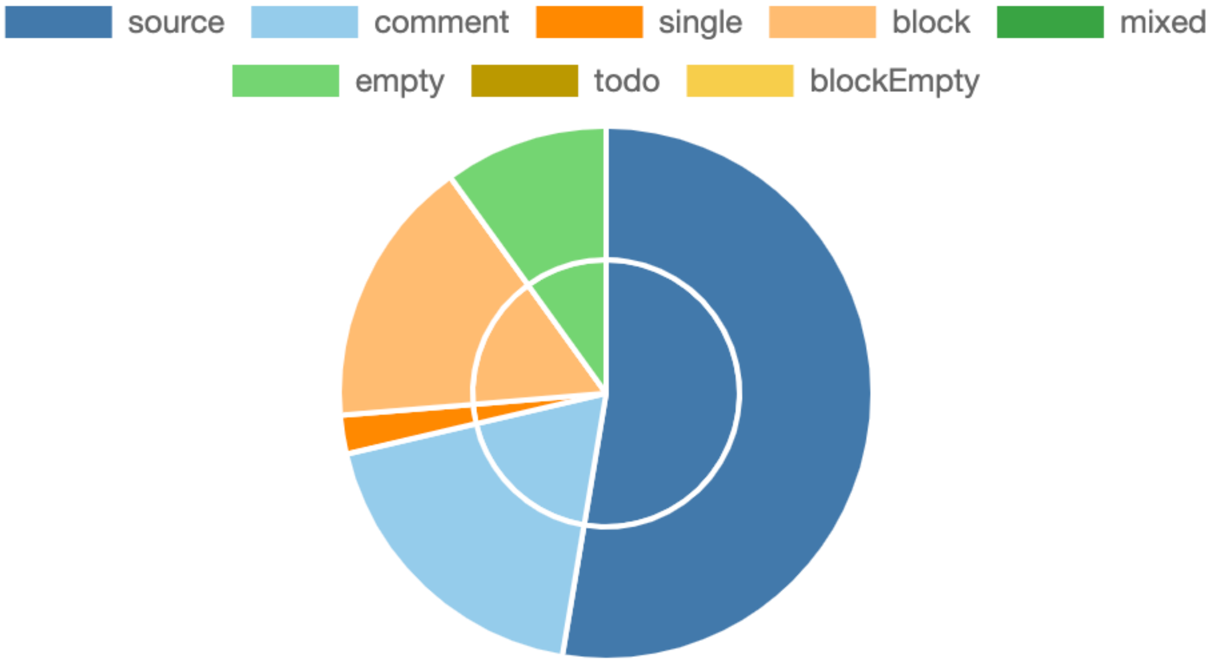
The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5)
GradualTokenSwap.sol	002c209c8997446abc07ae8e7d7d8412
RCOMBO.sol	a6db71cd63fec3cbd5b53a638eaa1fbc

4.4 Metrics / CallGraph



4.5 Metrics / Source Lines



4.6 Metrics / Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts

`0.7.6`
`>=0.4.25 <0.9.0`
			****
			(0 asm blocks)	

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2

🌐Public	💰Payable
7	0

External	Internal	Private	Pure	View
5	7	0	0	2

StateVariables

Total	🌐Public
6	6

4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/RCOMBO.sol	1		43	43	31	7	26	
🎨	contracts/hegic/GradualTokenSwap/contracts/ERC20Recovery.sol	1		12	12	9	1	10	
📝	contracts/hegic/GradualTokenSwap/contracts/GradualTokenSwap.sol	1		84	84	50	24	35	
📝 🎨	**Totals**	**3**		**139**	**139**	**90**	**32**	**71**	

Legend: [➖]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

5. Scope of Work

The Furucombo Team provided us with the files that needs to be tested. The scope of the audit is the RCOMBO contract.

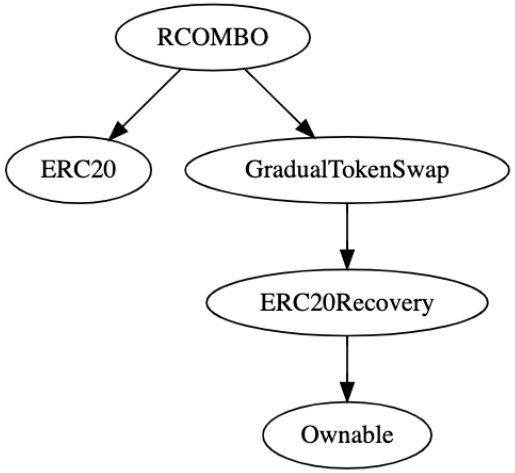
Following contracts with the direct imports been tested

- RCOMBO.sol

The team put forward the following assumptions regarding the security, usage of the contracts:

- Check overall smart contract security

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.



5.1 Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, Chainsulting‘s experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, Chainsulting’s experts found **no High issues** in the code of the smart contract.

MEDIUM ISSUES

During the audit, Chainsulting’s experts found **no Medium issues** in the code of the smart contract.

LOW ISSUES

5.1.1 SPDX license identifier
Severity: LOW
Status: ADDRESSED
File(s) affected: RCOMBO.sol

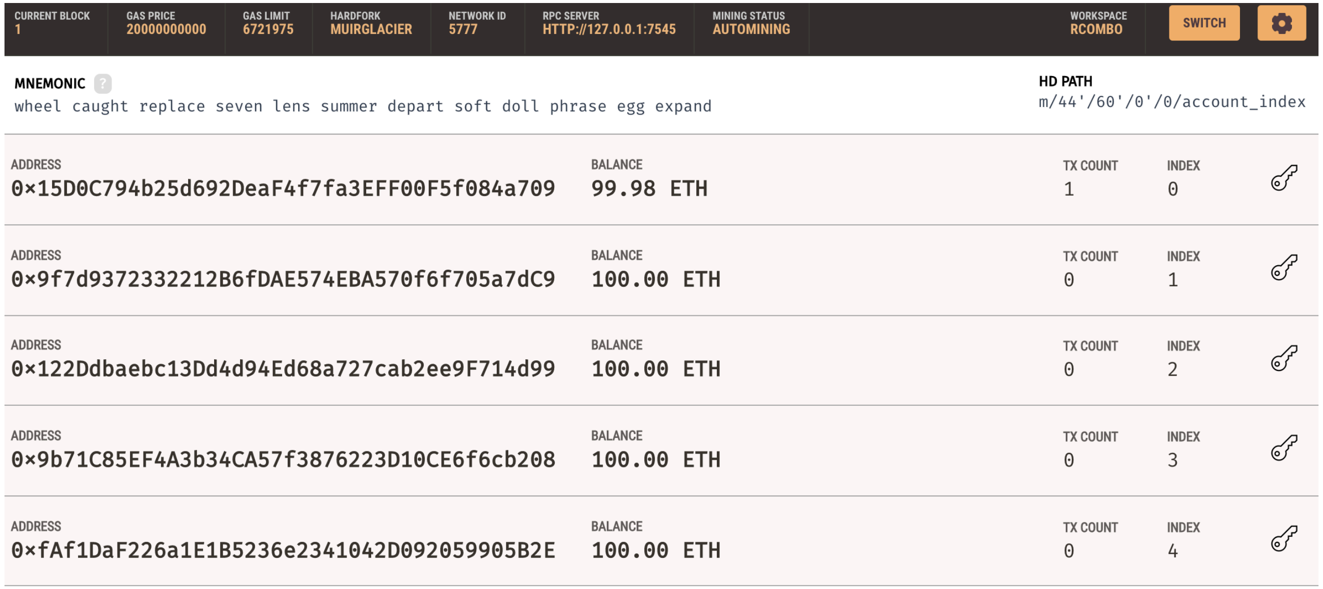
Attack / Description	Code Snippet	Result/Recommendation
Warning: SPDX license identifier not provided in source file.	Line NA	Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.

INFORMATIONAL ISSUES

During the audit, Chainsulting’s experts found **no Informational issues** in the code of the smart contract.

6. Test Deployment & Verify Claims

Environment: Ganache / localhost



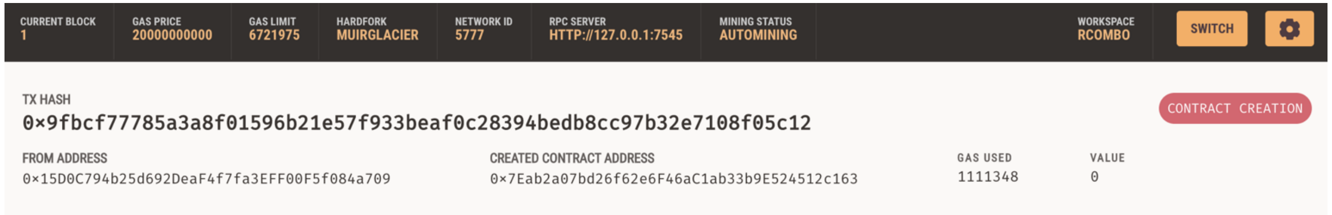
CURRENT BLOCK 1 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MUIRGLACIER | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE RCOMBO

MNEMONIC
wheel caught replace seven lens summer depart soft doll phrase egg expand

HD PATH
m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×15D0C794b25d692DeaF4f7fa3EFF00F5f084a709	99.98 ETH	1	0
0×9f7d9372332212B6fDAE574EBA570f6f705a7dC9	100.00 ETH	0	1
0×122Ddbaebc13Dd4d94Ed68a727cab2ee9F714d99	100.00 ETH	0	2
0×9b71C85EF4A3b34CA57f3876223D10CE6f6cb208	100.00 ETH	0	3
0×fAf1DaF226a1E1B5236e2341042D092059905B2E	100.00 ETH	0	4

6.1.1 Deployment of COMBO Token



CURRENT BLOCK 1 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MUIRGLACIER | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE RCOMBO

TX HASH
0×9fbcf77785a3a8f01596b21e57f933beaf0c28394bedb8cc97b32e7108f05c12

CONTRACT CREATION

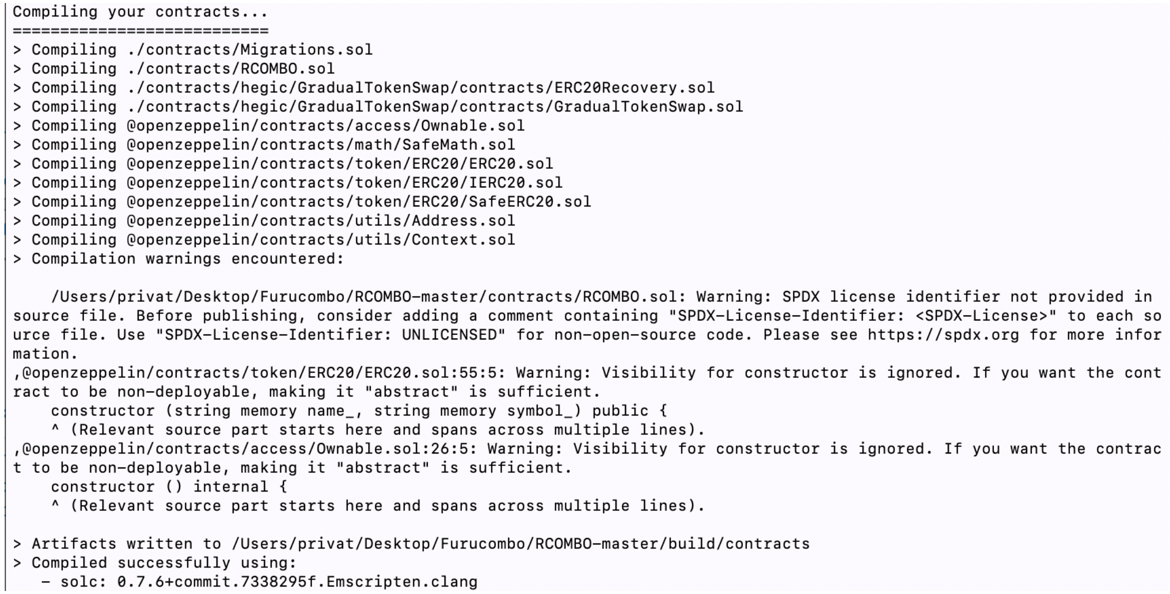
FROM ADDRESS
0×15D0C794b25d692DeaF4f7fa3EFF00F5f084a709

CREATED CONTRACT ADDRESS
0×7Eab2a07bd26f62e6F46aC1ab33b9E524512c163

GAS USED
1111348

VALUE
0

6.1.2 Compiling

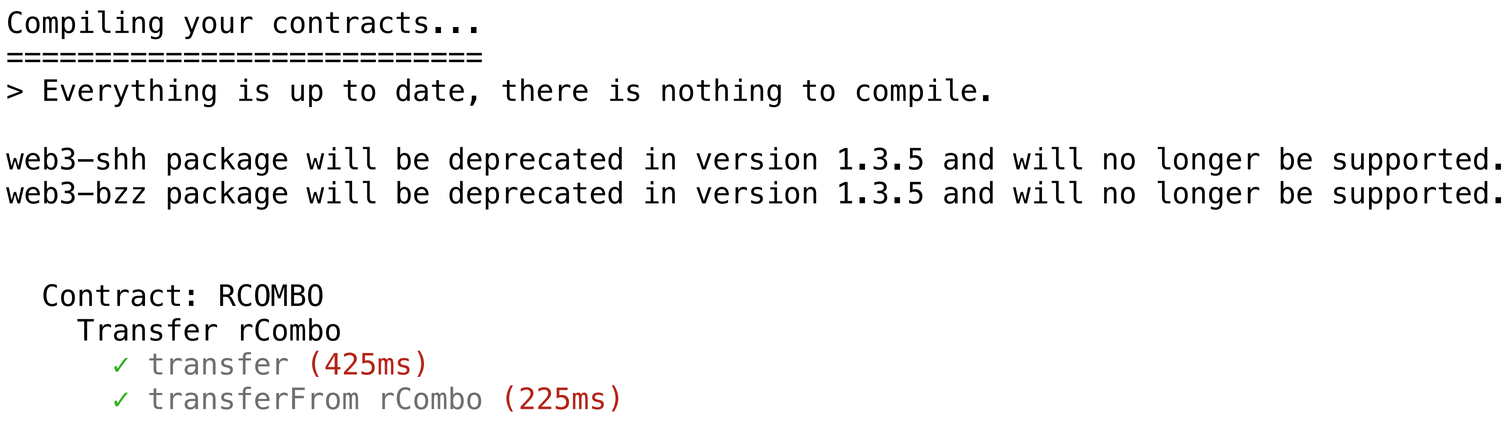


```
Compiling your contracts...
===========================
> Compiling ./contracts/Migrations.sol
> Compiling ./contracts/RCOMBO.sol
> Compiling ./contracts/hegic/GradualTokenSwap/contracts/ERC20Recovery.sol
> Compiling ./contracts/hegic/GradualTokenSwap/contracts/GradualTokenSwap.sol
> Compiling @openzeppelin/contracts/access/Ownable.sol
> Compiling @openzeppelin/contracts/math/SafeMath.sol
> Compiling @openzeppelin/contracts/token/ERC20/ERC20.sol
> Compiling @openzeppelin/contracts/token/ERC20/IERC20.sol
> Compiling @openzeppelin/contracts/token/ERC20/SafeERC20.sol
> Compiling @openzeppelin/contracts/utils/Address.sol
> Compiling @openzeppelin/contracts/utils/Context.sol
> Compilation warnings encountered:

    /Users/privat/Desktop/Furucombo/RCOMBO-master/contracts/RCOMBO.sol: Warning: SPDX license identifier not provided in
source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each so
urce file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more infor
mation.
,@openzeppelin/contracts/token/ERC20/ERC20.sol:55:5: Warning: Visibility for constructor is ignored. If you want the cont
ract to be non-deployable, making it "abstract" is sufficient.
    constructor (string memory name_, string memory symbol_) public {
    ^ (Relevant source part starts here and spans across multiple lines).
,@openzeppelin/contracts/access/Ownable.sol:26:5: Warning: Visibility for constructor is ignored. If you want the contrac
t to be non-deployable, making it "abstract" is sufficient.
    constructor () internal {
    ^ (Relevant source part starts here and spans across multiple lines).

> Artifacts written to /Users/privat/Desktop/Furucombo/RCOMBO-master/build/contracts
> Compiled successfully using:
   - solc: 0.7.6+commit.7338295f.Emscripten.clang
```

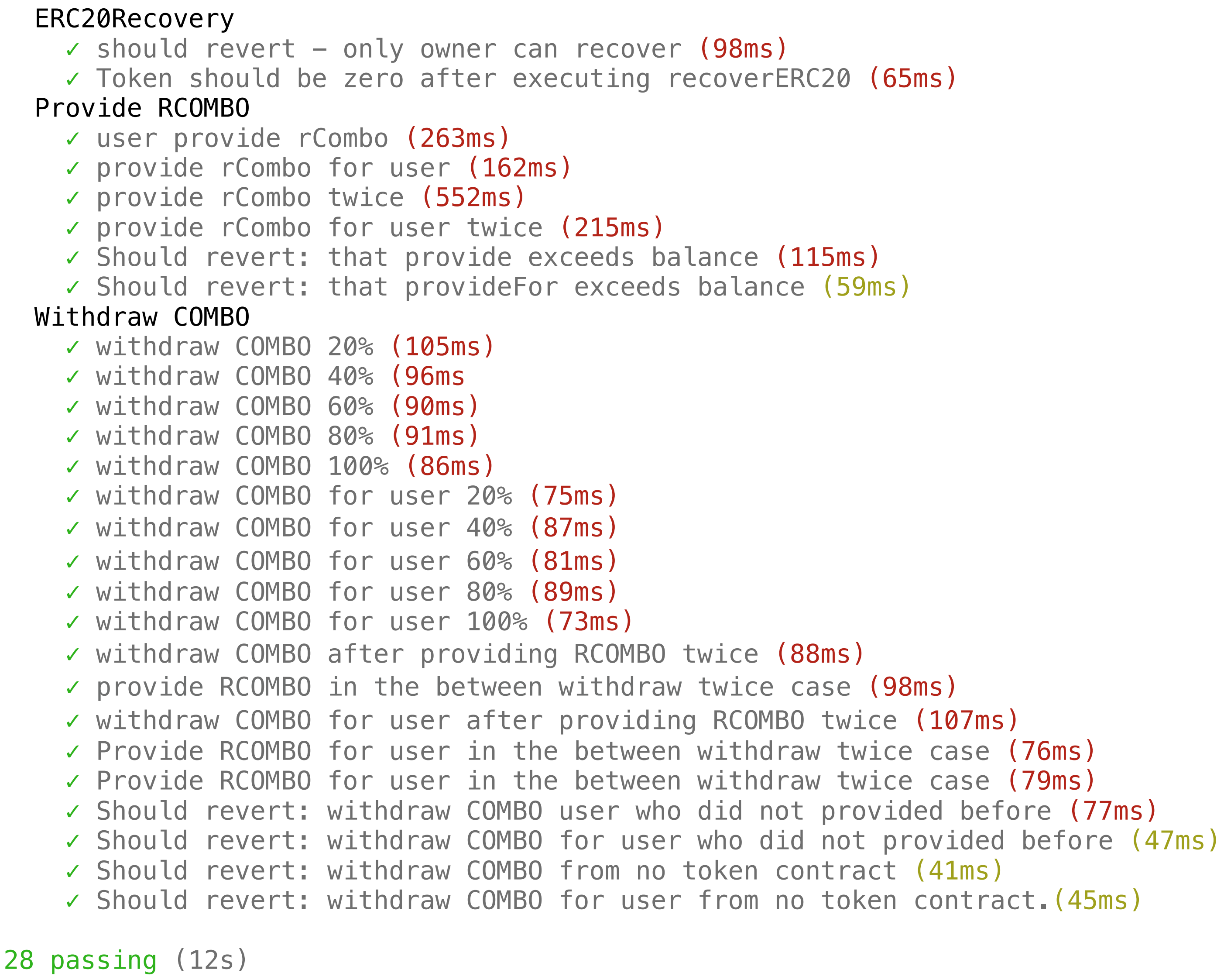
6.1.3 Unit Test



```
Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.

web3-shh package will be deprecated in version 1.3.5 and will no longer be supported.
web3-bzz package will be deprecated in version 1.3.5 and will no longer be supported.


  Contract: RCOMBO
    Transfer rCombo
      ✓ transfer (425ms)
      ✓ transferFrom rCombo (225ms)
```



```
  ERC20Recovery
    ✓ should revert - only owner can recover (98ms)
    ✓ Token should be zero after executing recoverERC20 (65ms)
  Provide RCOMBO
    ✓ user provide rCombo (263ms)
    ✓ provide rCombo for user (162ms)
    ✓ provide rCombo twice (552ms)
    ✓ provide rCombo for user twice (215ms)
    ✓ Should revert: that provide exceeds balance (115ms)
    ✓ Should revert: that provideFor exceeds balance (59ms)
  Withdraw COMBO
    ✓ withdraw COMBO 20% (105ms)
    ✓ withdraw COMBO 40% (96ms
    ✓ withdraw COMBO 60% (90ms)
    ✓ withdraw COMBO 80% (91ms)
    ✓ withdraw COMBO 100% (86ms)
    ✓ withdraw COMBO for user 20% (75ms)
    ✓ withdraw COMBO for user 40% (87ms)
    ✓ withdraw COMBO for user 60% (81ms)
    ✓ withdraw COMBO for user 80% (89ms)
    ✓ withdraw COMBO for user 100% (73ms)
    ✓ withdraw COMBO after providing RCOMBO twice (88ms)
    ✓ provide RCOMBO in the between withdraw twice case (98ms)
    ✓ withdraw COMBO for user after providing RCOMBO twice (107ms)
    ✓ Provide RCOMBO for user in the between withdraw twice case (76ms)
    ✓ Provide RCOMBO for user in the between withdraw twice case (79ms)
    ✓ Should revert: withdraw COMBO user who did not provided before (77ms)
    ✓ Should revert: withdraw COMBO for user who did not provided before (47ms)
    ✓ Should revert: withdraw COMBO from no token contract (41ms)
    ✓ Should revert: withdraw COMBO for user from no token contract.(45ms)

28 passing (12s)
```

7. Executive Summary

The overall code quality of the project is good and not overloaded with unnecessary functions, these is greatly benefiting the security of the contract. It correctly implemented widely-used and reviewed contracts from OpenZeppelin.

The main goal of the audit was to verify the claims regarding the security of the smart contract. During the audit, no critical issues were found after the manual and automated security testing and the claims been successfully verified.

8. Deployed Smart Contract

VERIFIED

Contract is deployed here:
https://etherscan.io/address/0x2DaDc3582C0655E8D21b1519baC30Bc40Ab14E9A#code