Smart contract security audit

MoonFarmer

v.1.2

Table of Contents

1.0 Introduction **3**

1.1 Project engagement 3

1.2 Disclaimer 3

2.0 Coverage **4**

2.1 Target Code and Revision 4

2.2 Attacks made to the contract 5

3.0 Security Issues **7**

3.1 High severity issues - Solved [1] 7

3.2 Medium severity issues [0] 7

3.3 Low severity issues [1] 7

4.0 Owner Privileges **8**

5.0 Summary of the audit **8**

1.0 Introduction

1.1 Project engagement

During June of 2021, MoonFarmer engaged CTDSec to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. MoonFarmer provided CTDSec with access to their code repository and whitepaper.

1.2 Disclaimer

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 network’s fast-paced and rapidly changing environment, we at CTDSec recommend that MoonFarmer team put in place a bug bounty program to encourage further and active analysis of the smart contract.

2.0 Coverage

2.1 Target Code and Revision

For this audit, we performed research, investigation, and review of the MoonFarmer contract followed by issue reporting, along with mitigation and remediation instructions outlined in this report. The following code files are considered in-scope for the review:

Source:

MoonFarmer.sol [SHA256] -

df990d9cfe316a3d48e09fe2bb469cef6a91f8a74e6f33708a6a09ab90061348

2.2 Attacks made to the contract

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices.

№	Issue description.	Checking status
1	Compiler warnings.	PASSED
2	Race conditions and Reentrancy. Cross-function race conditions.	PASSED
3	Possible delays in data delivery.	PASSED
4	Oracle calls.	PASSED
5	Front running.	PASSED
6	Timestamp dependence.	PASSED
7	Integer Overflow and Underflow.	PASSED
8	DoS with Revert.	PASSED
9	DoS with block gas limit.	LOW ISSUES
10	Methods execution permissions.	PASSED
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	PASSED
12	The impact of the exchange rate on the logic.	PASSED
13	Private user data leaks.	PASSED
14	Malicious Event log.	PASSED
15	Scoping and Declarations.	PASSED
16	Uninitialized storage pointers.	PASSED
17	Arithmetic accuracy.	PASSED

18	Design Logic.	SOLVED BY DEV TEAM
19	Cross-function race conditions.	PASSED
20	Safe Zeppelin module.	PASSED
21	Fallback function security.	PASSED
22	Overpowered functions / Owner privileges	SOLVED BY DEV TEAM

3.0 Security Issues

3.1 High severity issues - Solved [1]

1. Wrong liquidity adding Issue:

The function swapAndLiquify() adds liquidity to tokens in the wrong proportion.

For 1 token it is three times smaller than the BNB amount.

Recommendation:

Recalculate to fit 1:1 proportion.

Dev team update: The function purpose is swap and Liquify divides the contract's token into 3 portions which are distributed among token holders and the ecosystem. Function is correctly applied.

3.2 Medium severity issues [0]

No medium severity issues found.

3.3 Low severity issues [1]

1. Out of gas Issue:

The function includeInReward() uses the loop to find and remove addresses from the _excluded list. Function will be aborted with OUT_OF_GAS exception if there will be a long excluded addresses list.

The function _getCurrentSupply also uses the loop for evaluating total supply. It also could be aborted with OUT_OF_GAS exception if there will be a long excluded addresses list.

Recommendation:

Check that the excluded array length is not too big.

4.0 Owner Privileges

Owner can change the tax and liquidity fee. [Fees are limited to max 11%].

Owner can change the maximum transaction amount.

Owner can be excluded from the fee.

Owner can be excluded from checking the max transaction amount.

Owner can reset some contract settings to default(activateContract).

Owner can change the reward cycle block.

Owner can change the marketing address.

Owner can disable and enable the reflection fee. [limited to 1% max].

Owner can withdraw tokens and BNBs.

Owner can change threshHoldTopUpRate.

Owner can lock and unlock. By the way, using these functions the owner could retake privileges even after the ownership was renounced. [This function is deleted now].

5.0 Summary of the audit

Smart contracts only contain low issues and it’s safe to be deployed.