Code Security Assessment

AirCash Finance

Jan 22nd, 2022

Table of Contents

Summary

Overview

- Project Summary
- Audit Summary
- Vulnerability Summary
- Audit Scope

Findings

- AirCash.finance-01 : Financial Models
- AirCash.finance-02 : Financial Models
- ACC-01 : Tautology or contradiction
- ACC-02 : Improper return value
- ACC-03 : Missing Input Validation
- ACC-04 : Unnecessary abicoder v2 pragma
- ACC-05 : Missing emit events
- ASA-01 : Logic issue in `changeHandler()`
- OSA-01 : Redundant Code Components
- RSA-01 : Centralization Risk in RecordStorage.sol
- RSA-02 : Unused Variables
- RSA-03 : Improper Usage of public and external type
- RSA-04 : Inaccurate function name
- RSA-05 : Logic issue in `subWitnessAvailable`
- RSA-06 : Logic issue in `applyWithdraw`
- RSA-07 : Privileged Ownership
- RSA-08 : `authFromContract()` Function Not Restricted
- USA-01 : Missing error messages

Appendix

Disclaimer

About

Summary

This report has been prepared for AirCash Finance to discover issues and vulnerabilities in the source code of the AirCash Finance 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;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	AirCash Finance
Description	AirCash is the first and largest decentralized OTC platform in the galaxy. Buy and sell crypto with fiat money in a decentralized way.
Platform	bsc
Language	Solidity
Codebase	https://github.com/Aircoin-official/AirCash
Commit	35240a26fa297b14ee1029f86d4504b39c7faec5

Audit Summary

Delivery Date	Jan 22, 2022
Audit Methodology	Static Analysis, Manual Review

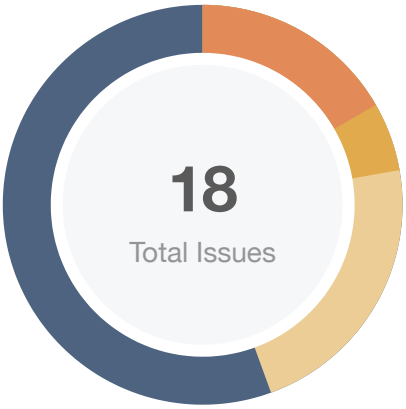
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	3	0	0	1	2	0
Medium	1	0	0	1	0	0
Minor	4	0	0	4	0	0
Informational	10	0	0	10	0	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
ASA	AppealStorage.sol	26f883dc69d0a46da74ff8a450016a22ebbde9a318a3473ea70987a11087b70c
OSA	OrderStorage.sol	717dbf7feb92dead7fb28158a452be1a4658294492e890e4c870fb96cf34911b
RSA	RecordStorage.sol	916c1ad200901177ef7dfa22dc0ac54057ea2bfada5a1d67a0eda182dcc28ed3
RSC	RestStorage.sol	710c991526f96aeae9e97f50aa13170bbf4d032805c9ec8c1b05392b170e4beb
USA	UserStorage.sol	7f16a32ad2eac9b28e1726b404ef0e9705a82d51718582616a13569d203a5226

Findings



Critical	**0** (0.00%)
Major	**3** (16.67%)
Medium	**1** (5.56%)
Minor	**4** (22.22%)
Informational	**10** (55.56%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
AirCash.finance-01	Financial Models	Logical Issue	Minor	Acknowledged
AirCash.finance-02	Financial Models	Logical Issue	Informational	Acknowledged
ACC-01	Tautology or contradiction	Mathematical Operations	Informational	Acknowledged
ACC-02	Improper return value	Gas Optimization	Informational	Acknowledged
ACC-03	Missing Input Validation	Volatile Code	Minor	Acknowledged
ACC-04	Unnecessary abicoder v2 pragma	Language Specific	Informational	Acknowledged
ACC-05	Missing emit events	Coding Style	Informational	Acknowledged
ASA-01	Logic issue in `changeHandler()`	Logical Issue	Minor	Acknowledged
OSA-01	Redundant Code Components	Volatile Code	Informational	Acknowledged
RSA-01	Centralization Risk in RecordStorage.sol	**Centralization / Privilege**	**Major**	Partially Resolved
RSA-02	Unused Variables	Gas Optimization	Informational	Acknowledged
RSA-03	Improper Usage of public and external type	Gas Optimization	Informational	Acknowledged
RSA-04	Inaccurate function name	Gas Optimization	Informational	Acknowledged

ID	Title	Category	Severity	Status
RSA-05	Logic issue in `subWitnessAvailable`	Logical Issue	Major	Acknowledged
RSA-06	Logic issue in `applyWithdraw`	Logical Issue	Medium	Acknowledged
RSA-07	Privileged Ownership	**Centralization / Privilege**	**Major**	Partially Resolved
RSA-08	`authFromContract()` Function Not Restricted	Control Flow	Minor	Acknowledged
USA-01	Missing error messages	Coding Style	Informational	Acknowledged

AirCash.finance-01 | Financial Models

Category	Severity	Location	Status
Logical Issue	Minor	Global	Acknowledged

Description

Regarding the AirCash project, there are some questions that I hope can be answered.

1. The **setPaidMoney** method in the OrderStorage.sol contract is not fully implemented, and there is not any implementation of buyer payment and seller collection.
2. In the RecordStorage.sol contract, users can stake other types of funds through the **tokenEscrow** method, but the method **applyUnfrozen** can only withdraw AIR token, so how to redeem other types of tokens?
3. There is a limit on the total number of congresses on the official website document, but there is no relevant limit in the contract. In addition, can the authority of the congress be supervised?
4. The **chanRole** method is to update the user role after the user's AIR balance is changed, but from the perspective of the code logic, the user role can only be changed to a lower level, but not to a higher level. Why is this designed?

Recommendation

Financial models of blockchain protocols need to be resilient to attacks. They need to pass simulations and verifications to guarantee the security of the overall protocol.

The financial model of this protocol is not in the scope of this audit.

Alleviation

[AirCash.finance Team]:

1. The setPaidMoney method is triggered after the buyer pays the fiat currency offline. The payment and collection are all performed offline, so the function of payment and collection will not be implemented in the method.
2. Currently, only AIR is used for staking and withdrawal. Other types of assets are transferred in the order, and the tokens will be transferred to the buyer's wallet address directly after the order is completed.

AirCash.finance-02 | Financial Models

Category	Severity	Location	Status
Logical Issue	Informational	Global	Acknowledged

Description

AIRCash is a decentralized order trading platform. Rest users create pre-requirements for sell or buy, and Order users select the corresponding rest requirements to match transactions. The seller in the transaction party needs to pledge the pre-sale coins to the current platform in advance.

After the transaction is completed, both parties to the transaction can file an appeal against the transaction. The witness user will be the first reviewer, and the congress user will be the final reviewer. And according to the appeal results, the credit limit and transaction data of both parties are updated, and rewards and punishments are given to witness and congress.

Considering that the core transaction transfer part is completed off-chain, the premise of our audit is that the logic of this part is complete and reliable.

Recommendation

Financial models of blockchain protocols need to be resilient to attacks. They need to pass simulations and verifications to guarantee the security of the overall protocol.

The financial model of this protocol is not in the scope of this audit.

Alleviation

AirCash team acknowledged this finding.

ACC-01 | Tautology or contradiction

Category	Severity	Location	Status
Mathematical Operations	Informational	RestStorage.sol: 294 OrderStorage.sol: 280 UserStorage.sol: 256	Acknowledged

Description

Detects expressions that are tautologies or contradictions.

Suppose a variable `x` is a uint256, then `x` >= 0 will be always true.

Recommendation

Fix the incorrect comparison by changing the value type or the comparison.

Alleviation

AirCash team acknowledged this finding.

ACC-02 | Improper return value

Category	Severity	Location	Status
Gas Optimization	Informational	OrderStorage.sol: 545, 559 UserStorage.sol: 277	Acknowledged

Description

The return results of these linked methods may not be continuous as there are some empty values in the return arrays. Which can be optimized.

Recommendation

It is recommended to use the push method to add the matched values to the array to ensure the continuity of array values.

Alleviation

AirCash team acknowledged this finding.

ACC-03 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	AppealStorage.sol: 63 OrderStorage.sol: 85, 97 RecordStorage.sol: 367, 368, 369, 370 RestStorage.sol: 72, 84 UserStorage.sol: 63, 64, 65, 66	Acknowledged

Description

The given inputs are missing the check for the non-zero address.

Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below:

```
require(_remoteAddr != address(0), "zero address");
```

Alleviation

AirCash team acknowledged this finding.

ACC-04 | Unnecessary abicoder v2 pragma

Category	Severity	Location	Status
Language Specific	Informational	OrderStorage.sol: 3 AppealStorage.sol: 3 RecordStorage.sol: 3 RestStorage.sol: 3 UserStorage.sol: 3	Acknowledged

Description

The linked line explicitly specifies `pragma abicoder v2`, which is unnecessary due to the feature being enabled by default in Solidity v0.8.0 and above.

Recommendation

Consider removing the explicit `pragma abicoder v2` specification on the linked line.

Alleviation

AirCash team acknowledged this finding.

ACC-05 | Missing emit events

Category	Severity	Location	Status
Coding Style	Informational	RecordStorage.sol: 82~84, 105~107, 113~115, 121~123, 129~131, 137~139, 145~147, 153~155, 161~163, 169~171, 177~179, 185~187, 193~195, 201~203, 209~211, 217~219, 225~227 OrderStorage.sol: 52~54	Acknowledged

Description

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

Recommendation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Alleviation

AirCash team acknowledged this finding.

ASA-01 | Logic issue in `changeHandler()`

Category	Severity	Location	Status
Logical Issue	Minor	AppealStorage.sol: 232	Acknowledged

Description

The `changeHandler` method has no limited caller, which may cause Appeal to fail.

Recommendation

A restriction should be added in this method.

Alleviation

[AirCash.finance Team]:

A restriction will be added in subsequent iterations that only buyers and sellers can call this method. Currently this method does not lead to financial security issues.

OSA-01 | Redundant Code Components

Category	Severity	Location	Status
Volatile Code	Informational	OrderStorage.sol: 90~93	Acknowledged

Description

The linked statements do not affect the functionality of the codebase and appear to be either leftovers from test code or older functionality.

Recommendation

We advise to remove the redundant statements for production environments.

Alleviation

AirCash team acknowledged this finding.

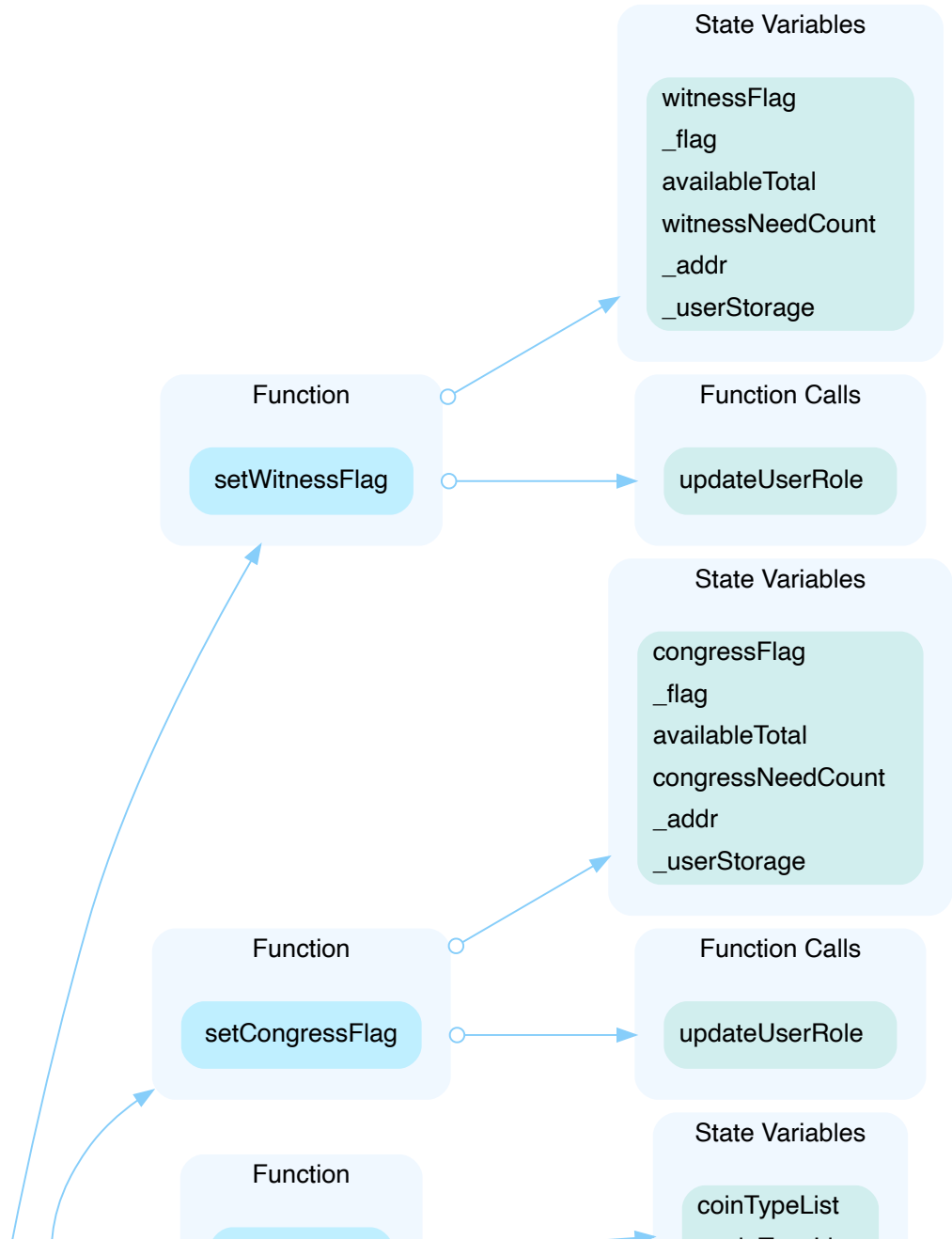
RSA-01 | Centralization Risk in RecordStorage.sol

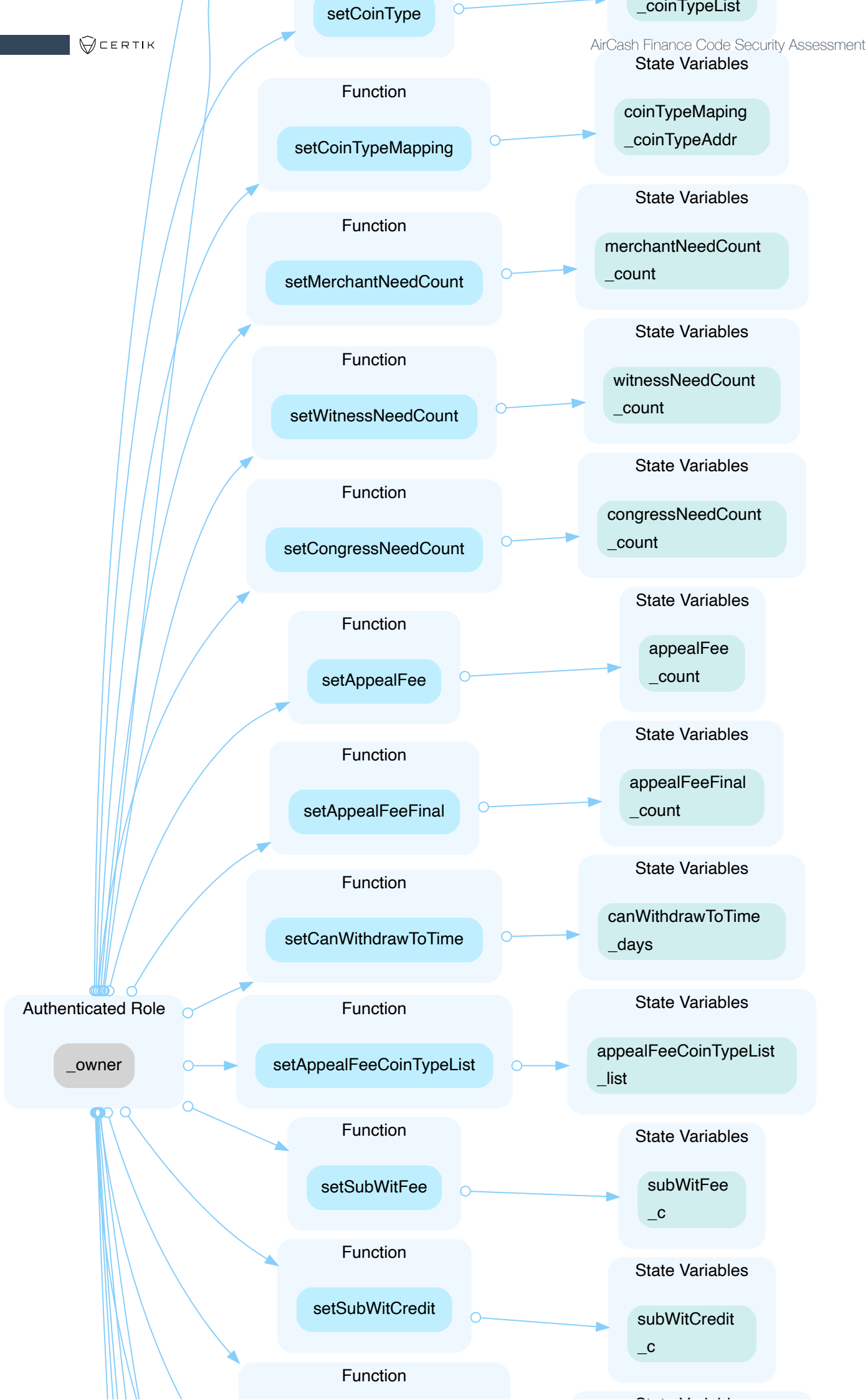
Category	Severity	Location	Status
Centralization / Privilege	Major	RecordStorage.sol: 52~61, 67~76, 82~84, 90~95, 105~107, 113~115, 121~123, 129~131, 137~139, 145~147, 153~155, 161~163, 169~171, 177~179, 185~187, 193~195, 201~203, 209~211, 217~219, 225~227, 233~258	Partially Resolved

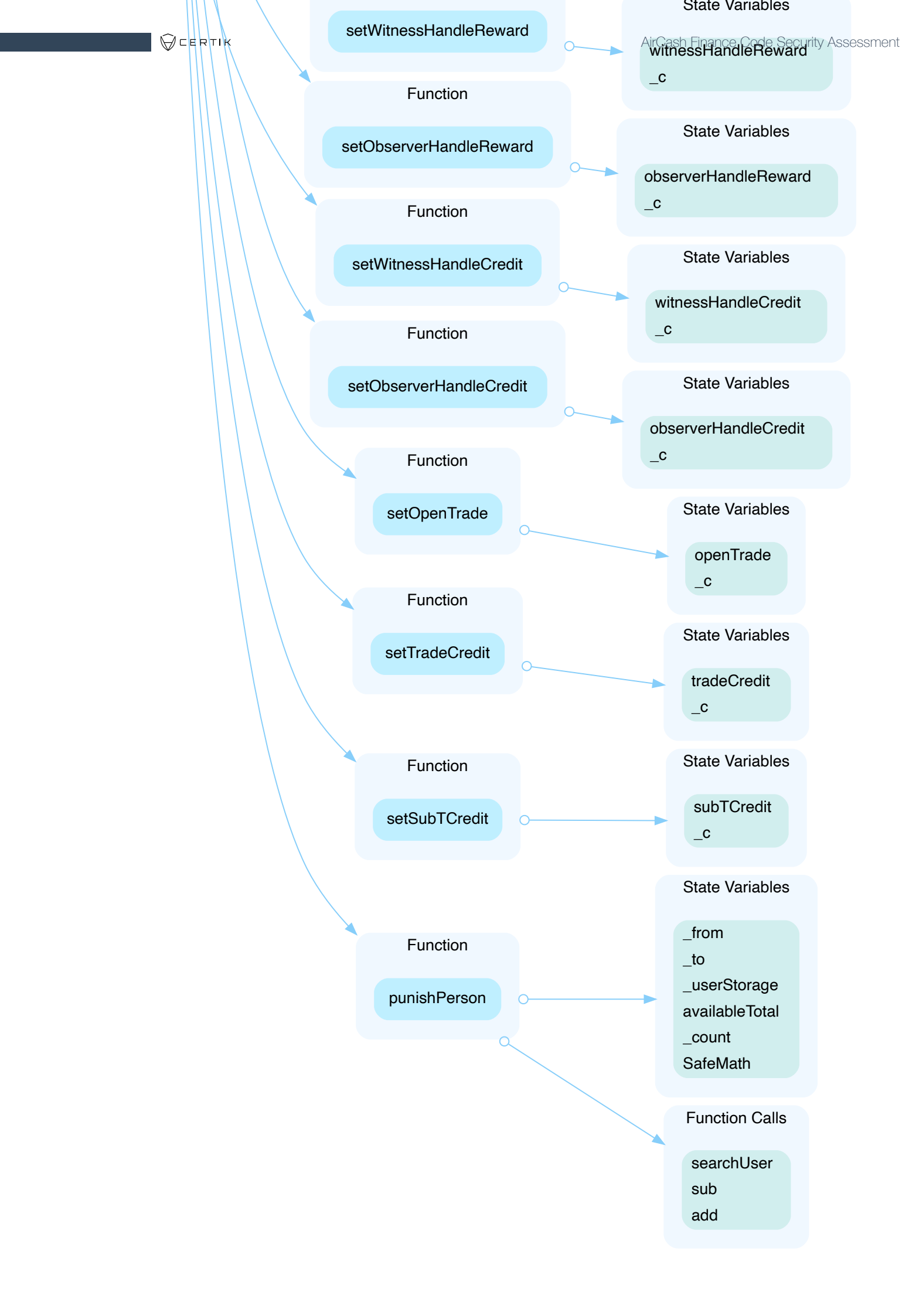
Description

In the contract, `RecordStorage`, the role, `_owner`, has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `_owner` may allow the hacker to take advantage of this.




setCoinType
_coinTypeList
Function
setCoinTypeMapping
State Variables
coinTypeMaping
_coinTypeAddr
Function
setMerchantNeedCount
State Variables
merchantNeedCount
_count
Function
setWitnessNeedCount
State Variables
witnessNeedCount
_count
Function
setCongressNeedCount
State Variables
congressNeedCount
_count
Function
setAppealFee
State Variables
appealFee
_count
Function
setAppealFeeFinal
State Variables
appealFeeFinal
_count
Function
setCanWithdrawToTime
State Variables
canWithdrawToTime
_days
Authenticated Role
_owner
Function
setAppealFeeCoinTypeList
State Variables
appealFeeCoinTypeList
_list
Function
setSubWitFee
State Variables
subWitFee
_c
Function
setSubWitCredit
State Variables
subWitCredit
_c
Function


setWitnessHandleReward
State Variables
witnessHandleReward
_c
Function
setObserverHandleReward
State Variables
observerHandleReward
_c
Function
setWitnessHandleCredit
State Variables
witnessHandleCredit
_c
Function
setObserverHandleCredit
State Variables
observerHandleCredit
_c
Function
setOpenTrade
State Variables
openTrade
_c
Function
setTradeCredit
State Variables
tradeCredit
_c
Function
setSubTCredit
State Variables
subTCredit
_c
Function
punishPerson
State Variables
_from
_to
_userStorage
availableTotal
_count
SafeMath
Function Calls
searchUser
sub
add

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR

- Remove the risky functionality.

Alleviation

Currently, this contract `RecordStorage` has been deployed at address `0xC32a027b45536809fbc50Dd8248F0201E349e598` on BSC chain. The owner of this contract is `0x75b297b807C79e69a57d1b8CAB9dada5A1781D2e` which is a Gnosis Multi sig (⅔). And the managers' addresses of this multis-sig wallet are as following list:

- 0xffa675ecF84e015c4E50cc42535547cCA6A2D08B
- 0x5fe0E48a2850da695680099D7196bBE3a70eb288
- 0x31347c59AbAB33DB91A77831a1BD3F71e5D82ea8

RSA-02 | Unused Variables

Category	Severity	Location	Status
Gas Optimization	Informational	RecordStorage.sol: 30, 35	Acknowledged

Description

The linked state variables are never truly used within this contract.

Recommendation

We advise removing the unused state variables.

Alleviation

AirCash team acknowledged this finding.

RSA-03 | Improper Usage of public and external type

Category	Severity	Location	Status
Gas Optimization	Informational	RecordStorage.sol: 82, 153	Acknowledged

Description

public functions that are never called by the contract could be declared external. When the inputs are arrays external functions are more efficient than `public` functions.

Recommendation

Consider using the external attribute for functions never called from the contract.

Alleviation

AirCash team acknowledged this finding.

RSA-04 | Inaccurate function name

Category	Severity	Location	Status
Gas Optimization	Informational	RecordStorage.sol: 307	Acknowledged

Description

The function of the method `setERC20Address` is inconsistent with its name. It should be named `getERC20Address`.

Recommendation

We recommend naming the function accurately.

Alleviation

AirCash team acknowledged this finding.

RSA-05 | Logic issue in `subWitnessAvailable`

Category	Severity	Location	Status
Logical Issue	Major	RecordStorage.sol: 625	Acknowledged

Description

The following code is used to calculate the deduction from the balance to be withdrawn when the available balance of the witness user is deducted and the balance is insufficient.

```
622 uint256 _draing = withdrawingTotal[_addr]["AIR"];
623 if (SafeMath.add(_availableTotal, _draing) >= subWitFee) {
624   _need = subWitFee;
625   subFromDraing = subWitFee - _availableTotal - _draing;
626   withdrawingTotal[_addr]["AIR"] = SafeMath.sub(
627     withdrawingTotal[_addr]["AIR"],
628     subFromDraing
629   );
630 }
```

Through analysis, there is a problem with the calculation of line 625, it should be:

```
subFromDraing = subWitFee - _availableTotal
```

Recommendation

It is suggested to review the logic of this part and correct the code.

Alleviation

AirCash team acknowledged this finding.

RSA-06 | Logic issue in `applyWithdraw`

Category	Severity	Location	Status
Logical Issue	Medium	RecordStorage.sol: 749	Acknowledged

Description

The method `applyWithdraw` to be called by any account to withdraw one's staked AIR token, but the variable `subFromDraing` is a global variable, which is meaningless here.

Recommendation

We recommend correcting the code logic.

Alleviation

AirCash team acknowledged this finding.

RSA-07 | Privileged Ownership

Category	Severity	Location	Status
Centralization / Privilege	**Major**	RecordStorage.sol: 233~237	Partially Resolved

Description

The `onlyOwner` account can transfer any amount of AIR token from witnesses or congresses to any address by calling the method `punishPerson()`.

Any compromise to the `onlyOwner` account may allow the hacker to take advantage of this and make the contract invoke malicious code, steal funds from normal accounts.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Alleviation

Currently, this contract `RecordStorage` has been deployed at address `0xC32a027b45536809fbc50Dd8248F0201E349e598` on BSC chain. The owner of this contract is `0x75b297b807C79e69a57d1b8CAB9dada5A1781D2e` which is a Gnosis Multi sig (⅔). And the managers' addresses of this multis-sig wallet are as following list:

- 0xffa675ecF84e015c4E50cc42535547cCA6A2D08B
- 0x5fe0E48a2850da695680099D7196bBE3a70eb288
- 0x31347c59AbAB33DB91A77831a1BD3F71e5D82ea8

RSA-08 | `authFromContract()` Function Not Restricted

Category	Severity	Location	Status
Control Flow	Minor	RecordStorage.sol: 357~362	Acknowledged

Description

This method is used to set the addresses of the other four contracts. Since it is declared as `external`, has no access restrictions, and can only be called once, hackers may call this method before the deployer to disrupt the normal use of the whole contract.

Recommendation

We recommend incorporating this method into the `constructor()`.

Alleviation

AirCash team acknowledged this finding.

USA-01 | Missing error messages

Category	Severity	Location	Status
Coding Style	Informational	UserStorage.sol: 200	Acknowledged

Description

The **require** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advise adding error messages to the linked **require** statements.

Alleviation

AirCash team acknowledged this finding.

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.

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 you ("Customer" or 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, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

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.