QuillAudits

Audit Report
April, 2022

For

FORMATION.FI

Contents

Scope of Audit

The scope of this audit was to analyze and document the Formation.fi smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis
Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption
In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
High	A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.
Medium	The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.
Low	Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.
Informational	These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	2	2	1
Closed	0	1	0	1

Introduction

During the period of **Mar 23, 2022 to April 01 , 2022** - QuillAudits Team performed a security audit for Formation.fi smart contracts.

The code for the audit was taken from following the official link:
https://github.com/Formation-Finance/form-staking

Version No.	Date	Commit ID/URL
01	Mar 23	6ef715111b0b4e9a7c205a38ce23e813f618367f
02	April 1	281ffd2512439ff0f19c89ea4ec2adcace3000b7

Issues Found – Code Review / Manual Testing

High severity issues

No issues found

Medium severity issues

1. Uncheck transfer

Description

The return value of an external transfer/transferFrom call is not checked since the external tokens do not revert in case of failure and return false. We've found the following return values are not checked.

1. L229: form.transfer(_to, formBal);
2. L231: form.transfer(_to, _amount);

Remediation

Please consider adding the require() check for those external calls or using SafeERC20, or ensure that the transfer/transferFrom return value is checked.

Status: Fixed in version 02

2. Inconsistent with code comments

Description

The following was added in the add() function, "XXX DO NOT add the same LP token more than once. Rewards will be messed up if you do". However, the code does not have a check for the same LP token added as recommended by the code comment.

We recommend adding a check to examine if a LP token already exists.

Status: Acknowledged by the auditee

3. Missing range check for Input Variable

Description

The owner can set the BONUS_MULTIPLIER state variable arbitrarily large or small via calling setBonusMultiplier() function, this might cause potential risks in rewarding and anti whale.

Remediation

We recommend setting ranges and check the above input variable.

Status: Acknowledged by the auditee

Low severity issues

4. Lack of event emissions

Description

The missing event makes it difficult to track off-chain liquidity fee changes. An event should be emitted for significant transactions calling the following functions:

- setBonusEndBlock()
- setBonusMultiplier()
- set()
- add()

Remediation

We recommend emitting an event to log the update of the above variables for those above-mentioned functions.

Status: Acknowledged by the auditee

5. Lack of Input Validation

Description

The following variables are not sanitized:

- The “form” token balance of the LPFarmV3 contract is not checked against empty balance in an if statement at line 185
- The user.amount is not checked against empty balance before transferring in the emergencyWithdraw()

Remediation
We recommend adding an empty balance check for the above variables, which helps to avoid the potential for erroneous values to result in unexpected behaviors or wasted gas.

Status: Acknowledged by the auditee

Informational issues

6. Conformance to Solidity naming conventions

Description
Local and State Variable Names should be in mixedCase. Examples: totalSupply, remainingSupply, balancesOf, creatorAddress, isPreSale, tokenExchangeRate.

Therefore, the BONUS_MULTIPLIER should be changed to bonusMultiplier.

Remediation
The Auditee should follow the Solidity naming convention.

Status: Acknowledged by the auditee

7. Public function that could be declared external

Description
The following public functions that are never called by the contract should be declared external to save gas:
- add()
- set()
- deposit()
- withdraw()
- emergencyWithdraw()

Remediation
Use the external attribute for functions never called from the contract.

Status: Fixed in version 02

Functional Testing Results

- bonusEndBlock() should return a correct number
- setBonusEndBlock() should be called only by owner (57ms)
- setBonusEndBlock() should return a correct number set by owner
- BONUS_MULTIPLIER should return correct number before set
- setBonusMultiplier() should be called only by owner
- setBonusMultiplier() should return a correct number set by owner
- add() should be called only by the owner
- add() with block.number > startBlock (41ms)
- add() with block.number <= startBlock (2099ms)
- set() should be called only by the owner
- set() should return a correct value set by the owner
- getMultiplier() should return a correct value if _to <= bonusEndBlock
- getMultiplier() should return a correct value if _from >= bonusEndBlock
- getMultiplier() should return a correct value if _to > bonusEndBlock && _from < bonusEndBlock
- deposit() called from lpowner (61ms)
- lpowner calls deposit() again (132ms)
- pendingForm() should return a correct value when lpSupply != 0
- withdraw() should revert when user.amount >= _amount
- withdraw() half of the amount (80ms)
- emergencyWithdraw() called by lpowner

Automated Tests

Slither

```
INFO:Detectors:
LPFarmingV3.safeFormTransfer(address,uint256) (LPFarmV3.sol#226-233) ignores return value by form.transfer(_to,formBal) (LPFarmV3.sol#229)
LPFarmingV3.safeFormTransfer(address,uint256) (LPFarmV3.sol#226-233) ignores return value by form.transfer(_to,_amount) (LPFarmV3.sol#231)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
LPFarmingV3.pendingForm(uint256,address) (LPFarmV3.sol#121-142) performs a multiplication on the result of a division:
        -formReward = (multiplier * formPerBlock * pool.allocPoint) / totalAllocPoint (LPFarmV3.sol#135-136)
        -accFormPerShare = accFormPerShare + ((formReward * 1e12) / lpSupply) (LPFarmV3.sol#137-139)
LPFarmingV3.updatePool(uint256) (LPFarmV3.sol#161-178) performs a multiplication on the result of a division:
        -formReward = (multiplier * formPerBlock * pool.allocPoint) / totalAllocPoint (LPFarmV3.sol#172-173)
        -pool.accFormPerShare = pool.accFormPerShare + ((formReward * 1e12) / lpSupply) (LPFarmV3.sol#174-176)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
LPFarmingV3.updatePool(uint256) (LPFarmV3.sol#161-178) uses a dangerous strict equality:
        - lpSupply == 0 (LPFarmV3.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in LPFarmingV3.deposit(uint256,uint256) (LPFarmV3.sol#181-198):
        External calls:
        - safeFormTransfer(msg.sender,pending) (LPFarmV3.sol#188)
                - form.transfer(_to,formBal) (LPFarmV3.sol#229)
                - form.transfer(_to,_amount) (LPFarmV3.sol#231)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (LPFarmV3.sol#190-194)
        State variables written after the call(s):
        - user.amount += _amount (LPFarmV3.sol#195)
        - user.rewardDebt = (user.amount * pool.accFormPerShare) / 1e12 (LPFarmV3.sol#196)
Reentrancy in LPFarmingV3.emergencyWithdraw(uint256) (LPFarmV3.sol#216-223):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),user.amount) (LPFarmV3.sol#219)
        State variables written after the call(s):
        - user.amount = 0 (LPFarmV3.sol#221)
        - user.rewardDebt = 0 (LPFarmV3.sol#222)
Reentrancy in LPFarmingV3.withdraw(uint256,uint256) (LPFarmV3.sol#201-213):
        External calls:
        - safeFormTransfer(msg.sender,pending) (LPFarmV3.sol#208)
                - form.transfer(_to,formBal) (LPFarmV3.sol#229)
                - form.transfer(_to,_amount) (LPFarmV3.sol#231)
        State variables written after the call(s):
        - user.amount -= _amount (LPFarmV3.sol#209)
        - user.rewardDebt = (user.amount * pool.accFormPerShare) / 1e12 (LPFarmV3.sol#210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

```
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - FormToken.transferOwnership(address) (FormToken.sol#15-17)
        - Ownable.transferOwnership(address) (@openzeppelin/contracts/access/Ownable.sol#62-65)
name() should be declared external:
        - ERC20.name() (@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
        - ERC20.symbol() (@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
        - ERC20.decimals() (@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)
totalSupply() should be declared external:
        - ERC20.totalSupply() (@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#113-117)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#136-140)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#158-167)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#181-185)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#201-210)
mint(address,uint256) should be declared external:
        - FormToken.mint(address,uint256) (FormToken.sol#11-13)
add(uint256,IERC20,bool) should be declared external:
        - LPFarmingV3.add(uint256,IERC20,bool) (LPFarmV3.sol#64-84)
set(uint256,uint256,bool) should be declared external:
        - LPFarmingV3.set(uint256,uint256,bool) (LPFarmV3.sol#87-100)
deposit(uint256,uint256) should be declared external:
        - LPFarmingV3.deposit(uint256,uint256) (LPFarmV3.sol#181-198)
withdraw(uint256,uint256) should be declared external:
        - LPFarmingV3.withdraw(uint256,uint256) (LPFarmV3.sol#201-213)
emergencyWithdraw(uint256) should be declared external:
        - LPFarmingV3.emergencyWithdraw(uint256) (LPFarmV3.sol#216-223)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
INFO:Detectors:
Parameter LPFarmingV3.add(uint256,IERC20,bool)._allocPoint (LPFarmV3.sol#65) is not in mixedCase
Parameter LPFarmingV3.add(uint256,IERC20,bool)._lpToken (LPFarmV3.sol#66) is not in mixedCase
Parameter LPFarmingV3.add(uint256,IERC20,bool)._withUpdate (LPFarmV3.sol#67) is not in mixedCase
Parameter LPFarmingV3.set(uint256,uint256,bool)._pid (LPFarmV3.sol#88) is not in mixedCase
Parameter LPFarmingV3.set(uint256,uint256,bool)._allocPoint (LPFarmV3.sol#89) is not in mixedCase
Parameter LPFarmingV3.set(uint256,uint256,bool)._withUpdate (LPFarmV3.sol#90) is not in mixedCase
Parameter LPFarmingV3.getMultiplier(uint256,uint256)._from (LPFarmV3.sol#103) is not in mixedCase
Parameter LPFarmingV3.getMultiplier(uint256,uint256)._to (LPFarmV3.sol#103) is not in mixedCase
Parameter LPFarmingV3.pendingForm(uint256,address)._pid (LPFarmV3.sol#121) is not in mixedCase
Parameter LPFarmingV3.pendingForm(uint256,address)._user (LPFarmV3.sol#121) is not in mixedCase
Parameter LPFarmingV3.setBonusEndBlock(uint256)._bonusEndBlock (LPFarmV3.sol#152) is not in mixedCase
Parameter LPFarmingV3.setBonusMultiplier(uint256)._bonusMultiplier (LPFarmV3.sol#156) is not in mixedCase
Parameter LPFarmingV3.updatePool(uint256)._pid (LPFarmV3.sol#161) is not in mixedCase
Parameter LPFarmingV3.deposit(uint256,uint256)._pid (LPFarmV3.sol#181) is not in mixedCase
Parameter LPFarmingV3.deposit(uint256,uint256)._amount (LPFarmV3.sol#181) is not in mixedCase
Parameter LPFarmingV3.withdraw(uint256,uint256)._pid (LPFarmV3.sol#201) is not in mixedCase
Parameter LPFarmingV3.withdraw(uint256,uint256)._amount (LPFarmV3.sol#201) is not in mixedCase
Parameter LPFarmingV3.emergencyWithdraw(uint256)._pid (LPFarmV3.sol#216) is not in mixedCase
Parameter LPFarmingV3.safeFormTransfer(address,uint256)._to (LPFarmV3.sol#226) is not in mixedCase
Parameter LPFarmingV3.safeFormTransfer(address,uint256)._amount (LPFarmV3.sol#226) is not in mixedCase
Variable LPFarmingV3.BONUS_MULTIPLIER (LPFarmV3.sol#25) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable LPFarmingV3.BONUS_MULTIPLIER (LPFarmV3.sol#25) is too similar to LPFarmingV3.setBonusMultiplier(uint256)._bonusMultiplier (LPFarmV3.sol#156)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

```
INFO:Detectors:
Address.functionCall(address,bytes) (@openzeppelin/contracts/utils/Address.sol#85-87) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (@openzeppelin/contracts/utils/Address.sol#114-120) is never used and should be removed
Address.functionDelegateCall(address,bytes) (@openzeppelin/contracts/utils/Address.sol#174-176) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#184-193) is never used and should be removed
Address.functionStaticCall(address,bytes) (@openzeppelin/contracts/utils/Address.sol#147-149) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#157-166) is never used and should be removed
Address.sendValue(address,uint256) (@openzeppelin/contracts/utils/Address.sol#60-65) is never used and should be removed
Context._msgData() (@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
ERC20._burn(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#280-295) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#45-58) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#69-80) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#60-67) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (@openzeppelin/contracts/access/Ownable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/ERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) necessitates a version too recent to be trusted. Consider deploying w
th 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12
0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12
0.7.6
Pragma version^0.8.1 (@openzeppelin/contracts/utils/Address.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (FormToken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (LPFarmV3.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (@openzeppelin/contracts/utils/Address.sol#60-65):
        - (success) = recipient.call{value: amount}() (@openzeppelin/contracts/utils/Address.sol#63)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (@openzeppelin/contracts/utils/Address.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (@openzeppelin/contracts/utils/Address.sol#137)
Low level call in Address.functionStaticCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#157-166):
        - (success,returndata) = target.staticcall(data) (@openzeppelin/contracts/utils/Address.sol#164)
Low level call in Address.functionDelegateCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#184-193):
        - (success,returndata) = target.delegatecall(data) (@openzeppelin/contracts/utils/Address.sol#191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
Reentrancy in LPFarmingV3.deposit(uint256,uint256) (LPFarmV3.sol#181-198):
        External calls:
        - safeFormTransfer(msg.sender,pending) (LPFarmV3.sol#188)
                - form.transfer(_to,formBal) (LPFarmV3.sol#229)
                - form.transfer(_to,_amount) (LPFarmV3.sol#231)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (LPFarmV3.sol#190-194)
        Event emitted after the call(s):
        - Deposit(msg.sender,_pid,_amount) (LPFarmV3.sol#197)
Reentrancy in LPFarmingV3.emergencyWithdraw(uint256) (LPFarmV3.sol#216-223):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),user.amount) (LPFarmV3.sol#219)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,user.amount) (LPFarmV3.sol#220)
Reentrancy in LPFarmingV3.withdraw(uint256,uint256) (LPFarmV3.sol#201-213):
        External calls:
        - safeFormTransfer(msg.sender,pending) (LPFarmV3.sol#208)
                - form.transfer(_to,formBal) (LPFarmV3.sol#229)
                - form.transfer(_to,_amount) (LPFarmV3.sol#231)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (LPFarmV3.sol#211)
        Event emitted after the call(s):
        - Withdraw(msg.sender,_pid,_amount) (LPFarmV3.sol#212)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.verifyCallResult(bool,bytes,string) (@openzeppelin/contracts/utils/Address.sol#201-221) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/Address.sol#213-216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.8.4', '^0.8.0', '^0.8.1']
        - ^0.8.0 (@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
        - ^0.8.1 (@openzeppelin/contracts/utils/Address.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4)
        - 0.8.4 (FormToken.sol#2)
        - 0.8.4 (LPFarmV3.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
```

Solhint Liner

```
LPFarmV3.sol
   3:1  error    Compiler version 0.8.4 does not satisfy the ^0.5.8 semver requirement                          compiler-version
  25:5  warning  Variable name must be in mixedCase                                                            var-name-mixedcase
  46:5  warning  Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)  func-visibility
```

Results

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the Formation.fi platform. We performed our audit according to the procedure described above.

The audit showed several medium, low, and informational severity issues. In the end, the majority of the issues were fixed and acknowledged by the Auditee.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the Formation.fi platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Formation.fi Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
April, 2022

For

FORMATION.FI

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com