HAECHI AUDIT

BasketDAO

Smart Contract

Security Audit Report

April 30th, 2021

Overview

Project Summary

Target	BasketDAO
Platform	Ethereum / Solidity
Codebase	Github Repositiory
Commit	9e8ff0ac862f08657172c143ef366d85add456fa

Audit Summary

Delivery Date	Apr. 30, 2021
Author	Jasper Lee
Version	1.0

Vulnerability Summary

Severety	**# of Findings**
Critical	0
Major	0
Medium	0
Minor	0
Infomational	2
Total	2

Key Findings

1. Mismatch between specifications in comment and actual implementation in `Logic.initialize()` Infomational

Description

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.

baskets-v0/Logic.sol:L77-L88

```
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          }
```

Recommendation

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

2. Dead code in `YieldFarmingV0._getTokenToCToken()` Infomational

Description

`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")` .

modules/YieldFarmingV0.sol:L64-70

```
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);
```

modules/YieldFarmingV0.sol:L198-L206

```
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     }
```

Recommendation

Please remove L68 to avoid duplicate code.

Appendix A - Files in Scope

File	SHA-1 Hash
baskets-v0/Constants.sol	c37cbf311091ca8cc5ca2c119bfb9937b45e1c52
baskets-v0/IO.sol	da82cab03bfd29c76e077aeca782370e9e6f63e0
baskets-v0/Logic.sol	f956ce31958e3f0a69dcb6e12b23165624c48ca5
baskets-v0/Storage.sol	03bd185be4115926b8566d2c8399ab396276702c
modules/YieldFarmingV0.sol	83da9cc5156b620e6fd0c6dba5be57269b93b433

Appendix B - Test Results

```
Logic
  #initialize()
    ✓ Should fail if INITIALIZED flag is on
    ✓ Should set up ERC20 token properly
    ✓ Should set assets properly
    ✓ Should set setup fees properly
    ✓ Should set setup roles properly
    ✓ Should be timelock only allowed to set the migrator
    ✓ Should be governance able to set governance
    ✓ Should set market maker roles properly
  #mintExact()
    ✓ Should fail when paused
    ✓ Should fail when _assets length differ from assets length
    ✓ Should fail when called by who is not migrator
    ✓ Should fail when _assets length differ from _amountsIn length
    ✓ Should fail when _amountsOut is smaller than 1,000,000
    ✓ Should mint properly
  #getOne()
    ✓ Should get the amount of assets properly
  #getFees()
    ✓ Should get fees properly
  #unpause()
    ✓ Should fail when unpaused already
    ✓ Should unpause properly
  #setFee()
    ✓ Should fail when called by who is not timelock
    ✓ Should fail when mint fee is bigger than fee divisor
    ✓ Should fail when burn fee is bigger than fee divisor
    ✓ Should fail when fee recipient address is 0x0
    ✓ Should set fees properly
  #setAssets()
    ✓ Should fail when called by who is not timelock
    ✓ Should fail when paused
    ✓ Should set assets properly
  #rescueERC20()
    ✓ Should fail when called by who is not market maker or governance
    ✓ Should fail when rescue target is listed
    ✓ Should rescue unlisted token properly
  #approveModule()
    ✓ Should fail when called by who is not timelock
    ✓ Should approve module properly
  #revokeModule()
    ✓ Should fail when called by who is not timelock
    ✓ Should revoke module properly
  #execute()
    ✓ Should fail when called by who is not governance or timelock
    ✓ Should fail when called unapproved module
  #mint()
    ✓ Should fail when paused
    ✓ Should fail when totalSupply is 0
    ✓ Should charge fee when called by who is not market maker
    ✓ Should mint properly
  #viewMint()
    ✓ Should preview the assets and amount required to mint properly
  #burn()
    ✓ Should fail when pause
    ✓ Should fail when totalSupply is 0
    ✓ Should fail when burn amount is less than 1e6
    ✓ Should charge fee when called by who is not market maker
```

```
    ✓ Should burn properly

YieldFarmingV0
  #enterMarkets()
    ✓ Should enter market properly
  #toCToken()
    ✓ Should fail if underlying token is not listed
    ✓ Should fail if underlying token is not UNI or COMP
    ✓ Should fail if underlying token balance is 0
    ✓ Should supplies assets to the Compound market properly
  #fromCToken()
    ✓ Should fail if underlying token is not listed
    ✓ Should fail if CToken is not CUNI or CCOMP
    ✓ Should fail if CToken balance is 0
    ✓ Should supplies assets to the Compound market properly
  #toATokenV1()
    ✓ Should fail if underlying token is not listed
    ✓ Should fail if underlying token is not UNI or COMP
    ✓ Should fail if underlying token balance is 0
    ✓ Should fail if fee recipient address is 0x0000000000000000000000000000000000000000
    ✓ Should supplies assets to the Aave market properly
  #fromATokenV1()
    ✓ Should fail if underlying token is not listed
    ✓ Should fail if underlying token balance is 0
    ✓ Should redeems assets from the Aave market properly
  #claimComp()
    ✓ Should fail if fee recipient address is 0x0000000000000000000000000000000000000000
    ✓ Should claims accrued COMP tokens
  #toXSushi()
    ✓ Should fail if underlying token balance is 0
    ✓ Should converts sushi to xsushi properly
  #fromXSushi()
    ✓ Should fail if underlying token balance is 0
    ✓ Should goes from xsushi to sushi properly
  #getClaimableComp()
    ✓ Should gets claimable comp for certain address properly
```

File	% Stmts	% Branch	% Funcs	% Lines
baskets-v0/Constants.sol	**100**	**100**	**100**	**100**
baskets-v0/IO.sol	**100**	**100**	**100**	**100**
baskets-v0/Logic.sol	**100**	**100**	**100**	**100**
baskets-v0/Storage.sol	**100**	**100**	**100**	**100**
modules/YieldFarmingV0.sol	**98.65**	**91.67**	**100**	**98.65**

Uncovered Lines

`require(ICToken(_ctoken).redeem(balance) == 0, "!ctoken-redeem");` in `YieldFarmingV0.fromCToken()`

It's hard to simulate CToken redeem() fails.

`require(ICToken(_ctoken).mint(balance) == 0, "!ctoken-mint");`

It's hard to simulate CToken mint() fails.

`revert("!supported-token-to-ctoken");`

It's dead code.