30th APRIL 2021

SMART CONTRACT AUDIT REPORT

-

version v1.0

Smart Contract Security Audit and General Analysis

HAECHI AUDIT

Table of Contents

1 Issues (0 Critical, 0 Major, 1 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 BasketDAO smart contract. HAECHI AUDIT conducted the audit focusing on whether BasketDAO 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 (https://github.com/basketdao/protocol). The last commit for the code audited is at "d8e79cacb267e7704d3038b41030e7cabf12ad7c".

Issues

HAECHI AUDIT has found 1Minor issue and 1 TIPS to improve the code quality

Severity	Issue	Status
MINOR	Mismatch between specifications in comment and actual implementation in Logic#initialize()	(Found v1.0)
TIPS	Dead code in YieldFarmingV0#_getTokenToCToken()	(Found v1.0)

03. Overview

Contracts Subject to Audit

- Constants.sol
- IO.sol
- Logic.sol
- Storage.sol
- YieldFarmingV0.sol

04. Issues Found

Minor : Mismatch between specifications in comment and actual implementation in Logic#initialize() (Found - v.1.0)

MINOR

```
77.       // Governance can set governance OR market maker
78.        _setRoleAdmin(GOVERNANCE, GOVERNANCE_ADMIN);
79.        _setRoleAdmin(MARKET_MAKER, GOVERNANCE_ADMIN);
80.        _setupRole(GOVERNANCE_ADMIN, _governance);
81.        _setupRole(GOVERNANCE, _governance);
82.
83.        // Market maker admin
84.        _setRoleAdmin(MARKET_MAKER, MARKET_MAKER_ADMIN);
85.        for (uint256 i = 0; i < _marketMakers.length; i++) {
86.            _setupRole(MARKET_MAKER_ADMIN, _marketMakers[i]);
87.            _setupRole(MARKET_MAKER, _marketMakers[i]);
88.        }
89.
```

Problem Statement

The comment in `Logic.initialize()` states that `_governance` is authorized to set governance or market makers. But in the actual code, only market makers are given permission because `_setRoleAdmin()` can grant permission to only one address.

Recommendation

Replace Logic.sol:L79 with `_setupRole (MARKET_MAKER_ADMIN, _governance);`

TIPS : Dead code in YieldFarmingV0#_getTokenToCToken() (Found - v.1.0)

TIPS

```
64.  function toCToken(address _token) external {
65.      _requireAssetData(_token);
66.
67.      // Only doing UNI or COMP for CTokens
68.      require(_token == UNI || _token == COMP, "!valid-to-ctoken");
69.
70.      address _ctoken = _getTokenToCToken(_token);
```

```
198.  function _getTokenToCToken(address _token) internal pure returns (address) {
199.      if (_token == UNI) {
200.          return CUNI;
201.      }
202.      if (_token == COMP) {
203.          return CCOMP;
204.      }
205.      revert("!supported-token-to-ctoken");
206.  }
```

Problem Statement

`YieldFarmingV0._getTokenToCToken()` is used only in `YieldFarmingV0.toCToken()`, but it checks for non-UNI and non-COMP cases before calling `YieldFarmingV0._getTokenToCToken()` preventing it from reaching `revert("!supported-token-to-ctoken")`.

Recommendation

Please remove L68 to avoid duplicate code.

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.