



Gravis Finance, Code Review and Security Analysis Report

Customer: Gravis Finance
Prepared on: 10th May 2022
Platform: BSC
Language: Solidity

rdauditors.com

Table of Contents

Disclaimer	2
Document	3
Introduction	5
Project Scope	6
Executive Summary	7
Code Quality	8
Documentation	9
Use of Dependencies	10
AS-IS Overview	11
Code Flow Diagram - Gravis Finance	16
Code Flow Diagram - Slither Results Log	20
Severity Definitions	37
Audit Findings	38
Conclusion	40
Note For Contract Users	41
Our Methodology	44
Disclaimers	46

Disclaimer

This document may contain confidential information about its systems and intellectual property of the customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the customer or it can be disclosed publicly after all vulnerabilities are fixed - upon the decision of the customer.

Document

Name	Smart Contract Code Review and Security Analysis Report of Gravis Finance
Platform	BSC / Solidity
File 1	StrategyFarmLP.sol
MD5 hash	0F7E4EE6A171E8B73E53E4E9114FBD8C
SHA256 hash	513C4CA7EBE6950E11B63DE7777C03E5EC7D79CB92DBBE708F59B EB259998AFD
File 2	StrategyRewardPool.sol
MD5 hash	0B2AF2A7FCA1689D47DBFD25743F52BB
SHA256 hash	36B8473C5302729932D2F0F83E81DF4C6ECFDAA63A82EAC5482BF D0CB2416578
File 3	StratManager.sol
MD5 hash	B686388EF6CE574BA0BBF0DB6F4F37B1
SHA256 hash	0B4770294045274CE83821A321B184223CDF1A1D57EED6B305B81A 94DDCFAAB7
File 4	Vault.sol

MD5 hash	6A77A960D4E1B1629E824473003C5951
SHA256 hash	F1BBEEF618913431F3C88C155BEE764BF8DA80BE9D1B29CA828997 6D96BCD5AB
Date	10/05/2022

Introduction

RD Auditors (Consultant) were contracted by Gravis Finance (Customer) to conduct a Smart Contracts Code Review and Security Analysis. This report represents the findings of the security assessment of the customer`s smart contracts and its code review conducted between 4th - 10th May 2022.

This contract consists of four files.

Project Scope

The scope of the project is a smart contract. We have scanned this smart contract for commonly known and more specific vulnerabilities, below are those considered (the full list includes but is not limited to):

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Byte array vulnerabilities
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference
- Implicit visibility level

Executive Summary

According to the assessment, the customer's solidity smart contract is **secured**.

You are Here



Insecure



Poorly Secured








Secure



Well-Secured

Automated checks are with smartDec, Mythril, Slither and remix IDE. All issues were performed by our team, which included the analysis of code functionality, the manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the AS-IS section and all issues found are located in the audit overview section.

We found the following;

Total Issues	0
 Critical	0
 High	0
 Medium	0
 Low	0
 Very Low	0

Code Quality

The libraries within this smart contract are part of a logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned to a specific address and its properties/methods can be reused many times by other contracts.

The Gravis Finance team has not provided scenario and unit test scripts, which would help to determine the integrity of the code in an automated way.

Overall, the code is almost commented. Commenting can provide rich documentation for functions, return variables and more. Use of the Ethereum Natural Language Specification Format (NatSpec) for commenting is recommended.

Documentation

We were given the Gravis Finance code as a file.

The hash of that file is mentioned in the table. As mentioned above, It's commented smart contract code, so anyone can quickly understand the programming flow as well as complex code logic.

Comments are very helpful in understanding the overall architecture of the protocol. It also provides a clear overview of the system components, including helpful details, like the lifetime of the background script.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure. Those were based on well known industry standard open source projects and even core code blocks that are written well and systematically.

AS-IS Overview

Gravis Finance

File And Function Level Report

File: StrategyFarmLP
Contract: StrategyFarmLP
Import: Ownable, Pausable, ReentrancyGuard, IERC20, SafeERC20, SafeMath, StratManager, UniswapInterfaces, StratManager
Inherit: StratManager
Observation: Passed
Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	initializeStrategy	onlyOwner	Passed	All Passed	No Issue	Passed
2	deposit	write	Passed	All Passed	No Issue	Passed
3	Withdraw	write	Passed	All Passed	No Issue	Passed
4	harvest	onlyEOA	Passed	All Passed	No Issue	Passed
5	removeLiquidity	internal	Passed	All Passed	No Issue	Passed
6	addLiquidity	internal	Passed	All Passed	No Issue	Passed
7	balanceOf	read	Passed	All Passed	No Issue	Passed
8	balanceOfwant	read	Passed	All Passed	No Issue	Passed
9	balanceOfPool	read	Passed	All Passed	No Issue	Passed

10	retireStrat	external	Passed	All Passed	No Issue	Passed
11	Panic	OnlyManager	Passed	All Passed	No Issue	Passed
12	Pause	OnlyManager	Passed	All Passed	No Issue	Passed
13	unpause	OnlyManager	Passed	All Passed	No Issue	Passed
14	_giveAllowance	internal	Passed	All Passed	No Issue	Passed

File: StratManager

Contract: StratManager

Import: Ownable, Pausable, IERC20, SafeERC20, SafeMath,
UniswapInterfaces

Inherit: Ownable, Pausable

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	setKeeper	onlyManager	Passed	All Passed	No Issue	Passed
2	setUnirouter	onlyOwner	Passed	All Passed	No Issue	Passed
3	setVault	onlyOwner	Passed	All Passed	No Issue	Passed
4	setCallFee	onlyOwner	Passed	All Passed	No Issue	Passed
5	setGravisFee	onlyOwner	Passed	All Passed	No Issue	Passed
6	setWithdrawFees	onlyOwner	Passed	All Passed	No Issue	Passed
7	chargeFees	internal	Passed	All Passed	No Issue	Passed
8	beforeDeposit	external	Passed	All Passed	No Issue	Passed

File: Vault
Contract: Vault
Import: Ownable, ERC20, IERC20, ReentrancyGuard, SafeMath, SafeERC20
Inherit: ERC20, Ownable, ReentrancyGuard
Observation: Passed
Test Report: Passed

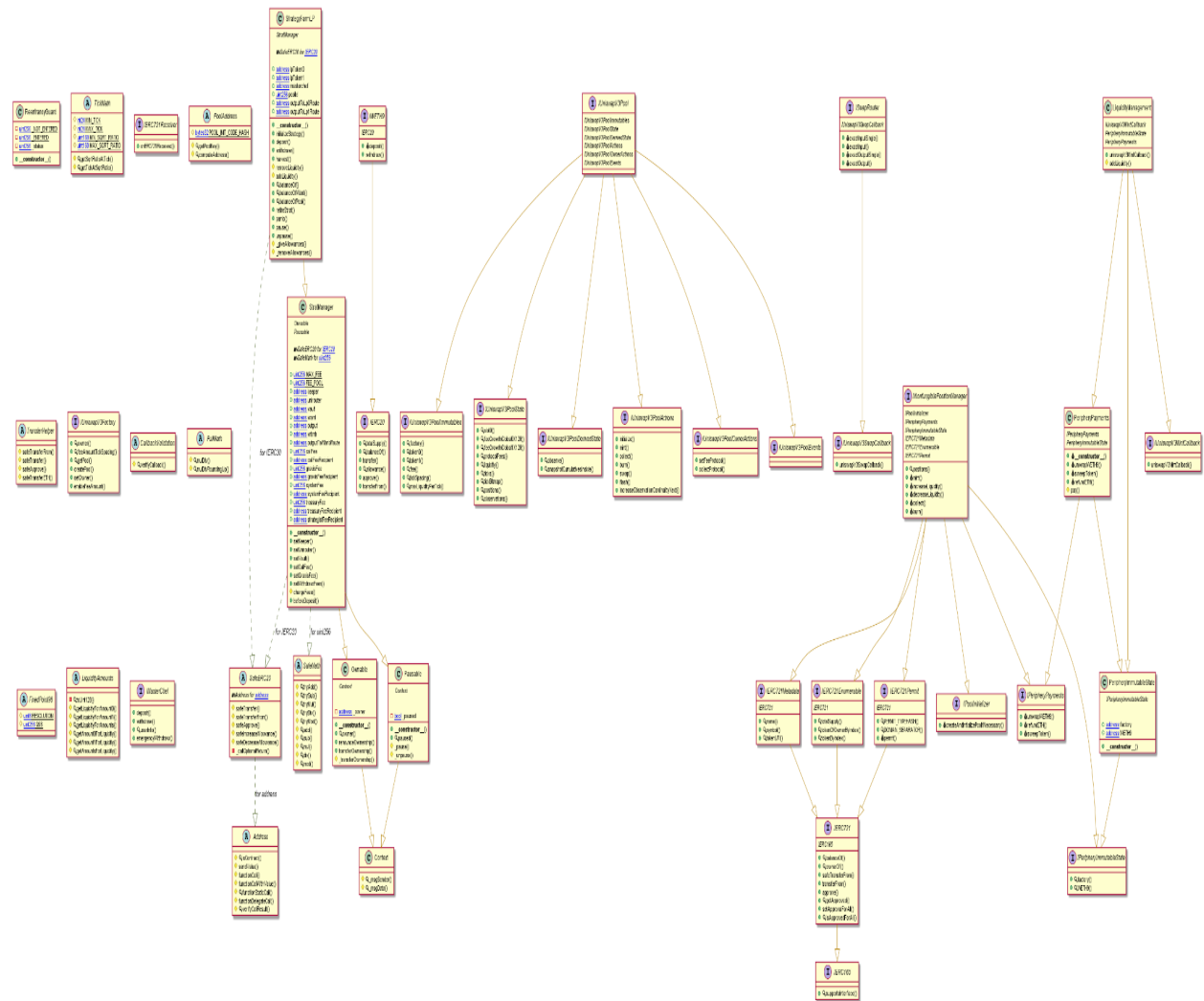
Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	want	read	Passed	All Passed	No Issue	Passed
2	balance	read	Passed	All Passed	No Issue	Passed
3	available	read	Passed	All Passed	No Issue	Passed
4	getPricePerFullShare	read	Passed	All Passed	No Issue	Passed
5	depositAll	write	Passed	All Passed	No Issue	Passed
6	deposit	write	Passed	All Passed	No Issue	Passed
7	earn	write	Passed	All Passed	No Issue	Passed
8	WithdrawAll	write	Passed	All Passed	No Issue	Passed
9	Withdraw	write	Passed	All Passed	No Issue	Passed
10	ProposesStrat	onlyowner	Passed	All Passed	No Issue	Passed
11	UpgradeStrat	onlyowner	Passed	All Passed	No Issue	Passed
12	inCaseTokenGetStuck	onlyowner	Passed	All Passed	No Issue	Passed

File: StrategyRewardPool
Contract: StrategyRewardPool
Import: Ownable, Pausable, ReentrancyGuard, IERC20, safeERC20,
SafeMath, UniswapInterfaces, StratManager
Inherit: StratManager
Observation: Passed
Test Report: Passed

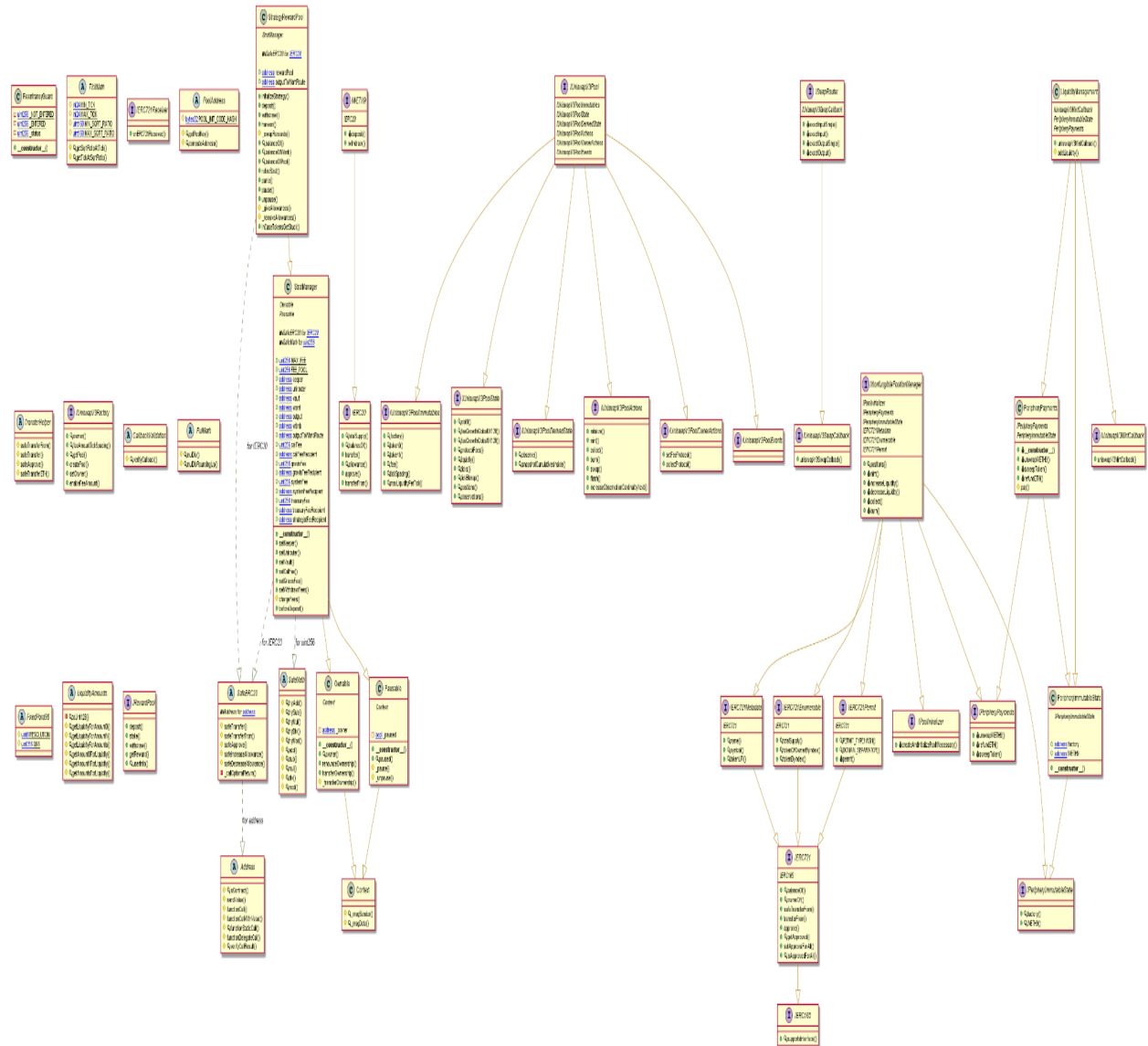
Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	initializeStrategy	onlyowner	Passed	All Passed	No Issue	Passed
2	deposit	write	Passed	All Passed	No Issue	Passed
3	withdraw	write	Passed	All Passed	No Issue	Passed
4	harvest	onlyEOA	Passed	All Passed	No Issue	Passed
5	_swapRewards	internal	Passed	All Passed	No Issue	Passed
6	balanceOf	read	Passed	All Passed	No Issue	Passed
7	balanceOfWant	read	Passed	All Passed	No Issue	Passed
8	balanceOfPool	read	Passed	All Passed	No Issue	Passed
9	retireStrat	write	Passed	All Passed	No Issue	Passed
10	Panic	onlyManager	Passed	All Passed	No Issue	Passed
11	Pause	onlyManager	Passed	All Passed	No Issue	Passed
12	unpause	onlyManager	Passed	All Passed	No Issue	Passed
13	_giveAllowances	internal	Passed	All Passed	No Issue	Passed

14	_removeAllowances	internal	Passed	All Passed	No Issue	Passed
15	incaseTokensGet Stuck	onlyManager	Passed	All Passed	No Issue	Passed

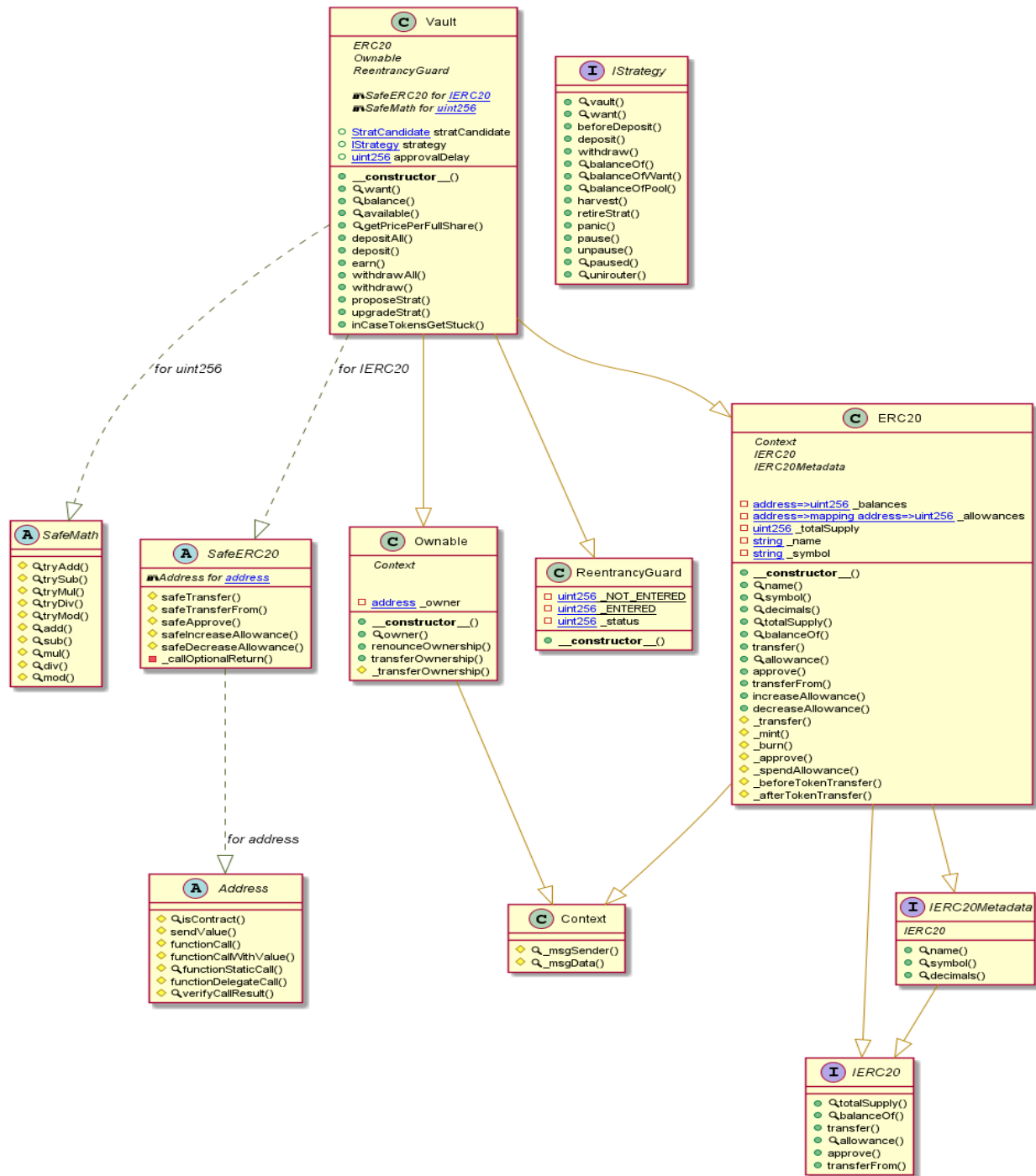
File: StrategyFarmLP



File: StrategyRewardPool.sol



File: Vault



[illegible]

Code Flow Diagram - Slither Results Log

File: StrategyFarmLP.sol

```
INFO:Detectors:
StratManager.setKeeper(address) (StrategyFarmLP.sol#2064-2066) should emit an event for:
- keeper = _keeper (StrategyFarmLP.sol#2065)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control
INFO:Detectors:
StratManager.setKeeper(address)._keeper (StrategyFarmLP.sol#2064) lacks a zero-check on :
- keeper = _keeper (StrategyFarmLP.sol#2065)
StratManager.setUnirouter(address)._unirouter (StrategyFarmLP.sol#2072) lacks a zero-check on :
- unirouter = _unirouter (StrategyFarmLP.sol#2073)
StratManager.setVault(address)._vault (StrategyFarmLP.sol#2080) lacks a zero-check on :
- vault = _vault (StrategyFarmLP.sol#2081)
StrategyFarmLP.initializeStrategy(address,uint256)._masterchef (StrategyFarmLP.sol#2173) lacks a zero-check on :
- masterchef = _masterchef (StrategyFarmLP.sol#2176)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in StratManager.chargeFees() (StrategyFarmLP.sol#2107-2126):
  External calls:
  - IERC20(wbnb).safeTransfer(callFeeRecipient,callFeeAmount) (StrategyFarmLP.sol#2116)
  - IERC20(wbnb).safeTransfer(gravisFeeRecipient,gravisFeeAmount) (StrategyFarmLP.sol#2119)
  - IERC20(wbnb).safeTransfer(strategistFeeRecipient,strategistFeeAmount) (StrategyFarmLP.sol#2122)
  Event emitted after the call(s):
  - ChargedFees(callFeeAmount,gravisFeeAmount,strategistFeeAmount) (StrategyFarmLP.sol#2124)
Reentrancy in StrategyFarmLP.harvest() (StrategyFarmLP.sol#2238-2245):
```

```
  Event emitted after the call(s):
  - StratHarvest(msg.sender) (StrategyFarmLP.sol#2244)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.verifyCallResult(bool,bytes,string) (StrategyFarmLP.sol#477-495) uses assembly
- INLINE ASM (StrategyFarmLP.sol#487-490)
TickMath.getTickAtSqrtRatio(uint160) (StrategyFarmLP.sol#1000-1142) uses assembly
- INLINE ASM (StrategyFarmLP.sol#1007-1011)
- INLINE ASM (StrategyFarmLP.sol#1012-1016)
- INLINE ASM (StrategyFarmLP.sol#1017-1021)
- INLINE ASM (StrategyFarmLP.sol#1022-1026)
- INLINE ASM (StrategyFarmLP.sol#1027-1031)
- INLINE ASM (StrategyFarmLP.sol#1032-1036)
- INLINE ASM (StrategyFarmLP.sol#1037-1041)
- INLINE ASM (StrategyFarmLP.sol#1042-1045)
- INLINE ASM (StrategyFarmLP.sol#1052-1057)
- INLINE ASM (StrategyFarmLP.sol#1058-1063)
- INLINE ASM (StrategyFarmLP.sol#1064-1069)
- INLINE ASM (StrategyFarmLP.sol#1070-1075)
- INLINE ASM (StrategyFarmLP.sol#1076-1081)
- INLINE ASM (StrategyFarmLP.sol#1082-1087)
- INLINE ASM (StrategyFarmLP.sol#1088-1093)
- INLINE ASM (StrategyFarmLP.sol#1094-1099)
- INLINE ASM (StrategyFarmLP.sol#1100-1105)
- INLINE ASM (StrategyFarmLP.sol#1106-1111)
- INLINE ASM (StrategyFarmLP.sol#1112-1117)
- INLINE ASM (StrategyFarmLP.sol#1118-1123)
- INLINE ASM (StrategyFarmLP.sol#1124-1129)
- INLINE ASM (StrategyFarmLP.sol#1130-1134)
FullMath.mulDiv(uint256,uint256,uint256) (StrategyFarmLP.sol#1661-1712) uses assembly
```

```
FullMath.mulDiv(uint256,uint256,uint256) (StrategyFarmLP.sol#1661-1712) uses assembly
- INLINE ASM (StrategyFarmLP.sol#1668-1672)
- INLINE ASM (StrategyFarmLP.sol#1676-1678)
- INLINE ASM (StrategyFarmLP.sol#1686-1688)
- INLINE ASM (StrategyFarmLP.sol#1689-1692)
- INLINE ASM (StrategyFarmLP.sol#1694-1695)
- INLINE ASM (StrategyFarmLP.sol#1697-1698)
- INLINE ASM (StrategyFarmLP.sol#1699-1700)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
- Version used: ['>=0.4.0', '>=0.5.0', '^0.8.4']
- ^0.8.4 (StrategyFarmLP.sol#2)
- >=0.4.0 (StrategyFarmLP.sol#1658)
- >=0.5.0 (StrategyFarmLP.sol#1735)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address.functionCall(address,bytes) (StrategyFarmLP.sol#361-363) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (StrategyFarmLP.sol#390-396) is never used and should be removed
Address.functionDelegateCall(address,bytes) (StrategyFarmLP.sol#450-452) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (StrategyFarmLP.sol#460-469) is never used and should be removed
Address.functionStaticCall(address,bytes) (StrategyFarmLP.sol#423-425) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (StrategyFarmLP.sol#433-442) is never used and should be removed
Address.sendValue(address,uint256) (StrategyFarmLP.sol#336-341) is never used and should be removed
CallbackValidation.verifyCallback(address,address,address,uint24) (StrategyFarmLP.sol#1637-1644) is never used and should be removed
Context.msgData() (StrategyFarmLP.sol#570-581) is never used and should be removed
```

```
INFO:Detectors:
TickMath.getSqrtRatioAtTick(int24) (StrategyFarmLP.sol#971-998) uses literals with too many digits:
- ratio = 0x100000000000000000000000000000000 (StrategyFarmLP.sol#974)
FixedPoint96.slitherConstructorConstantVariables() (StrategyFarmLP.sol#1728-1731) uses literals with too many digits:
- Q96 = 0x100000000000000000000000000000000 (StrategyFarmLP.sol#1730)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
TickMath.MAX_TICK (StrategyFarmLP.sol#966) is never used in TickMath (StrategyFarmLP.sol#964-1143)
PoolAddress.POOL_INIT_CODE_HASH (StrategyFarmLP.sol#1434) is never used in PoolAddress (StrategyFarmLP.sol#1433-1454)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
StrategyFarmLP.lpToken0 (StrategyFarmLP.sol#2135) should be constant
StrategyFarmLP.lpToken1 (StrategyFarmLP.sol#2136) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (StrategyFarmLP.sol#617-619)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (StrategyFarmLP.sol#625-628)
unwrapWETH9(uint256,address) should be declared external:
- PeripheryPayments.unwrapWETH9(uint256,address) (StrategyFarmLP.sol#1850-1858)
sweepToken(address,uint256,address) should be declared external:
- PeripheryPayments.sweepToken(address,uint256,address) (StrategyFarmLP.sol#1860-1871)
initializeStrategy(address,uint256) should be declared external:
- StrategyFarmLP.initializeStrategy(address,uint256) (StrategyFarmLP.sol#2172-2193)
withdraw(address,uint256,uint256) should be declared external:
- StrategyFarmLP.withdraw(address,uint256,uint256) (StrategyFarmLP.sol#2203-2236)
panic() should be declared external:
- StrategyFarmLP.panic() (StrategyFarmLP.sol#2281-2284)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:StrategyFarmLP.sol analyzed (43 contracts with 75 detectors), 154 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

File: StrategyRewardPool

```
INFO:Detectors:
StratManager.setKeeper(address) (StrategyRewardPool.sol#1539-1541) should emit an event for:
- keeper = _keeper (StrategyRewardPool.sol#1540)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control
INFO:Detectors:
StratManager.constructor(address,address,address,address,address,address,uint256,uint256,address,address,address)._keeper (StrategyRewardPool.sol#1494) lacks a zero-check on :
- keeper = _keeper (StrategyRewardPool.sol#1509)
StratManager.constructor(address,address,address,address,address,uint256,uint256,address,address,address)._unirouter (StrategyRewardPool.sol#1495) lacks a zero-check on :
- unirouter = _unirouter (StrategyRewardPool.sol#1510)
StratManager.constructor(address,address,address,address,address,address,uint256,uint256,address,address,address)._vault (StrategyRewardPool.sol#1496) lacks a zero-check on :
- vault = _vault (StrategyRewardPool.sol#1511)
StratManager.constructor(address,address,address,address,address,address,uint256,uint256,address,address,address)._want (StrategyRewardPool.sol#1497) lacks a zero-check on :
- want = _want (StrategyRewardPool.sol#1513)
StratManager.constructor(address,address,address,address,address,address,uint256,uint256,address,address,address)._output (StrategyRewardPool.sol#1498) lacks a zero-check on :
- output = _output (StrategyRewardPool.sol#1514)
StratManager.constructor(address,address,address,address,address,address,uint256,uint256,address,address,address)._wnnb (StrategyRewardPool.sol#1499) lacks a zero-check on :
- wbnb = _wnnb (StrategyRewardPool.sol#1515)
StratManager.constructor(address,address,address,address,address,address,uint256,uint256,address,address,address)._callFeeRecipient (StrategyRewardPool.sol#1504) lacks a zero-check on :
- callFeeRecipient = _callFeeRecipient (StrategyRewardPool.sol#1522)
StratManager.constructor(address,address,address,address,address,address,uint256,uint256,address,address,address)._gravisFeeRecipient (StrategyRewardPool.sol#1505) lacks a zero-check on :
```

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

```
INFO:Detectors:
Address.verifyCallResult(bool,bytes,string) (StrategyRewardPool.sol#195-213) uses assembly
- INLINE ASM (StrategyRewardPool.sol#205-208)
TickMath.getTickAtSqrtRatio(uint160) (StrategyRewardPool.sol#634-776) uses assembly
- INLINE ASM (StrategyRewardPool.sol#641-645)
- INLINE ASM (StrategyRewardPool.sol#646-650)
- INLINE ASM (StrategyRewardPool.sol#651-655)
- INLINE ASM (StrategyRewardPool.sol#656-660)
- INLINE ASM (StrategyRewardPool.sol#661-665)
- INLINE ASM (StrategyRewardPool.sol#666-670)
- INLINE ASM (StrategyRewardPool.sol#671-675)
- INLINE ASM (StrategyRewardPool.sol#676-679)
- INLINE ASM (StrategyRewardPool.sol#686-691)
- INLINE ASM (StrategyRewardPool.sol#692-697)
- INLINE ASM (StrategyRewardPool.sol#698-703)
- INLINE ASM (StrategyRewardPool.sol#704-709)
- INLINE ASM (StrategyRewardPool.sol#710-715)
- INLINE ASM (StrategyRewardPool.sol#716-721)
- INLINE ASM (StrategyRewardPool.sol#722-727)
- INLINE ASM (StrategyRewardPool.sol#728-733)
- INLINE ASM (StrategyRewardPool.sol#734-739)
- INLINE ASM (StrategyRewardPool.sol#740-745)
- INLINE ASM (StrategyRewardPool.sol#746-751)
- INLINE ASM (StrategyRewardPool.sol#752-757)
- INLINE ASM (StrategyRewardPool.sol#758-763)
- INLINE ASM (StrategyRewardPool.sol#764-768)
FullMath.mulDiv(uint256,uint256,uint256) (StrategyRewardPool.sol#1161-1206) uses assembly
- INLINE ASM (StrategyRewardPool.sol#1166-1170)
```


File: Vault

info@rdauditors.com


```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Vault.deposit(uint256) (Vault.sol#569-584):
  External calls:
    - strategy.beforeDeposit() (Vault.sol#570)
    - want().safeTransferFrom(msg.sender,address(this),_amount) (Vault.sol#573)
    - earn() (Vault.sol#574)
      - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (Vault.sol#272)
      - want().safeTransfer(address(strategy),_bal) (Vault.sol#588)
      - strategy.deposit() (Vault.sol#589)
      - (success,returndata) = target.call{value: value}(data) (Vault.sol#162)
  External calls sending eth:
    - earn() (Vault.sol#574)
      - (success,returndata) = target.call{value: value}(data) (Vault.sol#162)
  Event emitted after the call(s):
    - Transfer(address(0),account,amount) (Vault.sol#460)
    - _mint(msg.sender,shares) (Vault.sol#583)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Vault.getPricePerFullShare() (Vault.sol#561-563) uses timestamp for comparisons
  Dangerous comparisons:
    - totalSupply() == 0 (Vault.sol#562)
Vault.deposit(uint256) (Vault.sol#569-584) uses timestamp for comparisons
  Dangerous comparisons:
    - totalSupply() == 0 (Vault.sol#578)
Vault.upgradeStrat() (Vault.sol#613-625) uses timestamp for comparisons
  Dangerous comparisons:
    - require(bool,string)(stratCandidate.implementation != address(0),There is no candidate) (Vault.sol#614)
    - require(bool,string)(stratCandidate.proposedTime.add(approvalDelay) < block.timestamp,Delay has not passed) (Vault.sol
15)
Vault.inCaseTokensGetStuck(address) (Vault.sol#627-632) uses timestamp for comparisons
  Dangerous comparisons:
    - require(bool,string)(_token != address(want()),!token) (Vault.sol#628)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.verifyCallResult(bool,bytes,string) (Vault.sol#196-214) uses assembly
  - INLINE ASM (Vault.sol#206-209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (Vault.sol#133-135) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (Vault.sol#145-151) is never used and should be removed
Address.functionDelegateCall(address,bytes) (Vault.sol#181-183) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (Vault.sol#185-194) is never used and should be removed
Address.functionStaticCall(address,bytes) (Vault.sol#166-168) is never used and should be removed
```

```
Parameter Vault.proposeStrat(address)._implementation (Vault.sol#602) is not in mixedCase
Parameter Vault.inCaseTokensGetStuck(address)._token (Vault.sol#627) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Vault.upgradeStrat() (Vault.sol#613-625) uses literals with too many digits:
- stratCandidate.proposedTime = 5000000000 (Vault.sol#622)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (Vault.sol#306-308)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (Vault.sol#310-313)
name() should be declared external:
- ERC20.name() (Vault.sol#367-369)
symbol() should be declared external:
- ERC20.symbol() (Vault.sol#371-373)
decimals() should be declared external:
- ERC20.decimals() (Vault.sol#375-377)
transfer(address,uint256) should be declared external:
- ERC20.transfer(address,uint256) (Vault.sol#387-391)
approve(address,uint256) should be declared external:
- ERC20.approve(address,uint256) (Vault.sol#397-401)
transferFrom(address,address,uint256) should be declared external:
- ERC20.transferFrom(address,address,uint256) (Vault.sol#403-412)
increaseAllowance(address,uint256) should be declared external:
- ERC20.increaseAllowance(address,uint256) (Vault.sol#414-418)
decreaseAllowance(address,uint256) should be declared external:
- ERC20.decreaseAllowance(address,uint256) (Vault.sol#420-429)
getPricePerFullShare() should be declared external:
- Vault.getPricePerFullShare() (Vault.sol#561-563)
proposeStrat(address) should be declared external:
- Vault.proposeStrat(address) (Vault.sol#602-610)
upgradeStrat() should be declared external:
- Vault.upgradeStrat() (Vault.sol#613-625)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Vault.sol analyzed (11 contracts with 75 detectors), 56 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

File: StratManager

```
INFO:Detectors:
StratManager.setKeeper(address) (StratManager.sol#1514-1516) should emit an event for:
- keeper = _keeper (StratManager.sol#1515)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control
INFO:Detectors:
StratManager.constructor(address,address,address,address,address,address,address,uint256,uint256,address,address,address)._keeper (StratManager.sol#1469) lacks a zero-check on :
- keeper = _keeper (StratManager.sol#1484)
StratManager.constructor(address,address,address,address,address,address,uint256,uint256,address,address,address)._unirouter (StratManager.sol#1470) lacks a zero-check on :
- unirouter = _unirouter (StratManager.sol#1485)
StratManager.constructor(address,address,address,address,address,address,address,uint256,uint256,address,address,address)._vault (StratManager.sol#1471) lacks a zero-check on :
- vault = _vault (StratManager.sol#1486)
StratManager.constructor(address,address,address,address,address,address,address,uint256,uint256,address,address,address)._want (StratManager.sol#1472) lacks a zero-check on :
- want = _want (StratManager.sol#1488)
StratManager.constructor(address,address,address,address,address,address,address,uint256,uint256,address,address,address)._output (StratManager.sol#1473) lacks a zero-check on :
- output = _output (StratManager.sol#1489)
StratManager.constructor(address,address,address,address,address,address,address,uint256,uint256,address,address,address)._wnbn (StratManager.sol#1474) lacks a zero-check on :
- wnbn = _wnbn (StratManager.sol#1490)
StratManager.constructor(address,address,address,address,address,address,address,uint256,uint256,address,address,address)._callFeeRecipient (StratManager.sol#1479) lacks a zero-check on :
- callFeeRecipient = _callFeeRecipient (StratManager.sol#1497)
StratManager.constructor(address,address,address,address,address,address,address,uint256,uint256,address,address,address)._gravisFeeRecipient (StratManager.sol#1480) lacks a zero-check on :
- gravisFeeRecipient = _gravisFeeRecipient (StratManager.sol#1499)
```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation>

```
INFO:Detectors:
Reentrancy in StratManager.chargeFees() (StratManager.sol#1549-1566):
  External calls:
  - IERC20(wnbn).safeTransfer(callFeeRecipient,callFeeAmount) (StratManager.sol#1556)
  - IERC20(wnbn).safeTransfer(gravisFeeRecipient,gravisFeeAmount) (StratManager.sol#1559)
  - IERC20(wnbn).safeTransfer(strategistFeeRecipient,strategistFeeAmount) (StratManager.sol#1562)
  Event emitted after the call(s):
  - ChargedFees(callFeeAmount,gravisFeeAmount,strategistFeeAmount) (StratManager.sol#1564)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.verifyCallResult(bool,bytes,string) (StratManager.sol#195-213) uses assembly
- INLINE ASM (StratManager.sol#205-208)
TickMath.getTickAtSqrtRatio(uint160) (StratManager.sol#613-755) uses assembly
- INLINE ASM (StratManager.sol#620-624)
- INLINE ASM (StratManager.sol#625-629)
- INLINE ASM (StratManager.sol#630-634)
- INLINE ASM (StratManager.sol#635-639)
- INLINE ASM (StratManager.sol#640-644)
- INLINE ASM (StratManager.sol#645-649)
- INLINE ASM (StratManager.sol#650-654)
- INLINE ASM (StratManager.sol#655-658)
- INLINE ASM (StratManager.sol#665-670)
- INLINE ASM (StratManager.sol#671-676)
```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage>

```
INFO:Detectors:
Different versions of Solidity is used:
- Version used: ['>=0.4.0', '>=0.5.0', '^0.8.4']
- ^0.8.4 (StratManager.sol#2)
- >=0.4.0 (StratManager.sol#1134)
- >=0.5.0 (StratManager.sol#1205)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address.functionCall(address,bytes) (StratManager.sol#132-134) is never used and should be removed
Address.functionCall(address,bytes,string) (StratManager.sol#136-142) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (StratManager.sol#144-150) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (StratManager.sol#152-163) is never used and should be removed
Address.functionDelegateCall(address,bytes) (StratManager.sol#180-182) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (StratManager.sol#184-193) is never used and should be removed
Address.functionStaticCall(address,bytes) (StratManager.sol#165-167) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (StratManager.sol#169-178) is never used and should be removed
Address.isContract(address) (StratManager.sol#120-123) is never used and should be removed
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
TickMath.getSqrtRatioAtTick(int24) (StratManager.sol#584-611) uses literals with too many digits:
  - ratio = 0x100000000000000000000000000000000 (StratManager.sol#587)
FixedPoint96.slitherConstructorConstantVariables() (StratManager.sol#1198-1201) uses literals with too many digits:
  - Q96 = 0x100000000000000000000000000000000 (StratManager.sol#1200)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
TickMath.MAX_TICK (StratManager.sol#579) is never used in TickMath (StratManager.sol#577-756)
PoolAddress.POOL_INIT_CODE_HASH (StratManager.sol#915) is never used in PoolAddress (StratManager.sol#914-933)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
  - Ownable.renounceOwnership() (StratManager.sol#306-308)
transferOwnership(address) should be declared external:
  - Ownable.transferOwnership(address) (StratManager.sol#310-313)
unwrapWETH9(uint256,address) should be declared external:
  - PeripheryPayments.unwrapWETH9(uint256,address) (StratManager.sol#1320-1328)
sweepToken(address,uint256,address) should be declared external:
  - PeripheryPayments.sweepToken(address,uint256,address) (StratManager.sol#1330-1341)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:StratManager.sol analyzed (40 contracts with 75 detectors), 154 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

File: StrategyFarmLP

Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.

[more](#)

Pos: 2624:34:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

[more](#)

Pos: 2181:10:

Gas & Economy

Gas costs:

Gas requirement of function `StratManager.setWithdrawFees` is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 2662:8:

Gas costs:

Gas requirement of function `StrategyFarmLP.unpause` is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 2865:8:

Miscellaneous

Similar variable names:

`LiquidityAmounts.getLiquidityForAmounts(uint160,uint160,uint160,uint256,uint256)` : Variables have very similar names "liquidity" and "liquidity0". Note: Modifiers are currently not considered by this static analysis.

Pos: 2306:28:

Similar variable names:

`LiquidityAmounts.getAmount0ForLiquidity(uint160,uint160,uint128)` : Variables have very similar names "sqrtRatioAX96" and "sqrtRatioBX96". Note: Modifiers are currently not considered by this static analysis.

Pos: 2322:81:

Similar variable names:

StratManager.

(address,address,address,address,address,address,uint256,uint256,address,address,address) :

Variables have very similar names "_want" and "_wbnb". Note: Modifiers are currently not considered by this static analysis.

Pos: 2603:19:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

[more](#)

Pos: 2845:12:

Data truncated:

Division of integer values yields an integer value again. That means e.g. $10 / 100 = 0$ instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 2808:48:

File: Vault

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in

`Address.functionCallWithValue(address,bytes,uint256,string)`: Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

[more](#)

Pos: 412:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in `Vault.upgradeStrat()`: Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

[more](#)

Pos: 1220:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

[more](#)

Pos: 1222:65:

Gas & Economy

Gas costs:

Gas requirement of function ERC20.name is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 744:4:

Gas costs:

Gas requirement of function Vault.inCaseTokensGetStuck is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 1238:4:

Miscellaneous

Constant/View/Pure functions:

IStrategy.unpause() : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

[more](#)

Pos: 1259:4:

Similar variable names:

ERC20._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.

Pos: 976:49:

No return:

IStrategy.unirouter(): Defines a return type but never explicitly returns a value.

Pos: 1261:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

[more](#)

Pos: 1239:8:

File: StratManager

Security**Transaction origin:**

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.

[more](#)

Pos: 2579:34:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

[more](#)

Pos: 2128:10:

Gas & Economy

Gas costs:

Gas requirement of function `StratManager.setWithdrawFees` is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 2617:8:

Miscellaneous

Similar variable names:

`LiquidityAmounts.getAmountsForLiquidity(uint160,uint160,uint160,uint128)` : Variables have very similar names "`sqrtRatioAX96`" and "`sqrtRatioBX96`". Note: Modifiers are currently not considered by this static analysis.

Pos: 2318:32:

Guard conditions:

Use "`assert(x)`" if you never ever want `x` to be false, not in any circumstance (apart from a bug in your code). Use "`require(x)`" if `x` can be false, due to e.g. invalid input or a failing external component.

[more](#)

Pos: 2624:12:

Data truncated:

Division of integer values yields an integer value again. That means e.g. $10 / 100 = 0$ instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 2280:16:

File: StrategyRewardPool

Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.

[more](#)

Pos: 2627:34:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.

[more](#)

Pos: 1994:27:

Gas & Economy

Gas costs:

Gas requirement of function StratManager.setWithdrawFees is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 2665:8:

Miscellaneous

Similar variable names:

StrategyRewardPool.withdraw(address,uint256,uint256) : Variables have very similar names "amount" and "account". Note: Modifiers are currently not considered by this static analysis.

Pos: 2751:26:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

[more](#)

Pos: 2832:12:

Data truncated:

Division of integer values yields an integer value again. That means e.g. $10 / 100 = 0$ instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 2755:40:

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to lost tokens etc.
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 lost tokens.
Low	Low level vulnerabilities are most related to outdated, unused etc. These code snippets cannot have a significant impact on execution.
Lowest Code Style/ Best Practice	Lowest level vulnerabilities, code style violations and information statements cannot affect smart contract execution and can be ignored.

Audit Findings

Critical:

No critical severity vulnerabilities were found.

High:

No medium severity vulnerabilities were found.

Medium:

No medium severity vulnerabilities were found.

Low:

Missing event logs in the StrategyFarmLP and vault

It is best practice to emit an event when a significant state change is happening. It helps users interact with the blockchain. We suggest adding events in following functions:

- Deposit
- Withdraw
- DepositAll
- WithdrawAll
- Earn

Resolution: Add appropriate events in above functions.

Update (10/05/2022): Acknowledged

Very Low:

No very low severity vulnerabilities were found.

Conclusion

We were given a contract file and have used all possible tests based on the given object. The contract is written systematically, it is ready to go for production.

We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

The security state of the reviewed contract is now “secured”.

Note For Contract Users

There are several owner only functions. Those can be called by the owner's wallet only. So, if the owner's wallet is compromised, then it carries the risk of the contract becoming vulnerable.

File: Vault.sol

ProposeStrat: The owner can set the address for the strat to use with this vault.

UpdateStrat: The owner can upgrade strat.

InCaseTokenGetStuck: Owner can withdraw all the token from contract

```
184     function inCaseTokensGetStuck(address _token) external onlyOwner {  
185         require(_token != address(want()), "!token");  
186  
187         uint256 amount = IERC20(_token).balanceOf(address(this));  
188         IERC20(_token).safeTransfer(msg.sender, amount);  
189     }  
190 }
```

File: StrategyFarmLP.sol

Withdraw: There is a function which would enable the owner to change the Vault address and move all tokens to another address.

```
92     function withdraw(address account, uint256 share, uint256 totalShares) public virtual {
93         require(msg.sender == vault, "!vault");
94
95         uint256 amount = balanceOf() * share / totalShares;
96         uint256 wantBal = IERC20(want).balanceOf(address(this));
97
98         if (wantBal < amount) {
99             IMasterChef(masterchef).withdraw(poolId, amount - wantBal);
100             wantBal = IERC20(want).balanceOf(address(this));
101         }
102
103         if (wantBal > amount) {
104             wantBal = amount;
105         }
106
107         uint256 systemFeeAmount = wantBal * systemFee / 100;
108         uint256 treasuryFeeAmount = wantBal * treasuryFee / 100;
109         uint256 withdrawalAmount = wantBal - systemFeeAmount - treasuryFeeAmount;
110
111         IERC20(want).safeTransfer(account, withdrawalAmount);
112     }
```

Retirestrat: There is a function which would enable the owner to withdraw all the funds from the contract.

```
205     function retireStrat() external {
206         require(msg.sender == vault, "!vault");
207
208         IMasterChef(masterchef).emergencyWithdraw(poolId);
209
210         uint256 wantBal = IERC20(want).balanceOf(address(this));
211         IERC20(want).transfer(vault, wantBal);
212     }
```

File: StratManager.sol

SetWithdrawFees: The owner can set withdraw fees.

SetGravisFee: The owner can set Gravis fees.

SetCallFee: The owner can set call fees.

SetVault: The owner can change the vault address.

SetUniRouter: Update router that will be used for updates.

Owner has full control over the smart contract. Thus, technical auditing does not guarantee the project's ethical side.

Please do your due diligence before investing. Our audit report is never an investment advice.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyse the feasibility of an attack in a live system.

Suggested Solutions

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinised by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

RD Auditors Disclaimer

The smart contracts given for audit have been analysed in accordance with the best industry practices at the date of this report, in relation to: 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).

Because the total number of test cases are unlimited, the audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. 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 security of smart contracts.

Technical Disclaimer

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



Email: info@rdauditors.com

Website: www.rdauditors.com

