Smart Contract Audit Report

June, 2023

DEFIMOON PROJECT

Audit and
Development

CONTACTS

defimoon.org
audit@defimoon.org
defimoon_org
defimoonorg
defimoon
defimoonorg

DEFIMOON

be secure

18 June 2023

This audit report was prepared by Defimoon for Token IN.

Audit information

Description	erc20-like coin, vesting, staking
Audited files	InCoin, InVesting, InStaking
Timeline	13-18 June 2023
Audited by	Ilya Vaganov
Approved by	Artur Makhnach, Kirill Minyaev
Deployed to	0xc32ba5d293577cbb1df390f35b2bc6369a593b736d0865fedec1a2b08565de8e 0xa711ef10df1d60f79396e3ae8b42d282afac2d26b29193cd2b5ad9807f313bd7 0x2fce9ccd4c9fc0bcf8fdadb6ab9204135254b7966ede329df6e788a2d10c613c
Languages	Move
Methods	Architecture Review, Unit Testing, Functional Testing, Manual Review
Chain	Aptos
Status	Passed

Disclaimer

Audit Information

Defimoon utilizes both manual and automated auditing approach to cover the most ground possible. We begin with generic static analysis automated tools to quickly assess the overall state of the contract. We then move to a comprehensive manual code analysis, which enables us to find security flaws that automated tools would miss. Finally, we conduct an extensive unit testing to make sure contract behaves as expected under stress conditions.

In our decision making process we rely on finding located via the manual code inspection and testing. If an automated tool raises a possible vulnerability, we always investigate it further manually to make a final verdict. All our tests are run in a special test environment which matches the "real world" situations and we utilize exact copies of the published or provided contracts.

While conducting the audit, the Defimoon security team uses best practices to ensure that the reviewed contracts are thoroughly examined against all angles of attack. This is done by evaluating the codebase and whether it gives rise to significant risks. During the audit, Defimoon assesses the risks and assigns a risk level to each section together with an explanatory comment.

Audit overview

No issues found.

All modules are implemented using the best development practices. The modules use the official aptos-framework toolkit to achieve greater security and standards compatibility.

During testing, the main scenarios for interacting with modules were checked for vulnerabilities, bugs, and design logic.

Modules are tested and can be used in production.

Tests

InCoin

Test	Status
Should fail if not an InCoin account try mint	✔
Should fail if mint is called to an unregistered account	✔
Should success if an InCoin account try mint	✔
Should fail if trying to mint when supply limit reached	✔
Should success if non-registered account trying register	✔
Should success if registered account again trying register	✔
Should fail if trying to burn with insufficient balance	✔
Should success if trying to burn with enough balance	✔
Should transfer coins with correct balances changes	✔
Should default coin.move functions working correctly	✔

InVesting

Test	Status
Should fail release_funds if account not vested	✔
Should fail total_vested if account not vested	✔
Should fail residue if account not vested	✔
Should fail next_release if account not vested	✔
Should fail vesting_funds with insufficient balance	✔
Should success vesting_funds with enough balance	✔
Should fail release_funds if not enough time has passed	✔
Should success release_funds if enough time has passed	✔
Should success total_vested if account vested	✔
Should success residue if account vested	✔
Should success next_release if account vested	✔
Should fail release_funds if vesting ended	✔
Should fail vesting_funds again to the same account	✔

InStaking

Test	Status
Should fail stake if not enough money in the treasury	✔
Should fail replenish_the_treasury with insufficient balance	✔
Should success replenish_the_treasury with enough balance	✔
Should fail stake if the wrong option is selected	✔
Should fail stake if the wrong stake amount	✔
Should fail stake with insufficient balance	✔
Should fail claim if not staked	✔
Should fail get_stakes_indexes if account not staked	✔
Should fail next_claim if account not staked	✔
Should fail until_unlock if account not staked	✔
Should success stake with correct params and enough balance	✔
Should success get_stakes_indexes if staked	✔
Should fail claim if not enough time has passed	✔
Should fail claim if wrong id	✔
Should fail next_claim if wrong id	✔
Should fail until_unlock if wrong id	✔
Should success claim with correct id and if enough time has passed	✔
Should success next_claim with correct id	✔
Should success until_unlock with correct id	✔
Should different periods correspond to different percentages	✔
Should claim returns the locked balance if staking has ended	✔
Should can stake an unlimited number of times	✔
Should different stakes have different id	✔
Should stake id is removed when the stake period ends	✔