CERTIK

Security Assessment

Retrograde Protocol

Apr 22nd, 2022

Table of Contents

Summary

This report has been prepared for Retrograde Protocol to discover issues and vulnerabilities in the source code of the Retrograde Protocol 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	Retrograde Protocol
Platform	Terra
Language	Rust
Codebase	• https://github.com/retrogradeprotocol/retrograde-contracts
Commit	• 74d264dd5cd2d05ba88ef9630ba1535a678f203b • 3feb3bf5f397d1e102b2a81f0951b3fb8460de8c • 5bd1d225bd70beac51f2700ca50f57350bbb313b

Audit Summary

Delivery Date	Apr 22, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	0	0	0	0	0	0	0
Major	3	0	0	3	0	0	0
Medium	0	0	0	0	0	0	0
Minor	2	0	0	1	0	0	1
Informational	8	0	0	2	0	0	6
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
MSR	contracts/staking/src/msg.rs	bf268555bb71994215687ccdaddb39c8942be67a9568a6ac23a8ef50649f0c0e
STT	contracts/ido/src/state.rs	d216315204cb4b83833043339c9c2ad277beda6acc7d36474c5fa75af5637df8
MSS	contracts/ido/src/msg.rs	a42d191f85c6b75610b2b6b96d54824170198fc5d06c447e8a6b7379a822913f
ERO	contracts/ido/src/error.rs	fb7fc77e25716018778a4ae250829ae301a8c868681bc21142eb2f3d60b916c2
STE	contracts/staking/src/state.rs	22be663c87470ab986de73fa1c37bac18ebee5e3b29b7679d7a170b02e0daff3
LIE	packages/retrograde/src/lib.rs	93ca5880ecffaf375b5ce6c708421e1e9fbd7d735929b5b303d6ecf98114688e
STS	contracts/treasury/src/state.rs	a0d518d2448838bf3737eb4c4aa9ed8f853ec35106f18d3200ec2f4c8f7530f7
MSG	contracts/bonding/src/msg.rs	2d189ded50f863073ef5b620b14d4936bf3b2e6603fadcb55c0aa67ad886f727
COR	contracts/treasury/src/contract.rs	5e27e5e2d8c6d92b94510e87e33d4886bc5a02420553c30320cd0493c46a36d9
MSC	contracts/treasury/src/msg.rs	27a2429e3d08f38340c8e4f1f6f16770609b13dacbbaf5e1e4dd8551a00fe1f8
QUE	contracts/staking/src/querier.rs	eb6e2784ba2d1fe14fd78f022c3cb55f8c3459efab8254d510971bf679d43234
COT	contracts/staking/src/contract.rs	518ab2fb447f46fbf05a8b09417ae98dbefcf3ad568aab743462a287997af5f3
ERR	contracts/bonding/src/error.rs	fb7fc77e25716018778a4ae250829ae301a8c868681bc21142eb2f3d60b916c2
LIS	contracts/ido/src/lib.rs	1001dbea515c706daedc5f5025ec2e186e9b5c91ef22ef92b64841a1b0a7edaf
LIR	contracts/staking/src/lib.rs	e9fa38352b2264287798efef60c54cd4e24b7c480505b7af3802730b9edc7c02
CKP	contracts/bonding/src/contract.rs	57944f4679bf7c7e92c6441816793398588d5398376e9dc8e2aaaa4662383f06
MAR	packages/retrograde/src/market.rs	de65742d8d266ef4d6db411c9a68c8f776403b1da536f0c07cd23a4f73ecdba8
CON	contracts/ido/src/contract.rs	56ae53790b7127437f256b832da7a190c8fbcc11dd54cb4f5dc57c1fb516fdf7
BON	packages/retrograde/src/bond_pricing.rs	a6d9bfc387780d39098bbf9decb5f618704850facf7cc29e3bcadb6abd8ffaaf
STA	contracts/bonding/src/state.rs	55452ad8c4d3b132ce797a0b0358bf7289aeadfb308099ca3e3f1ae5ca05baf3
ERS	contracts/treasury/src/error.rs	aa28d75b3955b840874dfaaccdb3b137187115d8f9134b3a6ad1103b3ee64a5e

ID	File	SHA256 Checksum
LIC	contracts/treasury/src/lib.rs	ac480c4fb3f9c7acfdcce64e206678274678357e0884fe3ea695414362c097b5
LIB	contracts/bonding/src/lib.rs	ac480c4fb3f9c7acfdcce64e206678274678357e0884fe3ea695414362c097b5
MSI	contracts/ido/src/msg.rs	930e67a76486d30764a2bf104f7675a16226323d62e7d2b171713dc41cccdf83
STR	contracts/bonding/src/state.rs	1592a9256c7da83ce1996b1856b6c371ee6afe64cdc01765d8fe941faa85a4da
ERC	contracts/bonding/src/error.rs	fb7fc77e25716018778a4ae250829ae301a8c868681bc21142eb2f3d60b916c2
ERT	contracts/treasury/src/error.rs	aa28d75b3955b840874dfaaccdb3b137187115d8f9134b3a6ad1103b3ee64a5e
STU	contracts/treasury/src/state.rs	a0d518d2448838bf3737eb4c4aa9ed8f853ec35106f18d3200ec2f4c8f7530f7
MSB	contracts/bonding/src/msg.rs	4a818a1b9a74e2e56db0296adf455e10675f012b16cf175079b67020295c6e92
QUR	contracts/staking/src/querier.rs	eb6e2784ba2d1fe14fd78f022c3cb55f8c3459efab8254d510971bf679d43234
LIO	contracts/bonding/src/lib.rs	ac480c4fb3f9c7acfdcce64e206678274678357e0884fe3ea695414362c097b5
COE	contracts/treasury/src/contract.rs	4bf81f3595779d3690ea231620ce0bbbed5f52f1aed4a1680a9afe5172e97db3
LIA	contracts/treasury/src/lib.rs	ac480c4fb3f9c7acfdcce64e206678274678357e0884fe3ea695414362c097b5
MST	contracts/staking/src/msg.rs	06ff143bec723ffe57e3e06c196f0550ed06f0ac42cdcccc88b1d14dc3dc1cd8
LII	contracts/ido/src/lib.rs	1001dbea515c706daedc5f5025ec2e186e9b5c91ef22ef92b64841a1b0a7edaf
STK	contracts/staking/src/state.rs	b5cae1bfa2f0019612149bd3ec821ba130ee78c73c39c61c9e6f43a27e2274ab
COC	contracts/ido/src/contract.rs	673ea2d754e2ac551e9e60c54a1ff78808bc7d9428f8d6b72cc9bf764fa6eb4f
STC	contracts/ido/src/state.rs	8d7bcbb8cf2c67bfb9f187b705b8f6fc3faa47f44e707cef1b567c94a93fde83
MSE	contracts/treasury/src/msg.rs	27a2429e3d08f38340c8e4f1f6f16770609b13dacbbaf5e1e4dd8551a00fe1f8
COA	contracts/bonding/src/contract.rs	5d31788cba24c4088d4f3edf218016f4795a9a07a555d12beeae74f63492ef4f
COS	contracts/staking/src/contract.rs	b6c1408e60b45cbf91bdd6a9150c18bfcb7799121bdba61d09c8d79e86ee8b3f
LIT	contracts/staking/src/lib.rs	e9fa38352b2264287798efef60c54cd4e24b7c480505b7af3802730b9edc7c02
ERI	contracts/ido/src/error.rs	fb7fc77e25716018778a4ae250829ae301a8c868681bc21142eb2f3d60b916c2
STD	contracts/discovery/src/state.rs	8bacc4fe4b4c640787e136a1a5e669af1b00a739cbcc40be588e950fb1a4909c

ID	File	SHA256 Checksum
LID	contracts/discovery/src/lib.rs	1001dbea515c706daedc5f5025ec2e186e9b5c91ef22ef92b64841a1b0a7edaf
COD	contracts/discovery/src/contract.rs	6b407533846eb902a303289c46bba9a9c473337e4b9f86c6071fbc4b2c12f354
ERD	contracts/discovery/src/error.rs	fb7fc77e25716018778a4ae250829ae301a8c868681bc21142eb2f3d60b916c2
MSD	contracts/discovery/src/msg.rs	11a69c35e366964a7e9773450de8460cf6af005a2881bd68554aac661465ed1c

Review Notes

Retrograde Protocol creates a protocol for bonding asset, **RETRO** fair price discovery, and staking with **RETRO** token as rewards. The **Retrograde Protocol** consists of the following four contracts and modules:

- bonding
- discovery
- staking
- treasury

The `bonding` contract implements a simple swap template for the users to send the certain tokens to bond with an asset.

The `discovery` contract acts as a role to perform the `RETRO` fair price discovery. The users can deposit the UST in exchange for the `RETRO` token.

The `staking` contract allows the users to stake tokens (i.e., RETRO, ASTRO, ANC, retroASTRO, etc.) with `RETRO` token as rewards according to a predetermined reward distribution schedule.

The `treasury` contract holds the treasury assets related to the project. For example, the UST deposited in the `discovery` will be sent to the `treasury`.

External Dependencies

There are a few depending injection contracts or addresses in the current project:

- `treasury`, `bounding_asset` and `derivative_asset` for the contract `bonding`;
- `retro_token` and `retrograde_treasury` for the contract `discovery`;
- `retro_token` and `staking_token` for the contract `staking`.

We assume these vulnerable actors are implementing proper logic to collaborate with the current project.

Privileged Functions

The role `owner` has been adopted in the contract `discovery` over the following messages:

- `Start` to begin fair price discovery for the Retrograde token launch.
- `OwnerWithdraw` to withdraw leftover Retrograde token from the contract.

The role `owner` has been adopted in the contract `staking` over the following messages:

- `UpdateConfig` to update the configuration.

- `UpdateOwner` to update the owner.
- `UpdateUnlockTime` to update the unlock time of the staking.

The role `owner` has been adopted in the contract `treasury` over the following messages:

- `UpdateOwner` to set a new owner of the contract;
- `AddNativeManager` to add new native asset manager;
- `AddTokenManager` to add new token manager;
- `RemoveNativeManager` to remove native asset manager;
- `RemoveTokenManager` to remove token manager.

The role `native asset manager` has been adopted in the contract `treasury` over the following message:

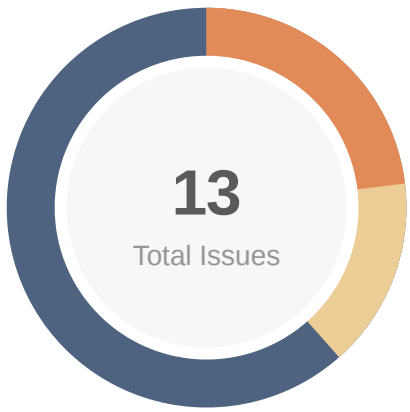
- `WithdrawNative` to withdraw native asset.

The role `token manager` has been adopted in the contract `treasury` over the following message:

- `WithdrawToken` to withdraw the token.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the `Timelock` contract.

Findings



Critical	**0** (0.00%)
Major	**3** (23.08%)
Medium	**0** (0.00%)
Minor	**2** (15.38%)
Informational	**8** (61.54%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
COD-01	Centralization Related Risks In Contract `discovery`	**Centralization / Privilege**	**Major**	Acknowledged
CON-01	Could Add More Details To Events	Coding Style	Informational	Resolved
CON-02	Redundant `clone()`	Language Specific	Informational	Resolved
CON-03	Unnecessary `Uint128::from()` - No Conversion Needed Form `Uint128` To `Uint128`	Coding Style	Informational	Resolved
CON-04	Permanently Locked `RETRO` Tokens	Logical Issue	Informational	Acknowledged
CON-05	Inconsistency Between Names, Comments And Implementations	Logical Issue, Inconsistency	Informational	Resolved
CON-06	Typo In Events	Coding Style	Informational	Resolved
COR-01	Centralization Related Risks In Contract `treasury`	**Centralization / Privilege**	**Major**	Acknowledged
COR-02	Inconsistency Between Act And Event	Coding Style	Minor	Resolved
COR-03	Unnecessary `.into()` - No Conversion Needed From `String` To `String`	Coding Style	Informational	Resolved
COS-01	Centralization Related Risks In Contract `staking`	**Centralization / Privilege**	**Major**	Acknowledged

ID	Title	Category	Severity	Status
COT-01	Lack Of Input Validation Of `distribution_schedule`	Volatile Code	Minor	Acknowledged
COT-02	Usage Of `env.block.time`	Logical Issue	Informational	Acknowledged

COD-01 | Centralization Related Risks In Contract `discovery`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	contracts/discovery/src/contract.rs (5bd1d22): 72, 75~77, 155, 158~160	Acknowledged

Description

In the contract `contracts/discovery/src/contract.rs`, the role `owner` has authority over the following message:

- `Start` to begin fair price discovery for the Retrograde token launch.
- `OwnerWithdraw` to withdraw leftover Retrograde token from the contract.

Any compromise to the `owner` account may allow a hacker to take advantage of this authority and manipulate the project.

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.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

[**Retrograde team**]: The team is implementing the short-term solution.

1. The team added a time delay to the `start` message, with a configurable delay time.
2. The privileged role will be a CW3 multi-sig contract, which can be verified to be a multi-sig on-chain.
3. The details of all timings will be shared publicly via twitter/medium/discord with our public audience.

CON-01 | Could Add More Details To Events

Category	Severity	Location	Status
Coding Style	Informational	contracts/ido/src/contract.rs (74d264d): 80, 125	Resolved

Description

In contract `contracts/ido/src/contract.rs`, the events in the functions `start_ido()` and `deposit_ust()` do not provide details of the execution messages. It would be better if the users can easily check the initial `RETRO` token deposited by Retrograde team for `start_ido()`, and the user account with the amount of UST deposited by the users for `deposit_ust()`.

Recommendation

Consider adding a more detailed description of deposits in the aforementioned emitted events, including the user and the deposited amount.

Alleviation

[**Retrograde team**]: The team heeded the advice and resolved the issue by adding the recommended attributes. The changes are reflected in the commit `3171d1f6c423795ec10d52bd88f7e97a3288765f`.

CON-02 | Redundant `clone()`

Category	Severity	Location	Status
Language Specific	Informational	contracts/ido/src/contract.rs (74d264d): 26~27	Resolved

Description

The variable `phase_two_length` and `phase_three_length` are of type `cosmwasm_std::Uint128`, which implements the `Copy` trait. It is unnecessary to use the `clone()`.

Reference: https://docs.rs/cosmwasm-std/0.16.2/cosmwasm_std/struct.Uint128.html

Recommendation

Recommend removing the `clone()` from the two aforementioned variables.

Alleviation

[**Retrograde team**]: The team heeded the advice and resolved the issue by removing the unnecessary `clone()`. The changes are reflected in the commit `3171d1f6c423795ec10d52bd88f7e97a3288765f`.

CON-03 | Unnecessary `Uint128::from()` - No Conversion Needed Form `Uint128` To `Uint128`

Category	Severity	Location	Status
Coding Style	Informational	contracts/ido/src/contract.rs (74d264d): 103	Resolved

Description

The variable `c.amount` is of type `cosmwasm_std::Uint128`, so there is no need to convert it to `cosmwasm_std::Uint128`.

Reference:

1. https://docs.rs/cosmwasm-std/0.16.2/cosmwasm_std/struct.MessageInfo.html
2. https://docs.rs/cosmwasm-std/0.16.2/cosmwasm_std/struct.Coin.html

Recommendation

Recommend removing the conversion `Uint128::from()` from the aforementioned line.

Alleviation

[**Retrograde team**]: The team heeded the advice and resolved the issue by removing the unnecessary `Uint128::from()`. The changes are reflected in the commit `3171d1f6c423795ec10d52bd88f7e97a3288765f`.

CON-04 | Permanently Locked `RETRO` Tokens

Category	Severity	Location	Status
Logical Issue	Informational	contracts/ido/src/contract.rs (74d264d): 246	Acknowledged

Description

In the function `claim_retro()`, users could claim their `RETRO` token based on the ratio of their existing deposit amount over the total deposit of `UST`. Due to the precision loss of rounding down when calculating the ratio,

```
246             let user_claim: Uint128 = Decimal::from_ratio(existing_amount,
current_deposits_total) * total_retro_deposits;
```

The sum of all `user_claim` would be less than the `total_retro_deposits`, resulting in permanently locked `RETRO` tokens in the `ido` contract. Such a condition would exacerbate when the `existing_amount` is small and `current_deposits_total` is large.

Recommendation

Recommend the Retrograde team to be able to withdraw or burn the `RETRO` after the end of IDO process.

Alleviation

[**Retrograde team**]: The team acknowledged that leftover RETRO will be permanently left in the contract, effectively burned, as nobody can withdraw it. The team decided not to change the current codebase.

CON-05 | Inconsistency Between Names, Comments And Implementations

Category	Severity	Location	Status
Logical Issue, Inconsistency	Informational	contracts/ido/src/contract.rs (74d264d): 295~304	Resolved

Description

The name and comments of function `send_ust_to_treasury()` indicate that the function is supposed to send the collected UST from the IDO to the `treasury` contract address. However, the linked code implementation actually sends the `RETRO` token to the `treasury` contract address.

Recommendation

Would like to check with the Retrograde team if this is the intended design.

Alleviation

[**Retrograde team**]: The team resolved the issue by changing the aforementioned lines of code to match with the function name and comments that the collected UST will be sent to the treasury. In particular, the following new implementation has been used to replace the linked lines of code:

```
303             // Transfer all UST to treasury
304             let send_ust_msg = BankMsg::Send {
305                 to_address: config.retrograde_treasury.to_string(),
306                 amount: coins(current_deposits_total.u128(), "uusd"),
307             };
308 
309             Ok(Response::new()
310                 .add_attribute("method", "send_ust_to_treasury")
311                 .add_attribute("amount", current_deposits_total.to_string())
312                 .add_message(send_ust_msg))
313         },
314         None => {
315             return Err(ContractError::Std(StdError::generic_err("IDO has not
started.")));
316         }
317     }
```

The changes are reflected in the commit `30a35420e276cf3faadc059f9ef307e8530f49e8`.

CON-06 | Typo In Events

Category	Severity	Location	Status
Coding Style	Informational	contracts/ido/src/contract.rs (74d264d): 39	Resolved

Description

The key in the linked event does not match with the value. It is supposed to be `"phase_three_length"` to represent the value of `phase_three_length`.

Recommendation

Recommend the Retrograde team to correct the typo.

Alleviation

[**Retrograde team**]: The team heeded the advice and resolved the issue by changing the key name `"phase_two_length"` into `"phase_three_length"`. The changes are reflected in the commit `e8abaa176ffe0119ca06e584a0e54ddea21f111f`.

COR-01 | Centralization Related Risks In Contract `treasury`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	contracts/treasury/src/contract.rs (74d264d): 59~66, 80~87, 101~109, 131~138, 152~160, 184~196, 215~228	Acknowledged

Description

In the contract `contracts/treasury/src/contract.rs`, the role `owner` has authority over the following messages:

- `UpdateOwner` to set a new owner of the contract;
- `AddNativeManager` to add new native asset manager;
- `AddTokenManager` to add new token manager;
- `RemoveNativeManager` to remove native asset manager;
- `RemoveTokenManager` to remove token manager.

The role `native asset manager` has authority over the following message:

- `WithdrawNative` to withdraw native asset.

The role `token manager` has authority over the following message:

- `WithdrawToken` to withdraw token.

Any compromise to these accounts may allow a hacker to take advantage of this authority and manipulate the project.

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.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

[**Retrograde team**]: The owner role will be assigned to a CW3 multisig contract, which can be verified to be a multisig on-chain. Both native asset managers and token managers will all only be assigned to smart contracts, never EOA addresses. Long term, all management will be passed to a governance smart contract.

COR-02 | Inconsistency Between Act And Event

Category	Severity	Location	Status
Coding Style	Minor	contracts/treasury/src/contract.rs (74d264d): 27, 36	Resolved

Description

In the function `instantiate()` in treasury, the `STATE.owner` is set as the `msg.owner`. However, in the response attributes, the `owner` field is broadcasted as the `info.sender`, where it is not necessarily the same as the `msg.owner`.

```
let state = State {
        owner: deps.api.addr_validate(&msg.owner)?, // Sets initial owner to specified
owner
...
    Ok(Response::new()
        .add_attribute("method", "instantiate")
        .add_attribute("owner", info.sender))
```

The inconsistency of the `owner` value in `STATE` and the response may cause misunderstanding over time and be prone to errors.

Recommendation

Recommend the Retrograde team to match the aforementioned two fields regarding to the `owner` to reflect the real status of `STATE.owner`.

Alleviation

[**Retrograde team**]: The team heeded the advice and resolved the issue by changing the `info.sender` into `msg.owner` in the attribute. The changes are reflected in the commit `0cd8c462019b1353288fdc86066c353d9e8272d3`.

COR-03 | Unnecessary `.into()` - No Conversion Needed From `String` To `String`

Category	Severity	Location	Status
Coding Style	Informational	contracts/treasury/src/contract.rs (74d264d): 204	Resolved

Description

The variable `token` is of type `String`, so there is no need to convert it to `String`.

Recommendation

Recommend removing the conversion `into()` from the aforementioned line.

Alleviation

[**Retrograde team**]: The team heeded the advice and resolved the issue by removing the unnecessary `into()`. The changes are reflected in the commit `0cd8c462019b1353288fdc86066c353d9e8272d3`.

COS-01 | Centralization Related Risks In Contract `staking`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	contracts/staking/src/contract.rs (3feb3bf): 211~216, 221~223, 239~244, 248~250, 264~269, 273~275	Acknowledged

Description

In the contract `contracts/staking/src/contract.rs`, the role `owner` has authority over the following messages:

- `UpdateConfig` to update the configuration.
- `UpdateOwner` to update the owner.
- `UpdateUnlockTime` to update the unlock time of the staking.

Any compromise to the `owner` account may allow a hacker to take advantage of this authority and manipulate the project.

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.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

[**Retrograde team**]: The owner role will be a CW3 multisig contract, which can be verified to be a multisig on-chain. The details of all timings will be shared publicly via twitter/medium/discord with our public audience.

COT-01 | Lack Of Input Validation Of `distribution_schedule`

Category	Severity	Location	Status
Volatile Code	Minor	contracts/staking/src/contract.rs (74d264d): 33, 206	Acknowledged

Description

In the function `instantiate()` and `update_config()`, the passed `distribution_schedule` has not been validated, which is declared as of type `Vec<(u64, u64, Uint128)>`. Each element `(u64, u64, Uint128)` is a tuple corresponding to a reward schedule, that is, `(start, end, reward amount)`. The test case in `contracts/staking/src/testing.rs` seems to assume the following:

1. suppose the `distribution_schedule` is <$(a_0, b_0, c_0), (a_1, b_1, c_1), (a_2, b_2, c_2), ..., (a_i, b_i, c_i), (a_{i+1}, b_{i+1}, c_{i+1}), ..., (a_n, b_n, c_n)$>;
2. for every two consecutive elements, the condition $a_i < b_i = a_{i+1} < b_{i+1}$ should be satisfied for any $0 <= i < n$, that is, the time intervals will perfectly cover the entire time schedule $[a_0, b_n]$ without any overlap or gap;
3. the total reward amount of `RETRO` token, $\sum_{i=0}^{n} c_i$ is less than the balance of the staking contract.

However, there is no validation to make sure that the input value will meet these requirements.

Recommendation

Recommend the Retrograde team to add the input validation for the aforementioned variable and provide appropriate documentation (also in form of unit testing for valid and invalid input sequences).

Alleviation

[**Retrograde team**]: This is forked from the Anchor staking contract. Despite the test cases, these conditions are not required to be true. Hence we are not going to add input validation for these conditions.

COT-02 | Usage Of `env.block.time`

Category	Severity	Location	Status
Logical Issue	Informational	contracts/staking/src/contract.rs (74d264d): 40, 98, 127, 169	Acknowledged

Description

The usage of the block timestamp `env.block.time` gives (small) room for a miner to influence it. That is, a miner can influence, to a certain degree, the outcome of a transaction in the block it mined.

Recommendation

Consider avoiding relying on the block timestamp - e.g. by switching to block height or locking operation for certain amount of blocks before possible to execute.

Alleviation

[**Retrograde team**]: The team acknowledged the finding and has chosen to use `env.block.time` throughout the contracts to be closer to clock time for time-based events rather than depend on block height, which has the potential for significant variance with respect to clock time.

Appendix

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.

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.

Language Specific

Language Specific findings are issues that would only arise within Rust, i.e. inappropriate usage of clone.

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.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code and possibly would not match with the function name, comment, or documentation.

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.