Terra - Airdrop Contracts - Audit Report

Prepared for Terra, 17 August 2022

Table of Contents

Introduction

SCV was engaged by Terra to assist in identifying security threats and vulnerabilities that have the potential to affect their security posture. Additionally, SCV will assist the team in understanding the risks and identifying potential mitigations.

Scope

SCV performed the security assessment on the following codebase:

- https://github.com/terra-money/airdrop/
- CodeFreeze hash: *9c8c7f063ef1a22f45812b288a0e7df09be29bb5*

Remediations were successfully applied into the following PRs:

- https://github.com/terra-money/airdrop/pull/40
- https://github.com/terra-money/airdrop/pull/45

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 Terra. Testing includes, but is not limited to, the following:

- Understanding the application and its code base 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 and Test Coverage

SCV used a scale from **0** to **10** that represents how **SUFFICIENT(6-10)** or **NOT SUFFICIENT(0-5)** each code criteria was during the assessment:

Criteria	Status	Scale Range	Notes
Provided Documentation	**Sufficient**	7-8	N/A
Code Coverage Test	**Sufficient**	7-8	N/A
Code Readability	**Sufficient**	6-8	N/A
Code Complexity	**Sufficient**	6-7	N/A

Vulnerabilities Summary

Airdrop Contracts

	Title and Summary	Risk	Status
1	Lack of validations during contract instantiation	**Medium**	**Remediated**
2	Missing validation on the admin address when updating configuration	**Medium**	**Remediated**
3	Confusing error message for incorrect signatures when claiming airdrop	**Informational**	**Remediated**
4	Consider removing either UpdateMerkleRoot or RegisterMerkleRoot	**Informational**	**Remediated**

Backend and Frontend

	Title and Summary	Risk	Status
5	Config variables can be fetched from the backend	**Medium**	**Remediated**
6	Dependencies should be pinned to exact versions in package.json	**Low**	**Remediated**
7	Vulnerable library in use.	**Low**	**Remediated**

Detailed Vulnerabilities: Airdrop Contracts

1. Lack of validations during contract instantiation

Likelihood	Impact	Risk
Unlikely	Severe	**Medium**

Notes

The *start_time* is expected to be lesser than *current_time* as team advices that would set vesting from genesis. Based on that, SCV consider this issue remediated.

Description

In *airdrop/contracts/airdrop/src/contract.rs:33,34*, there are no validations to make sure the provided *msg.start_time* and *msg.vesting_periods* are both positive values while *msg.claim_end_time* is in future time.

If *msg.start_time* and *msg.vesting_periods* are incorrectly instantiated as negative values, it would cause users to be unable to claim airdrop due to the transaction failing in *airdrop/contracts/airdrop/src/submsg.rs:92-95*. This is because the *ValidateBasic* method in *MsgCreatePeriodicVestingAccount* will reject invalid start time and vesting period lengths as per reference:

- https://github.com/cosmos/cosmos-sdk/blob/main/x/auth/vesting/types/msgs.go#L177-L185

As for *msg.claim_end_time*, misconfiguring its value to the past timestamp would cause users to be unable to claim airdrop and the admin unable to call *end_airdrop* as seen in lines 151 and 278, respectively.

Additionally, *msg.start_time* is not validated to be in the future of the current timestamp. If it is instantiated in the past, there's a chance that the vested tokens would be unlocked immediately, which defeats the whole purpose of the vesting periods.

Recommendations

Consider applying the following validations:

- Validate value of *msg.start_time* and *msg.vesting_periods* to be positive values (larger than 0)
- Verify value of *msg.start_time* and *msg.claim_end_time* is not in the past (larger than *env.block.time seconds()*)

2. Missing validation on the admin address when updating configuration

Likelihood	Impact	Risk
Unlikely	Moderate	**Medium**

Description

When updating the admin address in *airdrop/contracts/airdrop/src/contract.rs:101*, there is no validation that the passed admin *String* is a valid address. If the newly updated admin is not a valid address, it will cause *UpdateMerkleRoot*, *RegisterMerkleRoot*, *UpdateConfig*, and *End* functionality to be inaccessible.

Recommendations

Consider validating the newly admin address to be valid using *deps.api.addr_validate* as seen in line 30.

3. Confusing error message for incorrect signatures when claiming airdrop

Likelihood	Impact	Risk
Rare	Informational	**Informational**

Description

In *airdrop/contracts/airdrop/src/contract.rs:177-183*, if the signature provided by the user is in the correct format but turns out to be invalid, the *verified* variable will become false, and the error message in lines 179 to 181 will be displayed to the user. This is problematic because the error message is only intended for the Terra chain (see *airdrop/contracts/airdrop/src/verification.rs:151-153*).

Suppose the airdrop contract is used for other chains such as Ethereum or Cosmos. In that case, it will emit a confusing error message that it expects the address being provided is a terra address but received a native account address, which is incorrect and confusing to the users.

Recommendations

Consider reworking the function to return the correct error message to users.

4. Consider removing either UpdateMerkleRoot or RegisterMerkleRoot

Likelihood	Impact	Risk
Rare	Informational	**Informational**

Description

The function *UpdateMerkleRoot* and *RegisterMerkleRoot* in *airdrop/contracts/airdrop/src/contract.rs:68* and line 114 are similar in terms of functionality other than their emitted action attributes. As both function's outcomes are the same, it is recommended only to have one of them in the contract as it is enough for the overall use cases.

Recommendations

Consider removing the *RegisterMerkleRoot* functionality in the contract and use *UpdateMerkleRoot* if there is a need to update the *MERKLE_ROOT* state.

Detailed Vulnerabilities: Backend and Frontend

5. Config variables can be fetched from the backend

Likelihood	Impact	Risk
Unlikely	Severe	**Medium**

Description

In the backend controller, there is an endpoint that allows environment variables to be fetch using a *GET /config* request.

This is problematic because environment variables typically holds sensitive information that should not be exposed

Recommendations

Remove the endpoint from the controller as it provides no utility to the presented solution.

6. Dependencies should be pinned to exact versions in package.json

Likelihood	Impact	Risk
Rare	Moderate	**Low**

Description

The backend and frontend contains over 45 dependencies that are not pinned to an exact version in the *package.json* file.

This can potentially allow dependency attacks, as seen with the flow of events package with *Copay Bitcoin Wallet* for example.

Recommendations

Considering pin dependencies to an exact versions in package.json. As an example, from *^1.1.0* or *~1.1.0* simply do *1.1.0* to specify it.

7. Vulnerable library in use.

Likelihood	Impact	Risk
Rare	Low	**Low**

Description

The frontend depends on vulnerable software versions that could be upgraded to the latest. Vulnerable versions were:

- axios <0.21.2 (Incorrect Comparison vulnerability)
- nth-check <2.0.1 (Inefficient Regular Expression Complexity)

Additional information from advisories can be found in the links below:

- https://github.com/advisories/GHSA-cph5-m8f7-6c5x
- https://github.com/advisories/GHSA-rp65-9cf3-cjxr

Recommendations

Upgrade the dependency to latest using `npm audit fix`. The https://github.com/features/security can also assist in dependency updates and monitoring advisories on dependencies in use.

Document control

Document changes

Version	Date	Name	Changes
0.1	2022-08-12	Vinicius Marino	Initial report
0.2	2022-08-12	Vinicius Marino	Team communication and Pre-Release
1.0	2022-08-17	Vinicius Marino	Remediations Review and Document Release

Document contributors

Name	Role	Email address
Vinicius Marino	Security Specialist	vini@scv.services

Appendices

Appendix A: Report Disclaimer

The content of this audit report is provided “As is”, without representations and warranties of any kind.

The author and their employer disclaim any liability for damage arising out of, or in connection with, this audit report.

Appendix B: Risk assessment methodology

A qualitative risk assessment is performed on each vulnerability to determine the impact and likelihood of each.

Risk rate will be calculated on a scale. As per criteria Likelihood vs Impact table below:

Impact \ Likelihood	Rare	Unlikely	Possible	Likely
Critical	Medium	High	Critical	Critical
Severe	Low	Medium	High	High
Moderate	Low	Medium	Medium	High
Low	Low	Low	Low	Medium
Informational	Informational	Informational	Informational	Informational

LIKELIHOOD:

- **Likely**: likely a security incident will occur;
- **Possible**: It is possible a security incident can occur;
- **Unlikely**: Low probability a security incident will occur;
- **Rare**: In rare situations, a security incident can occur;

IMPACT:

- **Critical**: May cause a significant and critical impact;
- **Severe**: May cause a severe impact;
- **Moderate**: May cause a moderated impact;
- **Low**: May cause low or none impact;
- **Informational**: May cause very low impact or none.


SCV