HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: HedgeFarm
Date: April 11, 2023

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 HedgeFarm
Approved By	Noah Jelich | Lead SC Auditor at Hacken OU
Type	Yield Farming
Platform	EVM
Language	Solidity
Methodology	Link
Website	https://hedgefarm.finance/smart-farmooor
Changelog	22.03.2023 - Initial Review 11.04.2023 - Second 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 HedgeFarm (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 includes review and security analysis of the following smart contracts from the provided repository:

Initial review scope

Repository	https://github.com/HedgeFarm/smart-farmooor-contracts
Commit	4706853dff0f2ca317f3c7c4b9a3bebb5e3df77c
Whitepaper	
Functional Requirements	https://docs.hedgefarm.finance/products/alpha-2-the-smart-farmooor
Technical Requirements	https://github.com/HedgeFarm/smart-farmooor-contracts/blob/master/docs/src/SUMMARY.md
Contracts	File: ./contracts/common/library/Modifier.sol SHA3: 1430861a06e79358d9fbd011caf907fa4db259b140e2d2456d7bc59a329a9907 File: ./contracts/common/Rescuable.sol SHA3: 97b54b33473f2af03021aa22ee5268084d1fb32599944d5c9fb090b1d882d861 File: ./contracts/common/RoleManagement.sol SHA3: f9d89cf3bd058c32c8acfd4fa634c0aedbbd17ad8ac7f0420a7f8d148c9c9519 File: ./contracts/dex/interface/IDex.sol SHA3: f3f4a107eebd3214e8211bbc26b8abc3fbb6d39ddbf6e10a7ce0de369ab8980a File: ./contracts/dex/interface/traderjoe/IJoeRouter01.sol SHA3: 9b0c6a417e948c8c8cc6d65b1d83f03fb072cb340a058c5877147c3a92b3e5b3 File: ./contracts/dex/interface/traderjoe/IJoeRouter02.sol SHA3: 8217d0acf0a9ec757020547ed7d47390b19f7503ed930abcc7e25a174d6118cd File: ./contracts/dex/TraderJoeDexModule.sol SHA3: 5f8cf9a3ac44e2320ef68968e383bb663b620f143513ca7de2c84523cc2f4f2a File: ./contracts/yield/interface/aave/IPool.sol SHA3: 01d297b8481bf2e0761f5a050afd97a7a6874033a1261b5f6b7dfb7a8574df8c File: ./contracts/yield/interface/aave/IPoolDataProvider.sol SHA3: f4c4aa8d7b5100415b10a6e9df3b8c9242214beb820a7109fa1510606380071f File: ./contracts/yield/interface/aave/IRewardsController.sol SHA3: 8f733113b42f2dcc9b07cd054944f6c17344b1f8b28e4b2b2c657986bc7e42d1 File: ./contracts/yield/interface/aave/IScaledBalanceToken.sol SHA3: 96e0647077a2ac3724878019929c1bc817db1466735fa39c3a85555e81946663

	File: ./contracts/yield/interface/aave/WadRayMath.sol SHA3: 450aa8d4d060de6f57c44cbfd8c56c8f63ef4b4a525170aec1d695faad7d78ea File: ./contracts/yield/interface/compoundV2/ComptrollerInterface.sol SHA3: 43b43014de3b0b22b25e73b2772693aad6d813c527582e3d4af49dfdf06dfd28 File: ./contracts/yield/interface/compoundV2/CTokenInterface.sol SHA3: de5e708acd3c8245a926d7018f5968d1c5968ba1aa042af4b2ecba882919dda3 File: ./contracts/yield/interface/compoundV2/ExponentialNoError.sol SHA3: 20b79b00a54d37ee6ce857bd68769d476c9d7d80b89025316a6577c9ab1b4065 File: ./contracts/yield/interface/INativeGateway.sol SHA3: 4acd43db0befcc6fdcb5c3dc8954a09bd5a16bd94bcc18d7be1d46b5237d5b6f File: ./contracts/yield/interface/ISmartFarmooor.sol SHA3: 420004090cee2218fd92304ef7bc1377d1db47c51c98480108c22c3b86839cd5 File: ./contracts/yield/interface/IYieldModule.sol SHA3: 42b131c517f3c34a1fee629bb6b49bb88474528920a423b61553bddff4f7f448 File: ./contracts/yield/interface/nativeasset/INativeWrapper.sol SHA3: 5b309a65e242fde28f987b6774ab3b6fd09f6ce5631d6572b52d3b0c6a2f3f45 File: ./contracts/yield/interface/stargate/ILpStaking.sol SHA3: d5100204d82780226bf2effc140b1034909bccd8077db49f3108339dfc09db91 File: ./contracts/yield/interface/stargate/IPool.sol SHA3: d358eba72cb606b1f2cb26bf4694a5fff5360b88b6ca02a72002b777761c50d3 File: ./contracts/yield/interface/stargate/IStargateRouter.sol SHA3: 99bd147e8706eb4e413ec58d9eff5cabea6bb624376191f84c6319481407d7ee File: ./contracts/yield/module/AaveYieldModule.sol SHA3: aa48a7a54d9793735129ba6666d1bedc200de8afe92b4b43c1914912e7c4cb63 File: ./contracts/yield/module/BaseModule.sol SHA3: dc68a40b220f8152998be9a75d8462e9c8231fdc8456286a2517d6a5ab66481a File: ./contracts/yield/module/CompoundV2Module.sol SHA3: 65345658f89b0f84a6ee22e6622f4c5e38b4f2748725ea499dd0dcf80537ab05 File: ./contracts/yield/module/StargateYieldModule.sol SHA3: a9ef504c371474d414adf64081cc1b4b17bb32c1a058687764f5465e38c94f6f File: ./contracts/yield/NativeGateway.sol SHA3: 9549493d88c1979aaa42d798de6707a178bec7491f8d83dc4010debe40f19841 File: ./contracts/yield/SmartFarmooor.sol SHA3: 7a181ffb174c1934643153cc63a03b906d1f10f7aa4ec9d851b2c51174134039

Second review scope

Repository	https://github.com/HedgeFarm/smart-farmooor-contracts
Commit	da2e6465c12a5bcbcb524abb85850044bb86db88
Whitepaper	

Functional Requirements	https://docs.hedgefarm.finance/products/alpha-2-the-smart-farmooor
Technical Requirements	https://github.com/HedgeFarm/smart-farmooor-contracts/blob/master/docs/src/SUMMARY.md
Contracts	File: common/Rescuable.sol SHA3: 79b4f013780bf75969e626deeb0fef5e7f7eb5d8abf34074eb827a6323d27550 File: common/RoleManagement.sol SHA3: f9d89cf3bd058c32c8acfd4fa634c0aedbbd17ad8ac7f0420a7f8d148c9c9519 File: common/library/Modifier.sol SHA3: 1430861a06e79358d9fbd011caf907fa4db259b140e2d2456d7bc59a329a9907 File: dex/TraderJoeDexModule.sol SHA3: d2fe5715bb5a187ae0ec67e37006ea9fe3a59c4d92bc39f2d2ba476b436ef2d8 File: dex/interface/IDex.sol SHA3: f3f4a107eebd3214e8211bbc26b8abc3fbb6d39ddbf6e10a7ce0de369ab8980a File: dex/interface/traderjoe/IJoeRouter01.sol SHA3: 9b0c6a417e948c8c8cc6d65b1d83f03fb072cb340a058c5877147c3a92b3e5b3 File: dex/interface/traderjoe/IJoeRouter02.sol SHA3: 8217d0acf0a9ec757020547ed7d47390b19f7503ed930abcc7e25a174d6118cd File: yield/NativeGateway.sol SHA3: 9549493d88c1979aaa42d798de6707a178bec7491f8d83dc4010debe40f19841 File: yield/SmartFarmooor.sol SHA3: 8fd793fb33b9403aff67ad366d014dd6df02f3edae01d8ce4f1161865c143dee File: yield/interface/INativeGateway.sol SHA3: 4acd43db0befcc6fdcb5c3dc8954a09bd5a16bd94bcc18d7be1d46b5237d5b6f File: yield/interface/ISmartFarmooor.sol SHA3: 420004090cee2218fd92304ef7bc1377d1db47c51c98480108c22c3b86839cd5 File: yield/interface/IYieldModule.sol SHA3: 42b131c517f3c34a1fee629bb6b49bb88474528920a423b61553bddff4f7f448 File: yield/interface/aave/IPool.sol SHA3: 01d297b8481bf2e0761f5a050afd97a7a6874033a1261b5f6b7dfb7a8574df8c File: yield/interface/aave/IPoolDataProvider.sol SHA3: f4c4aa8d7b5100415b10a6e9df3b8c9242214beb820a7109fa1510606380071f File: yield/interface/aave/IRewardsController.sol SHA3: 8f733113b42f2dcc9b07cd054944f6c17344b1f8b28e4b2b2c657986bc7e42d1 File: yield/interface/aave/IScaledBalanceToken.sol SHA3: 96e0647077a2ac3724878019929c1bc817db1466735fa39c3a85555e81946663 File: yield/interface/aave/WadRayMath.sol SHA3: 450aa8d4d060de6f57c44cbfd8c56c8f63ef4b4a525170aec1d695faad7d78ea File: yield/interface/compoundV2/ComptrollerInterface.sol SHA3: 43b43014de3b0b22b25e73b2772693aad6d813c527582e3d4af49dfdf06dfd28 File: yield/interface/compoundV2/CTokenInterface.sol SHA3: de5e708acd3c8245a926d7018f5968d1c5968ba1aa042af4b2ecba882919dda3

	File: yield/interface/compoundV2/ExponentialNoError.sol SHA3: 20b79b00a54d37ee6ce857bd68769d476c9d7d80b89025316a6577c9ab1b4065 File: yield/interface/nativeasset/INativeWrapper.sol SHA3: 5b309a65e242fde28f987b6774ab3b6fd09f6ce5631d6572b52d3b0c6a2f3f45 File: yield/interface/stargate/ILpStaking.sol SHA3: d5100204d82780226bf2effc140b1034909bccd8077db49f3108339dfc09db91 File: yield/interface/stargate/IPool.sol SHA3: d358eba72cb606b1f2cb26bf4694a5fff5360b88b6ca02a72002b777761c50d3 File: yield/interface/stargate/IStargateRouter.sol SHA3: 99bd147e8706eb4e413ec58d9eff5cabea6bb624376191f84c6319481407d7ee File: yield/module/AaveYieldModule.sol SHA3: 7fb06897ec20f9b0e5bcbe57244b14680f857696dabb9b878549132e218565d6 File: yield/module/BaseModule.sol SHA3: 002339f41f12f60bf1c17eb50038558cbd3b0e30c145fc093edea3fdad317117 File: yield/module/CompoundV2Module.sol SHA3: 677439ad7379e2127fc770ff38d0fd9c56a08a65ca8141765529cdb60d0969b0 File: yield/module/StargateYieldModule.sol SHA3: 2d37953e7958c37834f73a6deb5b4d751e83a716865582b9bb20f25ccbdfbf73

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 the loss of user funds or contract state manipulation by external or internal actors.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation by external or internal actors.
Medium	Medium vulnerabilities are usually limited to state manipulations but cannot lead to asset loss. Major deviations from best practices are also in this category.
Low	Low vulnerabilities are related to outdated and unused code or minor Gas optimization. These issues won't have a significant impact on code execution but affect code quality

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 scoring methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**.

- Functional documentation is satisfactory.
- Technical documentation is satisfactory.
- NatSpecs are present and useful.

Code quality

The total Code Quality score is **9** out of **10**.

- Solidity style guidelines are not fully followed.
- There are some low code quality issues.

Test coverage

Code coverage of the project is **100%** (branch coverage).

- Deployment and basic user interactions are covered with tests.
- Negative cases coverage is present.
- Interactions with several users are tested.

Security score

As a result of the audit, the code contains **3** low severity issues. 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: **9.8**.

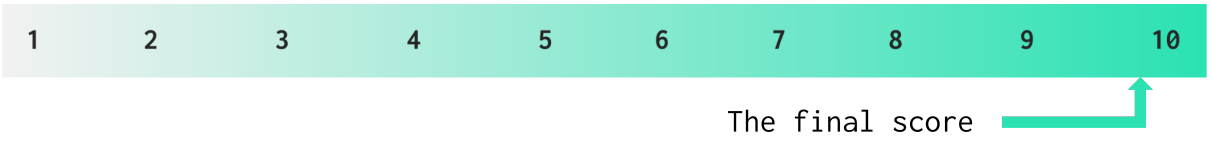

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
22 March 2023	13	4	1	0
11 April 2023	3	0	0	0

Risks

- The system claims to use multi sigs (GnosisSafe implementation) to manage the privileged roles. This can not be verified in this audit as this is **out of the scope**.
- The system claims to use an “openZeppelin TimeLock Controller with 48h retention, used to trigger admin actions on SmartFarmooor”. This can not be verified in this audit as this is **out of the scope**.
- “Dynamic allocation is at the heart of SmartFarmooor. An **off-chain bot calculates the optimal allocation**. Based on the result and quant review the Manager multisig applies the allocation to SmartFarmooor by calling setModuleAllocation().” This **functionality depends on off-chain logic** and is therefore **out of the scope** of this audit.
- The system deposits users’ funds into other protocols. These protocols are **out of the audit scope**.
- Contracts are upgradeable and are subject to future modification.
- **The rewards are swapped through the DEX module without setting slippage**.

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

System Overview

Smart Farmooor is a vault that helps users earn a higher return on their cryptocurrency by analyzing and allocating their assets across multiple yield markets.
It uses complex calculations to ensure assets are allocated in the most optimal way, taking into account factors like performance, strength, and safety. It's a great choice for those looking to maximize their returns while minimizing risk.
Each of the SmartFarmooor contracts is responsible for managing one asset. The main contract interacts with users who can deposit/withdraw a given asset. It mints/burns IOU for users and deploys assets to underlying yield modules according to a precalculated allocation.
Yield modules are connectors to external protocols where users' assets are deposited.

- The vault is fully composable and built on top of trusted protocols (Aave, BenQi, Stargate, TraderJoe Lend).
- The vault should be able to take the weight as parameters and do a reallocation of assets to supported yield markets.
- The vault design should be modular and allow for the addition of new protocol supports (new modules) without redeploying the core of the vault.
- Everything should be done on-chain and no access to the funds by the team should be possible (unruggable).
- The vault must be fully liquid at any time without any lockup period, funds should be accessible to withdraw at all times (asynchronous withdrawal on the yield market should be supported).
- The vault should support the following assets on the Avalanche Blockchain: BTC.b, USDC, USDT, sAVAX and AVAX.
- The vault should have a method accessible (with access rights) to harvest the rewards/fees

The files in the scope:

- **SmartFarmooor.sol** - Upgradable contract. This is where all the modules are added.
- **BaseModule.sol** - Abstract contract that inherits Rescuable.sol. The base module is inherited by all the modules (AaveYieldModule, CompoundV2Module, StargateYieldModule).
- **AaveYieldModule.sol** - The Aave module is responsible for interacting with the Aave lending protocol.
- **CompoundV2Module.sol** - The CompoundV2 abstraction is responsible for abstracting the logic of all forks of CompoundV2.
- **StargateYieldModule.sol** - The Stargate Yield module is responsible for interacting with the Stargate protocol and managing liquidity on it.
- **TraderJoeDexModule.sol** - Module to interact with Trader Joe DEX.

- **NativeGateway.sol** - gateway to convert native tokens to wrapped native tokens.
- **Rescuable.sol** - Can rescue tokens and native currency.
- **RoleManagement.sol** - manages the different roles.
- **Modifier.sol** - contains 2 modifiers.
- **IDex.sol** - Interface which is a face for Dex handlers implementations.
- **IJoeRouter01.sol** - Trader Joe's DEX interfaces.
- **IJoeRouter02.sol** - Trader Joe's DEX interfaces.
- **INativeGateway.sol** - Interface inherited by NativeGateway.sol.
- **ISmartFarmooor.sol** - Interface inherited by SmartFarmooor.sol.
- **IYieldModule.sol** - Interface inherited by BaseModule.sol.
- **IPool.sol** - Aave interface used in AaveYieldModule.sol.
- **IPoolDataProvider.sol** - Aave interface used in AaveYieldModule.sol.
- **IRewardsController.sol** - Aave interface used in AaveYieldModule.sol.
- **IScaledBalanceToken.sol** - Aave interface used in AaveYieldModule.sol.
- **WadRayMath.sol** - Aave library that provides functions to perform calculations with Wad and Ray units.
- **ComptrollerInterface.sol** - Compound v2 interface used in CompoundV2Module.sol.
- **CTokenInterface.sol** - Compound v2 interface used in CompoundV2Module.sol.
- **ExponentialNoError.sol** - Exponential module for storing fixed-precision decimals (from BenQi).
- **INativeWrapper.sol** - Wrapped native token (wAVAX) interface used in CompoundV2Module.sol and NativeGateway.sol.
- **ILpStaking.sol** - Stargate interface used in StargateYieldModule.sol.
- **IPool.sol** - Stargate interface used in StargateYieldModule.sol.
- **IStargateRouter.sol** - Stargate interface used in StargateYieldModule.sol.

Privileged roles

- SmartFarmoor :
  - DEFAULT_ADMIN_ROLE : can authorize contract upgrade, can add modules, can set the fee manager, can set the performance fee, can set the minimum and the maximum (cap) amount for deposits, can set automation rules, can pause/unpause, can rescue tokens and native tokens.
  - MANAGER_ROLE : can remove modules, can set module allocation, can finish panic, can set the minimum harvest threshold
  - PANICOOOR_ROLE : can panic.
  - PRIVATE_ACCESS_ROLE : deposit and withdrawal functions can be open or limited to private access roles.

- NativeGateway :
  - Owner : can rescue tokens and native tokens.
- AaveYieldModule :
  - Owner : can set DEX contract address, can set a new rewards token, can approve a DEX, can rescue tokens and native tokens.
  - Smart Farmoor : can deposit, withdraw and harvest.
- CompoundV2Module :
  - Owner : can set DEX contract address, can set a new rewards token, can approve a DEX, can rescue tokens and native tokens.
  - Smart Farmoor : can deposit, withdraw and harvest.
- StargateYieldModule :
  - Owner : can set DEX contract address, can set a new rewards token, can approve a DEX, can rescue tokens and native tokens. Also has all the manager roles.
  - Smart Farmoor : can deposit, withdraw and harvest.
  - Manager : can set the stargate pool id, can set the Stargate redeem Chain id, can set the Stargate staking pool id, can set the minimum LP profit threshold to withdraw.
- TraderJoeDexModule :
  - Owner : can set routes, can delete routes, can rescue tokens and native tokens.

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

Checked Items

We have audited the Customers' smart contracts for commonly known and specific vulnerabilities. Here are some items 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.	Passed
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.	Passed
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.	Not Relevant
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.	Passed
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.	Passed
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	Passed

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

Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	Not Relevant
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Not Relevant
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155 EIP-712	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. All parameters from the signature should be used in signer recovery. EIP-712 should be followed during a signer verification.	Not Relevant
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.	Not Relevant
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 or be locked on the contract.	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

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

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.	Passed
Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	Passed
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.	Passed
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 sufficient, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Passed
Stable Imports	**Custom**	The code should not reference draft contracts, which may be changed in the future.	Passed

Findings

Critical

No critical issues were found.

High

H01. Requirements Violation

In the requirements section of the documentation, it is initially specified that "All processes should be executed on-chain." However, the allocation calculation serves as an exception, being performed off-chain.

Additionally, the documentation specifies that “funds should be accessible to withdraw at any time” while it is not possible for the users to withdraw their funds when the contract is on pause.

Path: ./contracts/yield/SmartFarmoor.sol

Recommendation: align implementation and requirements.

Status: Fixed (Revised commit: da2e6465c12a5bcbcb524abb85850044bb86db88)

Medium

M01. Non-Finalized Code - Requirements Violation

In the SmartFarmoor contract, there is a variable automationRules (“The address of the contract containing automation rules”). There is also an external setter function and an internal one.

This part of the code is never used and is not documented on the website.

Path: ./contracts/yield/SmartFarmoor.sol : setAutomationRules()

Recommendation: provide documentation for the desired use cases of the automation rule contract and implement it in the code base or remove the unused functionality.

Status: Fixed (Revised commit: da2e6465c12a5bcbcb524abb85850044bb86db88)

M02. Data Consistency

The *_setRewards* function is responsible for setting initial reward tokens or updating reward tokens array. It’s possible to add one reward token to the *rewards* array multiple times.

Path: ./contracts/yield/module/BaseModule.sol : _setRewards()

Recommendation: add a check that one reward token can be added to the *rewards* array only once.

Status: Fixed (Revised commit: da2e6465c12a5bcbcb524abb85850044bb86db88)

M03. Best Practice Violation - Uninitialized Implementation

It's not recommended to leave an implementation contract uninitialized. An uninitialized implementation contract can be taken over by an attacker.

Path: ./contracts/yield/SmartFarmooor.sol

./contracts/yield/module/AaveYieldModule.sol

./contracts/yield/module/CompoundV2Module.sol

./contracts/yield/module/StargateYieldModule.sol

Recommendation: invoke the *_disableInitializers* function in the *constructor* to automatically lock the contract when it is deployed.

Status: Fixed (Revised commit: da2e6465c12a5bcbcb524abb85850044bb86db88)

M04. Redundant Inheritance

All modules have a variable *executionFee* inherited from BaseModule. This variable is defined as "The fee in native token that has to be paid in case of async withdrawal".

This fee is therefore needed for the Stargate module but not for the Aave module or the CompoundV2 module. The *getExecutionFee* function should always return 0 in these modules.

Path: ./contracts/yield/module/AaveYieldModule.sol

./contracts/yield/module/CompoundV2YieldModule.sol

Recommendation: Override the getExecutionFee function of AaveYieldModule and CompoundV2YieldModule to always return 0.

Status: Fixed (Revised commit: da2e6465c12a5bcbcb524abb85850044bb86db88)

Low

L01. NatSpecs Contradiction

The NatSpec of the function *getLastUpdatedBalance* says *"Get last updated balance on CompoundV2 fork"* while it should be Aave.

The NatSpec of the function *_rewardsProfit* says *"Collects the rewards tokens earned on CompoundV2 fork"* while it should be Aave.

The NatSpec comments for *getLastUpdatedBalance* in StargateYieldModule.sol contradict actual code behavior as this module is designed to work with the Stargate protocol.

Paths: ./contracts/yield/module/AaveModule.sol : getLastUpdatedBalance(), _rewardsProfit()

./contracts/yield/module/StargateYieldModule.sol: getLastUpdatedBalance()

Recommendation:.

Status: Fixed (Revised commit: da2e6465c12a5bcbcb524abb85850044bb86db88)

L02. Function State Mutability Can Be Changed To View

The function *_allocationIsCorrect* does not modify the state variables and should be declared as view.

This can lower Gas taxes.

Paths: ./contracts/yield/SmartFarmoor.sol : _allocationIsCorrect()

Recommendation: Change function's state mutability to view.

Status: Fixed (Revised commit: da2e6465c12a5bcbcb524abb85850044bb86db88)

L03. Variable Shadowing

In CompoundVsModule.sol, in the function deposit(), the variable uint256 error (L88) shadows uint256 error (L83).

Path: ./contracts/yield/module/CompoundV2Module.sol

Recommendation: If they are two different variables, rename one of them. If they are the same, do not declare it twice.

Status: Fixed (Revised commit: da2e6465c12a5bcbcb524abb85850044bb86db88)

L04. Unused Variable

The variable *expectedAmount* is never used.

Path: ./contracts/yield/module/CompoundV2Module.sol : _swapTokenRewardsForBaseToken()

Recommendation: Remove the unused variable.

Status: Fixed (Revised commit: da2e6465c12a5bcbcb524abb85850044bb86db88)

L05. Missing Zero Address Validation

Address parameters (*_cToken* & *_comptroller*) are used without checking against the possibility of 0x0.

This can lead to unwanted external calls to 0x0.

Path: ./contracts/yield/module/CompoundV2Module.sol : initialize()

Recommendation: Implement zero address checks.

Status: Fixed (Revised commit: da2e6465c12a5bcbcb524abb85850044bb86db88)

L06. Style Guide Violation

The provided projects should follow the official guidelines.

Inside each contract, library or interface, use the following order:

1. Type declarations
2. State variables
3. Events
4. Modifiers
5. Functions

Functions should be grouped according to their visibility and ordered:

1. constructor
2. receive function (if exists)
3. fallback function (if exists)
4. external
5. public
6. internal
7. private

Within a grouping, place the view and pure functions last.

Path: ./contracts/

Recommendation: Follow the official Solidity guidelines.

Status: Reported

L07. Redundant State Variable Update

When removing a module, the loop makes one unnecessary interaction.

for (uint i = _moduleId; i <= numberOfModules; i += 1) {

can be replaced by :

for (uint i = _moduleId; i < numberOfModules; i += 1) {

Path: ./contracts/yield/SmartFarmoor.sol : removeModule()

Recommendation: Replace <= by <.

Status: Fixed (Revised commit: da2e6465c12a5bcbcb524abb85850044bb86db88)

L08. Use of Hard-Coded Values

Hard-coded values are used in computations.

In the function *setModuleAllocation()*, the hard-coded value 100 is used.

In the function *finishPanic()*, the hard-coded value 10 is used.

Path: ./contracts/yield/SmartFarmoor.sol : setModuleAllocation(), finishPanic()

Recommendation: Convert these variables into constants.

Status: Fixed (Revised commit: da2e6465c12a5bcbcb524abb85850044bb86db88)

L09. Misleading Error Message

The function _rescueNative() gives the following error message : “Failed to send Ether” while the project is supposed to run on the Avalanche blockchain. Therefore, the native token will not always be Ether.

Path: ./contracts/common/Rescuable.sol : _rescueNative()

Recommendation: Modify messages in require conditions to fit code behavior.

Status: Fixed (Revised commit: da2e6465c12a5bcbcb524abb85850044bb86db88)

L10. Unused Code

In the harvest function the variable netProfit is first assigned :

```
uint netProfit = profit - fee;
```

It is then reassigned to another value without using the first one :

```
netProfit = IERC20(baseToken).balanceOf(address(this));
```

Path: ./contracts/yield/SmartFarmoor.sol : _harvest()

Recommendation: Either use the first value (for validation) or remove the first assignment.

Status: Fixed (Revised commit: da2e6465c12a5bcbcb524abb85850044bb86db88)

L11. Functions that can be declared external

“public” functions that are never called by the contract should be declared “external” to save gas.

Path: ./contracts/dex/TraderJoeDexModule.sol : deleteRoutes()

./contracts/yield/module/AaveYieldModule.sol : initialize(), getLastUpdatedBalance()

./contracts/yield/module/CompoundV2Module.sol : initialize(), getLastUpdatedBalance()

./contracts/yield/module/StargateYieldModule.sol : initialize(), getLastUpdatedBalance()

./contracts/yield/SmartFarmooor.sol : initialize(), pricePerShare(),

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

Status: Reported (All have been fixed except the 3 getLastUpdatedBalance())

L12. Missing Empty String Check

The `_setName` function of BaseModule.sol doesn't check if the `name` of the module is empty.

Path: ./contracts/yield/module/BaseModule.sol : _setName()

Recommendation: Implement empty string checks. The preferred way of checking whether a string is empty is to check if its length is equal to zero.

Status: Fixed (Revised commit: da2e6465c12a5bcbcb524abb85850044bb86db88)

L13. Missing Events

Events for critical state changes should be emitted for tracking things off-chain.

Paths: ./contracts/yield/SmartFarmooor.sol : addModule(), removeModule(), panic(), finishPanic(), _setFeeManager(), _setCap(), _setMinAmount(), _setBaseToken(), _setMinHarvestThreshold(), _setPerformanceFee(), _setAutomationRules()

./contracts/common/Rescuable.sol : _rescueToken(), _rescueNative()

./contracts/yield/module/BaseModule.sol : _setSmartFarmooor() _setDex(), _setManager(), _setBaseToken(), _setExecutionFee(), _setRewards(), _setWrappedNativeToken(), _setName()

./contracts/yield/module/StargateYieldModule.sol : _setLpStaking(), _setStargateRouter(), _setPool(), _setRouterPoolId(), _setRedeemFromChainId(), _setLpStakingPoolId(), _setLpProfitWithdrawalThreshold()

Recommendation: Create and emit related events.

Status: Reported

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

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing 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 report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and 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.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

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 Consultant cannot guarantee the explicit security of the audited smart contracts.