22th December 2020

FURUCOMBO

SMART CONTRACT AUDIT REPORT

-

version v2.0

Smart Contract Security Audit and General Analysis

HAECHI AUDIT

Table of Contents

2 Issues (0 Critical, 0 Major, 2 Minor) Found

About HAECHI AUDIT

HAECHI AUDIT is a global leading smart contract security audit and development firm operated by HAECHI LABS. HAECHI AUDIT consists of professionals with years of experience in blockchain R&D and provides the most reliable smart contract security audit and development services.

So far, based on the HAECHI AUDIT's security audit report, our clients have been successfully listed on the global cryptocurrency exchanges such as Huobi, Upbit, OKEX, and others.

Our notable portfolios include SK Telecom, Ground X by Kakao, and Carry Protocol while HAECHI AUDIT has conducted security audits for the world's top projects and enterprises.

Trusted by the industry leaders, we have been incubated by Samsung Electronics and awarded the Ethereum Foundation Grants and Ethereum Community Fund.

Contact : audit@haechi.io
Website : audit.haechi.io

01. Introduction

This report was written to provide a security audit for the Furucombo smart contract. HAECHI AUDIT conducted the audit focusing on whether Furucombo smart contract is designed and implemented in accordance with publicly released information and whether it has any security vulnerabilities.

The issues found are classified as CRITICAL, MAJOR, MINOR or TIPS according to their severity.

CRITICAL	Critical issues are security vulnerabilities that MUST be addressed in order to prevent widespread and massive damage.
MAJOR	Major issues contain security vulnerabilities or have faulty implementation issues and need to be fixed.
MINOR	Minor issues are some potential risks that require some degree of modification.
TIPS	Tips could help improve the code's usability and efficiency

HAECHI AUDIT advises addressing all the issues found in this report.

02. Summary

The code used for the audit can be found at GitHub

1. https://github.com/dinngodev/compound-actions-contract
   commit hash : “d223d77b1eec1905ec8884e7a913b3aafa561cd0”
2. https://github.com/dinngodev/staking-adapter-contract
   commit hash : “61b1f0adf7e37d15b471c55f0c9735398a030b73”.

Issues

HAECHI AUDIT has 0 Critical Issues, 0 Major Issues, and 2 Minor Issue; also, we included 1 Tip category that would improve the usability and/or efficiency of the code.

Severity	Issue	Status
MINOR	StakingRewards/CurveRewards#notifyRewardAmount() does not check if it received reward	(Found - v1.1) (Out-of-Scope - v2.0)
MINOR	StakingRewards/CurveRewards#notifyRewardAmount() can decrease rewardRate	(Found - v1.1) (Out-of-Scope - v2.0)
TIPS	FCompoundActions does not have function to deposit ether when used without DSProxy or Proxy	(Found - v1.0) (Acknowledged - v2.0)

03. Overview

Contracts Subject to Audit

- Compound-Actions
    - Config.sol
    - Migrations.sol
    - Cache.sol
    - Registry.sol
    - Proxy.sol
    - LibCache.sol
    - DSAuth.sol
    - DSGuard.sol
    - DSGuardFactory.sol
    - FCompoundActions.sol
    - HSCompound.sol
    - HandlerBase.sol
- Staking-Adapter
    - StakingRewardsAdapterFactory.sol
    - StakingRewardsAdapter.sol
    - CurveRewards.sol
    - StakingRewards.sol
    - Owned.sol
    - RewardsDistributionRecipient.sol
    - SynthetixPausable.sol

04. Issues Found

MINOR : StakingRewards/CurveRewards#notifyRewardAmount() does not check if it received reward. (Found - v1.1)(Out-of-Scope - v2.0)

MINOR

```
function notifyRewardAmount(uint256 reward)
    external
    onlyRewardsDistribution
    updateReward(address(0))
{
    if (block.timestamp >= periodFinish) {
        rewardRate = reward.div(rewardsDuration);
    } else {
        uint256 remaining = periodFinish.sub(block.timestamp);
        uint256 leftover = remaining.mul(rewardRate);
        rewardRate = reward.add(leftover).div(rewardsDuration);
    }

    // Ensure the provided reward amount is not more than the balance in the contract.
    // This keeps the reward rate in the right range, preventing overflows due to
    // very high values of rewardRate in the earned and rewardsPerToken functions;
    // Reward + leftover must be less than 2^256 / 10^18 to avoid overflow.
    uint256 balance = rewardsToken.balanceOf(address(this));
    require(
        rewardRate <= balance.div(rewardsDuration),
        "Provided reward too high"
    );

    lastUpdateTime = block.timestamp;
    periodFinish = block.timestamp.add(rewardsDuration);
    emit RewardAdded(reward);
}
```

[StakingRewards.sol]

```
function notifyRewardAmount(uint256 reward)
    external
    onlyRewardDistribution
    updateReward(address(0))
{
    if (block.timestamp >= periodFinish) {
        rewardRate = reward.div(DURATION);
    } else {
        uint256 remaining = periodFinish.sub(block.timestamp);
        uint256 leftover = remaining.mul(rewardRate);
        rewardRate = reward.add(leftover).div(DURATION);
    }
    lastUpdateTime = block.timestamp;
    periodFinish = block.timestamp.add(DURATION);
    emit RewardAdded(reward);
}
```

[CurveRewards.sol]

Problem Statement

StakingRewards/CurveRewards#notifyRewardAmount() does not check if it has received the reward to distribute.

StakingRewards do check if rewardRate is too high but CurveRewards does not.

It can lead to a high reward rate for farmers who get rewards faster than others. And can make others unable to earn the rewards.

Since this function is designed to be only called by admin, this error can only be done by admin.

Recommendation

Receive reward token by transferFrom when function is called.

Update

[v2.0] - Furucombo team has confirmed that StakingRewards/CurveRewards contract is added for the unit test purpose.

MINOR : StakingRewards/CurveRewards#notifyRewardAmount() can decrease rewardRate (Found - v1.1)(Out-of-Scope - v2.0)

MINOR

Problem Statement

StakingRewards/CurveRewards#notifyRewardAmount() does not check if the rewardRate decreases after notification. Since it updates rate to be *(leftoverRate + notified reward)/duration* when previous reward is not finished, if admin keeps notifying with zero reward, it can lead to continuous decrease on reward rate,

Since this function is designed to be only called by admin, this error can only be done by admin.

Recommendation

Check if rewardRate increases after notifying reward.

Update

[v2.0] - Furucombo team has confirmed that StakingRewards/CurveRewards contract is added for the unit test purpose.

TIPS : FCompoundActions does not have function to deposit ether when used without DSProxy or Proxy (Found - v1.0)(Acknowledged - v2.0)

TIPS

Although FCompoundActions is intended to work as a delegated function logic instead of a standalone smart contract, it should be noted that FCompoundActions does not have any function to deposit Ether. Since the expected scenario of using the FCompoundActions is through proxies, it is not an issue that requires change.

Update

[v2.0] - Furucombo team has confirmed that this behavior is intended.

05. Disclaimer

This report is not an advice on investment, nor does it guarantee adequacy of a business model and/or a bug-free code. This report should be used only to discuss known technical problems. The code may include problems on Ethereum that are not included in this report. It will be necessary to resolve addressed issues and conduct thorough tests to ensure the safety of the smart contract.