CERTIK

Security Assessment

kassandra

Oct 31st, 2021

Table of Contents

About

Summary

This report has been prepared for kassandra to discover issues and vulnerabilities in the source code of the kassandra 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	kassandra
Description	Kassandra Core and Governance
Platform	BSC
Language	Solidity
Codebase	https://github.com/KassandraFinance/kassandra-governance/tree/b937a9e3e50ef4e47076c9dc0dd4609c6fc189e4 https://github.com/KassandraFinance/kassandra-core/tree/f42068005abad5a4d7a63b32c7469a830c9ef693
Commit	

Audit Summary

Delivery Date	Oct 31, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

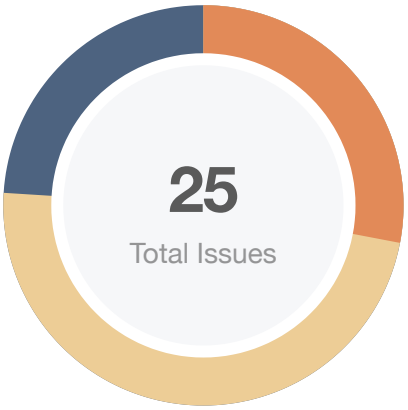
Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	7	0	0	5	1	1
Medium	0	0	0	0	0	0
Minor	12	0	0	0	0	12
Informational	6	0	0	0	0	6
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
FCK	core-contracts/contracts/core/Factory.sol	a81146e8911a9f20c6f13facb947dabb411b5bfb2fa0c496fdabdec093c33a07
MCK	core-contracts/contracts/core/Math.sol	fffc386fcf05b4f2d054de2374d378a6cffae1b92a47bb081315230f56618316
PCK	core-contracts/contracts/core/Pool.sol	33744823490 81e6fb58778d4b827eee55348e3aca3699f272d948bbe9d05828c
TCK	core-contracts/contracts/core/Token.sol	af23f164d6bc397c410aae4e809712aa2682fda0bfc589663cfd99cbafbab623
OCK	core-contracts/contracts/utils/Ownable.sol	dd83d1040d7d875c4a1e85dd0b3ecb8f8511b385225ca5439637465 12ad3996c
RGC	core-contracts/contracts/utils/ReentrancyGuard.sol	9d4f94e4cd876dd2c4e1b8fc7208cfb633ae80c597a7bd8a9fe53eafe8b4fcd7
CRP	core-contracts/contracts/CRPFactory.sol	de9b2592264c604f326b5d1702ffe208987822edf3fabeeb2dc24f47e2b4452c
CRC	core-contracts/contracts/ConfigurableRightsPool.sol	902ed2dee78cd646c96cd527e1a01c0b0619fef232bb33ade2238702ad585f49
PCT	core-contracts/contracts/PCToken.sol	ffce70d3877487827fea44c71844c830c1bfd205feaf805c911cd7d9f910393c
KCC	core-contracts/libraries/KassandraConstants.sol	721bc03c46757f6c85bbac3852dd19a7c6dadb4c0019e74df82bc69f7911351a
KSM	core-contracts/libraries/KassandraSafeMath.sol	f4c8cdf7c67afb4afde39f46be371964a0955c58e37a76721db6ba2309217451
RMC	core-contracts/libraries/RightsManager.sol	099302e139d96c110b503c7738bf979b3415f021ee8d907d583936f31bc25704
SAC	core-contracts/libraries/SafeApprove.sol	2a7dbc600e87b3f09c14b766815ad1bfe48765c932a19832049074be5bfe3e1b
SPM	core-contracts/libraries/SmartPoolManager.sol	9612874769d6faf89a426f56271c0065b783ee84a9e161c6362288a5890d198a
HEI	core-contracts/strategies/HEIM.sol	6bdf31e99a91bd3177b01f6a2b76ac75a44f746df1cc94422f0b992f82e1a2c4

ID	File	SHA256 Checksum
GAC	governance-contracts/GovernorAlpha.sol	2f596163adc749c80820132bb80ab3dc1436b87714545283d322b85966c16ef2
KCK	governance-contracts/Kacy.sol	ad4c578d57f1062c44124c5a39349a5e8b71c2fc2f524a432d9a6f5d4fd69bd4
SCK	governance-contracts/Staking.sol	1f3301415a3507398b6681fde8436e9b6fd288c94f2dba837eb53dedc80ed6b0
SGC	governance-contracts/StakingGov.sol	cb36a6b2b281d9562a508a05297c68f5d3199718504a8f11ee012ce1b74d1ced
SSC	governance-contracts/StakingStorage.sol	0bca9ebb57bbb2d64fa748b39eccc5d807ab1559e0d9901eb67ad2007556ae84
TCP	governance-contracts/Timelock.sol	775f0fcc60fd2b6f6f34880b9d37a07358619478894877622649942 6fcbf704a

Findings



Critical	**0** (0.00%)
Major	**7** (28.00%)
Medium	**0** (0.00%)
Minor	**12** (48.00%)
Informational	**6** (24.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
CRC-01	Missing Zero Address Checks	Volatile Code	Minor	Resolved
CRC-02	Missing Emit Event	Coding Style	Informational	Resolved
CRC-03	Privileged Roles	**Centralization / Privilege**	**Major**	Acknowledged
FCK-01	Missing Zero Address Checks	Volatile Code	Minor	Resolved
FCK-02	Centralized Risk	**Centralization / Privilege**	**Major**	Acknowledged
GAC-01	Missing Zero Address Checks	Volatile Code	Minor	Resolved
GAC-02	Missing checks for return values	Volatile Code	Informational	Resolved
GAC-03	Strange `cancel()` logic	Logical Issue	Minor	Resolved
GAC-04	Lack of input validation	Logical Issue	Minor	Resolved
HEI-01	Missing Zero Address Checks	Volatile Code	Minor	Resolved
HEI-02	Third Party Dependencies	**Centralization / Privilege**	**Major**	Partially Resolved
HEI-03	Privileged Roles	**Centralization / Privilege**	**Major**	Acknowledged
HEI-04	Usage of Magic Number	Coding Style	Informational	Resolved
PCK-01	Missing Emit Event	Coding Style	Informational	Resolved
PCK-02	Centralized Risk	**Centralization / Privilege**	**Major**	Acknowledged
PCK-03	Missing Inheritance	Coding Style	Informational	Resolved

ID	Title	Category	Severity	Status
SCK-01	Missing Zero Address Checks	Volatile Code	Minor	Resolved
SCK-02	addPool() Function Not Restricted	Volatile Code	Minor	Resolved
SCK-03	Centralized Risk	**Centralization / Privilege**	**Major**	Acknowledged
SCK-04	Missing Emit Event	Coding Style	Informational	Resolved
SCK-05	Wrong calculation	Logical Issue	Minor	Resolved
SGC-01	Pid can be manipulated	Logical Issue	Major	Resolved
TCP-01	Missing Zero Address Checks	Volatile Code	Minor	Resolved
TCP-02	Lack of input validation	Logical Issue	Minor	Resolved
TCP-03	Lack of input validation	Logical Issue	Minor	Resolved

CRC-01 | Missing Zero Address Checks

Category	Severity	Location	Status
Volatile Code	Minor	core-contracts/contracts/ConfigurableRightsPool.sol: 302	Resolved

Description

Lacks sanity check for ensuring parameters are not zero address in mentioned functions.

Recommendation

We advise to add zero address checking in functions like `setTimelock()`, `setStakingPools()`, `setVotingDelay()` and `setVotingPeriod()`.

Alleviation

Fixed by `02727b7ebdff5d3ca457818b03e19aba82c4b9c6`, `9c266826e5f0130b42fddd917db5143b768b0436`, `cacc40c4a374bf8b13c484aa3d240415384362cf`, `442624ff40a5c1ec914b423e57d1f6310ea4cb91`, `fd182a20fb2a44f67756111cb4d35e2ced969715`, `09d3aaa5a766c42e419da853968ff0e9526398d1` and `01a93036f986cea2efeb29754b37395b1d95e828`.

CRC-02 | Missing Emit Event

Category	Severity	Location	Status
Coding Style	Informational	core-contracts/contracts/ConfigurableRightsPool.sol: 302	Resolved

Description

Function that affect the status of sensitive variables should be able to emit events as notifications to customers, e.g

- setQuorum(uint256)
- setProposer(uint256)
- setVotingPeriod(uint256)
- setVotingDelay(uint256)

And all other mentioned functions.

Recommendation

We advise the client to consider adding events for sensitive actions and emit them in the corresponding functions.

Alleviation

Fixed by `01b92791cbe487c25661e4f389fd888135dcb71e`, `11db66701e339428ee34ee90809e0069651a5cfa` and `e6c2daeb3e7de44a6a9aceeab5d334689bb82704`.

CRC-03 | Privileged Roles

Category	Severity	Location	Status
Centralization / Privilege	**Major**	core-contracts/contracts/ConfigurableRightsPool.sol: 369, 412, 472, 502, 526	Acknowledged

Description

There is `strategyUpdater` role that can modify critical configurations, if an attacker takes control over this role, the actions he can perform may endanger users' funds.

The overly powerful owner is a centralization risk as he can perform actions without obtaining the consensus of the community.

Recommendation

We advise the client to handle these privileged roles carefully avoid any potential hack. We also advise the client to consider the following solutions:

1. `Timelock` with reasonable latency for community awareness on privileged operations;
2. Multisig with community-voted 3rd-party independent co-signers;
3. DAO or Governance module increasing transparency and community involvement.

Alleviation

The `strategyUpdater` role is entirely controlled by the governance timelock, it should be a contract the governance will also approve and have administrator powers over.

FCK-01 | Missing Zero Address Checks

Category	Severity	Location	Status
Volatile Code	Minor	core-contracts/contracts/core/Factory.sol: 73	Resolved

Description

Lacks sanity check for ensuring parameters are not zero address in mentioned functions.

Recommendation

We advise to add zero address checking in functions like `setTimelock()`, `setStakingPools()`, `setVotingDelay()` and `setVotingPeriod()`.

Alleviation

Fixed by `02727b7ebdff5d3ca457818b03e19aba82c4b9c6`, `9c266826e5f0130b42fddd917db5143b768b0436`, `cacc40c4a374bf8b13c484aa3d240415384362cf`, `442624ff40a5c1ec914b423e57d1f6310ea4cb91`, `fd182a20fb2a44f67756111cb4d35e2ced969715`, `09d3aaa5a766c42e419da853968ff0e9526398d1` and `01a93036f986cea2efeb29754b37395b1d95e828`.

FCK-02 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	core-contracts/contracts/core/Factory.sol: 57, 65, 71, 77	Acknowledged

Description

The owner of the account with the `Owner` role has the privilege to update the sensitive variables and conduct sensitive operations in the project.

Hackers who compromise the account with an `Owner` role may take advantage of these centralized privileges and manipulate the project for profits.

Recommendation

We advise the client to carefully manage the role Owner's account private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract-based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO / governance/voting module to increase transparency and user involvement.

Alleviation

Once the contracts are deployed, the first ETF ($HEIM) is created and all the staking pools are created we are going to change the owner to a timelock contract that is managed by a DAO. The timelock has a 48h minimum waiting time and the DAO also has a minimum 48h voting time, with thedefaults on deploy being 48h and 5 days.

GAC-01 | Missing Zero Address Checks

Category	Severity	Location	Status
Volatile Code	Minor	governance-contracts/GovernorAlpha.sol: 140~141, 174, 183, 192, 201	Resolved

Description

Lacks sanity check for ensuring parameters are not zero address in mentioned functions.

Recommendation

We advise to add zero address checking in functions like `setTimelock()`, `setStakingPools()`, `setVotingDelay()` and `setVotingPeriod()`.

Alleviation

Fixed by `02727b7ebdff5d3ca457818b03e19aba82c4b9c6`, `9c266826e5f0130b42fddd917db5143b768b0436`, `cacc40c4a374bf8b13c484aa3d240415384362cf`, `442624ff40a5c1ec914b423e57d1f6310ea4cb91`, `fd182a20fb2a44f67756111cb4d35e2ced969715`, `09d3aaa5a766c42e419da853968ff0e9526398d1` and `01a93036f986cea2efeb29754b37395b1d95e828`.

GAC-02 | Missing checks for return values

Category	Severity	Location	Status
Volatile Code	Informational	governance-contracts/GovernorAlpha.sol: 300~306, 320~326	Resolved

Description

There are missing checks for return values of not void-returning functions. Ignoring the return value might cause some unexpected exceptions.

Recommendation

We recommend checking the output of the aforementioned functions before continuing processing.

Alleviation

Fixed by `9988868c1befd16bba01a703847b47470b3c2fcf`.

GAC-03 | Strange `cancel()` logic

Category	Severity	Location	Status
Logical Issue	Minor	governance-contracts/GovernorAlpha.sol: 342~345	Resolved

Description

Anyone can cancel a proposal when the proposer's voting power is below `proposalThreshold()` EVEN IF the proposal gets enough votes to succeed. And even the proposer himself/herself can NOT cancel the proposal when his/her voting power is above `proposalThreshold()`.

Recommendation

We advise the client to carefully review the `cancel()` function and make sure it works as intended.

Alleviation

Fixed by 762ba679ba9ec08795d13c4df1c7943a7939c99d

GAC-04 | Lack of input validation

Category	Severity	Location	Status
Logical Issue	Minor	governance-contracts/GovernorAlpha.sol: 339	Resolved

Description

The function should check if the transaction is already canceled. Otherwise, the transaction can be canceled many times and the event can be generated many times.

Recommendation

We advise the client to add a check to make sure the transaction is not already canceled, otherwise, the code should revert.

Alleviation

Fixed by 7076fc4af31798366890aca907fa087bb6bbd117.

HEI-01 | Missing Zero Address Checks

Category	Severity	Location	Status
Volatile Code	Minor	core-contracts/strategies/HEIM.sol: 61~66, 85~88, 100, 112, 124	Resolved

Description

Lacks sanity check for ensuring parameters are not zero address in mentioned functions.

Recommendation

We advise to add zero address checking in functions like `setTimelock()`, `setStakingPools()`, `setVotingDelay()` and `setVotingPeriod()`.

Alleviation

Fixed by `02727b7ebdff5d3ca457818b03e19aba82c4b9c6`, `9c266826e5f0130b42fddd917db5143b768b0436`, `cacc40c4a374bf8b13c484aa3d240415384362cf`, `442624ff40a5c1ec914b423e57d1f6310ea4cb91`, `fd182a20fb2a44f67756111cb4d35e2ced969715`, `09d3aaa5a766c42e419da853968ff0e9526398d1` and `01a93036f986cea2efeb29754b37395b1d95e828`.

HEI-02 | Third Party Dependencies

Category	Severity	Location	Status
Centralization / Privilege	**Major**	core-contracts/strategies/HEIM.sol: 55, 221	Partially Resolved

Description

The code depends on `Airnode` contract for update the weights of the crpPool, while the scope of the audit would treat those third-party entities as black boxes and assume their functional correctness and return honest results. However in the real world, third parties may be compromised that led to assets being lost or stolen. In addition, upgrades of third parties are possible to lead to severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of StrategyHEIM requires interaction with API3 Airnode solution to connect Heimdall API social score data and update the pool weights. We encourage the team to constantly monitor the statuses of those 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

Somewhat aleviated by `24f6771af4c25943ebabdb2481b0dcf0304a61c0`. The key problem here is that API3/Airnode and/or Heimdall could manipulate the data that changes the weights (just like any other oracle could). To increase the security of the funds against such an attack the following mitigations exist:

- An `updaterRole` is responsible for initiating the requests for Heimdall data, they cannot modify the request in any way, the only thing they are responsible for is to ping the strategy so that it updates the pool. The foundation will be responsible for this role initially but the DAO can appoint whoever they wish.
- The strategy has a `suspectDiff` parameter that defines the difference in percentage of the social score between the current and previous calls that will trigger an automatic suspension of the strategy. Or, in other words, if the social score of the current call deviates `suspectDiff` percent from the previous call from ~24h ago, the strategy will automatically enter a suspended state and will not commit the new weights. A `StrategyPaused` event is raised in this case with reason `ERR_SUSPECT_REQUEST`.
- A `watcherRole` is responsible for checking the strategy is working as intended, their power is to pause or resume the strategy when they think the strategy is being attacked or receiving bad data - when paused no calls to Airnode can be made or received and the weight update is also paused in

the pool - and to accept or decline automatically suspended calls - the watcher is responsible for checking the new weights and making sure they are correct and accept or reject them. If the watcher pauses the pool a `StrategyPaused` event is raised with reason `WATCHER_PAUSED`, when it's resumed a `StrategyResumed` is raised with the following possible reasons, `WATCHER_RESUMED`, `ACCEPTED_SUSPENDED_REQUEST`, `REJECTED_SUSPENDED_REQUEST`. The foundation will also be responsible for this role initially but the DAO can appoint whoever they wish.

- Even if the watcher fails with their obligations or is bribed to accept or not pause bad calls, the function that updates the weights is gradual and takes 24 hours to completely change to the new weights. We hope some investors will keep an eye on the strategy and alert everybody to redeem their $HEIM in such case in far less than 24 hours frustrating the attack as quickly as possible.
- All functions that change the parameters of the strategy can only be called by the DAO and all of them contain events alerting of the changes.
- The strategy is decoupled from the pool and can be replaced by another contract, more decentralised and cryptoeconomically sound. We raised an issue on GitHub for collecting discussions on how we can create a better strategy to replace the current one. https://github.com/KassandraFinance/kassandra-core/issues/1

HEI-03 | Privileged Roles

Category	Severity	Location	Status
Centralization / Privilege	**Major**	core-contracts/strategies/HEIM.sol: 83, 98, 110, 122, 148, 245	Acknowledged

Description

There is `onlyOwner()` role that can modify critical configurations and `onlyAirnode()` that can update pool weights, if an attacker takes control over these roles, he may manipulate the project for profits.

Recommendation

We advise the client to handle these privileged roles carefully avoid any potential hack. We also advise the client to consider the following solutions:

1. `Timelock` with reasonable latency for community awareness on privileged operations;
2. Multisig with community-voted 3rd-party independent co-signers;
3. DAO or Governance module increasing transparency and community involvement.

Alleviation

These roles are entirely controlled by the governance timelock, it should be a contract the governance will also approve and have administrator powers over.

HEI-04 | Usage of Magic Number

Category	Severity	Location	Status
Coding Style	Informational	core-contracts/strategies/HEIM.sol: 257~273	Resolved

Description

There are magic numbers for scores and weights calculation, and also input `data` interpretation which may be wrong once the numbers and data format from the caller gets updated.

```
257             for (uint i = 0; i < 14; i++) {
258                 scores[i] = data >> (i * 18) & 0x3FFFF;
259                 if (scores[i] == 0x3FFFF) {
260                     emit RequestFailed(requestId, "ERR_SCORE_OVERFLOW");
261                     return;
262                 }
263                 totalScore += scores[i];
264             }
265
266             uint minimumKacy = coreFactory.minimumKacy();
267             uint kacyPercentage = scores[kacy] * KassandraConstants.ONE /
totalScore;
268             uint totalWeight = 40;
269
270             if (kacyPercentage < minimumKacy) {
271                 totalScore -= scores[kacy];
272                 totalWeight = 38;
273             }
```

Recommendation

We advise the team adds proper documentation specifying the purpose of the linked number and monitor third-party format updates.

Alleviation

Fixed by `24f6771af4c25943ebabdb2481b0dcf0304a61c0`.

PCK-01 | Missing Emit Event

Category	Severity	Location	Status
Coding Style	Informational	core-contracts/contracts/core/Pool.sol: 82, 643~646	Resolved

Description

Function that affect the status of sensitive variables should be able to emit events as notifications to customers, e.g

- setQuorum(uint256)
- setProposer(uint256)
- setVotingPeriod(uint256)
- setVotingDelay(uint256)

And all other mentioned functions.

Recommendation

We advise the client to consider adding events for sensitive actions and emit them in the corresponding functions.

Alleviation

Fixed by `01b92791cbe487c25661e4f389fd888135dcb71e`, `11db66701e339428ee34ee90809e0069651a5cfa` and `e6c2daeb3e7de44a6a9aceeab5d334689bb82704`.

PCK-02 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	core-contracts/contracts/core/Pool.sol: 77, 89, 125, 105, 147, 631	Acknowledged

Description

The owner of the account with the `Owner` role has the privilege to update the sensitive variables and conduct sensitive operations in the project.

Hackers who compromise the account with an `Owner` role may take advantage of these centralized privileges and manipulate the project for profits.

Recommendation

We advise the client to carefully manage the role Owner's account private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract-based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO / governance/voting module to increase transparency and user involvement.

Alleviation

Once the contracts are deployed, the first ETF ($HEIM) is created and all the staking pools are created we are going to change the owner to a timelock contract that is managed by a DAO. The timelock has a 48h minimum waiting time and the DAO also has a minimum 48h voting time, with thedefaults on deploy being 48h and 5 days.

PCK-03 | Missing Inheritance

Category	Severity	Location	Status
Coding Style	Informational	core-contracts/contracts/core/Pool.sol: 15~702	Resolved

Description

Missing inheritance from the interface `IPool`.

Recommendation

Consider inheriting from the interface, like below:

```
contract Pool is Ownable, ReentrancyGuard, Token, Math, IPool {
...
}
```

Alleviation

Fixed by `052dc68e30e04f23ff86807a527cf11fc0cffcaf`.

SCK-01 | Missing Zero Address Checks

Category	Severity	Location	Status
Volatile Code	Minor	governance-contracts/Staking.sol: 430	Resolved

Description

Lacks sanity check for ensuring parameters are not zero address in mentioned functions.

Recommendation

We advise to add zero address checking in functions like `setTimelock()`, `setStakingPools()`, `setVotingDelay()` and `setVotingPeriod()`.

Alleviation

Fixed by `02727b7ebdff5d3ca457818b03e19aba82c4b9c6`, `9c266826e5f0130b42fddd917db5143b768b0436`, `cacc40c4a374bf8b13c484aa3d240415384362cf`, `442624ff40a5c1ec914b423e57d1f6310ea4cb91`, `fd182a20fb2a44f67756111cb4d35e2ced969715`, `09d3aaa5a766c42e419da853968ff0e9526398d1` and `01a93036f986cea2efeb29754b37395b1d95e828`.

SCK-02 | addPool() Function Not Restricted

Category	Severity	Location	Status
Volatile Code	Minor	governance-contracts/Staking.sol: 359~368	Resolved

Description

The current implementation is relying on the trust of the owner to avoid repeatedly adding the same staking token to the pool, as the function will only be called by the owner.

Recommendation

Detect whether the given pool for addition is a duplicate of an existing pool. The pool addition is only successful when there is no duplicate. Using mapping of `addresses` -> `booleans`, which can restrict the same address being added twice.

Alleviation

Having pools with the same staking token is a desired feature, since each pool may have different vesting schedules, so this was not changed. Despite that, some new requires have been address to the addPool() function by `1f3f75ef240e2ab044f24a4b15be0d06f5ccd8ad`.

SCK-03 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	governance-contracts/Staking.sol: 366, 387, 407, 413, 422, 430	Acknowledged

Description

The owner of the account with the `Owner` role has the privilege to update the sensitive variables and conduct sensitive operations in the project. For example, addPool(), addReward(), updatePeriodFinish(), recoverERC20(), setRewardsDuration() and setKacy().

Hackers who compromise the account with an `Owner` role may take advantage of these centralized privileges and manipulate the project for profits.

Recommendation

We advise the client to carefully manage the role Owner's account private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract-based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO / governance/voting module to increase transparency and user involvement.

Alleviation

Once the contracts are deployed, the first ETF ($HEIM) is created and all the staking pools are created we are going to change the owner to a timelock contract that is managed by a DAO. The timelock has a 48h minimum waiting time and the DAO also has a minimum 48h voting time, with thedefaults on deploy being 48h and 5 days.

SCK-04 | Missing Emit Event

Category	Severity	Location	Status
Coding Style	Informational	governance-contracts/Staking.sol: 153, 165, 174, 183	Resolved

Description

Function that affect the status of sensitive variables should be able to emit events as notifications to customers, e.g

- setQuorum(uint256)
- setProposer(uint256)
- setVotingPeriod(uint256)
- setVotingDelay(uint256)

And all other mentioned functions.

Recommendation

We advise the client to consider adding events for sensitive actions and emit them in the corresponding functions.

Alleviation

Fixed by `01b92791cbe487c25661e4f389fd888135dcb71e`, `11db66701e339428ee34ee90809e0069651a5cfa` and `e6c2daeb3e7de44a6a9aceeab5d334689bb82704`.

SCK-05 | Wrong calculation

Category	Severity	Location	Status
Logical Issue	Minor	governance-contracts/Staking.sol: 199~201	Resolved

Description

The noted calculation is wrong. It will cause revert when multiple withdrawals are made during lock+vesting period.

Recommendation

We advise the client to use the amount at deposit time instead of user.amount.

Alleviation

Fixed by 918d74565df559fb262b32a17be5f6f3727190f4.

SGC-01 | Pid can be manipulated

Category	Severity	Location	Status
Logical Issue	Major	governance-contracts/StakingGov.sol: 90~93	Resolved

Description

The `pid` is not involved when generating/verifying signature. So we don't know the signature should be used to approve delegation for which pid. The hacker can change pid at his/her will.

Recommendation

We advise the client to add pid in the to-be-signed message when generating signature such that pid is checked when verifying signature.

Alleviation

Fixed by 4bfe2902ac3c9d6651f25d2f661620e73243ba7c.

TCP-01 | Missing Zero Address Checks

Category	Severity	Location	Status
Volatile Code	Minor	governance-contracts/Timelock.sol: 53, 78	Resolved

Description

Lacks sanity check for ensuring parameters are not zero address in mentioned functions.

Recommendation

We advise to add zero address checking in functions like `setTimelock()`, `setStakingPools()`, `setVotingDelay()` and `setVotingPeriod()`.

Alleviation

Fixed by `02727b7ebdff5d3ca457818b03e19aba82c4b9c6`, `9c266826e5f0130b42fddd917db5143b768b0436`, `cacc40c4a374bf8b13c484aa3d240415384362cf`, `442624ff40a5c1ec914b423e57d1f6310ea4cb91`, `fd182a20fb2a44f67756111cb4d35e2ced969715`, `09d3aaa5a766c42e419da853968ff0e9526398d1` and `01a93036f986cea2efeb29754b37395b1d95e828`.

TCP-02 | Lack of input validation

Category	Severity	Location	Status
Logical Issue	Minor	governance-contracts/Timelock.sol: 97	Resolved

Description

The function should check if the transaction is queued already, otherwise, transactions may be lost if 2 transactions from different users have identical target, value, signature, data, eta.

Recommendation

We advise the client to add a check to make sure the transaction is NOT queued already; if already queued, the code should revert.

Alleviation

Fixed by be7560a76410673ab9b736cbce55779be0c88274 and 02b7e868b1c50f99b5af3b0a5f3a70cba0120aad. Included the `proposalId` in the hash to allow two proposals with the same command to queue in the same block.

TCP-03 | Lack of input validation

Category	Severity	Location	Status
Logical Issue	Minor	governance-contracts/Timelock.sol: 115	Resolved

Description

The function should check if the transaction is actually queued: if not, it should not emit event.

Recommendation

We advise the client to add a check to make sure the transaction is actually queued, otherwise, it should not emit event.

Alleviation

Fixed by be7560a76410673ab9b736cbce55779be0c88274 and 02b7e868b1c50f99b5af3b0a5f3a70cba0120aad. Included the `proposalId` in the hash to allow two proposals with the same command to queue in the same block.

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.

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.

Volatile Code

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

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.