SCV SECURITY

AUDIT REPORT

TerraForm Labs

Warp Contracts

Prepared by SCV-Security

Prepared on 14th August 2023

Table of Contents

Introduction

SCV has been engaged by TerraForm Labs 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

The following CosmWasm contracts and packages were in scope:

- **Contracts**
  - warp-account
  - warp-controller
  - warp-resolver
  - warp-templates
- **Packages**
  - account
  - controller
  - resolver
  - templates

Functionality

Warp is a decentralized and open-source event handler designed for Cosmos blockchains. It enables users to add various messages, known as "jobs," into a queue. These jobs are executed when specific conditions are met. Users have the flexibility to define complex conditions by combining multiple expressions. Each job message includes a reward for its execution. Once the specified condition is met, any participant can execute the job and claim the associated reward.

Submitted Codebase

- **Repository**
  - https://github.com/terra-money/warp-contracts/tree/chore/contract-split
- **Commit**
  - `74b278f9c470521fa9e898a7f7d98359ca61f46d`

Revisions Remediations

Remediations recommendations were applied by TerraForm Labs and revised by SCV team on the following:

- **Repository**
  - https://github.com/terra-money/warp-contracts/tree/audit/aug7-scv
- **Commit**
  - `345cc9b6853daa7161f4f53c6b6a56b9859559e3`

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 TerraForm Labs 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.
- Analyze 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 organization.
- **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 categorized 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**	The documentation available at docs.warp.money is comprehensive, providing rich details on protocol mechanics from both technical and user perspective.
Coverage	**NOT-SUFFICIENT**	The codebase did not have sufficient testing coverage. Many components were not tested at all and a majority of the tests were empty.
Readability	**SUFFICIENT**	Overall the code was readable, although the codebase lacked code comments and consistent variable naming conventions
Complexity	**SUFFICIENT**	N/A

Findings Summary

Summary Title	Risk Impact	Status
Queue length not decremented when job evicted leading to premature evictions	SEVERE	RESOLVED
`State.q` is reduced twice when a job is executed	SEVERE	RESOLVED
Updating `fee_denom` in `update_config` will introduce issues with existing jobs	SEVERE	RESOLVED
`HydrateVars` doesn't work for nested base64 messages	SEVERE	RESOLVED
Hardcoded fee denom in Warp Templates	MODERATE	RESOLVED
Outstanding `TODO` comments in the codebase	INFO	RESOLVED
Replace magic numbers	INFO	RESOLVED
Remove commented code blocks	INFO	RESOLVED
`edit_template` function emits the wrong event attribute	INFO	RESOLVED
Contract should use two-step ownership transfer	INFO	ACKNOWLEDGED
Variable naming can be improved	INFO	RESOLVED
Unimplemented query entrypoint	INFO	RESOLVED

Audit Observations

The audit observations section is intended to present potential findings that are related to the underlying design of the protocol and would require underlying design changes to remediate that may change the overall functioning of the protocol. SCV asks that the client formulate responses to add context to validate or invalidate the following concerns.

1. Inadequate protection from spamming jobs

To execute messages, keepers query the controller with a paginated query. An attacker could create multiple failing jobs with high fees to slow the process of keepers and avoid the execution of real jobs, this attack would be relatively inexpensive to execute as the rewards for jobs are relatively cheap.

Example:

- An attacker creates `10,000 jobs` with a reward of `0.5 Luna` for a total of `2500$.`
  To fetch the jobs keepers will use `QueryJobs` that has a limit of `50 jobs` per page.
- Every keeper will need to call the endpoint `200` times and simulate `10,000 jobs` each block.
- Depending on their hardware, keepers will probably need more than 6 seconds to evaluate all those jobs.

Potential Remediation:

Consider adjusting the query queueing to have a reputation based field. Store the amount of successful jobs for a specific account. And add a query that allows keepers to get the highest combination or reward and reputation.

2. Protocol does not prevent front running of jobs

Currently, the protocol does not prevent jobs from being front run. An example of this can be Astroport limit orders. This can be done by executing the Limit Order on Warp using a smart contract that crafts a message with a buy on Astroport followed by the execution of the job and a sell message on Astroport. Since the transaction will be atomically executed this operation is executed to ensure the actor is guaranteed a profit.

Due to the transparent nature of the queued jobs, this issue may be difficult to remediate. One potential solution is to obfuscate the job message information. This would prevent attackers from enumerating the potential messages to front run, but it may not be in line with the current transparent design principles of the warp protocol

Findings Technical Details

1. Queue length not decremented when job evicted leading to premature evictions

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

The `EvictJob` function does not decrement the queue length `q` when a job is evicted. This may create a situation where `q` could become inflated over time as evictions occur but `q` is not decreased. This could lead to `t` being calculated lower than it should be which could ultimately create a situation where jobs may be prematurely evicted.

Additionally this would also create a situation where the value of `a` could be incorrectly determined due to an inflated `q` size. This would result in the job account being charged more than they should be based on the protocol design

Recommendation

We recommend decreasing the value of `q` when a job is evicted.

2. `State.q` is reduced twice when a job is executed

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

The `q` state variable is used to measure the number of jobs in the queue. When jobs are created `state.q` is increased and when jobs are executed or deleted, the value is decreased. Currently `state.q` is decreased in `execute_job` in `contracts/warp-controller/src/execute/job.rs:389` which is intended, but then it is decreased a second time in `contracts/warp-controller/src/contract.rs:238` when the job execute message reply is handled.

This will result in a situation where the `state.q` value does not accurately reflect the amount of jobs which will directly impact the eviction functionality.

Recommendation

We recommend removing the second instance of `state.q` being decreased.

3. Updating `fee_denom` in `update_config` will introduce issues with existing jobs

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

The `update_config` function in `contracts/warp-controller/src/execute/controller.rs:22` allows the contract owner to update the `fee_denom`. This action is performed without the consideration of existing jobs which may introduce a number of unintended consequences.

When a job is created, its reward is stored as a `Uint128` amount in the current fee denom (`fee_denom0`). This reward in `fee_denom0` is sent to the controller contract to ensure proper functioning of the job. However, if the admin uses the `update_config` function to change the fee denom to `fee_denom1`, the existing jobs that were already rewarded in `fee_denom0` will cause the reward amount to be sent in `fee_denom1` instead. This may work for some jobs as long as the contract has enough balance in `fee_denom1`, but there might come a point where the controller contract doesn't have sufficient funds in the new fee denom to pay the reward, resulting in a protocol-wide blockage until more `fee_denom1` funds are added.

Furthermore, this situation could lead to a scenario where a job is created with one denom, but later, when the denom is changed, using `delete_job` or `evict_job` could return a different denom than the one initially used to create the job. Depending on the reward amount and the difference between the denoms, this could result in the job owner receiving more or less value than the original reward amount they sent.

Recommendation

We recommend removing the ability for the `fee_denom` to be updated or alternatively updating the job struct to hold the denom that was defined at the initial job creation and then ensuring this denom is used for all operations with the job rather than the config denom.

4. `HydrateVars` doesn't work for nested base64 messages

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

In case of nested base64 encoded messages in `QueryHydrateVars` located in `contracts/warp-resolver/src/contract.rs:98`
the contract will not correctly replace variables in nested base64 encoded strings. This will cause jobs with nested base64 messages to fail.

Recommendation

Consider implementing a mechanism to decode nested base64 messages or consider preventing nested base64 messages.

5. Hardcoded fee denom in Warp Templates

RISK IMPACT: MODERATE	STATUS: RESOLVED

Description

`uLuna` is hard coded as a fee denom in multiple locations, this can lead to problems if warp is deployed on other chains (such as injective).

- `contracts/warp-templates/src/contract.rs:92`
- `contracts/warp-templates/src/contract.rs:140`

Recommendation

Consider using the field `fee_denom` instead.

6. Outstanding TODO comments in the codebase

RISK IMPACT: INFO	**STATUS:** RESOLVED

Description

The following code lines contain outstanding TODO comments:

- `contracts:/warp-controller/src/contract.rs:307`
- `contracts/warp-controller/src/execute/job.rs:44`
- `contracts/warp-controller/src/execute/job.rs:342`
- `contracts/warp-controller/src/execute/job.rs:353`
- `contracts/warp-resolver-src/util/variable.rs:368`
- `contracts/warp-template/src/contract.rs:114`
- `contracts/warp-template/src/contract.rs:264`

Recommendation

Consider resolving the TODO comments.

7. Replace magic numbers

RISK IMPACT: INFO	**STATUS:** RESOLVED

Description

There are several instances within the scope of the audit that use hard-coded number literals to represent the maximum number of characters allowed for the name of a job. It is best practice to create variables that represent this value as it improves code readability and prevents developers from making inconsistent changes across the codebase.

The following lines of code uses magic numbers to represent the above value :

- `contracts/warp-controller/src/execute/job.rs:24`
- `contracts/warp-controller/src/execute/job.rs:222`
- `contracts/warp-templates/src/contract.rs:98`
- `contracts/warp-templates/src/contract.rs:106`
- `contracts/warp-templates/src/contract.rs:166`

Recommendation

We recommend replacing the magic numbers mentioned above with a constant that is descriptive of its value and use case.

8. Remove commented code blocks

RISK IMPACT: INFO	STATUS: RESOLVED

Description

There are a number of commented code blocks within the scope of this audit. It is best practice to remove these non implemented codeblocks before the contracts are released to improve the readability and maintainability of the codebase.

The following lines of code are commented and unused:

- `contracts/warp-controller/src/execute/job.rs:58-61`
- `contracts/warp-controller/src/execute/job.rs:329-334`
- `contracts/warp-controller/src/execute/job.rs:345`

Recommendation

We recommend removing the commented code blocks that are unused.

9. `edit_template` function emits the wrong event attribute

RISK IMPACT: INFO	STATUS: RESOLVED

Description

In `contracts/warp-templates/src/contract.rs:188`, the `edit_template` function emits the "`action`" attribute "`submit_msg_template`".

Recommendation

We recommend replacing the "`action`" attribute to "`edit_template`".

10. Contract should use two-step ownership transfer

RISK IMPACT: INFO	STATUS: ACKNOWLEDGED

Description

The current ownership transfer for the contract is executed in one step, which imposes a risk that if the new owner is incorrect, then the admin privileges of the contract are effectively transferred and lost. It is a good practice to implement a two-step ownership transfer. The two-step process provides an additional layer of security and control during the migration.

Recommendation

We recommend implementing a two-step ownership transfer where the current owner proposes a new owner address, and then that new owner address must call the contract to accept ownership within a finite time frame. SCV suggests the following implementation:

- https://docs.rs/cw-controllers/latest/cw_controllers/index.html

11. Variable naming can be improved

RISK IMPACT: INFO	STATUS: RESOLVED

Description

Throughout the codebase there are ambiguous variable names and other naming conventions that do not follow best practices. To increase the readability of the codebase we recommend implementing consistent naming conventions and utilising descriptive variable names:

- In `contracts/warp-controller/src/execute/job.rs:46`, the variable name `q` overlaps with the state variable `q`.
- The variables `q` and `a` are not descriptive of their function and do not have comments describing its functionality in `evict_jobs`
- The config variables `t_max`, `t_min`, `a_max`, `a_min` and `q_max` and the state variable `q` have non-descriptive names and do not have any documentation to support their functionality.

Recommendation

We recommend updating the variables mentioned above to improve their readability.

12. Unimplemented query entrypoint

RISK IMPACT: INFO	STATUS: RESOLVED

Description

The `warp_account` contract’s `query` function is currently not implemented. It is best practice to implement a query entrypoint so that the contract conforms to the generic contract interface and its config values can be queried.

Recommendation

We recommend updating the query to return the configuration information of the contract.

Document Control

Version	Date	Notes
-	31st July 2023	Security audit commencement date.
0.1	7th August 2023	Initial report with identified findings delivered.
0.5	12th August 2023	Performed revisions on implemented remediations.
1.0	14th August 2023	Final report delivery and engagement completed.

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

THANK YOU FOR CHOOSING

SCV
SECURITY

scv.services

contact@scv.services