# Code Security Assessment

# Defiai

Jan 19th, 2022

# Table of Contents

**Summary**

**Overview**

- Project Summary
- Audit Summary
- Vulnerability Summary
- Audit Scope

**Findings**

- GLOBAL-01 : Centralization Risk
- GLOBAL-02 : Unlocked Compiler Version
- GLOBAL-03 : Missing Emit Events
- GLOBAL-04 : Function Visibility Optimization
- CCK-01 : Incorrect Error Message
- CCK-02 : Missing Input Validation
- CRC-01 : Missing Input Validation
- DCK-01 : Potential Sandwich Attacks
- DCK-02 : Missing Update `totalAllocPoint` in Function `Set()`
- DFA-01 : Delegation Not Moved Along With Token
- DFA-02 : Token Minted To Centralized Address
- DFI-01 : Incorrect Value Setting
- DFI-02 : Missing Input Validation
- DFI-03 : Transfer in Function `updatePool`
- DFS-01 : Potential Sandwich Attacks
- LMM-01 : Missing Input Validation
- VMC-01 : No `vestingToken` after the Lock Period
- VMC-02 : Missing Input Validation
- VMC-03 : Lack of Access Control

**Appendix**

**Disclaimer**

**About**

# Summary

This report has been prepared for Defiai to discover issues and vulnerabilities in the source code of the Defiai project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Additionally, this audit is based on a premise that all external contracts were implemented safely.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

# Overview

## Project Summary

| | |
|---|---|
| Project Name | Defiai |
| Platform | BSC |
| Language | Solidity |
| Codebase | https://github.com/DEFIAI2021/defiai/tree/main/contracts<br>https://github.com/DEFIAI2021/defiai/blob/main/contracts/base/Distributor.sol |
| Commit | 3e34837763a316b40cc7e3f44d2b1c0a1923433b<br>5a7e098303e8791361e5424a4db8ced9f3a6f6f7 |

## Audit Summary

| | |
|---|---|
| Delivery Date | Jan 19, 2022 |
| Audit Methodology | Static Analysis, Manual Review |

## Vulnerability Summary

| Vulnerability Level | Total | Pending | Declined | Acknowledged | Partially Resolved | Resolved |
|---|---|---|---|---|---|---|
| Critical | 0 | 0 | 0 | 0 | 0 | 0 |
| Major | 5 | 0 | 0 | 2 | 0 | 3 |
| Medium | 1 | 0 | 0 | 0 | 0 | 1 |
| Minor | 4 | 0 | 0 | 2 | 0 | 2 |
| Informational | 9 | 0 | 0 | 1 | 0 | 8 |
| Discussion | 0 | 0 | 0 | 0 | 0 | 0 |

# Audit Scope

| ID | File | SHA256 Checksum |
|---|---|---|
| DAO | contracts/base/DAOToken.sol | 436780e2072202770e1c409a9487ab20a18471a7bab4dad9799d9eca0015b522 |
| DFA | contracts/base/DeFiAiToken.sol | 920d7e064556d76d6fed3129b85607e09fe2588afe585a58e22c9d9e2b7c0753 |
| DCK | contracts/base/Distributor.sol | 3ccb6fd59ae22c7ac41ea453c6852307e3f3f88c7820b4f6f7a23dde65f8d40d |
| LMM | contracts/base/LiquidityMiningMaster.sol | ef66f04b42fd1a32807ca6dfabf188889c47e05471498ae9f03848740e741a11 |
| VMC | contracts/base/VestingMaster.sol | 956cde8c117c5e4f00c77cef70aac5fa5c97a690c175a16d168ef45c311a41c6 |
| VMO | contracts/base/VestingMasterOld.sol | 60ff6237052c2a4049eba0ad9aae42a27f483973982e0863e56b7d44c6913baf |
| ICC | contracts/interfaces/ICore.sol | ca2cd4d4aa0e6b6c2196f5e9de4be2e6e9ca2aa28e144d26d9b438c0a8fed93f |
| ICR | contracts/interfaces/ICoreRef.sol | 0dbfb637a224bcb5916d22a01dee280e629c1c6debaea91f8872b84ec2ddbcde |
| IDA | contracts/interfaces/IDAOToken.sol | 8fbe96d3fbb112eb700735aae489c421e6790ab33dc8285bf11abf5aa70d12a0 |
| IDF | contracts/interfaces/IDeFiAIFarm.sol | 9bb0529b67b37db1121d16f15ce3b08d13097f7776fb33e47a10ed6bf88c4bc6 |
| IDT | contracts/interfaces/IDeFiAiToken.sol | 626fa964d73bde197b66a6b83ca98be161d276ba83d7375cb913371a194fce0a |
| ILM | contracts/interfaces/ILiquidityMiningMaster.sol | edef8dbf0099ec3d0f5a63d65cf776c1bffeb503829746c4da3f53d99bf36a74 |
| IPF | contracts/interfaces/IPancakeswapFarm.sol | cf995333162df38b2a8f0ac719a2045bca4e0114f090e280acd0ba23f404ec8d |
| IPC | contracts/interfaces/IPermissions.sol | fdc721cc82ece17cfa8f1343b824dbcc5f8f1bed796aff936fed118914cb4af3 |

| ID | File | SHA256 Checksum |
| --- | --- | --- |
| ISC | contracts/interfaces/IStrategy.sol | f6c2d460e865d0d99abc9a36a8db75bab8807c29fc35b8fc96ed9489cb0f6819 |
| IUS | contracts/interfaces/IUSDMToken.sol | d1d328f2594fb5943762ab30cd6af3034c85579d1ead8731d67e670112bffef6 |
| IVM | contracts/interfaces/IVestingMaster.sol | 3048dee5dcbe0cd959ef86d6d05f40fe1328bde1349feb6e14700ac8f69f22ff |
| CCK | contracts/refs/Core.sol | 544b20def81473ec7dd3a8dbd0e15fb8f90029dcd19b4e5cc0b6e1258c0cad1a |
| CRC | contracts/refs/CoreRef.sol | 41e72ff4b6a1ab923ec9ee4bcb1f39b470a83df487e0a1bd29b9de238dc6a314 |
| PCK | contracts/refs/Permissions.sol | 6b626d92eb0c802a0d4ea423902af44e7c12fcba26ff35a46f2eb97cddcb0d9b |
| DFI | contracts/DeFiAIFarm.sol | 137e1b58acf4908d2af10ea3bb21b0833b70a0a980f95f048e0a3b3fc26a74e6 |
| DFS | contracts/DeFiAIStratX2.sol | a6da265c258e354175773469b23ca88fad1d67a7d89e1644021929983cbc061f5 |
| DFX | contracts/DeFiAIStratX2_PCS.sol | b3fca321a53ca6f066d6a4059b6a3d03f7aaa8cd79834a8122101f6d3e980ba1 |

# Understandings

## Overview

`DeFiAiToken` is decentralized finance (DeFi) token deployed on the Binance smart chain.

There are 2 types of mining pools in the `deifai` protocol. If the `vestingMaster` contract is set, part of the user's reward will be locked, and the locked part of the reward will be released linearly. The pool in `DeFiAIFarm` has a corresponding strategy contract.

## Privileged Functions

The contract contains the following privileged functions that are restricted by some modifiers. They are used to modify the contract configurations and address attributes. We grouped these functions below:

### The `onlyGuardianOrGovernor` modifier:

Contract `LiquidityMiningMaster`:

- addPool( uint256 _allocPoint, IERC20 _lpToken, bool _locked, bool _withUpdate )
- setPool( uint256 _pid, uint256 _allocPoint, bool _locked, bool _withUpdate )
- updateTokenPerBlock(uint256 _tokenPerBlock)
- updateEndBlock(uint256 _endBlock)

Contract `CoreRef`:

- pause()
- unpause()

Contract `DeFiAIFarm`:

- add(uint256 _allocPoint, IERC20 _want, bool _withUpdate, address _strat )
- set(uint256 _pid, uint256 _allocPoint, bool _withUpdate )
- updateTokenPerBlock(uint256 _tokenPerBlock)
- updateEndBlock(uint256 _endBlock)

### The `onlyGovernor` modifier:

Contract `LiquidityMiningMaster`:

- updateVestingMaster(address _vestingMaster)

Contract `Core`:

- setDeFiAISupportRatio(uint256 _DeFiAISupportRatio)
- setDeFiAI(address token)
- allocateDeFiAI(address to, uint256 amount)
- allocateToken( address _token, address to, uint256 amount )
- approveDeFiAI(address to, uint256 amount)
- approveToken( address _token, address to, uint256 amount )
- setApprovedPairAndContract(address _pair, address _contract)
- removeApprovedPairAndContract(address _pair, address _contract)

Contract `CoreRef`:

- setCore(address core_)

Contract `Permissions`:

- createRole(bytes32 role, bytes32 adminRole)
- grantMinter(address minter)
- grantBurner(address burner)
- grantPCVController(address pcvController)
- grantGovernor(address governor)
- grantGuardian(address guardian)
- revokeMinter(address minter)
- revokeBurner(address burner)
- revokePCVController(address pcvController)
- revokeGovernor(address governor)
- revokeGuardian(address guardian)
- revokeOverride(bytes32 role, address account)

Contract `DeFiAIFarm`:

- setVestingMaster(address _vestingMaster)
- setDevSupply(uint256 _devSupply)

Contract `DeFiAIStratX2`:

- setDevAddress(address _devAddress)
- setBuyBackRate(uint _buyBackRate)

## The `onlyFarms` modifier:

Contract `VestingMaster`:

- lock(address account, uint256 amount)

Contract `DeFiAIStratX2`:

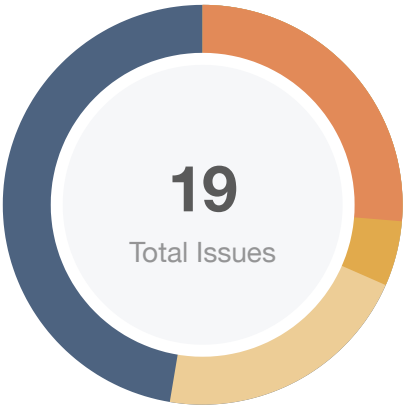
- deposit(uint256 _wantAmt)
- withdraw(uint256 _wantAmt)

## The `onlyGovernance` modifier:

Contract `Distributor`:

- function add(address _token, uint256 _allocPoint, address _pool, address[] calldata _wbnbToTokenPath)
- function set(uint256 pid, address _token, uint256 _allocPoint, address _pool, address[] calldata _wbnbToTokenPath)
- function allocate()
- function distribute()
- function buyBack()
- function setSlippageFactor(uint256 _slippageFactor)
- function setFloorRatio(uint256 _floorRatio)
- function setLmpShares(uint256 _lmpShares)
- function setOperatorFee(uint256 _operatorFee)
- function setWbnbToDefiAIPath(address[] memory _defiaiToWbnbPath)

# Findings




| | |
|---|---|
| **Critical** | **0** (0.00%) |
| **Major** | **5** (26.32%) |
| **Medium** | **1** (5.26%) |
| **Minor** | **4** (21.05%) |
| **Informational** | **9** (47.37%) |
| **Discussion** | **0** (0.00%) |

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| **GLOBAL-01** | Centralization Risk | **Centralization / Privilege** | **Major** | Acknowledged |
| GLOBAL-02 | Unlocked Compiler Version | Language Specific | Informational | Resolved |
| GLOBAL-03 | Missing Emit Events | Coding Style | Informational | Resolved |
| GLOBAL-04 | Function Visibility Optimization | Gas Optimization | Informational | Resolved |
| CCK-01 | Incorrect Error Message | Logical Issue | Informational | Resolved |
| CCK-02 | Missing Input Validation | Volatile Code | Informational | Resolved |
| CRC-01 | Missing Input Validation | Volatile Code | Informational | Resolved |
| DCK-01 | Potential Sandwich Attacks | Logical Issue | Minor | Acknowledged |
| DCK-02 | Missing Update `totalAllocPoint` in Function `Set()` | Logical Issue | Minor | Resolved |
| DFA-01 | Delegation Not Moved Along With Token | Logical Issue | Major | Resolved |
| **DFA-02** | Token Minted To Centralized Address | **Centralization / Privilege** | **Major** | Acknowledged |
| DFI-01 | Incorrect Value Setting | Logical Issue | Major | Resolved |
| DFI-02 | Missing Input Validation | Volatile Code | Informational | Resolved |
| DFI-03 | Transfer in Function `updatePool` | Logical Issue | Minor | Resolved |

| ID | Title | Category | Severity | Status |
| --- | --- | --- | --- | --- |
| DFS-01 | Potential Sandwich Attacks | Logical Issue | Minor | Acknowledged |
| LMM-01 | Missing Input Validation | Volatile Code | Informational | Acknowledged |
| VMC-01 | No `vestingToken` after the Lock Period | Logical Issue | Major | Resolved |
| VMC-02 | Missing Input Validation | Volatile Code | Informational | Resolved |
| VMC-03 | Lack of Access Control | Logical Issue | Medium | Resolved |

# GLOBAL-01 | Centralization Risk

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Centralization / Privilege | Major | Global | Acknowledged |

## Description

In the contract `LiquidityMiningMaster`, the role `guardian/governor` has the authority over the following function:

- addPool()
- setPool()
- updateTokenPerBlock()
- updateEndBlock()
- updateVestingMaster()

In the contract `VestingMaster`, the role `farm` has the authority over the following function:

- lock()

In the contract `Core`, the role `governor` has the authority over the following function:

- setDeFiAISupportRatio()
- setDeFiAI()
- allocateToken()
- approveDeFiAI()
- approveToken()
- setApprovedPairAndContract()
- removeApprovedPairAndContract()

In the contract `CoreRef`, the role `guardian/governor` has the authority over the following function:

- setCore()
- pause()
- unpause()

In the contract `Permissions`, the role `governor` has the authority over the following function:

- createRole()
- grantMinter()

- grantBurner()
- grantPCVController()
- grantGovernor()
- grantGuardian()
- revokeMinter()
- revokeBurner()
- revokePCVController()
- revokeGovernor()
- revokeGuardian()
- revokeOverride()

In the contract `DeFiAIFarm`, the role `guardian/governor` has the authority over the following function:

- setVestingMaster()
- setDevSupply()
- add()
- set()
- updateTokenPerBlock()
- updateEndBlock()

In the contract `DeFiAIStratX2`, the role `farm/governor` has the authority over the following function:

- deposit()
- withdraw()
- setDevAddress()
- setBuyBackRate()

In the contract `Distributor`, the role `governance` has the authority over the following function:

- function add(address _token, uint256 _allocPoint, address _pool, address[] calldata _wbnbToTokenPath)
- function set(uint256 pid, address _token, uint256 _allocPoint, address _pool, address[] calldata _wbnbToTokenPath)
- function allocate()
- function distribute()
- function buyBack()
- function setSlippageFactor(uint256 _slippageFactor)
- function setFloorRatio(uint256 _floorRatio)
- function setLmpShares(uint256 _lmpShares)
- function setOperatorFee(uint256 _operatorFee)

- function setWbnbToDefiAIPath(address[] memory _defiaiToWbnbPath)

Any compromise to these accounts may allow the hacker to manipulate the project through these functions.

## Recommendation

## Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

## Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

## Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles. OR
- Remove the risky functionality.

## Alleviation

[Team]:

1. We will transfer the ownership of the contract to a timelock contract and will implement DAO voting for changes in the future.
2. These contracts `Core`, `CoreRef` and `Permissions` have been removed.

# GLOBAL-02 | Unlocked Compiler Version

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Language Specific | Informational | Global | Resolved |

## Description

The following contracts have unlocked compiler versions. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to ambiguity when debugging as compiler specific bugs may occur in the codebase that would be difficult to identify over a span of multiple compiler versions rather than a specific one.

- token.sol

## Recommendation

We advise that the compiler version is alternatively locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.7.6` the contract should contain the following line:

```
pragma solidity 0.7.6;
```

## Alleviation

The development team resolved this issue in commit `23303576db0feb0cacc95d9f27d0b759bc0c577e`.

# GLOBAL-03 | Missing Emit Events

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Coding Style | Informational | Global | Resolved |

## Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

**contract `VestingMaster`**

- `setFarm()`

**contract `Core`**

- `setApprovedPairAndContract()`
- `removeApprovedPairAndContract()`

**contract `DeFiAIFarm`**

- `setVestingMaster()`
- `setDevSupply()`

**contract `DeFiAIStratX2`**

- `setBuyBackRate()`

## Recommendation

We advise the client to add events for sensitive actions, and emit them in the function.

## Alleviation

The development team resolved this issue in commit `23303576db0feb0cacc95d9f27d0b759bc0c577e`.

# GLOBAL-04 | Function Visibility Optimization

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Gas Optimization | Informational | Global | Resolved |

## Description

The following functions are declared as `public` and are not invoked in any of the contracts contained within the project's scope. The functions that are never called internally within the contract should have external visibility.

In the contract `LiquidityMiningMaster`:

- `addPool()` in L97
- `setPool()` in L128
- `deposit()` in L231
- `withdraw()` in L279
- `emergencyWithdraw()` in L327
- `updateTokenPerBlock()` in L368
- `updateEndBlock()` in L378
- `updateVestingMaster()` in L398

In the contract `VestingMaster`:

- `claim()` in L85

In the contract `Core`:

- `approveDeFiAI()` in L82
- `setApprovedPairAndContract()` in L119
- `removeApprovedPairAndContract()` in L137

In the contract `CoreRef`:

- `pause()` in L93
- `unpause()` in L98

In the contract `DeFiAIFarm`:

- `setVestingMaster()` in L87

- `setDevSupply()` in L91
- `add()` in L99
- `set()` in L133
- `deposit()` in L238
- `withdraw()` in L285
- `withdrawAll()` in L340
- `emergencyWithdraw()` in L344
- `updateTokenPerBlock()` in L381
- `updateEndBlock()` in L391

## Recommendation

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

## Alleviation

The development team resolved this issue in commit `23303576db0feb0cacc95d9f27d0b759bc0c577e`.

# CCK-01 | Incorrect Error Message

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Informational | contracts/refs/Core.sol: 144 | Resolved |

## Description

The error message in `require(approvedContractExisted[_pair][_contract], "Core::setApprovedPairAndContract: Not exist")` does not describe the error correctly.

## Recommendation

The message "Core::setApprovedPairAndContract: Not exist" can be changed to "Core::removeApprovedPairAndContract: Not exist" .

## Alleviation

The development team removed this contract in commit `23303576db0feb0cacc95d9f27d0b759bc0c577e`.

# CCK-02 | Missing Input Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Volatile Code | Informational | contracts/refs/Core.sol: 176 | Resolved |

## Description

The given input is missing the sanity check.

## Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below:
_setDeFiAI():

```
require(token != address(0), "token can not be zero address.");
```

## Alleviation

The development team removed this contract in commit `23303576db0feb0cacc95d9f27d0b759bc0c577e`.

# CRC-01 | Missing Input Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Volatile Code | Informational | contracts/refs/CoreRef.sol: 21, 87 | Resolved |

## Description

The given input is missing the sanity check.

## Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below:

constructor():

```
require(core_ != address(0), "core_ can not be zero address.");
```

setCore():

```
require(core_ != address(0), "core_ can not be zero address.");
```

## Alleviation

The development team removed this contract in commit `23303576db0feb0cacc95d9f27d0b759bc0c577e`.

# DCK-01 | Potential Sandwich Attacks

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Minor | contracts/base/Distributor.sol: 222~229 | Acknowledged |

## Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by backrunning (after the transaction being attacked) a transaction to sell the asset.

The following functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

- `_safeSwap()`
- `buyBack()`

## Recommendation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

## Alleviation

[Team]:Assured the functions swapping with 0 outputs are swapping intentionally with no regard to price.

# DCK-02 | Missing Update `totalAllocPoint` in Function `Set()`

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Minor | contracts/base/Distributor.sol: 167~171 | Resolved |

## Description

```
167         if (_poolInfo.allocPoint > _allocPoint) {
168             totalAllocPoint.sub(_poolInfo.allocPoint.sub(_allocPoint));
169         } else {
170             totalAllocPoint.add(_allocPoint.sub(_poolInfo.allocPoint));
171         }
```

When calling function `set()`, missing update `totalAllocPoint`.

## Recommendation

We advise the client to update `totalAllocPoint` as below:

```
        totalAllocPoint = totalAllocPoint.sub(_poolInfo.allocPoint).add(_allocPoint);
```

## Alleviation

The development team resolved this issue in commit `23303576db0feb0cacc95d9f27d0b759bc0c577e`.

# DFA-01 | Delegation Not Moved Along With Token

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Major | contracts/base/DeFiAiToken.sol: 53 | Resolved |

## Description

The voting power of delegation is not moved from token sender to token recipient along with the `constructor()`.

## Recommendation

We advise the client to consider moving delegation along with the constructor.

```
53 constructor(address _treasury, address _core) CoreRef(_core) {
54     _balances[msg.sender] = uint96(totalSupply);
55     // _balances[_treasury] = uint96(totalSupply);
56     _moveDelegates(address(0), _delegates[msg.sender], uint96(totalSupply));
57     emit Transfer(address(0), msg.senger, totalSupply);
58 }
```

## Alleviation

The development team changed the logic of the contract in commit `23303576db0feb0cacc95d9f27d0b759bc0c577e`.

# DFA-02 | Token Minted To Centralized Address

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| **Centralization / Privilege** | **Major** | contracts/base/DeFiAiToken.sol: 54 | Acknowledged |

## Description

The amount of `totalSupply` tokens that are minted to the centralized address `msg.sender` who is deployer, may raise the community's concerns about the centralization issue.

## Recommendation

We advise the client to carefully manage the deployer account's private key and avoid any potential risks of being hacked. We also advise the client to adopt Multisig, Timelock, and/or DAO in the project to manage this specific account in this case.

## Alleviation

[Team]:The minted tokens will be transferred immediately to the farm contract.

# DFI-01 | Incorrect Value Setting

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Major | contracts/DeFiAIFarm.sol: 323~324 | Resolved |

## Description

When re-setting `user.shares`, `realAmt` should be set to `user.shares` first.

## Recommendation

We advise the client to modify as bellow:

```
323  realAmt = user.shares;
324  user.shares = 0;
```

## Alleviation

The development team resolved this issue in commit `23303576db0feb0cacc95d9f27d0b759bc0c577e`.

# DFI-02 | Missing Input Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Volatile Code | Informational | contracts/DeFiAIFarm.sol: 99 | Resolved |

## Description

The given input is missing the sanity check.

## Recommendation

We advise adding the check for the passed-in values to prevent unexpected errors as below:

add():

```
require(_strat != address(0), "_strat can not be zero address.");
```

## Alleviation

The development team resolved this issue in commit `23303576db0feb0cacc95d9f27d0b759bc0c577e`.

## DFI-03 | Transfer in Function `updatePool`

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Minor | contracts/DeFiAIFarm.sol: 231 | Resolved |

## Description

Currently, `devSupply.div(FEE_DENOM.sub(devSupply))` is not less than 1, so the amount transferred to `devAddress` is greater than `tokenReward`. Could you please tell us more details about this?

## Recommendation

We advise the client to check the transfer logic.

## Alleviation

Logic error was fixed in commit `23303576db0feb0cacc95d9f27d0b759bc0c577e`, and the intended behavior of the function is to mint an additional percentage to the dev.

# DFS-01 | Potential Sandwich Attacks

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Minor | contracts/DeFiAIStratX2.sol: 432~439 | Acknowledged |

## Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by backrunning (after the transaction being attacked) a transaction to sell the asset.

The following functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

- `_safeSwap()`
- `buyBack()`

## Recommendation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

## Alleviation

[Team]:Assured the functions swapping with 0 outputs are swapping intentionally with no regard to price.

## LMM-01 | Missing Input Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Volatile Code | Informational | contracts/base/LiquidityMiningMaster.sol: 53 | Acknowledged |

### Description

The given input is missing the sanity check.

### Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below:

constructor():

```
require(_core != address(0), "_core can not be zero address.");
require(rewardToken != address(0), "rewardToken can not be zero address.");
```

### Alleviation

No alleviation.

# VMC-01 | No `vestingToken` after the Lock Period

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Major | contracts/base/VestingMaster.sol: 89, 113 | Resolved |

## Description

Currently, when the lock period is end, users can't claim their asset throw functions.

## Recommendation

We advise the client to change functions `_claim()` and `getVestingAmount()` as below:

```
function _claim(address account) internal {
    LockedRewardV2 storage lockedRewards = userLockedRewards[account];
    uint256 claimableAmount;
    uint256 totalLockedPeriod = period.mul(lockedPeriodAmount);
    if (block.timestamp < lockedRewards.start.add(totalLockedPeriod)) {
      uint256 diff = block.timestamp
          .sub(lockedRewards.lastClaimed)
          .div(period).mul(period);
      claimableAmount = lockedRewards.vesting
          .mul(diff)
          .div(totalLockedPeriod);
    } else {
      claimableAmount = lockedRewards.pending;
    }
    if (claimableAmount > totalLockedRewards) {
        claimableAmount = totalLockedRewards;
    }
    lockedRewards.pending = lockedRewards.pending.sub(claimableAmount);
    lockedRewards.lastClaimed = block.timestamp;
    totalLockedRewards = totalLockedRewards.sub(claimableAmount);
    if(claimableAmount > 0){
      vestingToken.safeTransfer(account, claimableAmount);
    }
    emit Claim(account, claimableAmount);
}

function getVestingAmount()
    public
    view
    override
    returns (uint256 lockedAmount, uint256 claimableAmount)
{
    LockedRewardV2 memory lockedRewards = userLockedRewards[msg.sender];
    uint256 totalLockedPeriod = period.mul(lockedPeriodAmount);
```

```
        if (block.timestamp < lockedRewards.start.add(totalLockedPeriod)) {
          uint256 diff = block.timestamp
              .sub(lockedRewards.lastClaimed)
              .div(period).mul(period);
          claimableAmount = lockedRewards.pending
              .mul(diff)
              .div(totalLockedPeriod);
        } else {
          claimableAmount = lockedRewards.pending;
        }
        if (claimableAmount > totalLockedRewards) {
            claimableAmount = totalLockedRewards;
        }
        lockedAmount = lockedRewards.pending.sub(claimableAmount);
    }
```

## Alleviation

The development team resolved this issue in commit `23303576db0feb0cacc95d9f27d0b759bc0c577e`.

## VMC-02 | Missing Input Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Volatile Code | Informational | contracts/base/VestingMaster.sol: 48, 136 | Resolved |

### Description

The given input is missing the sanity check.

### Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below:

constructor():

```
require(_governor != address(0), "_governor can not be zero address.");
```

setFarm():

```
require(_farmAddress != address(0), "_farmAddress can not be zero address.");
```

### Alleviation

The development team resolved this issue in commit `23303576db0feb0cacc95d9f27d0b759bc0c577e`.

## VMC-03 | Lack of Access Control

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Medium | contracts/base/VestingMaster.sol: 136 | Resolved |

### Description

The following functions can be called by anyone to update the sensitive stats of the contract:

- `setFarm()`

### Recommendation

We recommend adding proper access control to this function or checking the status of initialization in the deployment process.

### Alleviation

The development team resolved this issue in commit `23303576db0feb0cacc95d9f27d0b759bc0c577e`.

# Appendix

## Finding Categories

### Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

### Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

### Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

### Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

### Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

### Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

## Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

# Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

# About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.