AUDIT REPORT

TerraForm Labs

Community Pool Vesting

Prepared by SCV-Security

On 18th October 2023

Table of Contents

Introduction

SCV has been engaged by TFL to conduct a comprehensive security review with the goal of identifying potential security threats and vulnerabilities within the codebase. The purpose of this audit is to evaluate the security posture of the codebase and provide actionable recommendations to mitigate any identified risks. This report presents an overview of the findings from our security audit, outlining areas of concern and proposing effective measures to enhance the codebase's security.

Scope Functionality

The community-pool-vesting contract is a cliff vesting contract with abilities to vest tokens for a specific period of time. The contract also supports native staking via delegations and un-delegations of tokens from designated validators of choice.

Submitted Codebase

Community Pool Vesting Contract	
Repository	https://github.com/terra-money/community-pool-vesting
Commit	`d60fe9462bf5873cdfb540b923ecf607cbfbaf5b`
Branch	`Main`

Revisions Codebase

Community Pool Vesting Contract	
Repository	https://github.com/terra-money/community-pool-vesting
Commit	`66f1830e7a25595673ce8fc45a88cc3343718a7a`
Branch	`unlocked-funds`

Methodologies

SCV performs a combination of automated and manual security testing based on the scope of testing. The testing performed is based on the extensive experience and knowledge of the auditor to provide the greatest coverage and value to TFL. Testing includes, but is not limited to, the following:

- Understanding the application and its functionality purpose.
- Deploying SCV in-house tooling to automate dependency analysis and static code review.
- Analyse each line of the code base and inspect application security perimeter.
- Review underlying infrastructure technologies and supply chain security posture.

Code Criteria

This section provides an evaluation of specific criteria aspects as described below:

- **Documentation:** Evaluating the presence and comprehensiveness of publicly available or provided explanatory information, diagram flowcharts, comments, and supporting documents to enhance code understanding.
- **Coverage:** Evaluating whether the code adequately addresses all necessary cases and scenarios, ensuring that the intended functionality or requirements are sufficiently covered.
- **Readability:** Assessing how easily the code can be understood and maintained, considering factors such as code structure, naming conventions, and overall organisation.
- **Complexity:** Evaluating the complexity of the code, including factors such as, number of lines, conditional statements, and nested structures.

The status of each criteria is categorised as either **SUFFICIENT** or **NOT-SUFFICIENT** based on the audit assessment. This categorisation provides insights to identify areas that may require further attention and improvement.

Criteria	Status	Notes
Documentation	**SUFFICIENT**	During this document report release, contract documentation details along with intended functionality were available here.
Coverage	**SUFFICIENT**	Testing coverage is considered sufficient, although there is room for improvement as the current coverage only extends to 94% of the code.
Readability	**SUFFICIENT**	The codebase had good readability overall and utilised many Rust and CosmWasm best practices.
Complexity	**SUFFICIENT**	N/A

Findings Summary

Summary Title	Risk Impact	Status
Improper handling of delegation rewards	CRITICAL	RESOLVED
Rewards are incorrectly sent to owner	CRITICAL	RESOLVED
Potential mishandling of contract funds	SEVERE	ACKNOWLEDGED
Incorrect vesting for non luna denominations	MODERATE	ACKNOWLEDGED
Instantiate lacking validations	MODERATE	PARTIALLY RESOLVED
Vesting does not implement cooldown period	LOW	ACKNOWLEDGED
Updating the owner does not update the whitelisted addresses	LOW	RESOLVED
Potential integer truncation in vesting formula	INFO	ACKNOWLEDGED
Addresses are not validated	INFO	RESOLVED
Event emitting wrong response	INFO	RESOLVED
Implement a two-step ownership transfer	INFO	ACKNOWLEDGED

Findings Technical Details

1. Improper handling of delegation rewards

RISK IMPACT: CRITICAL	STATUS: RESOLVED

Description

The `_withdraw_delegation_rewards` function in `src/contract.rs:326` performs queries to get the delegation amount and then sends the delegation amount in a bank message to the caller, but does not actually pass the `DistributionMsg::WithdrawDelegatorReward`. Only the `claim_delegator_reward` passes the `WithdrawDelegatorReward` message before it calls `_withdraw_delegation_rewards`, but there are many other functions that call `_withdraw_delegation_rewards` which do not first perform the withdrawal. The result of this is that the contract's balance will be sent instead of the delegation reward balance.

Any time `undelegate_funds`, `redelegate_funds`, and `delegate_funds` are called the contract's balance will incorrectly pay the amount from the contract's funds, or if the balance is not enough or the reward denom is not in the contract's balance it will block the execution of the calling message which can create a situation where the owner cannot perform any delegation related actions on the contract unless they manually send it funds to bypass the error. Additionally, in the finding below we cover that the rewards are also sent to the incorrect address.

Recommendation

We recommend adding the distribution message mentioned above to the `_withdraw_delegation_rewards` function. Additionally, the `DistributionMsg::WithdrawDelegatorReward` message can be removed from `claim_delegator_reward` since it calls `_withdraw_delegation_rewards`.

2. Rewards are incorrectly sent to owner

RISK IMPACT: CRITICAL	STATUS: RESOLVED

Description

The `_withdraw_delegation_rewards` function in `src/contract.rs:346` sends the accumulated rewards to `info.sender`. In the context of the transaction where this is called, `info.sender` is `config.owner`. `config.owner` is the only caller of `undelegate_funds`, `redelegate_funds`, `claim_delegator_reward`, and `delegate_funds`. This is because the `MessageInfo` passed to `_withdraw_delegation_rewards` specifies `config.owner` as `info.sender`. The result of this is that the owner will receive delegation rewards that should have been directed towards the recipient. This will happen in any call related to the delegation functionality. While there is no included documentation or code comments to support this, it is likely that the rewards should be sent to the vesting contract itself or the recipient.

Recommendation

We recommend adjusting the `to_address` in `line 346` to be the vesting contract, and also documenting the expected flow in a code comment or project documentation.

3. Potential mishandling of contract funds

RISK IMPACT: SEVERE	STATUS: ACKNOWLEDGED

Revision Notes

The team has mentioned that this approach is intentional. They intend to carry out the delegation, undelegation, redelegation, and reward withdrawals throughout the vesting period. In doing so, they aim to emulate the behavior of existing Luna vesting implementations for airdropped wallets post-genesis.

Description

The vesting contract performs loose accounting of the contracts balance, with no differentiation between rewards, vested funds, or funds to be delegated. This may cause situations where funds are unintentionally mishandled. For example, the owner may call `delegate_funds`, which delegates a specified amount from the contract's balance. This can cause a situation where vested funds are inappropriately delegated before the recipient can withdraw them. Additionally, there is no documentation or code comments to describe these fund flows.

Recommendation

We recommend documenting the expected funds flows. If it falls within the expected design of the contract, we recommend adding logic to handle info.funds sent with the delegation message, so that only funds passed in the message are delegated and not an uncapped amount from the contracts balance.

4. Incorrect vesting for non luna denominations

RISK IMPACT: MODERATE	STATUS: ACKNOWLEDGED

Revision Notes

The team has mentioned that this is expected since they are exclusively vesting Luna. In the event that any other tokens find their way into the contract, they retain the right to promptly withdraw them.

Description

The `withdraw_vested_funds` function in `src/contract.rs:441` does not implement a linear vesting schedule if the denom of the vesting token is not `uluna`. This means that any vesting schedule with a non luna denom will immediately vest the current balance of the contract regardless of where the contract is in its vesting timeline.

We classify this issue as a moderate severity because withdrawing vested non luna denoms does not impact the `last_withdrawn_time` or the luna vesting schedule and appears to be added to withdraw additional denoms from the contract. If a vesting contract is deployed with the intention for the main vesting denom to not be luna, then this issue is more problematic.

Recommendation

We recommend documenting the expected funds flows. If it falls within the expected design of the contract, we recommend adding logic to handle `info.funds` sent with the delegation message, so that only funds passed in the message are delegated and not an uncapped amount from the contracts balance.

5. Instantiate lacking validations

RISK IMPACT: MODERATE	STATUS: PARTIALLY RESOLVED

Revision Notes

The validation `cliff_amount` being less than `vesting_amount` was not implemented.

Description

The `instantiate` function in `src/contract.rs:14` is lacking validations to ensure that misconfigurations that affect the contracts functionality are not introduced.

The following validations should be implemented:

- Ensure that the `start_time` is greater than or equal to `env.block.time.seconds`
  - Impact: if the `start_time` is less than the current time, it will have unintended consequences on the vesting schedules
- Validate that `start_time` less than `end_time`
  - Impact: if the `start_time` is equal to or greater than `end_time` then it will cause a divide by zero panic in `withdraw_vested_funds`.
- Validate that `cliff_amount` is less than `vesting_amount`
  - Impact: if the `cliff_amount` exceeds or equals the `vesting_amount`, it means that there is no vesting left for the recipient beyond the cliff period.

Recommendation

We recommend implementing the validations mentioned above.

6. Vesting does not implement cooldown period

RISK IMPACT: LOW	STATUS: ACKNOWLEDGED

Revision Notes

The team has mentioned that due to withdrawals only being executed by them (with TFL being the probable sole whitelist member), they will personally oversee the cooldown period, which will not be enforced by contracts.

Description

The `withdraw_vested_funds` function in `src/contract.rs:400` does not implement a cooldown period. This means that the whitelisted user can call the `withdraw_vested_funds` function at any interval. This may potentially introduce scenarios where integer division truncation causes the `NothingToWithdraw` to be raised or also call it very frequently and receive minimal funds. It is best practice to allow a sufficient amount of time to pass between withdrawals.

Recommendation

We recommend implementing a minimum period that must pass between calls to `withdraw_vested_funds`.

7. Updating the owner does not update the whitelisted addresses

RISK IMPACT: LOW	STATUS: RESOLVED

Description

In the `update_owner` function in `src/contract.rs:109`, when updating to a new owner address, the contract retains the previous owner's address in the `whitelisted_addresses`. The previous owner would still have access to `withdraw_cliff_vested_funds` and `withdraw_vested_funds`. As of now, if the owner were to be changed, it would require a two-step process to update the whitelist and the owner.

Recommendation

We recommend adding the new owner's address into `whitelisted_addresses` and removing the old owner's address from it when updating ownership.

8. Potential integer truncation in vesting formula

RISK IMPACT: INFO	STATUS: ACKNOWLEDGED

Description

The community vesting contract utilizes a linear vesting formula to calculate the `amount_withdrawable` by the user in `src/contract.rs:428`. The formula currently employs integer arithmetic for its calculations. Due to the nature of integer division in Rust (u64), the formula may result in incorrect calculations, specifically when the numerator is smaller than the denominator.

The formula can be summarized as:

```
Amount_withdrawable = Total Vesting Amount - Amount since last vesting - Vesting amount remaining
```

While the `vesting_amount` is used to pad the numerator, there are still cases where the numerator can be sufficiently small to cause an integer truncation. For example, if the vesting token is a low amount and has a small decimal, then this could lead to full funds being prematurely being vested before the vesting period ends.

This issue is very unlikely to occur within normal vesting schedule timeframes and with the uluna decimal of 10^6, but we raise the finding as an informational to ensure that the same formula is not used for denoms with a smaller decimal or expected amount.

Recommendation

In future iterations of the contract, if additional denominations are added, we recommend implementing a scaling factor so that the numerator cannot be less than the denominator resulting in integer truncation.

9. Addresses are not validated

RISK IMPACT: INFO	STATUS: RESOLVED

Description

The `addToWhitelistMsg` function in `src/contract.rs:179` does not adequately validate the addresses that are being added to the whitelist. The function does not ensure the addresses specified are valid `Addr`.

Recommendation

We recommend validating the addresses with the `addr_validate` function.

10. Event emitting wrong response

RISK IMPACT: INFO	STATUS: RESOLVED

Description

In the `withdraw_cliff_vested_funds` function in `src/contract.rs:394`, the value of the "*action*" attribute key emitted "`withdraw_vested_funds`" when it should be "`withdraw_cliff_vested_funds`".

Additionally, in the `claim_delegator_reward` function in `src/contract.rs:286`, the value of the "*action*" attribute key emitted "`withdraw_delegator_rewards`". This is misleading because other emitted actions follow the executed function name.

Recommendation

We recommend modifying the values to the associated function name.

11. Implement a two-step ownership transfer

RISK IMPACT: INFO	STATUS: ACKNOWLEDGED

Description

It is a good practice to implement a two-step ownership transfer. The two-step process provides additional security and control during the migration.

Using a two-step ownership transfer mechanism helps provide a window of opportunity for the current owner to cancel the transfer if they did not intend to initiate it or if there were any unintended actions.

Recommendation

We recommend modifying the code to implement two-step ownership transfer.

Document Control

Version	Date	Notes
-	13th October	Security audit commencement date.
0.1	17th October	Initial report with identified findings delivered.
0.5	18th October	Fixes remediations implemented and reviewed.
1.0	18th October	Audit completed, final report delivered.

Appendices

A. Appendix – Risk assessment methodology

SCV-Security employs a risk assessment methodology to evaluate vulnerabilities and identified issues. This approach involves the analysis of both the LIKELIHOOD of a security incident occurring and the potential IMPACT if such an incident were to happen. For each vulnerability, SCV-Security calculates a risk level on a scale of 5 to 1, where 5 denotes the highest likelihood or impact. Consequently, an overall risk level is derived from combining these two factors, resulting in a value from 10 to 1, with 10 signifying the most elevated level of security risk

Risk Level	Range
CRITICAL	10
SEVERE	From 9 to 8
MODERATE	From 7 to 6
LOW	From 5 to 4
INFORMATIONAL	From 3 to 1

LIKELIHOOD and **IMPACT** would be individually assessed based on the below:

Rate	LIKELIHOOD	IMPACT
5	**Extremely Likely**	Could result in severe and irreparable consequences.
4	**Likely**	May lead to substantial impact or loss.
3	**Possible**	Could cause partial impact or loss on a wide scale.
2	**Unlikely**	Might cause temporary disruptions or losses.
1	**Rare**	Could have minimal or negligible impact.

B. Appendix – Report Disclaimer

This report should not be regarded as an "endorsement" or "disapproval" of any specific project or team. These reports do not indicate the economics or value of

any "product" or "asset" created by a team or project that engages SCV-Security for a security review. The audit report does not make any statements or warranties about the code's utility, safety, suitability of the business model, regulatory compliance of the business model, or any other claims regarding the fitness of the implementation for its purpose or its bug-free status. The audit documentation is intended for discussion purposes only. The content of this audit report is provided "as is," without representations and warranties of any kind, and SCV-Security disclaims any liability for damages arising from or in connection with this audit report. Copyright of this report remains with SCV-Security.

THANK YOU FOR CHOOSING

SCV SECURITY

scv.services

contact@scv.services