Furucombo

Vesting Contract

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
0.6 (31.03.2021)	Testing SWC Checks
0.7 (31.03.2021)	Verify Claims
0.9 (01.04.2021)	Summary and Recommendation
1.0 (01.04.2021)	Final document
1.2 (07.04.2021)	Added deployed contract addresses

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.

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)

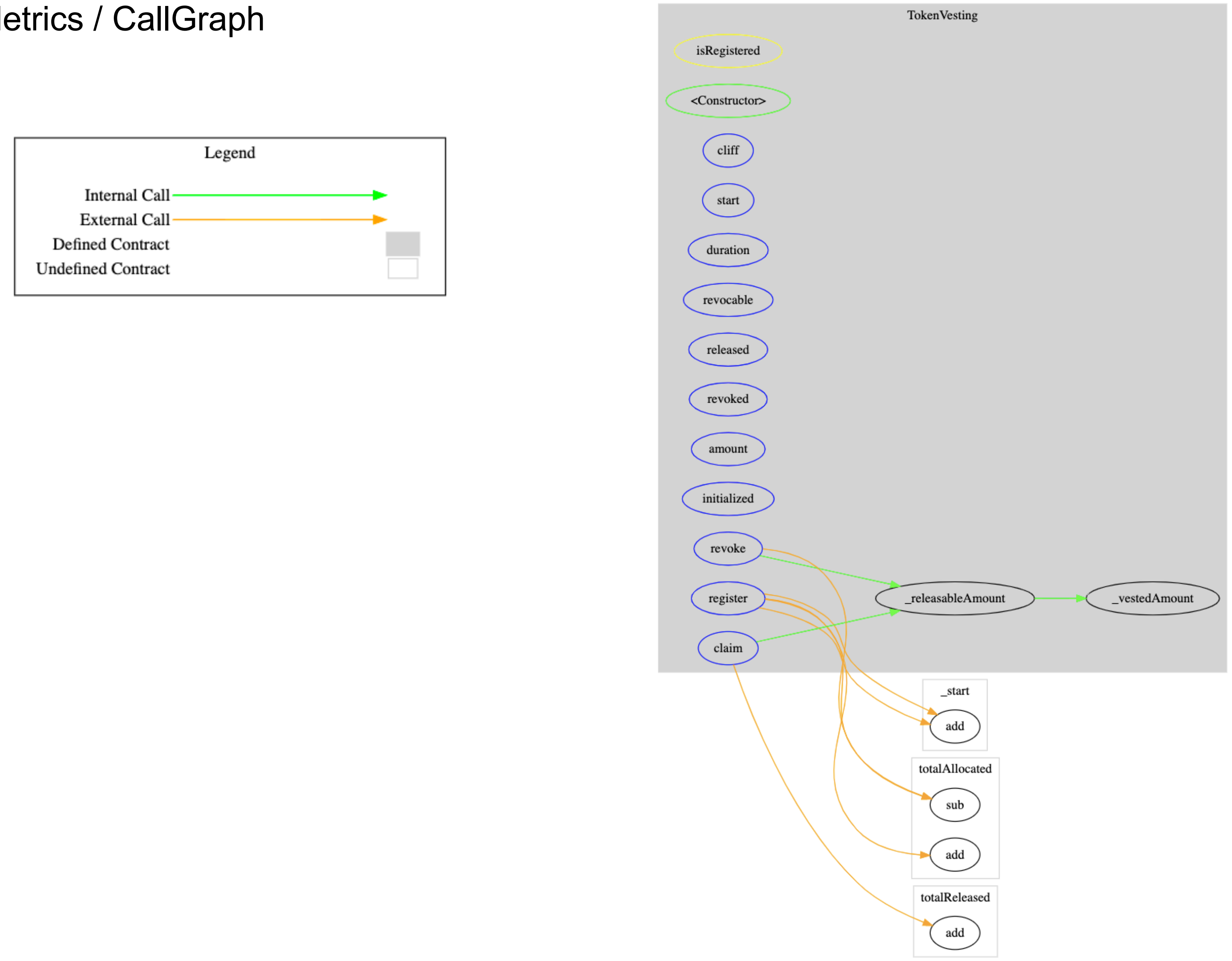
1. SafeMath.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.2.0/contracts/math/SafeMath.sol
2. IERC20.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.2.0/contracts/token/ERC20/IERC20.sol
3. SafeERC20.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.2.0/contracts/token/ERC20/SafeERC20.sol
4. Ownable.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.2.0/contracts/access/Ownable.sol
5. Context.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.2.0/contracts/GSN/Context.sol
6. Address.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.2.0/contracts/utils/Address.sol

4.3 Tested Contract Files

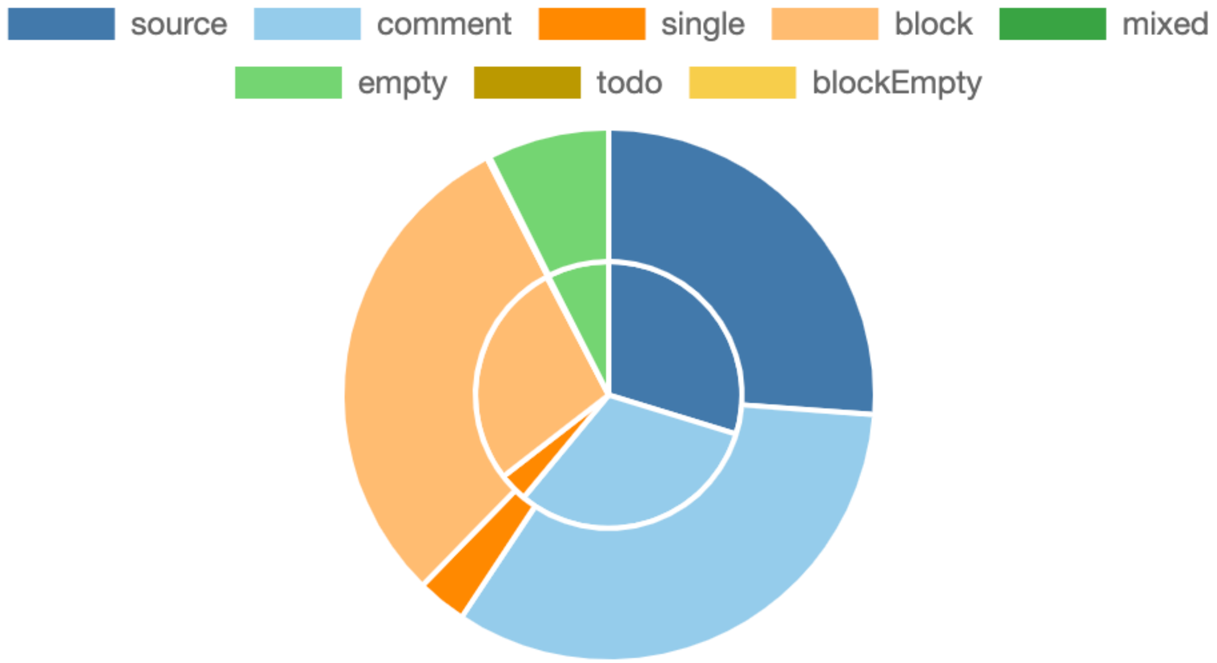
The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability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

File	Fingerprint (MD5)
TokenVesting.sol	d4ddb624a49d9271294bf66e87b93b38

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
`^0.6.0`			`yes` (2 asm blocks)	

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

🌐Public	💰Payable
21	0

External	Internal	Private	Pure	View
17	44	4	8	17

StateVariables

Total	🌐Public
5	3

4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍🎨	TokenVesting.sol	6	1	793	703	306	395	184	🖥☀
📝📚🔍🎨	**Totals**	**6**	**1**	**793**	**703**	**306**	**395**	**184**	🖥☀

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 Vesting Token contract.

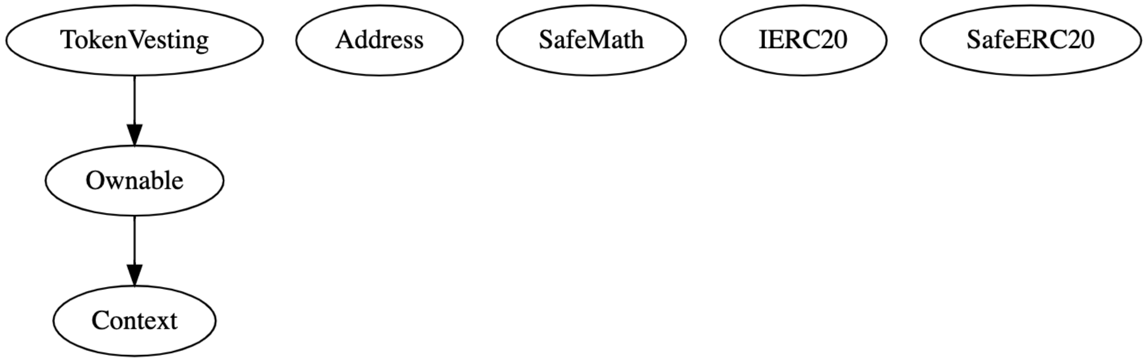
Following contracts with the direct imports been tested

- TokenVesting.sol

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

- It's possible to add beneficiary with related information to the vesting contract
- Deployer cannot burn any vested funds during the vesting period
- Deployer cannot pause the contract
- Beneficiaries are able to withdraw tokens after vesting period ends
- Deployer cannot withdraw token from beneficiaries during or after the vesting period
- Revoke should fail, if set as false

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

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

INFORMATIONAL ISSUES

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

5.2. SWC Attacks & Special Checks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✓
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✓
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✓
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✓
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✓
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✓
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✓
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✓

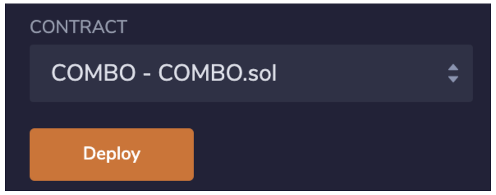
ID	Title	Relationships	Test Result
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✓
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✓
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✓
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✓
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	✓
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✓

ID	Title	Relationships	Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✓
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✓
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✓
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✓
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✓
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✓
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✓
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✓

ID	Title	Relationships	Test Result
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	✓
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✓
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✓
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓

7. Test Deployment

7.1.1 Deploy COMBO Token



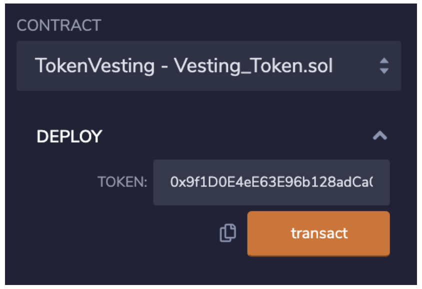
CONTRACT

COMBO - COMBO.sol

Deploy

Tx: https://ropsten.etherscan.io/tx/0x1bfd6c43ca3358e34d6319ccb42ee3aa65c03b4f14ec8b37075c07218914abc9
Contract: https://ropsten.etherscan.io/address/0xbe6e6a18f2897d46613e4779904a3430a70f550c#code

7.1.2 Deploy Vesting contract



CONTRACT

TokenVesting - Vesting_Token.sol

DEPLOY

TOKEN: 0x9f1D0E4eE63E96b128adCa(

transact

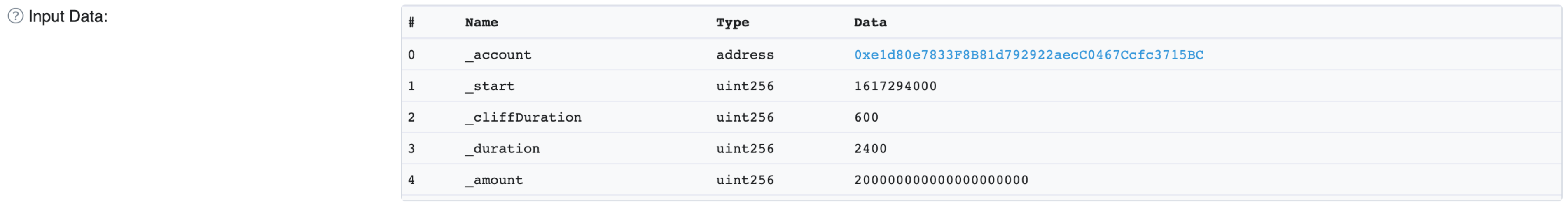
Tx: https://ropsten.etherscan.io/tx/0x837584cce63f40004211db772f2e69916adf4581ab1cc64fe5a22acaa29f7cb1
Contract: https://ropsten.etherscan.io/address/0xe92a115cf4a69ecc77c4793f5b33553eb9a0d587#code

7.1.3 Register vesting period for one beneficiary

Send funds to vesting contract 500 COMBO
Tx: https://ropsten.etherscan.io/tx/0x15990c5ffbf2d06a33104dd2e79c90237e23a5dc3124906b11f5c5d3984b2c7d

Registered vesting
Start: 18:20 1/4/2021
Cliff duration: 600 seconds
Duration: 2400 seconds (40 minutes)
Amount: 200 COMBO

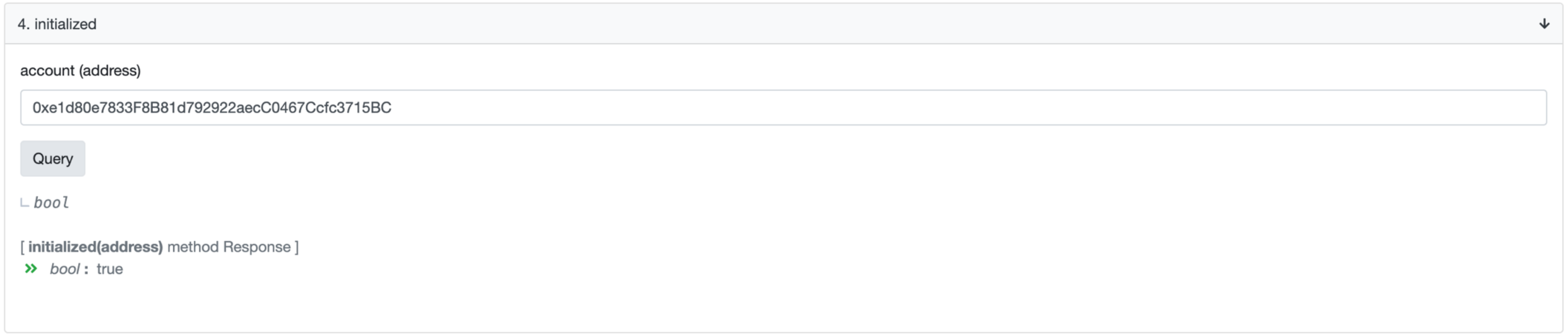
Tx: https://ropsten.etherscan.io/tx/0x8ba6261bb0d0a283fba94dcaa3409fe96f58bc17bac0ed7f249891baaca73f7f



Input Data:

#	Name	Type	Data
0	_account	address	0xe1d80e7833F8B81d792922aecC0467Ccfc3715BC
1	_start	uint256	1617294000
2	_cliffDuration	uint256	600
3	_duration	uint256	2400
4	_amount	uint256	200000000000000000000

7.1.4 Check if beneficiary is initialized


4. initialized

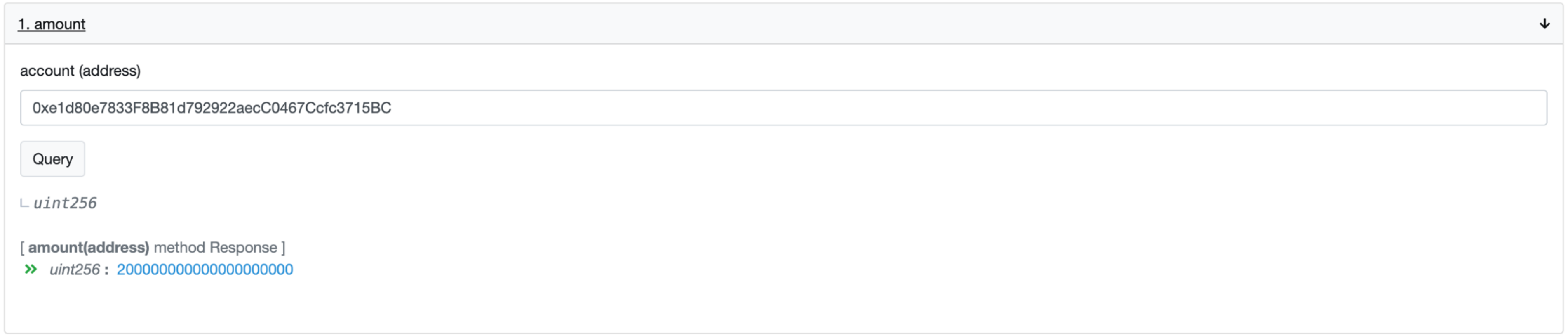
account (address)

0xe1d80e7833F8B81d792922aecC0467Ccfc3715BC

Query

∟ bool

[**initialized(address)** method Response]
» *bool* : true


1. amount

account (address)

0xe1d80e7833F8B81d792922aecC0467Ccfc3715BC

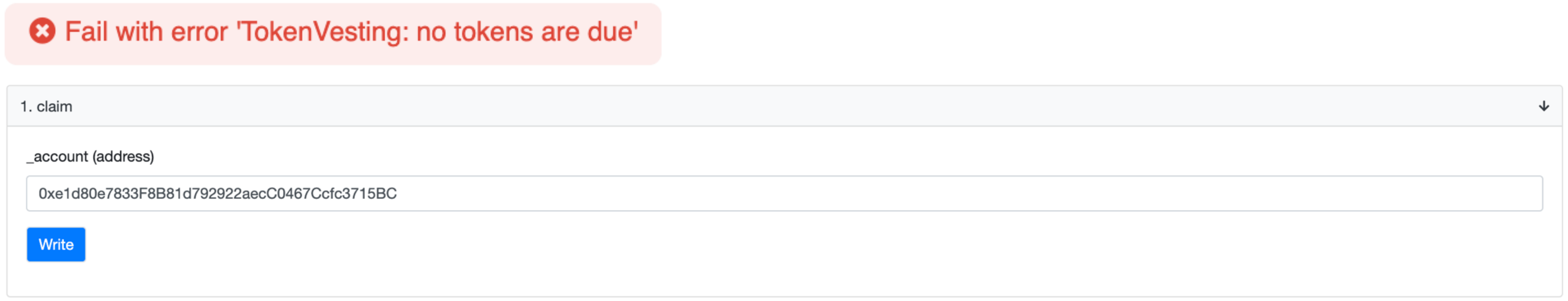
Query

∟ uint256

[**amount(address)** method Response]
» *uint256* : 20000000000000000000

7.1.4 Try to claim COMBO Token before vesting period starts

Tx: https://ropsten.etherscan.io/tx/0x82b70feced2dac2c20e6caaf164a8499cd002df67c3461a45d9976a708bb7a64



Fail with error 'TokenVesting: no tokens are due'

1. claim

_account (address)

0xe1d80e7833F8B81d792922aecC0467Ccfc3715BC

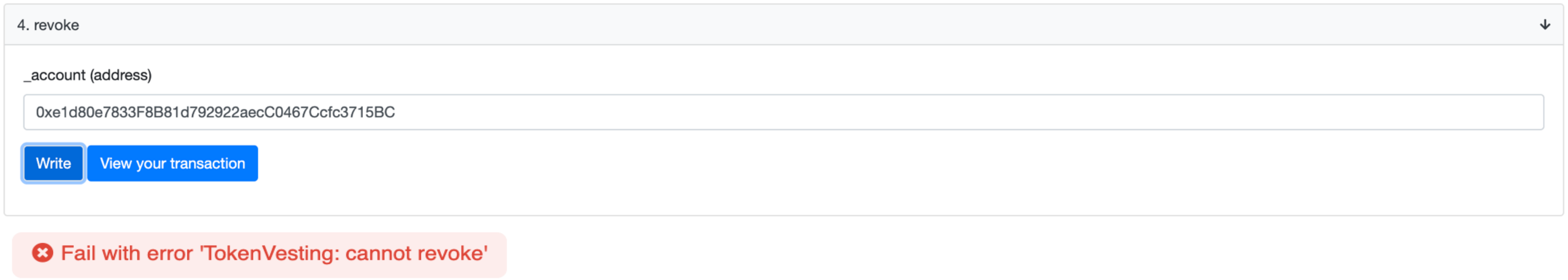
7.1.5 Try to burn any vested funds, during the vesting period

Tx: https://ropsten.etherscan.io/tx/0x8a59e651455306ce4e945364de08328d735c76b56c3a7a1ee00a1ddaaa5afc60
No token burn function or similar, at COMBO token or Vesting contract. Transfer out didn’t worked as well.

7.1.6 Try to pause contract during the vesting period

No functions to pause the contract.

7.1.6 Revoke vesting period should fail, if bool was set to false


4. revoke

_account (address)

0xe1d80e7833F8B81d792922aecC0467Ccfc3715BC

Fail with error 'TokenVesting: cannot revoke'

Tx: https://ropsten.etherscan.io/tx/0xf2f79c6f6fbb5d633ad3e0cdb2d9327829ef2a1444954c1c2f71827caf156ff8

7.1.7 Beneficiaries are able to withdraw tokens after vesting period ends

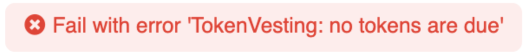
Tx: https://ropsten.etherscan.io/tx/0xf89b2a3b344e7bc1f976f54ece35eb806af7156e278b1492348fec362c7c9204
61 COMBO been possible to claim, after 13 minutes of vesting and a total duration of 40 minutes.

Tx: https://ropsten.etherscan.io/tx/0x9125626e3e8cc6b63286135175b490099183f985d49515c7f06d663829e5aa4a
46.583333333333333333 COMBO been possible to claim after 21 minutes of vesting and a total duration of 40 minutes.

Tx: https://ropsten.etherscan.io/tx/0x566a3783bd972385d75b55f441f5080fab9e53a08683e76fe0d5f1021ece40a4
92.416666666666666667 COMBO been possible to claim after 40 minutes of vesting and a total duration of 40 minutes.
200 of 200 COMBO been successful claimed / withdrawn from the vesting contract

7.1.8 Beneficiaries are not able to withdraw more tokens after vesting period ends and all tokens are withdrawn.

Tx: https://ropsten.etherscan.io/tx/0x18e469edd1ff103f22f452e70696a1406ef4154b4750261e0fd8aa0a1f0d635b


Fail with error 'TokenVesting: no tokens are due'

7.2. Verify Claims

7.2.1 It’s possible to add beneficiary with related information to the vesting contract
Status: tested and verified ✅

7.2.2 Deployer cannot burn any vested funds during the vesting period
Status: tested and verified ✅

7.2.3 Deployer cannot pause the contract
Status: tested and verified ✅

7.2.4 Beneficiaries are able to withdraw tokens after vesting period ends
Status: tested and verified ✅

7.2.5 Deployer cannot withdraw token from beneficiaries during or after the vesting period
Status: tested and verified ✅

7.2.6 Revoke should fail, if set as false
Status: tested and verified ✅

7.2.7 Checking the overall coding quality and security
Status: tested and verified ✅

8. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase. The final debrief took place on the April 01, 2021. The overall code quality of the project is very good, 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 and the functions. During the audit, no issues were found after the manual and automated security testing and the claims been successfully verified.

9. Deployed Smart Contract

VERIFIED

Contract is deployed here:

1.) https://etherscan.io/address/0x5A7434f0579354fB51EaB6f848cbdA4EAA53756f#code
2.) https://etherscan.io/address/0x8299cC8234D9539eF258A4c2600A2D94ef1042E3#code
3.) https://etherscan.io/address/0xb61B8EF639209a8292f88956319172337dFC0Ca5#code