Lendefi

Vesting

SMART CONTRACT AUDIT

26.08.2021

Made in Germany by Chainsulting.de

Table of contents

1. Disclaimer 3
2. About the Project and Company 4
   2.1 Project Overview 5
3. Vulnerability & Risk Level 6
4. Auditing Strategy and Techniques Applied 7
   4.1 Methodology 7
   4.2 Tested Contract Files 8
   4.3 Metrics / CallGraph 9
   4.4 Metrics / Source Lines & Risk 10
   4.5 Metrics / Capabilities 11
5. Scope of Work 13
   5.1 Manual and Automated Vulnerability Test 14
   5.2. SWC Attacks 15
   5.3. Verify Claims 19
6. Executive Summary 22
7. Deployed Smart Contract 22

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 DOGON SIRIUS LIMITED (Lendefi). 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 (22.08.2021)	Layout
0.2 (25.08.2021)	Test Deployment
0.5 (23.08.2021)	Automated Security Testing Manual Security Testing
0.6 (25.08.2021)	Testing SWC Checks
0.7 (25.08.2021)	Verify Claims
0.9 (25.08.2021)	Summary and Recommendation
1.0 (25.08.2021)	Final document
1.1 (26.08.2021)	Added deployed contract address

2. About the Project and Company

Company address:

DOGON SIRIUS LIMITED
Unit 3A-16, Level 3A, Labuan Times Square
Jalan Merdeka, 87000 Labuan
Malaysia

Website: https://www.lendefi.finance

Twitter: https://twitter.lendefi.finance

Telegram: https://telegram.lendefi.finance

Medium: https://medium.lendefi.finance

GitHub: https://github.lendefi.finance

LinkedIn: https://linkedin.lendefi.finance

Facebook: https://facebook.lendefi.finance

2.1 Project Overview

The Lendefi protocol (the “Protocol”) allows secured lending, giving the much-needed confidence to the lenders in a highly volatile crypto market. Secure lending options will open up lending opportunities for traditional and private lenders to access higher interest rates without getting direct exposure to the crypto market fluctuations.

Lendefi protocol cuts the middle-man out of the lending process and eliminates the red tape involved with the lending and borrowing. This removes any counterparty risk between the borrower and the lender, who then can deal on a trustless basis. The lender will receive a variable interest and be secured by the liquidity provided on the DeFi ecosystem in such protocols as Uniswap . Hence, if the borrower is not able to maintain their loan, the Protocol will ensure the lender is repaid and the borrower credited with the remaining equity. Borrowers can select from a wide variety of supported assets to invest by borrowing funds from the Protocol.

Supported assets can be added and removed via Lendefi’s decentralized governance mechanism (the “DAO”). The base currency for lending and borrowing is USDC, hence making it more user-friendly and fostering mainstream adoption. Lendefi has specifically chosen USDC because it is the safest stable coin from a custody and reputation perspective, given it is a collaboration between Coinbase and Circle, and undergoes regular audits and is subject to regulatory compliance.

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:

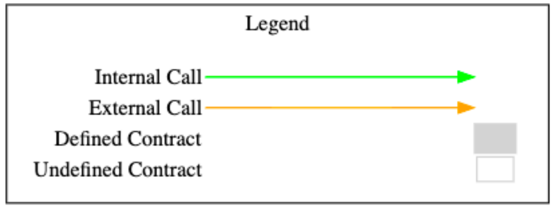
1. Code review that includes the following:
   1. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
   2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   3. 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:
   1. 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.
   2. 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.

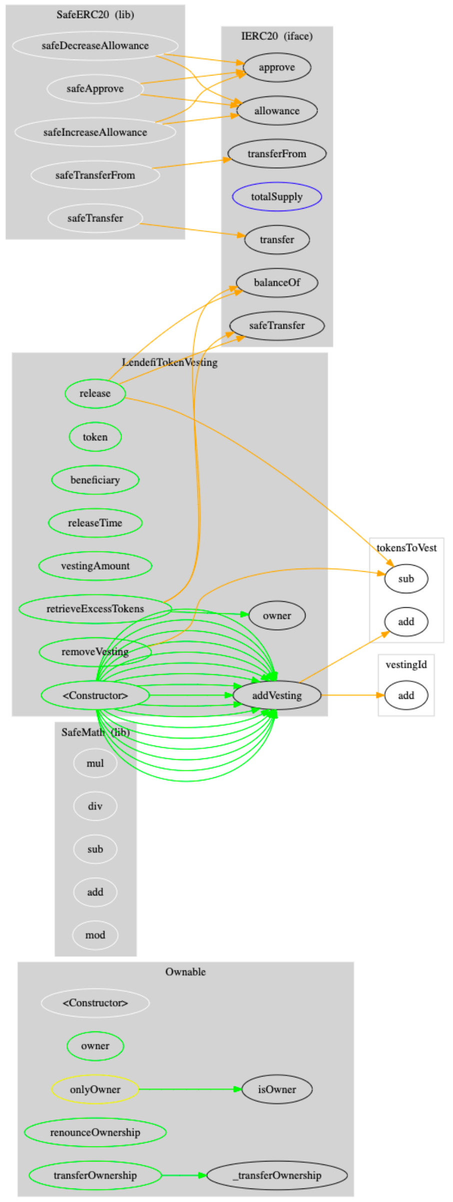
4.2 Tested Contract Files

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

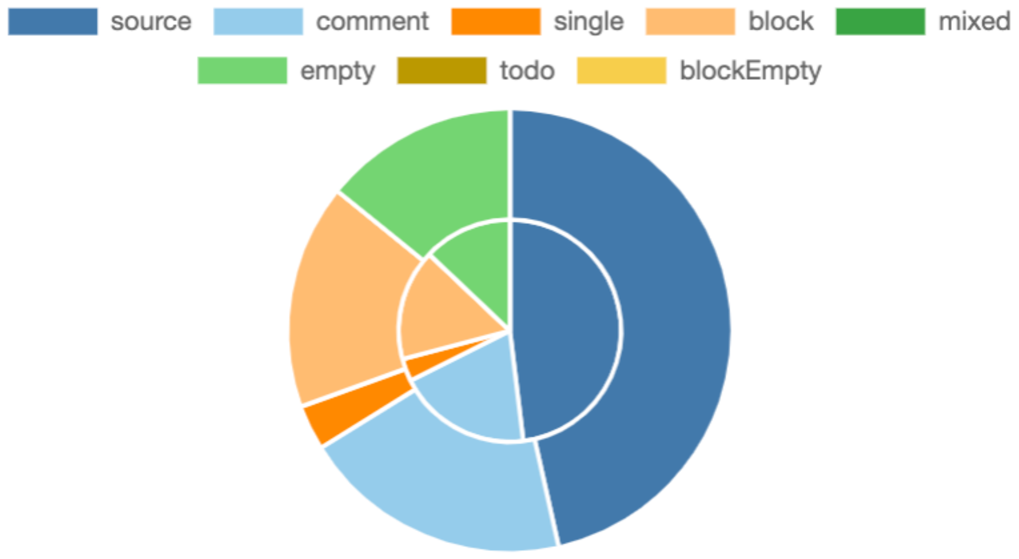
File	Fingerprint (MD5)
./vesting_new.sol	ec433a28473a7e679a72a172018f6238

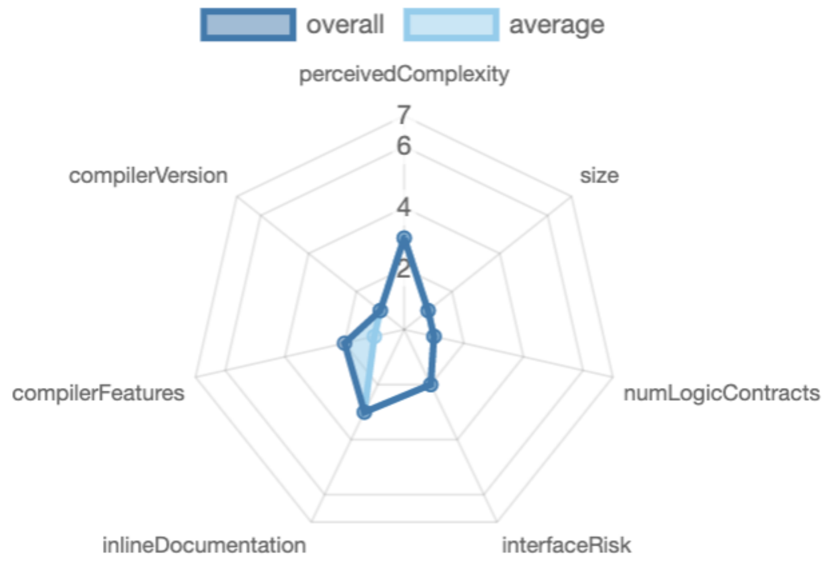
4.3 Metrics / CallGraph





4.4 Metrics / Source Lines & Risk





4.5 Metrics / Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`0.5.2`			**** (0 asm blocks)	

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

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐Public	💰Payable
19	0

External	Internal	Private	Pure	View
6	29	0	5	9

StateVariables

Total	🌐Public
10	1

4.6 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍	Vesting Contract/vesting_new.sol	4	1	318	293	175	78	142	📤☀️
📝📚🔍	**Totals**	**4**	**1**	**318**	**293**	**175**	**78**	**142**	📤☀️

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, ...)

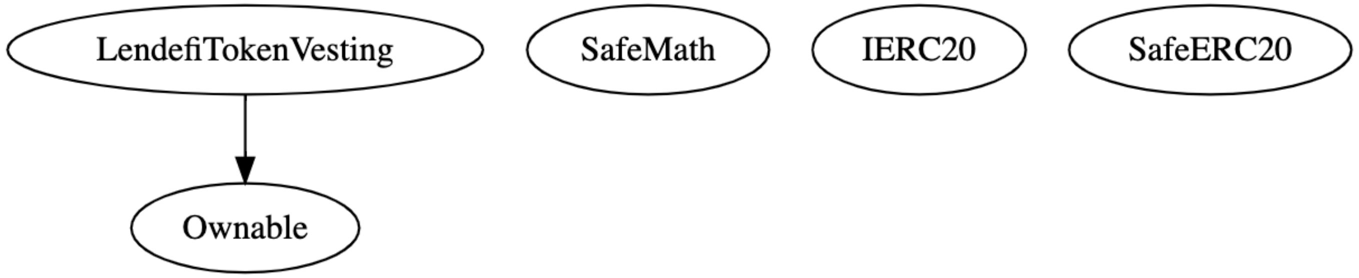
5. Scope of Work

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

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

- Deployer/Owner can’t burn any vested funds during the vesting periods
- Beneficiaries can’t withdraw vested tokens before a vesting period ends.
- Deployer/Owner can’t update beneficiary during vesting period
- The smart contract is coded according to the newest standards and in a secure way.

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

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	✓

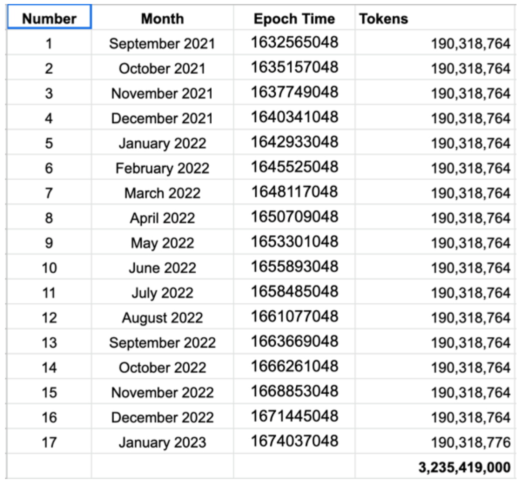
ID	Title	Relationships	Test Result
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	✓
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✓
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✓
SWC-100	Function Default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

5.3. Verify Claims

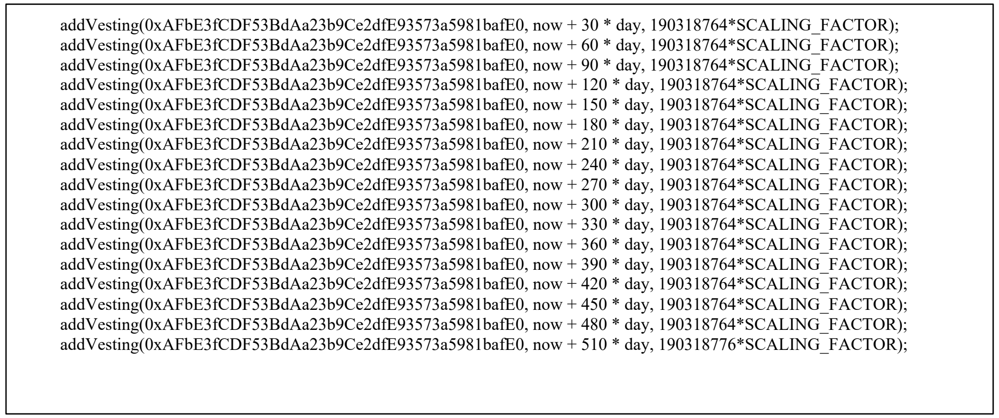
5.3.1 The vesting release duration is correctly calculated and working

Status: tested and verified ✅

deployment to testnet
Contract: https://testnet.bscscan.com/address/0x888c4af1794c56b71cddbaeb7310f0fc200b2c8b#code
Tx: https://testnet.bscscan.com/tx/0x4eddb03bd92379c1685e176d8878ae6958a5c2f9f6d62771b4fd6c608d645587
Token: https://testnet.bscscan.com/token/0xff88d03a430d08de01a6ca5918ae370fb2b29e24



Number	Month	Epoch Time	Tokens
1	September 2021	1632565048	190,318,764
2	October 2021	1635157048	190,318,764
3	November 2021	1637749048	190,318,764
4	December 2021	1640341048	190,318,764
5	January 2022	1642933048	190,318,764
6	February 2022	1645525048	190,318,764
7	March 2022	1648117048	190,318,764
8	April 2022	1650709048	190,318,764
9	May 2022	1653301048	190,318,764
10	June 2022	1655893048	190,318,764
11	July 2022	1658485048	190,318,764
12	August 2022	1661077048	190,318,764
13	September 2022	1663669048	190,318,764
14	October 2022	1666261048	190,318,764
15	November 2022	1668853048	190,318,764
16	December 2022	1671445048	190,318,764
17	January 2023	1674037048	190,318,776
			3,235,419,000



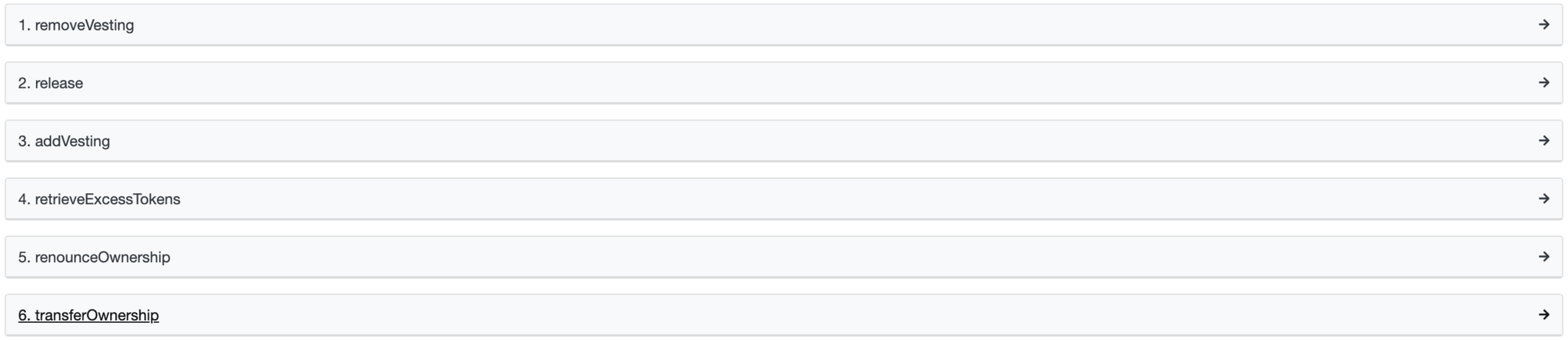
```
addVesting(0xAFbE3fCDF53BdAa23b9Ce2dfE93573a5981bafE0, now + 30 * day, 190318764*SCALING_FACTOR);
addVesting(0xAFbE3fCDF53BdAa23b9Ce2dfE93573a5981bafE0, now + 60 * day, 190318764*SCALING_FACTOR);
addVesting(0xAFbE3fCDF53BdAa23b9Ce2dfE93573a5981bafE0, now + 90 * day, 190318764*SCALING_FACTOR);
addVesting(0xAFbE3fCDF53BdAa23b9Ce2dfE93573a5981bafE0, now + 120 * day, 190318764*SCALING_FACTOR);
addVesting(0xAFbE3fCDF53BdAa23b9Ce2dfE93573a5981bafE0, now + 150 * day, 190318764*SCALING_FACTOR);
addVesting(0xAFbE3fCDF53BdAa23b9Ce2dfE93573a5981bafE0, now + 180 * day, 190318764*SCALING_FACTOR);
addVesting(0xAFbE3fCDF53BdAa23b9Ce2dfE93573a5981bafE0, now + 210 * day, 190318764*SCALING_FACTOR);
addVesting(0xAFbE3fCDF53BdAa23b9Ce2dfE93573a5981bafE0, now + 240 * day, 190318764*SCALING_FACTOR);
addVesting(0xAFbE3fCDF53BdAa23b9Ce2dfE93573a5981bafE0, now + 270 * day, 190318764*SCALING_FACTOR);
addVesting(0xAFbE3fCDF53BdAa23b9Ce2dfE93573a5981bafE0, now + 300 * day, 190318764*SCALING_FACTOR);
addVesting(0xAFbE3fCDF53BdAa23b9Ce2dfE93573a5981bafE0, now + 330 * day, 190318764*SCALING_FACTOR);
addVesting(0xAFbE3fCDF53BdAa23b9Ce2dfE93573a5981bafE0, now + 360 * day, 190318764*SCALING_FACTOR);
addVesting(0xAFbE3fCDF53BdAa23b9Ce2dfE93573a5981bafE0, now + 390 * day, 190318764*SCALING_FACTOR);
addVesting(0xAFbE3fCDF53BdAa23b9Ce2dfE93573a5981bafE0, now + 420 * day, 190318764*SCALING_FACTOR);
addVesting(0xAFbE3fCDF53BdAa23b9Ce2dfE93573a5981bafE0, now + 450 * day, 190318764*SCALING_FACTOR);
addVesting(0xAFbE3fCDF53BdAa23b9Ce2dfE93573a5981bafE0, now + 480 * day, 190318764*SCALING_FACTOR);
addVesting(0xAFbE3fCDF53BdAa23b9Ce2dfE93573a5981bafE0, now + 510 * day, 190318776*SCALING_FACTOR);
```

withdraw first vesting amount with modified period 0 and vesting id 1
Tx: https://testnet.bscscan.com/tx/0xa0caa738ab949cbb3cc286cad62d964a338afea33c73dc443d1d364a72ec24e8

5.3.2 Deployer/Owner can’t burn any vested funds during the vesting periods

Status: tested and verified ✅

There is no burn function

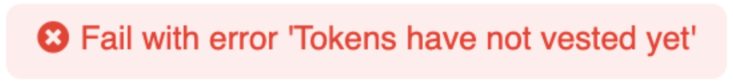

1. removeVesting →
2. release →
3. addVesting →
4. retrieveExcessTokens →
5. renounceOwnership →
6. transferOwnership →

5.3.3 Beneficiaries can’t withdraw vested tokens before a vesting period ends.

Status: tested and verified ✅

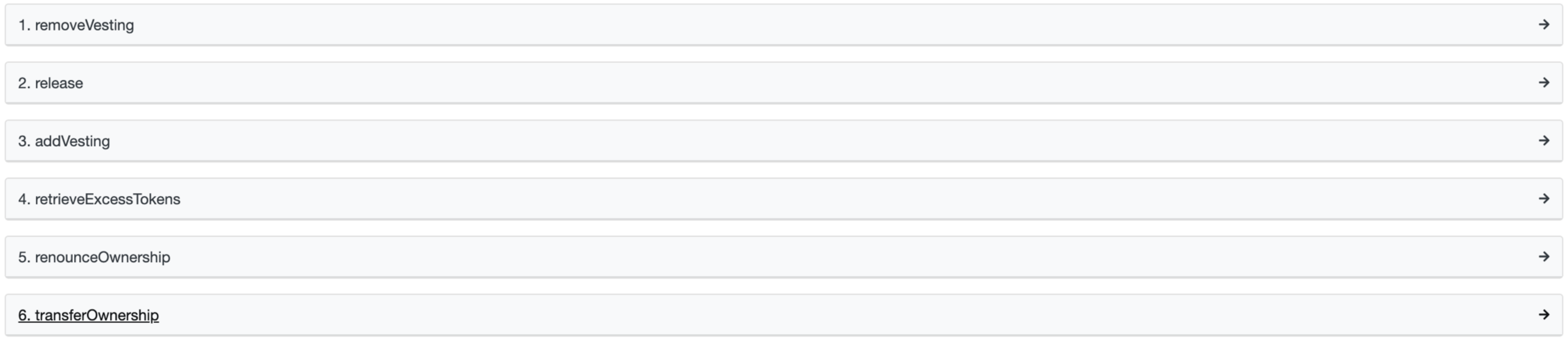
Tx: https://testnet.bscscan.com/tx/0x892113340b127c43d70edbef9de4d2f4e3c1c73af749a499da332a2841e798f1

Try to release vesting with id 2


Fail with error 'Tokens have not vested yet'

5.3.4 Deployer/Owner can’t update beneficiary during vesting period

Status: tested and verified ✅



1. removeVesting →
2. release →
3. addVesting →
4. retrieveExcessTokens →
5. renounceOwnership →
6. transferOwnership →

5.3.5 The smart contract is coded according to the newest standards and in a secure way.

Status: tested and verified ✅

6. Executive Summary

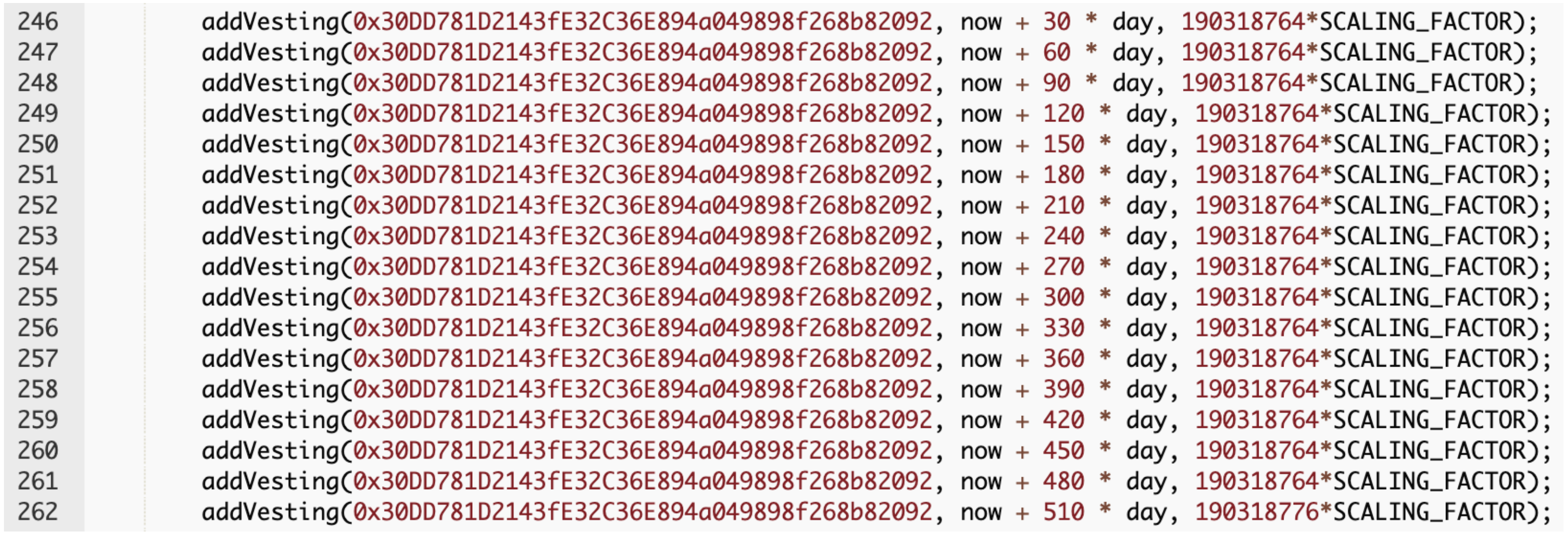
Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase. The final debriefs took place on the August 25, 2021.

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 critical issues were found after the manual and automated security testing and the claims been successfully verified.

7. Deployed Smart Contract

VERIFIED

https://bscscan.com/address/0xc598d81c62f6391b2412d02a78fa3f3affe58b52#code



```
        addVesting(0x30DD781D2143fE32C36E894a049898f268b82092, now + 30 * day, 190318764*SCALING_FACTOR);
        addVesting(0x30DD781D2143fE32C36E894a049898f268b82092, now + 60 * day, 190318764*SCALING_FACTOR);
        addVesting(0x30DD781D2143fE32C36E894a049898f268b82092, now + 90 * day, 190318764*SCALING_FACTOR);
        addVesting(0x30DD781D2143fE32C36E894a049898f268b82092, now + 120 * day, 190318764*SCALING_FACTOR);
        addVesting(0x30DD781D2143fE32C36E894a049898f268b82092, now + 150 * day, 190318764*SCALING_FACTOR);
        addVesting(0x30DD781D2143fE32C36E894a049898f268b82092, now + 180 * day, 190318764*SCALING_FACTOR);
        addVesting(0x30DD781D2143fE32C36E894a049898f268b82092, now + 210 * day, 190318764*SCALING_FACTOR);
        addVesting(0x30DD781D2143fE32C36E894a049898f268b82092, now + 240 * day, 190318764*SCALING_FACTOR);
        addVesting(0x30DD781D2143fE32C36E894a049898f268b82092, now + 270 * day, 190318764*SCALING_FACTOR);
        addVesting(0x30DD781D2143fE32C36E894a049898f268b82092, now + 300 * day, 190318764*SCALING_FACTOR);
        addVesting(0x30DD781D2143fE32C36E894a049898f268b82092, now + 330 * day, 190318764*SCALING_FACTOR);
        addVesting(0x30DD781D2143fE32C36E894a049898f268b82092, now + 360 * day, 190318764*SCALING_FACTOR);
        addVesting(0x30DD781D2143fE32C36E894a049898f268b82092, now + 390 * day, 190318764*SCALING_FACTOR);
        addVesting(0x30DD781D2143fE32C36E894a049898f268b82092, now + 420 * day, 190318764*SCALING_FACTOR);
        addVesting(0x30DD781D2143fE32C36E894a049898f268b82092, now + 450 * day, 190318764*SCALING_FACTOR);
        addVesting(0x30DD781D2143fE32C36E894a049898f268b82092, now + 480 * day, 190318764*SCALING_FACTOR);
        addVesting(0x30DD781D2143fE32C36E894a049898f268b82092, now + 510 * day, 190318776*SCALING_FACTOR);
```