HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: DEFIAI
Date: July 29th, 2022

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for DEFIAI
Approved By	Evgeniy Bezuglyi | SC Department Head at Hacken OU Noah Jelich | Senior Solidity SC Auditor at Hacken OU
Type	Farming
Platform	BSC
Network	Binance
Deployed Contract	https://bscscan.com/address/0x6548a320d3736920cad8a2cfbfefdb14db6376ea
Language	Solidity
Methods	Manual Review, Automated Review, Architecture Review
Website	https://dfai.app/
Timeline	25.05.2022 - 29.07.2022
Changelog	25.05.2022 - Initial Review 06.06.2022 - Second Review 21.06.2022 - Third Review 06.07.2022 - Fourth Review 18.07.2022 - Fifth Review 27.07.2022 - Sixth Review 29.07.2022 - Seventh Review

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by DEFIAI (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

Scope

The scope of the project is smart contracts in the repository:

Initial review scope

Repository:
https://github.com/DEFIAI2021/defiai-v2
Commit:
3eb1924
Technical Documentation: No
Integration and Unit Tests: No
Deployed Contracts Addresses: No
Contracts:
File: ./DeFiAIFarmV2.sol
SHA3: 9b1b8f0466723bd1ee49d1903324986d45dbfe21c850e394f4121bb69eb10dbf

File: ./interfaces/IDeFiAIMultiStrat.sol
SHA3: f9d4d11931e1f888b18a28aa617df05ec8285a5fa67022459b9d76aaa56fb5ab

File: ./interfaces/IDeFiAIStrat.sol
SHA3: 157ed02d02d04cd3c33fdf662fd9b1bcac38165d21b156a5094b339206f6e2b9

Second review scope

Repository:
https://github.com/DEFIAI2021/defiai-v2
Commit:
6d0441a
Technical Documentation: No
Integration and Unit Tests: No
Deployed Contracts Addresses: No
Contracts:
File: ./DeFiAIFarmV2.sol
SHA3: 73d6f3ba390783778ceea31dd1620ff9ef473c12b97cda259f973b117de96147

File: ./DeFiAIStableStrat.sol
SHA3: 2e85461614a8b5649c513c6ef11e56f00f84255ffd109c3b6fb0b07e7a2991d7

File: ./interfaces/IDeFiAIMultiStrat.sol
SHA3: 8046912d44fba3943d0a9ffaeb2f446b6480e2867b801bab173cb4a2bc942d74

File: ./interfaces/IDeFiAIStrat.sol
SHA3: 438d263593fd680476424070d927554a8b7666d4625aaceb89abff5068c7102b

Third review scope

Repository:
https://github.com/DEFIAI2021/defiai-v2
Commit:
cfe4eca
Technical Documentation: No

Integration and Unit Tests: Yes
Deployed Contracts Addresses: No
Contracts:
File: ./DeFiAIFarmV2.sol
SHA3: 91cfbf7983122e074e55d8d8454bbaeaaed56b38075394f623249755aed9fc9c

File: ./DeFiAIStableStrat.sol
SHA3: 76a62c7b8f5433685bedc9171e4e05aed685ba2b35fb770d26ac3d725b39fb25

File: ./DeFiAIStrat.sol
SHA3: a6a084b9eb8166ac9b866150f63e440b0c7a5482623ee64856f1d6b030e9c730

File: ./interfaces/IDeFiAIMultiStrat.sol
SHA3: f9d4d11931e1f888b18a28aa617df05ec8285a5fa67022459b9d76aaa56fb5ab

File: ./interfaces/IDeFiAIStrat.sol
SHA3: 6fcc69d9c1a5fbcf9e0638bf4c8f6c3b2a0e8a35a09097358877a6f5a4e81b36

Fourth review scope

Repository:
https://github.com/DEFIAI2021/defiai-v2
Commit:
06dd601
Technical Documentation:
Type: Whitepaper (partial functional requirements provided)
https://dfai.app/documentation/DEFIAI-whitepaper-1.0-en.pdf

Type: Technical description
https://github.com/DEFIAI2021/defiai-v2/blob/master/README.md
Integration and Unit Tests: Yes
Deployed Contracts Addresses: No
Contracts:
File: ./contracts/solc_0.8/defiai/DeFiAIFarmV2.sol
SHA3: 270db24c3f9154d67b6bac4f2c95869ab6b3e18e9cf29bfc0babd9ae836e3b1d

File: ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol
SHA3: e007056074b2aefda30290772f2fded6667f12441020129672f3192b78400a3d

File: ./contracts/solc_0.8/defiai/interfaces/IDeFiAIMultiStrat.sol
SHA3: d6fe45659041ff3ac40330634f44c619847512be955db5f6dfd870c15fd9908c

File: ./contracts/solc_0.8/defiai/interfaces/IDeFiAIStrat.sol
SHA3: 2f3344b82c41d8b35b7c9f7a4bac20545b66bd8b4cc930fdb3d97bb34bbc836b

Fifth review scope

Repository:
https://github.com/DEFIAI2021/defiai-v2
Commit:
f4aab82
Technical Documentation:
Type: Whitepaper (partial functional requirements provided)
https://dfai.app/documentation/DEFIAI-whitepaper-1.0-en.pdf

Type: Technical description
https://github.com/DEFIAI2021/defiai-v2/blob/master/README.md
Integration and Unit Tests: Yes
Deployed Contracts Addresses:
https://bscscan.com/address/0x6548a320d3736920cad8a2cfbfefdb14db6376ea

Contracts:
File: ./contracts/solc_0.8/defiai/DeFiAIFarmV2.sol
SHA3: 43bc39931e1d1578b893e399655e2a378de188c7a03d1604d759c78ba3b9ff5a

File: ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol
SHA3: 5af9994bf6499b9f77137c1e2322f0688405e7de53fc4d7abb505bd253e63e64

File: ./contracts/solc_0.8/defiai/interfaces/IDeFiAIMultiStrat.sol
SHA3: d6fe45659041ff3ac40330634f44c619847512be955db5f6dfd870c15fd9908c

File: ./contracts/solc_0.8/defiai/interfaces/IDeFiAIStrat.sol
SHA3: 2f3344b82c41d8b35b7c9f7a4bac20545b66bd8b4cc930fdb3d97bb34bbc836b

Sixth review scope

Repository:
https://github.com/DEFIAI2021/defiai-v2
Commit:
5066229
Technical Documentation:
Type: Whitepaper (partial functional requirements provided)
https://dfai.app/documentation/DEFIAI-whitepaper-1.0-en.pdf

Type: Technical description
https://github.com/DEFIAI2021/defiai-v2/blob/master/README.md
Integration and Unit Tests: Yes
Deployed Contracts Addresses:
https://bscscan.com/address/0x6548a320d3736920cad8a2cfbfefdb14db6376ea
Contracts:
File: ./contracts/solc_0.8/defiai/DeFiAIFarmV2.sol
SHA3: 1c804454e55117c4adc361d7043b206e2ab72c64edcfc0392fb517594387416d

File: ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol
SHA3: 8b8075242626ec3f30b73ec26f88b6ecac75f4d74dcf37870093cb88749153f3

File: ./contracts/solc_0.8/defiai/interfaces/IDeFiAIStableStrat.sol
SHA3: a08dbe0c139882c22eb6988179866ede08af0115515ae9adabf8b565ed836f28

File: ./contracts/solc_0.8/defiai/interfaces/IDeFiAIStrat.sol
SHA3: 2b8321b824b084ef79a102cc72b4d2c791119f81d6129e9f68eed86b6e1e8bfe

Seventh review scope

Repository:
https://github.com/DEFIAI2021/defiai-v2
Commit:
4a98f13
Technical Documentation:
Type: Whitepaper (partial functional requirements provided)
https://dfai.app/documentation/DEFIAI-whitepaper-1.0-en.pdf

Type: Technical description
https://github.com/DEFIAI2021/defiai-v2/blob/master/README.md
Integration and Unit Tests: No
Deployed Contracts Addresses:
https://bscscan.com/address/0x6548a320d3736920cad8a2cfbfefdb14db6376ea
Contracts:
File: ./contracts/solc_0.8/defiai/DeFiAIFarmV2.sol
SHA3: 1c804454e55117c4adc361d7043b206e2ab72c64edcfc0392fb517594387416d

```
File: ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol
SHA3: 8b8075242626ec3f30b73ec26f88b6ecac75f4d74dcf37870093cb88749153f3

File: ./contracts/solc_0.8/defiai/interfaces/IDeFiAIStableStrat.sol
SHA3: a08dbe0c139882c22eb6988179866ede08af0115515ae9adabf8b565ed836f28

File: ./contracts/solc_0.8/defiai/interfaces/IDeFiAIStrat.sol
SHA3: 2b8321b824b084ef79a102cc72b4d2c791119f81d6129e9f68eed86b6e1e8bfe
```

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions.
Medium	Medium-level vulnerabilities are important to fix; however, they cannot lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

The score measurement details can be found in the corresponding section of the methodology.

Documentation quality

The Customer provided superficial technical and functional requirements. The total Documentation Quality score is **5** out of **10**.

Code quality

The total CodeQuality score is **2.5** out of **10**. The code partially follows official language style guides, and tests are not functioning.

Architecture quality

The architecture quality score is **10** out of **10**. Clean and clear architecture.

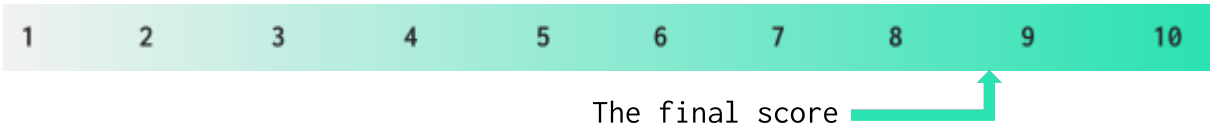
Security score

As a result of the audit, the code contains **1** low severity issue. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **8.75**.



Checked Items

We have audited provided smart contracts for commonly known and more specific vulnerabilities. Here are some of the items that are considered:

Item	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Not Relevant
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Passed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Not Relevant
Access Control & Authorization	CWE-284	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	Passed
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Not Relevant
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Not Relevant
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless it is required.	Passed
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed
Authorization through	SWC-115	tx.origin should not be used for authorization.	Passed

tx.origin			
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Passed
Signature Unique Id	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifier should always be used. All parameters from the signature should be used in signer recovery	Passed
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	Passed
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed
Calls Only to Trusted Addresses	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of unused variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Passed
EIP standards violation	EIP	EIP standards should not be violated.	Not Relevant
Assets integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions.	Passed
User Balances manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed
Flashloan Attack	**Custom**	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Passed
Token Supply manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the customer.	Not Relevant
Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There	Passed

		should not be any cases when execution fails due to the block Gas limit.	
Style guide violation	**Custom**	Style guides and best practices should be followed.	Failed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	**Custom**	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Failed
Secure Oracles Usage	**Custom**	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Not Relevant
Tests Coverage	**Custom**	The code should be covered with unit tests. Test coverage should be 100%, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Failed
Stable Imports	**Custom**	The code should not reference draft contracts, that may be changed in the future.	Passed

System Overview

DeFiAI V2 is a pool-system, which allows users to deposit tokens into the specified pool and withdraw after. The system collects fees for withdrawal. It has the following contracts:

- *DeFiAIFarmV2* – contract responsible for transferring funds between users and *DeFiAIStableStrat*.
- *DeFiAIStableStrat* – contract responsible for depositing and withdrawing tokens into specified DEX platforms to farm. The contract has 3 different farming strategies, which can be set once after deployment. When a user withdrawals or deposits their fund, it sends 30 percent of the user's accumulated reward to the developer address.

Privileged roles

DeFiAIFarmV2 contract has 1 privileged role, which is governance. Governance is used to represent owner and developer addresses.

- The governance can:
  - initialize contract.

DeFiAIStableStrat contract has 2 privileged roles, which are farm and governance. Governance is used to represent owner and developer addresses.

- The farm can:
  - deposit to the specified DEX platform.
  - withdraw from the specified DEX platform.
- The governance can:
  - initialize contract.
  - change the strategy of the contract.
  - change developer address.
  - withdraw missend tokens.

Findings

Critical

1. **Time-independent accumulated reward distribution.**

   withdraw function of DefiAIStableStrat calculates the deserved reward using the deposited balance but not considering the deposit time.

   This issue leads to unfair reward distribution.

   File: ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol

   Contract: DeFiAIStableStrat

   Function: withdraw

   Recommendation: Apply a logic where the accumulated reward will be distributed depending on both the passed time and the amount of tokens deposited.

   Status: Fixed (Revised commit: f4aab82)

High

1. **Dev address gets all rewards during strategy change.**

   In the reward distribution logic, the dev address takes 30 percent of the reward and transfers the remaining to the user who withdrew its funds, depending on the deposit amount. However, in case of a strategy change, the dev address takes all accumulated rewards of all users.

   File: ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol

   Contract: DeFiAIStableStrat

   Function: changeActiveStrategy

   Recommendation: Do not transfer rewards to the dev address in the changeActiveStrategy function. Instead, let users collect their rewards after strategy change and take the developer fee during collection.

   Status: Fixed (Revised commit: 06dd601)

Medium

No medium severity issues were found.

Low

1. **Style guide violation.**

   The contract does not follow the Solidity code style guide.

   File: ./contracts/solc_0.8/defiai/DeFiAIFarmV2.sol

   Contract: DeFiAIFarmV2

Recommendation: Follow the official Solidity code style guide.

Status: Mitigated

2. **Unused variables.**

   FEE_DENOM variable of DefiAIFarmV2 and *SLIPPAGE_FACTOR_MAX* variable of DeFIStableStrat are defined but never used.

   Files: ./contracts/solc_0.8/defiai/DeFiAIFarmV2.sol, ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol

   Contracts: DeFiAIFarmV2, DeFiAIStableStrat

   Recommendation: Remove mentioned variables.

   Status: Fixed (Revised commit: 9760015)

3. **Unused events.**

   Events *UpdateWithdrawalFee, UpdateMinWithdrawalFee* are defined but never used.

   File: ./contracts/solc_0.8/defiai/DeFiAIFarmV2.sol

   Contract: DeFiAIFarmV2

   Recommendation: Remove these events.

   Status: Fixed (Revised commit: 6d0441a)

4. **Redundant modifier.**

   Modifier *nonReentrant* is redundant as long as no Ether is sent.

   File: ./contracts/solc_0.8/defiai/DeFiAIFarmV2.sol

   Contract: DeFiAIFarmV2

   Functions: withdraw, deposit

   Recommendation: Remove this modifier.

   Status: Fixed (Revised commit: 6d0441a)

5. **Redundant if statement.**

   Specified functions do nothing if *_wantAmt* is zero, only emit events.

   File: ./contracts/solc_0.8/defiai/DeFiAIFarmV2.sol

   Contract: DeFiAIFarmV2

   Functions: deposit, withdraw

   Recommendation: Instead of *if* condition, use *require* statement at the beginning of the function to check *_wantAmt.*

   Status: Fixed (Revised commit: cfe4eca)

6. **Missing zero address validation.**

   Address parameters are being used without checking against the possibility of 0x0.

   This can lead to unwanted external calls to 0x0.

   Files: ./contracts/solc_0.8/defiai/DeFiAIFarmV2.sol, DeFiAIStableStrat.sol

   Contracts: DeFiAIFarmV2, DeFiAIStableStrat

   Functions: constructor, initialize

   Recommendation: Implement zero address checks.

   Status: Fixed (Revised commit: cfe4eca)

7. **Missing return value check.**

   deposit function of the *pool.strat* contract is called in the *deposit* function, but the return value was never used. *removeLiquidity* and *addLiquidity* functions of Router contracts have return statements. However, they are ignored in the DeFiAIStableStrat contract.

   Files: ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol, ./contracts/solc_0.8/defiai/DeFiAIFarmV2.sol

   Contracts: DeFiAIStableStrat, DeFiAIFarmV2

   Functions: _converLpToWant, _convertWantToLp, changeActiveStrategy, deposit

   Recommendation: Add return value checks.

   Status: Reported

8. **No check for allowance before transfer.**

   safeTransferFrom function requires allowance to transfer tokens. If the allowance is insufficient, it throws an error. However, the safeTransferFrom function is called in the deposit function without checking the allowance before.

   File: ./contracts/solc_0.8/defiai/DeFiAIFarmV2.sol

   Contract: DeFiAIFarmV2

   Function: deposit

   Recommendation: Check for allowance before transfer.

   Status: Fixed (Revised commit: cfe4eca)

9. **Public functions can be external.**

   getPoolInfo function is never called in the DeFiAIFarmV2, and *balances* is never called in the DeFiAIStableStrat. In order to save

Gas, public functions that are never called in the contract should be declared as external.

Files: ./contracts/solc_0.8/defiai/DeFiAIFarmV2.sol, ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol

Contracts: DeFiAIFarmV2, DeFiAIStableStrat

Functions: getPoolInfo, balances

Recommendation: Use the external attribute for functions never called from the contract.

Status: Fixed (Revised commit: 9760015)

10. Variables can be declared immutable.

No setter functions for *withdrawalFee* and *devAddress* state variables in the DeFiAIFarmV2 and *swapRouterAddress, busd, usdt, defiaiFarmAddress, withdrawalMultiplier,* and *stratAddress* state variables in the DeFiAIStableStrat. They can be declared immutable.

Immutable variables decrease storage costs.

Files: ./contracts/solc_0.8/defiai/DeFiAIFarmV2.sol, ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol

Contracts: DeFiAIFarmV2, DeFiAIStableStrat

Recommendation: Change variables to immutable.

Status: Fixed (Revised commit: 06dd601)

11. Redundant variable.

realAmt variable in the *withdraw* function is redundant.

File: ./contracts/solc_0.8/defiai/DeFiAIFarmV2.sol

Contract: DeFiAIFarmV2

Function: withdraw

Recommendation: Change *realAmt* with *_wantAmt*.

Status: Fixed (Revised commit: cfe4eca)

12. Usage of “transfer” instead of “safeTransfer”.

SafeERC20 contract is imported in DeFiAIStableStrat contract; however, transfer method is used instead of safeTransfer.

File: ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol

Contract: DeFiAIStableStrat

Functions: _collect, changeActiveStrategy

Recommendation: Use safeTransfer instead of transfer, or implement a return value check for transfers.

Status: Fixed (Revised commit: cfe4eca)

13. **Missing balance check.**

In the withdraw function of DeFiAIStableStrat contract, the user's balance is reduced as _wantAmt without checking against the insufficient balance.

File: ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol

Contract: DeFiAIStableStrat

Function: withdraw

Recommendation: Add a "require" statement, which checks the user's balance before reducing the balance.

Status: Fixed (Revised commit: cfe4eca)

14. **Missing array length check.**

_pcsAddresses, _mdexAddresses, _bswAddresses input parameters' lengths should be equal to each other, and 4.

File: ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol

Contract: DeFiAIStableStrat

Function: constructor

Recommendation: Add event emitting.

Status: Fixed (Revised commit: cfe4eca)

15. **Missing event emitting.**

Contracts should emit events in the critical state changes.

File: ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol

Contract: DeFiAIStableStrat

Functions: setDevAddress, changeActiveStrategy

Recommendation: Create and emit related events.

Status: Fixed (Revised commit: cfe4eca)

16. **Missing return value.**

In the DeFiAIMultiStrat contract, the _safeStableSwap function is specified as "returns uint256", but the function does not return anything.

File: ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol

Contract: DeFiAIStableStrat

Function: _safeStableSwap

Recommendation: Remove "returns" statement.

Status: Fixed (Revised commit: cfe4eca)

17. **Incorrect usage of a memory pointer.**

In the DeFiAIStableStrat contract init function, the pointer of _pcsAddresses, _mdexAddresses, and _bswAddresses are memory, but it should be calldata.

File: ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol

Contract: DeFiAIStableStrat

Function: init

Recommendation: Change memory pointers to calldata.

Status: Fixed (Revised commit: cfe4eca)

18. **Missing override for “balances” function.**

DeFiAIStableStrat contract inherits IDeFiAIMultiStrat contract however does not override its balances function.

This issue causes “Missing Implementation Error” during compilation.

File: ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol

Contract: DeFiAIStableStrat

Recommendation: Add override keyword to the balances mapping.

Status: Fixed (Revised commit: cfe4eca)

19. **Unused libraries.**

ReentrancyGuard, Pausable, and console libraries in DeFiAIStableStrat are unused.

File: ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol

Contract: DeFiAIStableStrat

Recommendation: Remove unused libraries.

Status: Fixed (Revised commit: 9760015)

20. **Redundant call.**

farmInfo[activePid].totalShare is assigned to the poolShare variable. However, after being assigned *farmInfo[activePid].totalShare* is called again in the if statement instead of using *poolShare*.

File: ./contracts/solc_0.8/defiai/DeFiAIStableStrat.sol

Contract: DeFiAIStableStrat

Recommendation: Use *poolshare* variable instead of calling *farmInfo[activePid].totalShare*

Status: Fixed (Revised commit: 9760015)

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on the security of the code. It also cannot be considered a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit cannot guarantee the explicit security of the audited smart contracts.