Security Assessment

Kalata

Jul 14th, 2021

Table of Contents

MCK-06 : Potential issues when opening position

MCK-07 : Missing Value Check

MCK-08 : "TODO" Comment within Codebase

MCK-09 : Variable Visibility Not Set

OCK-01 : Functions Declared As Virtual

OCK-02 : Missing Event Emission

OCK-03 : Variable Visibility Not Set

SCK-01 : Return Value Not Handled

SCK-02 : Functions Declared As Virtual

SCK-03 : Potential Arithmetic Error

SCK-04 : Privileged Function Access

SCK-05 : Missing Event Emission

SCK-06 : Variable Visibility Not Set

SCK-07 : Potential reward distribution error

Appendix

Disclaimer

About

Summary

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

The auditing process pays special attention to the following considerations:

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

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

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

Overview

Project Summary

Project Name	Kalata
Platform	BSC
Language	Solidity
Codebase	https://github.com/kalata-io/kalata-contracts
Commit	fb8055b0f5a2c07c29fe6dd78d792935a1214e9b

Audit Summary

Delivery Date	Jul 14, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

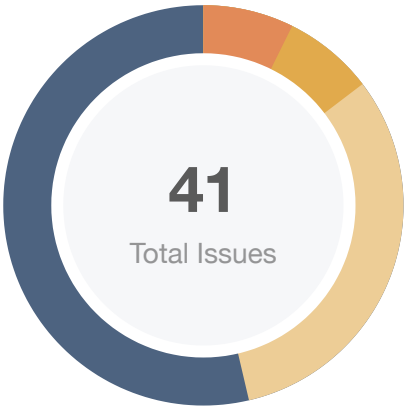
Vulnerability Summary

Vulnerability Level	Total	Pending	Partially Resolved	Resolved	Acknowledged	Declined
Critical	0	0	0	0	0	0
Major	3	0	0	1	2	0
Medium	3	0	0	3	0	0
Minor	13	0	0	11	2	0
Informational	22	0	0	22	0	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	file	SHA256 Checksum

Findings



Critical	**0** (0.00%)
Major	**3** (7.32%)
Medium	**3** (7.32%)
Minor	**13** (31.71%)
Informational	**22** (53.66%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Privileged Ownership	**Centralization / Privilege**	**Minor**	**Acknowledged**
GLOBAL-02	3rd party dependencies	Control Flow	Minor	Acknowledged
ABE-01	Functions Declared As Virtual	Language Specific	Informational	Resolved
BEP-01	Functions Declared As Virtual	Language Specific	Informational	Resolved
CCK-01	Return Value Not Handled	Volatile Code	Informational	Resolved
CCK-02	Functions Declared As Virtual	Language Specific	Informational	Resolved
CCK-03	Missing Event Emission	Volatile Code	Minor	Resolved
CCK-04	Privileged Function Access	**Centralization / Privilege**	**Medium**	**Resolved**
CCP-01	Return Value Not Handled	Volatile Code	Informational	Resolved
CCP-02	Functions Declared As Virtual	Language Specific	Informational	Resolved
CCP-03	Community Tokens	Control Flow	Minor	Resolved
FCK-01	Functions Declared As Virtual	Language Specific	Informational	Resolved
FCK-02	Ambiguous for loop	Coding Style	Minor	Resolved
FCK-03	Unused State Variables	Gas Optimization	Informational	Resolved
FCK-04	Variable Visibility Not Set	Volatile Code	Informational	Resolved

ID	Title	Category	Severity	Status
GCK-01	Return Value Not Handled	Volatile Code	Informational	Resolved
GCK-02	Functions Declared As Virtual	Language Specific	Informational	Resolved
GCK-03	Potential Error Handling Mistake	Data Flow	Minor	Resolved
GCK-04	Logic Mistake	Logical Issue	Medium	Resolved
GCK-05	Missing Event Emission	Volatile Code	Minor	Resolved
GCK-06	Execute and Expire Poll Difference	Coding Style	Minor	Resolved
GCK-07	Unclear Comment within Codebase	Coding Style	Informational	Resolved
MCK-01	Return Value Not Handled	Volatile Code	Informational	Resolved
MCK-02	Functions Declared As Virtual	Language Specific	Informational	Resolved
MCK-03	Misspelled Method Name	Volatile Code	Informational	Resolved
MCK-04	Missing Event Emission	Volatile Code	Minor	Resolved
MCK-05	Potential reentrancy	Volatile Code	Minor	Resolved
MCK-06	Potential issues when opening position	Volatile Code	Minor	Resolved
MCK-07	Missing Value Check	Logical Issue	Major	Acknowledged
MCK-08	"TODO" Comment within Codebase	Coding Style	Informational	Resolved
MCK-09	Variable Visibility Not Set	Volatile Code	Informational	Resolved
OCK-01	Functions Declared As Virtual	Language Specific	Informational	Resolved
OCK-02	Missing Event Emission	Volatile Code	Minor	Resolved
OCK-03	Variable Visibility Not Set	Volatile Code	Informational	Resolved
SCK-01	Return Value Not Handled	Volatile Code	Informational	Resolved
SCK-02	Functions Declared As Virtual	Language Specific	Informational	Resolved
SCK-03	Potential Arithmetic Error	Mathematical Operations	Medium	Resolved

ID	Title	Category	Severity	Status
SCK-04	Privileged Function Access	**Centralization / Privilege**	**Major**	**Resolved**
SCK-05	Missing Event Emission	Volatile Code	Minor	Resolved
SCK-06	Variable Visibility Not Set	Volatile Code	Informational	Resolved
SCK-07	Potential reward distribution error	Control Flow	Major	Acknowledged

GLOBAL-01 | Privileged Ownership

Category	Severity	Location	Status
Centralization / Privilege	Minor	Global	Acknowledged

Description

The owner of contracts is able to :

- Change the address of governance contract, uniswapFactory, `govToken`, `baseToken` and uniswapRouter in the collector contract.
- Update configuration in community contract and send tokens in the community contract to an arbitrary address.
- Change the distribution schedule, create a new `mAssets` and change the token weight in the factory contract.
- Register/Update/Migrate asset in the mint contract.
- Register asset in the oracle contract.
- Set factory address and register asset in the staking contract.

without obtaining consensus from the community.

The price feeder of each asset is able to set the price of the asset without obtaining consensus from the community.

Recommendation

Renounce ownership when it is the right time to do so, or gradually migrate to a timelock plus multisig governing procedure and let the community monitor in respect to transparency considerations.

Alleviation

The team adds a timelock contract, which will be the owner of other contracts.

GLOBAL-02 | 3rd party dependencies

Category	Severity	Location	Status
Control Flow	Minor	Global	Acknowledged

Description

The contract is serving as the underlying entity to interact with the third party PancakeSwap protocols. The scope of the audit would treat those 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties may be compromised and therefore may lead to assets being lost or stolen. Furthermore, the contract uses a third party price oracle. The price feeder could potentially be manipulated by flash-loan attack.

Recommendation

We understand that the business logic of the Kalata protocol requires the interaction with the PancakeSwap protocol for swapping tokens and creating token pairs. We encourage the team to constantly monitor the statuses of those 3rd parties to mitigate side effects when unexpected activities are observed. We also recommend the team makes sure that the price feeder is able to provide an accurate price for the asset.

ABE-01 | Functions Declared As Virtual

Category	Severity	Location	Status
Language Specific	Informational	projects/contracts/implements/AbstractBEP20Token.sol: 26, 228, 250	Resolved

Description

All of these methods had been initialized as virtual, meaning they can be overridden by inheriting contracts to change their behavior.

Recommendation

Please refer to the documentation of the Solidity language to certify that all of them are correctly labeled as “virtual” and should have the ability to be overridden.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

BEP-01 | Functions Declared As Virtual

Category	Severity	Location	Status
Language Specific	Informational	projects/contracts/implements/BEP20Token.sol: 10	Resolved

Description

All of these methods had been initialized as virtual, meaning they can be overridden by inheriting contracts to change their behavior.

Recommendation

Please refer to the documentation of the Solidity language to certify that all of them are correctly labeled as “virtual” and should have the ability to be overridden.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

CCK-01 | Return Value Not Handled

Category	Severity	Location	Status
Volatile Code	Informational	projects/contracts/implements/Collector.sol: 61, 78, 99	Resolved

Description

The return value of the function `IBEP20Token.approve()` and `IBEP20Token.transfer()` are not properly handled in the linked function.

Recommendation

We recommend to use variables to receive the return values of the functions mentioned above and to handle both success and failure cases if needed by the business logic.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

CCK-02 | Functions Declared As Virtual

Category	Severity	Location	Status
Language Specific	Informational	projects/contracts/implements/Collector.sol: 24, 34	Resolved

Description

All of these methods had been initialized as virtual, meaning they can be overridden by inheriting contracts to change their behavior.

Recommendation

Please refer to the documentation of the Solidity language to certify that all of them are correctly labeled as “virtual” and should have the ability to be overridden.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

CCK-03 | Missing Event Emission

Category	Severity	Location	Status
Volatile Code	Minor	projects/contracts/implements/Collector.sol: 30~32	Resolved

Description

There are a bunch of functions can change state variables. However, these functions do not emit events to pass the changes out of chain.

Recommendation

Recommend emitting events, for all the essential state variables that can be changed during runtime.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

CCK-04 | Privileged Function Access

Category	Severity	Location	Status
Centralization / Privilege	Medium	projects/contracts/implements/Collector.sol: 95~103, 51~91	Resolved

Description

The functions `distribute()` and `convert()` can be called by anyone. Although a user cannot directly benefit by calling these functions, he could potentially disrupt the operation of the governance contract. Furthermore, according to the comment, the `distribute()` function is used for supplying trading fee rewards for KALA stakers. The governance contract does not have such functionality.

Recommendation

We recommend restrict access to the linked functions.

Alleviation

[Kalata Team]: Delete Collector.sol, since it will be available in release 2

CCP-01 | Return Value Not Handled

Category	Severity	Location	Status
Volatile Code	Informational	projects/contracts/implements/Community.sol: 45	Resolved

Description

The return value of the function `IBEP20Token.approve()` and `IBEP20Token.transfer()` are not properly handled in the linked function.

Recommendation

We recommend to use variables to receive the return values of the functions mentioned above and to handle both success and failure cases if needed by the business logic.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

CCP-02 | Functions Declared As Virtual

Category	Severity	Location	Status
Language Specific	Informational	projects/contracts/implements/Community.sol: 22	Resolved

Description

All of these methods had been initialized as virtual, meaning they can be overridden by inheriting contracts to change their behavior.

Recommendation

Please refer to the documentation of the Solidity language to certify that all of them are correctly labeled as “virtual” and should have the ability to be overridden.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

CCP-03 | Community Tokens

Category	Severity	Location	Status
Control Flow	Minor	projects/contracts/implements/Community.sol	Resolved

Description

We have noticed that `Governance.sol` does not directly call the `spend()` function in `Community.sol`, and there is no functions in the contract that send tokens to the community contract. How does `Community.sol` receive tokens? How does `Governance.sol` spend tokens?

Alleviation

[Kalata Team]: Delete Community.sol, since it will be available in release 2

FCK-01 | Functions Declared As Virtual

Category	Severity	Location	Status
Language Specific	Informational	projects/contracts/implements/Factory.sol: 61	Resolved

Description

All of these methods had been initialized as virtual, meaning they can be overridden by inheriting contracts to change their behavior.

Recommendation

Please refer to the documentation of the Solidity language to certify that all of them are correctly labeled as “virtual” and should have the ability to be overridden.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

FCK-02 | Ambiguous for loop

Category	Severity	Location	Status
Coding Style	Minor	projects/contracts/implements/Factory.sol: 152~161	Resolved

Description

The second for loop within the `distribute()` method seems redundant and is unnecessary. It could be simplified to

```
1 IBEP20Token(_config.govToken).mint(_config.staking, distributedAmount);
2 IStaking(_config.staking).depositReward(_config.govToken, distributedAmount);
```

Recommendation

Simplify the logic in the function.

Alleviation

[Kalata Team]: This was a bug and has already been fixed

`IStaking(_config.staking).depositReward(_config.govToken, amount);`

Should be:

`IStaking(_config.staking).depositReward(token, amount);"`

FCK-03 | Unused State Variables

Category	Severity	Location	Status
Gas Optimization	Informational	projects/contracts/implements/Factory.sol	Resolved

Description

Variable `KALATA_TOKEN_WEIGHT` is never used in the contract.

Alleviation

[Kalata Team]: Remove KALATA_TOKEN_WEIGHT

FCK-04 | Variable Visibility Not Set

Category	Severity	Location	Status
Volatile Code	Informational	projects/contracts/implements/Factory.sol	Resolved

Description

Variable visibilities are not set.

Recommendation

Although the default variable visibility is set as `internal`, it is best practice to define the visibility explicitly for each variable.

Alleviation

The issue is resolved in commit "f0c70ede2335cc89e1d59aa70b631334a6e6b993"

GCK-01 | Return Value Not Handled

Category	Severity	Location	Status
Volatile Code	Informational	projects/contracts/implements/Governance.sol: 185, 227	Resolved

Description

The return value of the function `IBEP20Token.approve()` and `IBEP20Token.transfer()` are not properly handled in the linked function.

Recommendation

We recommend to use variables to receive the return values of the functions mentioned above and to handle both success and failure cases if needed by the business logic.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

GCK-02 | Functions Declared As Virtual

Category	Severity	Location	Status
Language Specific	Informational	projects/contracts/implements/Governance.sol: 54, 417, 433, 445 , 459	Resolved

Description

All of these methods had been initialized as virtual, meaning they can be overridden by inheriting contracts to change their behavior.

Recommendation

Please refer to the documentation of the Solidity language to certify that all of them are correctly labeled as “virtual” and should have the ability to be overridden.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

GCK-03 | Potential Error Handling Mistake

Category	Severity	Location	Status
Data Flow	Minor	projects/contracts/implements/Governance.sol: 75~115	Resolved

Description

In the function `createPoll()`, if the `pollId` exists, the poll won't created. The poll creator then would not get his deposit back, as the function continues and executes successfully.

Recommendation

The function should revert to a state before the deposit upon failure.

Alleviation

[Kalata Team]: Delete Governance.sol, since it will be available in release 2

GCK-04 | Logic Mistake

Category	Severity	Location	Status
Logical Issue	Medium	projects/contracts/implements/Governance.sol: 231	Resolved

Description

In the endPoll() function, the `totalDeposit` is subtracted even if the deposit is not returned to the poll creator.

Recommendation

We recommend the team to further explain the plan on handling deposits that are not returned to the pool creator.

Alleviation

[Kalata Team]: Delete Governance.sol, since it will be available in release 2

GCK-05 | Missing Event Emission

Category	Severity	Location	Status
Volatile Code	Minor	projects/contracts/implements/Governance.sol: 148~157, 160~186, 248~259, 261~268	Resolved

Description

There are a bunch of functions can change state variables. However, these functions do not emit events to pass the changes out of chain.

Recommendation

Recommend emitting events, for all the essential state variables that can be changed during runtime.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

GCK-06 | Execute and Expire Poll Difference

Category	Severity	Location	Status
Coding Style	Minor	projects/contracts/implements/Governance.sol	Resolved

Description

A passed poll can either be set to executed and expired by calling function `executePoll` and `expirePoll` respectively.

Recommendation

We recommend clarifying when a poll should be set to executed and when a poll should be set to expired.

Alleviation

[Kalata Team]: Delete Governance.sol, since it will be available in release 2

GCK-07 | Unclear Comment within Codebase

Category	Severity	Location	Status
Coding Style	Informational	projects/contracts/implements/Governance.sol: 122	Resolved

Description

We had found an unclear comment within the codebase that needs clarification:

```
// user->KalaToken.tansfer(govAddress,amount) -> forward to this method”
```

Alleviation

[Kalata Team]: Delete Governance.sol, since it will be available in release 2

MCK-01 | Return Value Not Handled

Category	Severity	Location	Status
Volatile Code	Informational	projects/contracts/implements/Mint.sol: 219, 222, 298	Resolved

Description

The return value of the function `IBEP20Token.approve()` and `IBEP20Token.transfer()` are not properly handled in the linked function.

Recommendation

We recommend to use variables to receive the return values of the functions mentioned above and to handle both success and failure cases if needed by the business logic.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

MCK-02 | Functions Declared As Virtual

Category	Severity	Location	Status
Language Specific	Informational	projects/contracts/implements/Mint.sol: 44, 86, 304, 460, 464, 469, 480, 490, 494, 498, 502, 506, 510, 524, 538	Resolved

Description

All of these methods had been initialized as virtual, meaning they can be overridden by inheriting contracts to change their behavior.

Recommendation

Please refer to the documentation of the Solidity language to certify that all of them are correctly labeled as “virtual” and should have the ability to be overridden.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

MCK-03 | Misspelled Method Name

Category	Severity	Location	Status
Volatile Code	Informational	projects/contracts/implements/Mint.sol: 524, 538	Resolved

Description

The method name deviates from what is expected due to a spelling error.

Recommendation

Method names should be consistent with the English language.

Alleviation

[Kalata Team]: Fix spelling error

MCK-04 | Missing Event Emission

Category	Severity	Location	Status
Volatile Code	Minor	projects/contracts/implements/Mint.sol: 98~104	Resolved

Description

There are a bunch of functions can change state variables. However, these functions do not emit events to pass the changes out of chain.

Recommendation

Recommend emitting events, for all the essential state variables that can be changed during runtime.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

MCK-05 | Potential reentrancy

Category	Severity	Location	Status
Volatile Code	Minor	projects/contracts/implements/Mint.sol	Resolved

Description

Function `deposit` and `openPosition` in the mint contract risk being vulnerable to a reentrancy attack. State variables are massively changed later in the external call of `IBEP20Token().mint` and `IERC20().transferFrom`. Since the real implementation of the external functions are unclear, and the address behind the interface is unclear, a reentrancy problem and attack may take place.

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts.

Alleviation

[Kalata Team]: Add ReentrancyGuard

MCK-06 | Potential issues when opening position

Category	Severity	Location	Status
Volatile Code	Minor	projects/contracts/implements/Mint.sol: 115~153	Resolved

Description

When a user opens a new position by calling function `openPosition`, array `postionIdxArray` is not updated. This may cause issues when querying positions by calling function `queryAllPositions` or `queryPositions`.

Recommendation

Update array `postionIdxArray` when opening a new position.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

MCK-07 | Missing Value Check

Category	Severity	Location	Status
Logical Issue	Major	projects/contracts/implements/Mint.sol: 216	Acknowledged

Description

In the function `withdraw` within the mint contract, the `protocolFeeRate` is not checked for validity, this may cause unintended asset losses to the users of the contract.

Recommendation

There should be a check for the protocol fee rate to be an allowed value, such as smaller than a reasonable amount.

Alleviation

[Kalata Team]: make sure `config.protocolFeeRate>0`.

However, we still recommend setting a reasonable upper limit for the fee rate.

MCK-08 | "TODO" Comment within Codebase

Category	Severity	Location	Status
Coding Style	Informational	projects/contracts/implements/Mint.sol: 287~288	Resolved

Description

The following comment was found within the codebase:

```
//require(assetConfig.endPrice > 0, "Asset is not in deprecated state");```
```

Recommendation

We recommend to delete unnecessary comments.

Alleviation

[Kalata Team]: delete unnecessary comments

MCK-09 | Variable Visibility Not Set

Category	Severity	Location	Status
Volatile Code	Informational	projects/contracts/implements/Mint.sol	Resolved

Description

Variable visibilities are not set.

Recommendation

Although the default variable visibility is set as `internal`, it is best practice to define the visibility explicitly for each variable.

Alleviation

The issue is resolved in commit "f0c70ede2335cc89e1d59aa70b631334a6e6b993"

OCK-01 | Functions Declared As Virtual

Category	Severity	Location	Status
Language Specific	Informational	projects/contracts/implements/Oracle.sol: 29, 51, 76	Resolved

Description

All of these methods had been initialized as virtual, meaning they can be overridden by inheriting contracts to change their behavior.

Recommendation

Please refer to the documentation of the Solidity language to certify that all of them are correctly labeled as “virtual” and should have the ability to be overridden.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

OCK-02 | Missing Event Emission

Category	Severity	Location	Status
Volatile Code	Minor	projects/contracts/implements/Oracle.sol: 40~45, 47~49, 58~64	Resolved

Description

There are a bunch of functions can change state variables. However, these functions do not emit events to pass the changes out of chain.

Recommendation

Recommend emitting events, for all the essential state variables that can be changed during runtime.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

OCK-03 | Variable Visibility Not Set

Category	Severity	Location	Status
Volatile Code	Informational	projects/contracts/implements/Oracle.sol	Resolved

Description

Variable visibilities are not set.

Recommendation

Although the default variable visibility is set as `internal`, it is best practice to define the visibility explicitly for each variable.

Alleviation

The issue is resolved in commit "f0c70ede2335cc89e1d59aa70b631334a6e6b993"

SCK-01 | Return Value Not Handled

Category	Severity	Location	Status
Volatile Code	Informational	projects/contracts/implements/Staking.sol: 121, 157	Resolved

Description

The return value of the function `IBEP20Token.approve()` and `IBEP20Token.transfer()` are not properly handled in the linked function.

Recommendation

We recommend to use variables to receive the return values of the functions mentioned above and to handle both success and failure cases if needed by the business logic.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

SCK-02 | Functions Declared As Virtual

Category	Severity	Location	Status
Language Specific	Informational	projects/contracts/implements/Staking.sol: 41	Resolved

Description

All of these methods had been initialized as virtual, meaning they can be overridden by inheriting contracts to change their behavior.

Recommendation

Please refer to the documentation of the Solidity language to certify that all of them are correctly labeled as “virtual” and should have the ability to be overridden.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

SCK-03 | Potential Arithmetic Error

Category	Severity	Location	Status
Mathematical Operations	Medium	projects/contracts/implements/Staking.sol: 74~75	Resolved

Description

An arithmetic or logical error when trying to calculate `reward.pendingReward` results in a constant value of 0. `reward.index` is subtracted from `pool.rewardIndex` which would result in 0 as in the line before it was established that they are equal. This would then result in the multiplication to be 0.

Recommendation

Check the logic flow of the contract and establish what `reward.pendingReward` should be.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

SCK-04 | Privileged Function Access

Category	Severity	Location	Status
Centralization / Privilege	Major	projects/contracts/implements/Staking.sol: 132~143	Resolved

Description

The `depositReward()` function can be called by anyone. An attacker could potentially manipulate the reward without sending tokens to the contract.

Recommendation

Restrict access to function `depositReward()`.

Alleviation

[Kalata Team]: This is a bug and it is fixed, add privilege checking logic:

```
require(_config.factory == msg.sender, ""unauthorized"");
```

SCK-05 | Missing Event Emission

Category	Severity	Location	Status
Volatile Code	Minor	projects/contracts/implements/Staking.sol: 132~143	Resolved

Description

There are a bunch of functions can change state variables. However, these functions do not emit events to pass the changes out of chain.

Recommendation

Recommend emitting events, for all the essential state variables that can be changed during runtime.

Alleviation

The issue is resolved in commit "fb8055b0f5a2c07c29fe6dd78d792935a1214e9b".

SCK-06 | Variable Visibility Not Set

Category	Severity	Location	Status
Volatile Code	Informational	projects/contracts/implements/Staking.sol	Resolved

Description

Variable visibilities are not set.

Recommendation

Although the default variable visibility is set as `internal`, it is best practice to define the visibility explicitly for each variable.

Alleviation

The issue is resolved in commit "f0c70ede2335cc89e1d59aa70b631334a6e6b993"

SCK-07 | Potential reward distribution error

Category	Severity	Location	Status
Control Flow	Major	projects/contracts/implements/Staking.sol	Acknowledged

Description

The reward distribution from the factory contract to the staking contract is as follows: The `distribute` function is called every 2 hours by the backend, which will send the staking reward to the staking contract. When the reward is sent, the `_stakes[assetToken]` variable is updated, which also updates user rewards. If an attacker knows when the `distribute` function is called, he could potentially deposit many tokens to the staking pool right before the function is called and withdraw all of his tokens from the pool as soon as the reward is updated. If he has a large number of tokens, he could claim the majority of the reward. Normal users would have far fewer rewards than they are supposed to.

Recommendation

We recommend using timestamp or block number to calculate the reward.

Appendix

Finding Categories

Centralization / Privilege

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

Gas Optimization

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

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

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

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

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

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in-storage one.

Language Specific

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

Coding Style

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

Checksum Calculation Method

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

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

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

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

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

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

About

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