Security Assessment

Chainge

May 7th, 2021

Summary

This report has been prepared for Chainge smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Chainge
Description	Chainge is a FRC758 token. And the FRC758 is a time-slicing smart contract standard on the Fusion blockchain. That different from ERC20 tokens in that it supports time slicing.
Platform	Fusion blockchain
Language	Solidity
Codebase	https://github.com/ChaingeFinance/ChaingeToken
Commits	85c3e11ed6fcf9c97c98d6fc50e7e07cebfe54a8

Audit Summary

Delivery Date	May 07, 2021
Audit Methodology	Manual Review, Static Analysis
Key Components	

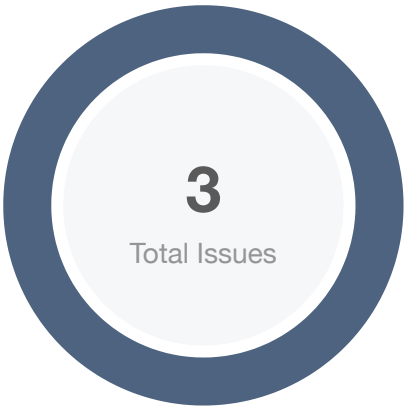
Vulnerability Summary

Total Issues	3
Critical	0
Major	0
Medium	0
Minor	0
Informational	3
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
CTC	ChaingeToken.sol	a4fcef2dd9a2ad151c530697049e767e9548c2e0fce2de82b9ef51eb496a8328
CCT	Controllable.sol	b9505db9b62bb2782c69c1bacdf4435bb99109eb45b94edbe759720904bbd6c6
FRC	FRC758.sol	2427d62545ab26f13669026a5e5b9caf898712d5c15c54088ac7f7099c99bcd4
OCT	Ownable.sol	96a3a5df189f55e3a39fdb1d2a4b95861a88de2d802553c370cc77b11ad94468
IFR	interfaces/IFRC758.sol	c387d2d9542d49ebfd1fde9d58d50606928d9f0b177719a141b2c5ac4a4dce55
ITS	interfaces/ITimeSlicedTokenReceiver.sol	a2e0b5b1227fc55baea77242e555f5cf1fa5625faefbdf673877be0f347a24c6
SMC	libraries/SafeMath256.sol	561687055acd6f7390b19d3b509e3286162b2677f01b0983eab65ca258078f27

Findings



Critical	0 (0.00%)
Major	0 (0.00%)
Medium	0 (0.00%)
Minor	0 (0.00%)
Informational	3 (100.00%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
FRC-01	Redundant Check	Logical Issue	Informational	Resolved
FRC-02	Local Variable Shadowing	Optimization	Informational	Resolved
FRC-03	Uninitialized state variables	Coding Style	Informational	Acknowledged

FRC-01 | Redundant Check

Category	Severity	Location	Status
Logical Issue	Informational	FRC758.sol: 286	Resolved

Description

The judgment condition is repeated.

```
if( currSt.tokenEnd > st.tokenStart && currSt.tokenEnd >= st.tokenStart) {
```

Recommendation

Please confirm and modify it.

Alleviation

The recommendation was applied in commit 736b2a4304ad7bf6b76b202040ca8dee4383004e.

FRC-02 | Local Variable Shadowing

Category	Severity	Location	Status
Optimization	Informational	FRC758.sol: 405	Resolved

Description

This declaration shadows an existing declaration(FRC758.sol:405:26).

```
uint256 index = _addSlice(addr, st.tokenEnd, currStTokenEnd, currStAmunt, currSt.next);
```

The shadowed declaration is here(FRC758.sol:383:17):

```
uint256 index = _addSlice(addr, currSt.tokenStart, st.tokenStart, currSt.amount,
current);
```

Recommendation

Rename the local variable.

Alleviation

The recommendation was applied in commit 736b2a4304ad7bf6b76b202040ca8dee4383004e.

FRC-03 | Uninitialized state variables

Category	Severity	Location	Status
Coding Style	Informational	FRC758.sol: 59	Acknowledged

Description

Uninitialized state variables.

Recommendation

Initialize all the variables. If a variable is meant to be initialized to zero, explicitly set it to zero to improve code readability.

Alleviation

The development team has acknowledged this finding.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings 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.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

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

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in-storage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is 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 assessment. This report does 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.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.