// HALBORN

Stader Labs LunaX for Terra2

CosmWasm Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **June 6th, 2022 - June 8th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY 5
CONTACTS 5
1 EXECUTIVE OVERVIEW 6
1.1 INTRODUCTION 7
1.2 AUDIT SUMMARY 7
1.3 TEST APPROACH & METHODOLOGY 8
RISK METHODOLOGY 8
1.4 SCOPE 10
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW 11
3 FINDINGS & TECH DETAILS 12
3.1 (HAL-01) DEPOSIT BOUNDS NOT VALIDATED - LOW 14
Description 14
Code Location 14
Risk Level 15
Recommendation 15
Remediation plan 15
3.2 (HAL-02) STATE SAVED WITHOUT ANY CHANGE - INFORMATIONAL 16
Description 16
Code Location 16
Risk Level 16
Recommendation 16
Remediation plan 17
3.3 (HAL-03) UNNECESSARY LOWERCASE ON ADDRESS - INFORMATIONAL 18
Description 18

Code Location 18
Risk Level 19
Recommendation 19
Remediation plan 19
3.4 (HAL-04) LACK OF ADDRESS VALIDATION - INFORMATIONAL 20
Description 20
Code Location 20
Risk Level 20
Recommendation 20
Remediation plan 21
3.5 (HAL-05) UNNECESSARY LOOPING OVER VECTOR - INFORMATIONAL 22
Description 22
Code Location 22
Risk Level 22
Recommendation 22
Remediation plan 23
3.6 (HAL-06) HARDCODED DENOM IN QUERY - INFORMATIONAL 24
Description 24
Code Location 24
Risk Level 24
Recommendation 24
Remediation plan 25
3.7 (HAL-07) ADDRESS VALIDATION DONE AT ACCEPT-MANAGER STEP - INFORMATIONAL 26
Description 26
Code Location 26

Risk Level 27
Recommendation 27
Remediation plan 27
3.8 (HAL-08) DUPLICATE CODE IMPACTS MAINTAINABILITY - INFORMATIONAL 28
Description 28
Code Location 28
Risk Level 29
Recommendation 29
Remediation plan 29
3.9 (HAL-09) OVERFLOW CHECKS NOT SET FOR PROFILE RELEASE - INFORMATIONAL 30
Description 30
Code Location 30
Risk Level 30
Recommendation 30
Remediation plan 30
3.10 (HAL-10) CONFIGURATION PARAMETER NOT SET UPON INSTANTIATION - INFORMATIONAL 31
Description 31
Code Location 31
Risk Level 31
Recommendation 32
Remediation plan 32
3.11 (HAL-11) UNMANTAINED DEPENDENCY - INFORMATIONAL 33
Description 33
Code Location 33

Risk Level 34

Recommendation 34

Remediation plan 34

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	06/06/2022	Jose C. Ramirez
0.2	Draft Version	06/06/2022	Jose C. Ramirez
0.3	Draft Review	06/08/2022	Gabi Urrutia
1.0	Remediation Plan	06/10/2022	Jose C. Ramirez
1.1	Remediation Plan Review	06/10/2022	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 smart contracts beginning on June 6th, 2022 and ending on June 8th, 2022 . The security assessment was scoped to the smart contracts provided in the GitHub repository stader-liquid-token, commit hashes and further details can be found in the Scope section of this report. LunaX contracts were audited in the past, but recently received relevant updates to prepare for the Terra 2 release.

1.2 AUDIT SUMMARY

The team at Halborn was provided three days for the engagement and assigned a full-time security engineer to audit the security of the smart contract. The security engineers are blockchain and smart-contract security experts with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended
- Identify potential security issues with the smart contracts

In summary, Halborn identified some improvements to reduce the likelihood and impact of risks, which were mostly addressed by Stader Labs team. The main ones are the following:

- Perform bounds validation on all the relevant configuration parameters.
- Remove unnecessary save operations on state variables.
- Remove unnecessary lowercase operations on addresses that are being validated.

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:

The following phases and associated tools were used throughout the term of the audit:

- Research into the architecture, purpose, and use of the platform.
- Smart contract manual code review and walk-through to identify any logic issue.
- Thorough assessment of safety and usage of critical Rust variables and functions in scope that could lead to arithmetic related vulnerabilities.
- Finding unsafe Rust code usage (cargo-geiger)
- Active Fuzz testing (honggfuzz).
- Test coverage review (cargo tarpaulin).
- Local or public Testnet deployment (LocalTerra or bombay-12)
- Scanning of Rust dependencies for known 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: 86f5617343f56a7b74f4b42e84ffa54ff6317c40
   (b) Contracts in scope:
      - Airdrops Registry contract (contracts/airdrops-registry)
      - Reward contract (contracts/reward)
      - Staking contract (contracts/staking)

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

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	1	10

LIKELIHOOD

IMPACT

(HAL-01)				
(HAL-06) (HAL-07) (HAL-08) (HAL-09) (HAL-10) (HAL-11)	(HAL-02) (HAL-03) (HAL-04) (HAL-05)			

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) DEPOSIT BOUNDS NOT VALIDATED	Low	PARTIALLY SOLVED - 06/10/2022
(HAL-02) STATE SAVED WITHOUT ANY CHANGE	Informational	SOLVED - 06/09/2022
(HAL-03) UNNECESSARY LOWERCASE ON ADDRESS	Informational	SOLVED - 06/09/2022
(HAL-04) LACK OF ADDRESS VALIDATION	Informational	NOT APPLICABLE
(HAL-05) UNNECESSARY LOOPING OVER VECTOR	Informational	ACKNOWLEDGED
(HAL-06) HARDCODED DENOM IN QUERY	Informational	SOLVED - 06/09/2022
(HAL-07) ADDRESS VALIDATION DONE AT ACCEPT-MANAGER STEP	Informational	ACKNOWLEDGED
(HAL-08) DUPLICATE CODE IMPACTS MAINTAINABILITY	Informational	ACKNOWLEDGED
(HAL-09) OVERFLOW CHECKS NOT SET FOR PROFILE RELEASE	Informational	ACKNOWLEDGED
(HAL-10) CONFIGURATION PARAMETER NOT SET UPON INSTANTIATION	Informational	ACKNOWLEDGED
(HAL-11) UNMANTAINED DEPENDENCY	Informational	SOLVED - 06/09/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) DEPOSIT BOUNDS NOT VALIDATED - LOW

Description:

The min_deposit and max_deposit configuration variables of the **staking** contract do not undergo any validation step. The aforementioned values are used to limit the maximum/minimum amounts that users are allowed to deposit.

If a typing error or a malicious admin would set the minimum to a very big number or the maximum to zero, the protocol would effectively suffer a denial of service. A similar situation applies to undelegation_cooldown, unbounding_period and reinvest_cooldown.

Code Location:

Listing 1: contracts/staking/src/contract.rs (Lines 59,60)

```
56 let config = Config {
57     manager: info.sender,
58     vault_denom: "uluna".to_string(),
59     min_deposit: msg.min_deposit,
60     max_deposit: msg.max_deposit,
```

Listing 2: contracts/staking/src/contract.rs

```
76 undelegation_cooldown: msg.undelegation_cooldown,
77 unbonding_period: msg.unbonding_period,
78 reinvest_cooldown: msg.reinvest_cooldown,
```

Listing 3: contracts/staking/src/contract.rs

```
283 config.min_deposit = update_config.min_deposit.unwrap_or(config.
 ↳ min_deposit);
284 config.max_deposit = update_config.max_deposit.unwrap_or(config.
 ↳ max_deposit);
```

Listing 4: contracts/staking/src/contract.rs

```
307 config.undelegation_cooldown = update_config
308     .undelegation_cooldown
309     .unwrap_or(config.undelegation_cooldown);
310 config.unbonding_period = update_config
311     .unbonding_period
312     .unwrap_or(config.unbonding_period);
313 config.reinvest_cooldown = update_config
314     .reinvest_cooldown
315     .unwrap_or(config.reinvest_cooldown);
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

A validation routine should be added inside the instantiate and update_config functions to enforce that the values of min_deposit, max_deposit, undelegation_cooldown, unbounding_period and reinvest_cooldown are within the expected ranges.

Remediation plan:

PARTIALLY SOLVED: The Stader Labs team acknowledged the lack of bounds for reinvest_cooldown. On the other hand, validation mechanisms for undelegation_cooldown, unbounding_period, min_deposit and max_deposit were implemented in commit b27a835330f81dcaeee0a8cb091b60a5b3e4b8e1.

3.2 (HAL-02) STATE SAVED WITHOUT ANY CHANGE - INFORMATIONAL

Description:

The queue_undelEgation function loads STATE to update other details with the current values of current_undelegation_batch_id and exchange_rate. Although none of its fields are actually updated on this function, the STATE is saved at the end.

This issue does not pose an actual security threat, but unnecessary code decreases readability and increases gas costs.

Code Location:

Listing 5: contracts/staking/src/contract.rs (Line 820)

```
816             batch_undelegation.undelegation_er = state.exchange_rate;
817             Ok(batch_undelegation)
818         },
819 )?;
820 STATE.save(deps.storage, &state)?;
821
822 Ok(Response::default())
```

Risk Level:

Likelihood - 2
Impact - 1

Recommendation:

Do not save state variables if nothing has been updated.

Remediation plan:

SOLVED: The issue was fixed in commit 8121307836d015f0a4ecdd790d38b2e16c674d01.

3.3 (HAL-03) UNNECESSARY LOWERCASE ON ADDRESS - INFORMATIONAL

Description:

The contracts within scope actively lowercase addresses that are later validated. This kind of operation made sense when the supported CosmWasm version didn't implement any fixes for CWA-2022-002. Terra 2 is based on CosmWasm 1.0.0 which implements a validation mechanism to prevent address normalization issues found in the past.

Although not an actual security issue, it is an unnecessary operation for addresses that are later validated through deps.api.addr_validate(). In addition to incurring in extra gas cost, it decreases readability.

Code Location:

Listing 6: contracts/airdrops-registry/src/contract.rs (Line 77)

```
74 TMP_MANAGER_STORE.save(
75     deps.storage,
76     &TmpManagerStore {
77         manager: manager.to_lowercase(),
78     },
79 )?;
```

Listing 7: Resources affected

```
1 contracts/airdrops-registry/src/contract.rs#77, 131, 134
2 contracts/reward/src/contract.rs#32, 89
3 contracts/staking/src/contract.rs#189, 275, 333, 364, 365, 430,
↳ 431, 1196, 1237, 1263
```

Risk Level:

Likelihood - 2
Impact - 1

Recommendation:

When storing addresses for later usage on Terra 2, validation through deps.api.addr_validate() is enough and explicit capitalization checks are not required anymore.

Remediation plan:

SOLVED: The issue was fixed in commit f51ab5b426fdc18aae1192d97f89b6d67d0ad5e7.

3.4 (HAL-04) LACK OF ADDRESS VALIDATION - INFORMATIONAL

Description:

The staking contracts doesn't validate addresses of validators on the `add_validator`, `rebalance_pool` and `remove_validator_from_pool` functions. This does not introduce an actual security threat in the contract as these addresses are later used to query if an actual validator exists, avoiding accounting for incorrect addresses.

However, this issue has been included for informational purposes, as the error raised from validating the address is more descriptive for a user inputting an incorrect address than the current `ContractError::ValidatorNotDiscoverable`. In addition, no lowercasing operating will be required if the address is validated as mentioned in (HAL-03) UNNECESSARY LOWERCASE ON ADDRESS.

Code Location:

Listing 8: Resources affected

```
1 contracts/staking/src/contract.rs#333,364,365,430,431
```

Risk Level:

Likelihood - 2
Impact - 1

Recommendation:

Validate the addresses using `deps.api.addr_validate()` without any lower-casing.

Remediation plan:

NOT APPLICABLE: The proposed recommendation was considered not suitable for Validator addresses as opposed to normal Terra addresses. Therefore, the best possible approach is to maintain the lowercase and check if it is part of the validator pool afterwards.

3.5 (HAL-05) UNNECESSARY LOOPING OVER VECTOR - INFORMATIONAL

Description:

The undelegate_stake function retrieves the amount of funds to be undelegated on current_undelegation_batch_id and then goes over the whole vector of stake_tuples to actually undelegate the required amount. However, instead of checking if there is a positive amount of funds to be delegated before looping over the vector, the check is done inside the for loop, effectively going over the whole vector unnecessarily.

Although not posing an actual security threat, the function will incur in unnecessary extra gas costs when there is nothing to undelegate.

Code Location:

Listing 9: contracts/staking/src/contract.rs (Lines 894,895)

```
892 for index in (0..stake_tuples.len()).rev() {
893     let tuple_val = stake_tuples.get(index).unwrap().clone();
894     if to_undelegate.is_zero() {
895         break;
896     }
897     let val_addr = Addr::unchecked(tuple_val.1);
```

Risk Level:

Likelihood - 2
Impact - 1

Recommendation:

Place the check prior to the for loop to avoid going over it unnecessarily. The following snippet illustrates an example of this.

Listing 10: Proposed fix

```
if !to_undelegate.is_zero() {
    for index in (0..stake_tuples.len()).rev() {
        let tuple_val = stake_tuples.get(index).unwrap().clone();
        // Additional code
        // Additional code
    }
}
```

Remediation plan:

ACKNOWLEDGED: The Stader Labs team acknowledged this finding.

3.6 (HAL-06) HARDCODED DENOM IN QUERY - INFORMATIONAL

Description:

The denom of the native coin managed by the protocol, uluna at the time of the audit, is stored as part of the Configuration to be later accessed as config.vault_denom when required. However, one line of code has does not follow this pattern and hard-codes uluna instead.

Although not posing an actual security threat, following the described pattern aids on code maintainability and readability in case the vault denom is to be changed in the future.

Code Location:

Listing 11: contracts/staking/src/contract.rs (Line 1206)

```
203 Ok(UserInfoResponse {
204     user_info: UserQueryInfo {
205         total_tokens: user_token_balance,
206         total_amount: Coin::new(user_amount.u128(), "uluna".
  ↳ to_string()),
207     },
208 })
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Use config.vault_denom instead, as shown below.

Listing 12: Proposed fix (Line 1206)

```
203 Ok(UserInfoResponse {
204     user_info: UserQueryInfo {
205         total_tokens: user_token_balance,
206         total_amount: Coin::new(user_amount.u128(), config.
 ↳ vault_denom),
207     },
208 })
```

Remediation plan:

SOLVED: The issue was fixed in commit 8121307836d015f0a4ecdd790d38b2e16c674d01.

3.7 (HAL-07) ADDRESS VALIDATION DONE AT ACCEPT-MANAGER STEP - INFORMATIONAL

Description:

The contracts within scope implement the recommended two-step pattern for transferring privileged addresses while validating the address. However, address validation is done on the `accept_manager` function instead of the `set_manager` function.

Although not posing an actual security threat, it will cause inconvenience to manager users. When an incorrect address is submitted, the error introduced in the first step will be actually checked on the second step instead, making them go back to the first step.

Code Location:

Listing 13: contracts/airdrops-registry/src/contract.rs (Lines 98,103)

```
84 pub fn accept_manager(
85     deps: DepsMut,
86     info: MessageInfo,
87     _env: Env,
88 ) -> Result<Response, ContractError> {
89     let mut config = CONFIG.load(deps.storage)?;
90
91     let tmp_manager_store =
92         if let Some(tmp_manager_store) = TMP_MANAGER_STORE.
↳ may_load(deps.storage)? {
93             tmp_manager_store
94         } else {
95             return Err(ContractError::TmpManagerStoreEmpty {});
96         };
97
98     let manager = deps.api.addr_validate(tmp_manager_store.manager
↳ .as_str())?;
99     if info.sender != manager {
```

```
100             return Err(ContractError::Unauthorized {});
101         }
102
103         config.manager = deps.api.addr_validate(tmp_manager_store.
    ↳ manager.as_str())?;
104         TMP_MANAGER_STORE.remove(deps.storage);
```

Listing 14: Resources affected

```
1 contracts/airdrops-registry/src/contract.rs#98,103
2 contracts/reward/src/contract.rs#110,115
3 contracts/staking/src/contract.rs#210
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to perform address validation upon storage as part of the set_manager function, instead of doing it at accept_manager.

Remediation plan:

ACKNOWLEDGED: The Stader Labs team acknowledged this finding.

3.8 (HAL-08) DUPLICATE CODE IMPACTS MAINTAINABILITY - INFORMATIONAL

Description:

The contracts within scope enforce access controls on the execution of multiple privileged messages that are manager only. The logic that enforces this mechanism is duplicated across multiple functions.

Although not posing an actual security threat, it negatively impacts code's readability and maintainability, as it is error-prone. This same issue applies to the operation paused controls of the staking contract.

It should be highlighted that the staking contract follows the proposed logic in most of its privileged functions.

Code Location:

Listing 15: contracts/airdrops-registry/src/contract.rs

```
 99 if info.sender != manager {
100     return Err(ContractError::Unauthorized {});
101 }
```

Listing 16: contracts/staking/src/contract.rs

```
522 let operation_controls = OPERATION_CONTROLS.load(deps.storage)?;
523
524 if operation_controls.deposit_paused {
525     return Err(ContractError::OperationPaused("deposit".to_string
 ↳ ()));
526 }
```

Listing 17: Resources affected

```
contracts/airdrops-registry/src/contract.rs#70,99,120
contracts/reward/src/contract.rs#82,111,135,173
contracts/staking/src/contract.rs#524, 636, 671, 683, 776, 832,
↳ 845, 934, 1016, 1106
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Refactor the access controls and pausing controls to a function of their own and use them across all the relevant functionalities.

Remediation plan:

ACKNOWLEDGED: The Stader Labs team acknowledged this finding.

3.9 (HAL-09) OVERFLOW CHECKS NOT SET FOR PROFILE RELEASE - INFORMATIONAL

Description:

Although the overflow-checks parameter is set to **true** in profile.release and is implicitly applied to all contracts and packages in the workspace, it is not explicitly enabled in **Cargo.toml** for each individual contract and package, which could have unexpected consequences if the project is refactored.

Code Location:

Listing 18: Resources affected

```
contracts/airdrops-registry/Cargo.toml
contracts/reward/Cargo.toml
contracts/staking/Cargo.toml
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended that you explicitly enable overflow checks on each individual contract and package. That measure helps when the project is refactored to avoid unintended consequences.

Remediation plan:

ACKNOWLEDGED: The Stader Labs team acknowledged this finding.

3.10 (HAL-10) 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 19: contracts/staking/src/contract.rs (Line 69)

```
68 reward_contract: deps.api.addr_validate(msg.reward_contract.as_str
↳ ())?,
69 cw20_token_contract: Addr::unchecked("0"),
70
71 protocol_fee_contract: deps.api.addr_validate(msg.
↳ protocol_fee_contract.as_str())?,
```

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: The Stader Labs team acknowledged this finding.

3.11 (HAL-11) UNMANTAINED DEPENDENCY - INFORMATIONAL

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. 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

Code Location:

Listing 20: Dependency tree

```
bigint 4.4.3
 cosmwasm-bignumber 2.2.0
     stader-utils 0.1.0
         staking 0.1.0
         reward 0.1.0
            staking 0.1.0
         airdrops-registry 0.1.0
             staking 0.1.0
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Beware of using dependencies and packages that are no longer supported by developers or have publicly known security flaws, even when they are not currently exploitable.

Remediation plan:

SOLVED: The issue was fixed in commit 6431dd2138e296e3d4358a7774bd826289874576.

THANK YOU FOR CHOOSING

// HALBORN