HALBORN

# Stader Labs LunaX Contracts (Updated code)

CosmWasm Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **March 16th, 2022 - March 17th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY 3
CONTACTS 3
1 EXECUTIVE OVERVIEW 4
1.1 INTRODUCTION 5
1.2 AUDIT SUMMARY 5
1.3 TEST APPROACH & METHODOLOGY 5
RISK METHODOLOGY 6
1.4 SCOPE 8
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW 9
3 FINDINGS & TECH DETAILS 10
3.1 (HAL-01) LACK OF ADDRESS NORMALIZATION - LOW 12
Description 12
Code Location 12
Risk Level 12
Recommendation 12
Remediation plan 13
3.2 (HAL-02) CONFIGURATION PARAMETER NOT SET UPON INSTANTIATION - INFORMATIONAL 14
Description 14
Code Location 14
Risk Level 14
Recommendation 15
Remediation plan 15
4 AUTOMATED TESTING 16
4.1 AUTOMATED ANALYSIS 17

Description 17

## DOCUMENT REVISION HISTORY

| VERSION | MODIFICATION | DATE | AUTHOR |
|---|---|---|---|
| 0.1 | Document Creation | 03/16/2022 | Jose C. Ramirez |
| 0.2 | Draft Version | 03/16/2022 | Jose C. Ramirez |
| 0.3 | Draft Review | 03/16/2022 | Gabi Urrutia |
| 1.0 | Remediation Plan | 03/17/2022 | Jose C. Ramirez |
| 1.1 | Remediation Plan Review | 03/17/2022 | Gabi Urrutia |

## CONTACTS

| CONTACT | COMPANY | EMAIL |
|---|---|---|
| Rob Behnke | Halborn | Rob.Behnke@halborn.com |
| Steven Walbroehl | Halborn | Steven.Walbroehl@halborn.com |
| Gabi Urrutia | Halborn | Gabi.Urrutia@halborn.com |
| Jose C. Ramirez | Halborn | jose.ramirez@halborn.com |

# EXECUTIVE OVERVIEW

## 1.1 INTRODUCTION

Stader Labs engaged Halborn to conduct a security audit on their updated smart contracts beginning on March 16th, 2022 and ending on March 17th, 2022 . The security assessment was scoped to the smart contracts provided to the Halborn team.

## 1.2 AUDIT SUMMARY

The team at Halborn assigned a full-time security engineer to audit the security of the latest update on the codebase of the LunaX smart contracts. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is:

- Ensure smart contract functions work as intended
- Identify potential security issues with smart contracts

In summary, Halborn identified some improvement to reduce the likelihood and impact of multiple risks, which has been addressed by Stader Labs. The main being the following:

- Some addresses still lacked lowercase normalization, although they did not present a great risk.

## 1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual review of the code and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the smart contract audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture, purpose, and use of the platform.
- Manual code read and walkthrough.
- Manual assessment of use and safety for the critical Rust variables and functions in scope to identify any contracts logic related vulnerability.
- Fuzz testing (Halborn custom fuzzing tool)
- Checking the test coverage (cargo tarpaulin)
- Scanning of Rust files for vulnerabilities (cargo audit)

## RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

**RISK SCALE - LIKELIHOOD**

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

**RISK SCALE - IMPACT**

5 - May cause devastating and unrecoverable impact or loss.

4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

| CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL |
|---|---|---|---|---|

**10** - CRITICAL
**9 - 8** - HIGH
**7 - 6** - MEDIUM
**5 - 4** - LOW
**3 - 1** - VERY LOW AND INFORMATIONAL

## 1.4 SCOPE

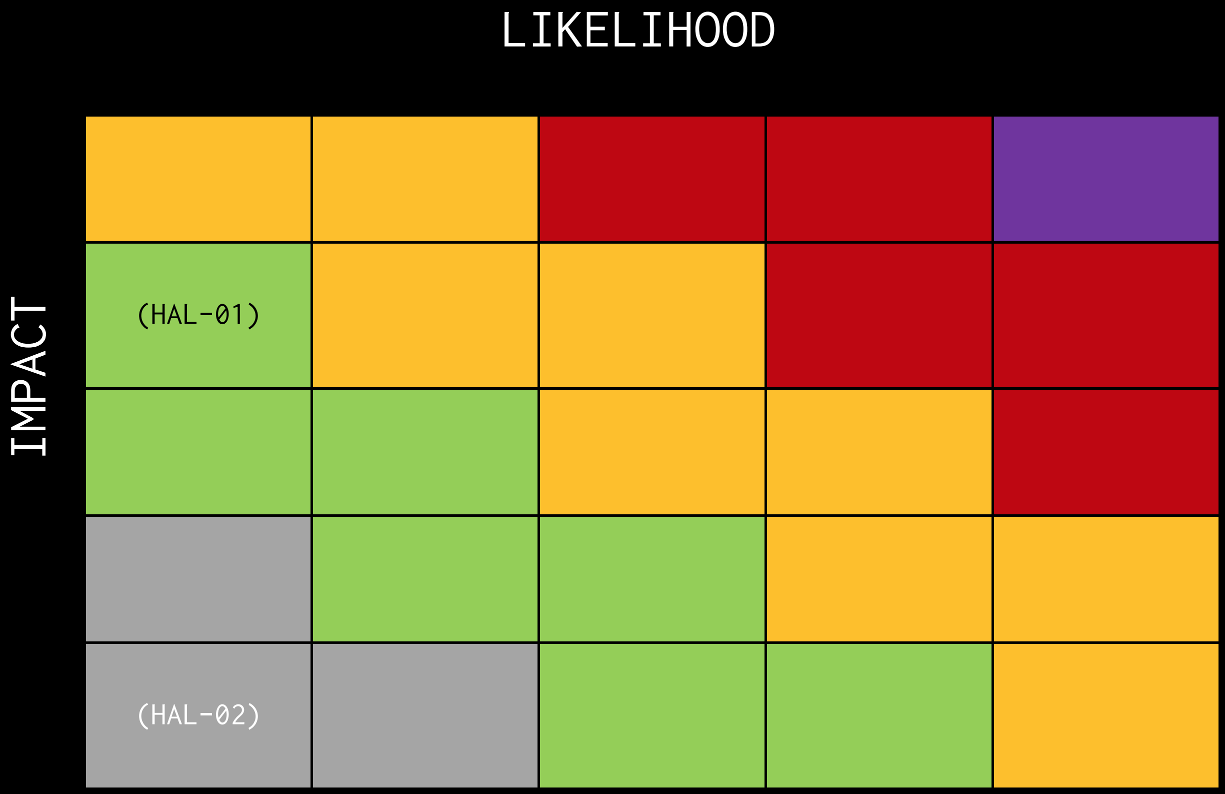
Code repository: https://github.com/stader-labs/stader-liquid-token

1. CosmWasm LunaX Smart Contracts
   (a) Commit ID: 2753227bf6f959b5b3eb0625aeea1a686997ec74
   (b) Contracts within scope:
      i. airdrops-registry
      ii. reward
      iii. staking

Out-of-scope: External libraries and financial related attacks.

## 2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

| CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |



| SECURITY ANALYSIS | RISK LEVEL | REMEDIATION DATE |
|---|---|---|
| (HAL-01) LACK OF ADDRESS NORMALIZATION | Low | SOLVED - 03/17/2022 |
| (HAL-02) CONFIGURATION PARAMETER NOT SET UPON INSTANTIATION | Informational | ACKNOWLEDGED |

## FINDINGS & TECH DETAILS

## 3.1 (HAL-01) LACK OF ADDRESS NORMALIZATION - LOW

### Description:

The multiple functionalities of the contracts within do not consider Terra addresses to be valid in both upper and all lower case. While valid, a strict comparison between the same address in its all uppercase version (e.g.: **TERRA1KG...XNL8**) and its all lowercase version (e.g.: **terra1kg...xnl8**) will fail. A clear example of when this can create a security issue is when `info.sender` is compared to a previously stored address to enforce access controls.

The risk of this issue has been reduced as the instances that could create the greatest impact are `manager` only functionalities, where the likelihood of failure is much lower than user-accessible features. The rest of the instances are in `query` functions, thus causing an annoyance to users rather than a security risk.

### Code Location:

**Listing 1: Affected resources**

```
contracts/reward/src/contracts.rs #32
contracts/staking/src/contracts.rs #285, 1142, 1265, 1304, 1330
```

### Risk Level:

**Likelihood - 1**
**Impact - 4**

### Recommendation:

This issue can be solved using the two different approaches detailed below:

- In addition to validation, addresses should be normalized to lower-case before being stored for future use.
- Instead of just validating, addresses should be stored in canonical format using deps.api.addr_canonicalize().

The following considerations should be taken into account when implementing the second option:

- To successfully compare a canonical address, both ends should be in canonical format. For example, when performing access controls, the sender (for example, info.sender or env.message.sender) should also be canonicalized beforehand.
- To send funds to a canonicalized address or include it in a message to a different contract, it should first be converted to its human-readable format via deps.api.addr_humanize()

Remediation plan:

**SOLVED:** On line 1142, the input was sanitized before the compute_withdrawable_funds function was called.

This issue was fixed in commit 3a7eb2c6b9806b806f15d761af37aa0c23d6130d.

## 3.2 (HAL-02) CONFIGURATION PARAMETER NOT SET UPON INSTANTIATION - INFORMATIONAL

Description:

The instantiate function did not set the cw20_token_contract address, as it did for other required contract addresses in the configuration. Instead, it relied on update_config being called post initialization, which could cause undesirable situations if this address is not set right after deployment.

It is worth noting that the update_config function only allowed the CW20 address to be set if it contained the initial value Addr::unchecked("0"). This effectively prohibited any future change after the first update.

Code Location:

**Listing 2: contracts/staking/src/contract.rs (Line 66)**

```
59 airdrop_registry_contract: deps
60     .api
61     .addr_validate(msg.airdrops_registry_contract.as_str())?,
62 airdrop_withdrawal_contract: deps
63     .api
64     .addr_validate(msg.airdrop_withdrawal_contract.as_str())?,
65 reward_contract: deps.api.addr_validate(msg.reward_contract.as_str
   ↳ ())?,
66 cw20_token_contract: Addr::unchecked("0"),
```

Risk Level:

**Likelihood - 1**
**Impact - 1**

Recommendation:

The cw20_token_contract variable should be set on instantiation, just like with the other contract addresses.

Remediation plan:

**ACKNOWLEDGED:** Stader Labs acknowledged this finding.

# AUTOMATED TESTING

## 4.1 AUTOMATED ANALYSIS

Description:

Halborn used automated security scanners to assist with detection of well-known security issues and vulnerabilities. Among the tools used was cargo audit, a security scanner for vulnerabilities reported to the RustSec Advisory Database. All vulnerabilities published in https://crates.io are stored in a repository named The RustSec Advisory Database. cargo audit is a human-readable version of the advisory database which performs a scanning on Cargo.lock. Security Detections are only in scope. All vulnerabilities shown here were already disclosed in the above report. However, to better assist the developers maintaining this code, the auditors are including the output with the dependencies tree, and this is included in the cargo audit output to better know the dependencies affected by unmaintained and vulnerable crates.

| ID | package | Short Description |
|---|---|---|
| RUSTSEC-2020-0025 | bigint | biginit is unmaintained, use uint instead |