CREDMARK

SMART CONTRACT AUDIT

ZOKYO.

June 26th, 2021 | v. 1.0

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

SCORE

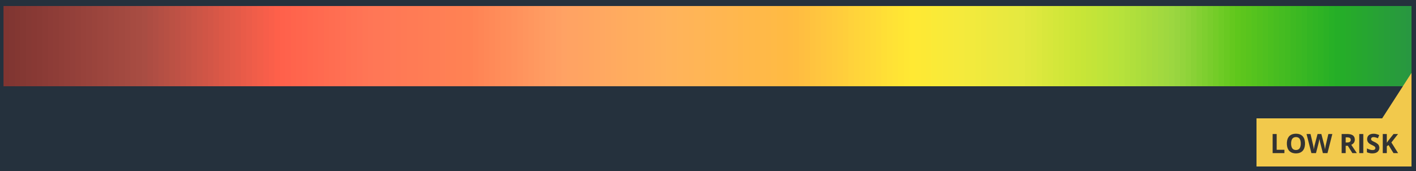
100

TECHNICAL SUMMARY

This document outlines the overall security of the Credmark smart contracts, evaluated by Zokyo's Blockchain Security team.

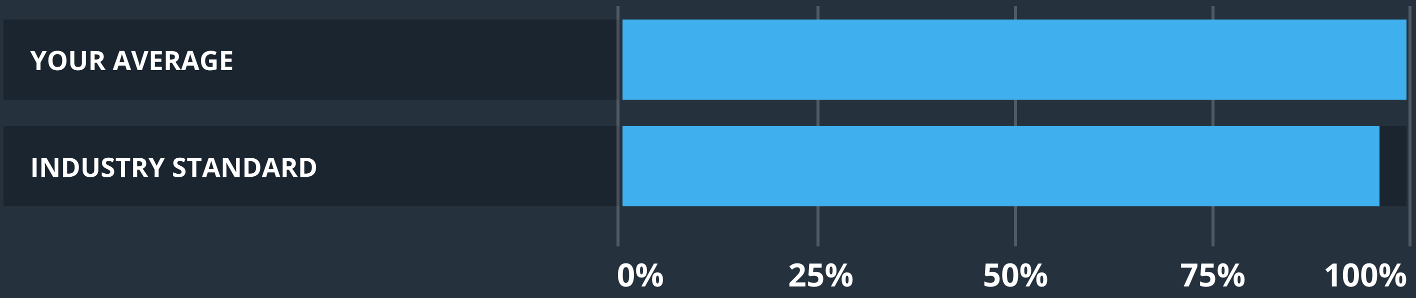
The scope of this audit was to analyze and document the Credmark smart contract codebase for quality, security, and correctness.

Contract Status



There were no critical issues found during the audit.

Testable Code



The testable code is 100%, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the Credmark team put in place a bug bounty program to encourage further and active analysis of the smart contract.

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Credmark smart contract's source code was taken from the repository provided by the Credmark team:
https://github.com/credmark/cmk-launch-contracts
Initial commit: 29416c2f95d4b5dd301b14af230490265e22ab75
Last reviewed commit: 3fcdaba3b1ed1b8a0f15f1ace17ca3668168078d

Within the scope of this audit Zokyo auditors have reviewed the following contract(s):
CMKToken.sol
VestingSchedule.sol

Throughout the review process, care was taken to ensure that the token contract:

- Implements and adheres to existing Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of Credmark smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1 Due diligence in assessing the overall code quality of the codebase.

2 Cross-comparison with other, similar smart contracts by industry leaders.

3 Testing contract logic against common and uncommon attack vectors.

4 Thorough, manual review of the codebase, line-by-line.

EXECUTIVE SUMMARY

There were no critical issues found during the audit. All mentioned findings may have an effect only in case of specific conditions performed by the contract owner and are connected to the code style and minor optimizations.

All issues were successfully fixed by the Credmark team.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

● Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

● High

The issue affects the ability of the contract to compile or operate in a significant way.

● Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

● Low

The issue has minimal impact on the contract’s ability to operate.

● Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

MEDIUM | RESOLVED

Solidity version update

Issue is classified as Medium, because it is included to the list of standard smart contracts' vulnerabilities. Currently used version (0.8.0) is not the last in the line, so it contradicts standard checklist.

Recommendation:
You need to update the solidity version to the latest one in the branch - consider 0.8.4.

LOW | RESOLVED

Methods should be declared as external

VestingScedule.sol:
addVestingSchedule(), claim(), cancel(), getTokenAddress(), getTotalAllocation(), getTotalClaimedAllocation(), getVestingSchedule()

Recommendation:
Declare methods as external.

LOW | RESOLVED

SafeMath can be removed

Since the contract has 0.8 version of Solidity, SafeMath library can be removed, as overflow and underflow errors are built-in. Removing the library will give gas savings by omitting extra function calls.

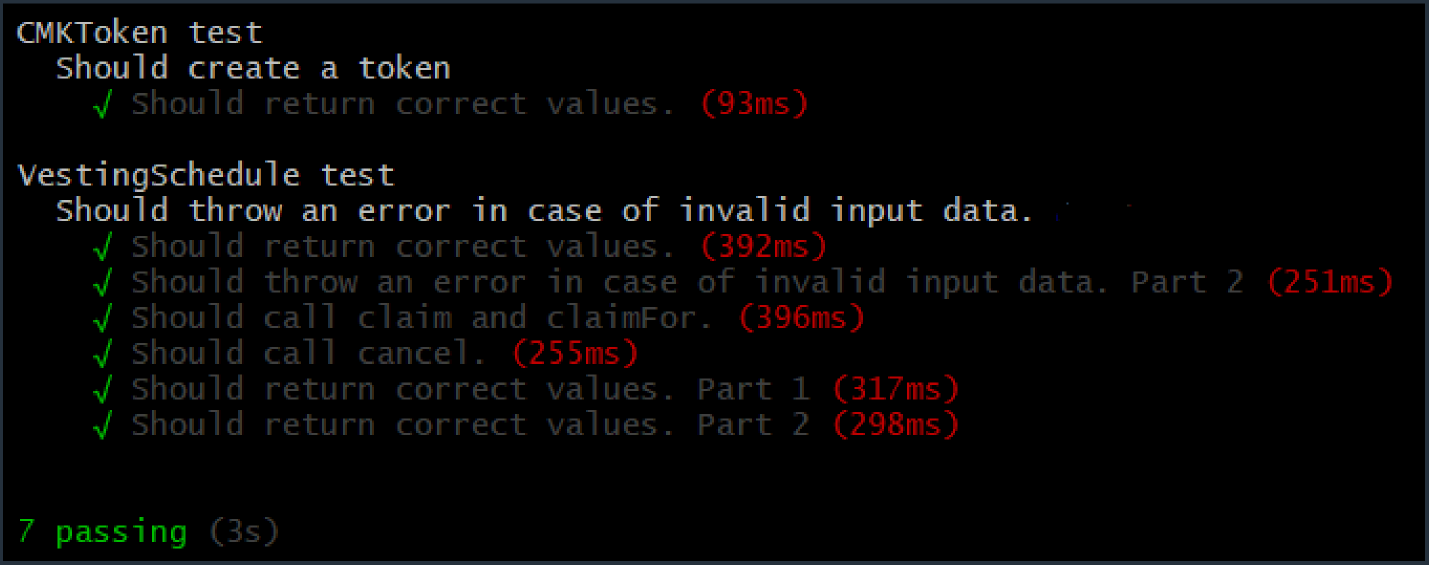
Recommendation:
Omit SafeMath library.

	CMKToken	VestingSchedule
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo team

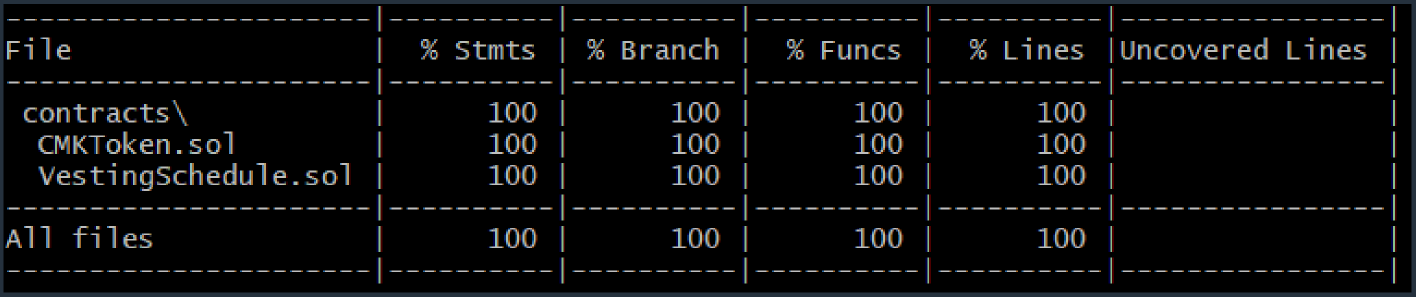
As part of our work assisting Credmark in verifying the correctness of their contract code, our team was responsible for writing integration tests using Truffle testing framework.
Tests were based on the functionality of the code, as well as review of the Credmark contract requirements for details about issuance amounts and how the system handles these.



```
CMKToken test
  Should create a token
    √ Should return correct values. (93ms)

VestingSchedule test
  Should throw an error in case of invalid input data.
    √ Should return correct values. (392ms)
    √ Should throw an error in case of invalid input data. Part 2 (251ms)
    √ Should call claim and claimFor. (396ms)
    √ Should call cancel. (255ms)
    √ Should return correct values. Part 1 (317ms)
    √ Should return correct values. Part 2 (298ms)


7 passing (3s)
```



```
----------------------|----------|----------|----------|----------|----------------|
File                  |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
----------------------|----------|----------|----------|----------|----------------|
 contracts\           |      100 |      100 |      100 |      100 |                |
  CMKToken.sol        |      100 |      100 |      100 |      100 |                |
  VestingSchedule.sol |      100 |      100 |      100 |      100 |                |
----------------------|----------|----------|----------|----------|----------------|
All files             |      100 |      100 |      100 |      100 |                |
----------------------|----------|----------|----------|----------|----------------|
```

We are grateful to have been given the opportunity to work with the Credmark team.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

Zokyo's Security Team recommends that the Credmark team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.