Astroport.fi Periphery Contracts

CosmWasm Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **November 22nd, 2021 - December 6th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	10/22/2021	Luis Quispe Gonzales
0.2	Document Updates	12/02/2021	Jose Ramirez
0.3	Document Updates	12/03/2021	Michal Bazyli
0.4	Document Updates	12/06/2021	Luis Quispe Gonzales
0.5	Draft Review	12/13/2021	Gabi Urrutia
1.0	Remediation Plan	12/16/2021	Luis Quispe Gonzales
1.1	Remediation Plan Review	12/17/2021	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
Luis Quispe Gonzales	Halborn	Luis.QuispeGonzales@halborn.com

EXECUTIVE OVERVIEW

1.1 AUDIT SUMMARY

Astroport.fi engaged Halborn to conduct a security assessment on CosmWasm smart contracts beginning on November 22nd, 2021 and ending December 6th, 2021.

The security engineers involved on the audit 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 achieve the following:

- Ensure that smart contract functions work 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 partially addressed by Astroport team. The main ones are the following:

- Restrict claiming in lockdrop contract during deposit and withdrawal windows.
- Split owner address transfer functionality to allow transfer to be completed by recipient.
- Validate that initial timestamps in airdrop and auction contracts are greater than current timestamp.
- Harden lockdrop contract to restrict the initialization of repeated pools.
- Restrict changes to incentives share in lockdrop contract.

External threats, such as financial related attacks, oracle attacks, and inter-contract functions and calls should be validated for expected logic and state.

1.2 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. It’s quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that was 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

1.3 SCOPE

1. CosmWasm Smart Contracts
    (a) Repository: astroport-periphery
    (b) Commit ID: d13df3bdbd2bce99862c63aa7564f8a2e6ebc2b7
    (c) Contracts in scope:
        i. contracts/airdrop
        ii. contracts/auction
        iii. contracts/lockdrop

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

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	1	3	6

LIKELIHOOD

IMPACT

🟨	🟨	🟥 (HAL-01)	🟥	🟪
🟩	🟨 (HAL-02)	🟨	🟥	🟥
🟩 (HAL-04) (HAL-05)	🟩 (HAL-03)	🟨	🟨	🟥
⬜ (HAL-06) (HAL-07) (HAL-08) (HAL-09)	🟩	🟩	🟨	🟨
⬜ (HAL-10) (HAL-11)	⬜	🟩	🟩	🟨

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) POSSIBILITY TO WITHDRAW TERRASWAP LP TOKENS AFTER CLAIMING REWARDS	High	SOLVED - 12/13/2021
(HAL-02) PRIVILEGED ADDRESS CAN BE TRANSFERRED WITHOUT CONFIRMATION	Medium	RISK ACCEPTED
(HAL-03) INITIAL TIMESTAMPS DO NOT HAVE A MINIMUM THRESHOLD	Low	RISK ACCEPTED
(HAL-04) OWNER CAN INITIALIZE REPEATED POOLS IN LOCKDROP CONTRACT	Low	RISK ACCEPTED
(HAL-05) INCENTIVES SHARE CAN BE CHANGED UNRESTRICTEDLY	Low	SOLVED - 12/13/2021
(HAL-06) MAXIMUM THRESHOLD FOR SLIPPAGE IS NOT ENFORCED WHEN ADDING LIQUIDITY	Informational	ACKNOWLEDGED
(HAL-07) AUCTION ADDRESS CANNOT BE CHANGED IF SET WITH AN ERRONEOUS VALUE	Informational	ACKNOWLEDGED
(HAL-08) ASSETS WITHDRAWN FROM TERRASWAP SHOULD MATCH ASSETS IN ASTROPORT POOL	Informational	ACKNOWLEDGED
(HAL-09) POTENTIAL INCONSISTENCY IN TOTAL AIRDROP SIZE	Informational	SOLVED - 12/06/2021
(HAL-10) POTENTIAL DIVISION BY ZERO	Informational	ACKNOWLEDGED
(HAL-11) INACCURATE CONDITIONAL WHEN CLAIMING REWARDS IN AUCTION CONTRACT	Informational	ACKNOWLEDGED

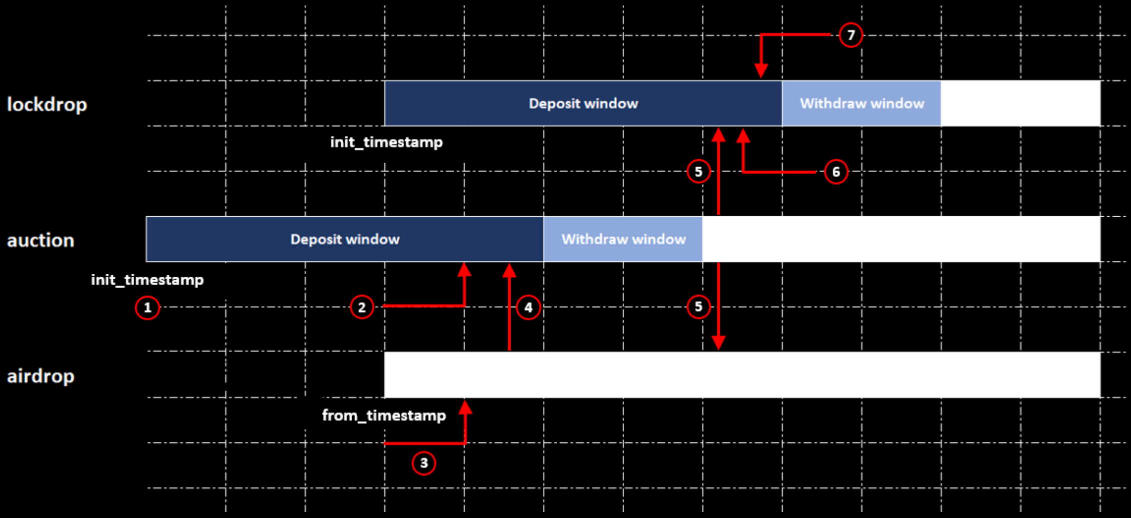
FINDINGS & TECH DETAILS

3.1 (HAL-01) POSSIBILITY TO WITHDRAW TERRASWAP LP TOKENS AFTER CLAIMING REWARDS - HIGH

Description:

handle_claim_rewards_and_unlock_for_lockup function in **contracts/lockdrop/src/contract.rs** allow users to claim their rewards if state.are_claims_allowed is set to **true**. However, if the claiming is enabled too early (i.e.: before **deposit** and **withdrawal** windows), users will even be able to withdraw their Terraswap LP tokens after claiming rewards.

Attack scenario:



1. Owner deploys **lockdrop**, **auction** and **airdrop** contracts. In particular, init_timestamp for **auction** contract is incorrectly set.
2. As part of usual protocol working, users deposit UST to **auction** contract during its deposit window.
3. As part of usual protocol working, users claim rewards (ASTRO tokens) in **airdrop** contract and their claimed_amount increase.

4. As part of usual protocol working, users delegate ASTRO tokens to **auction** contract.
5. Once **deposit** and **withdrawal** windows finish for **auction** contract, owner can initialize pool, which enables claims for **lockdrop** and **airdrop** contracts.
6. Users increase lockup position size in **lockdrop** contract and claim rewards (ASTRO tokens).
7. Finally, users can withdraw their Terraswap LP tokens from **lockdrop** contract and keep their rewards.

A proof of concept video showing how to exploit this security issue is included in the report.

It is important to mention that this situation can happen under the following circumstances:

- init_timestamp in **auction** contract is incorrectly set, see HAL-03 for more details.
- init_timestamp in **lockdrop** contract is incorrectly set.
- **Deposit** / **withdrawal** windows are incorrectly set in **lockdrop** or **auction** contracts.

Code Location:

The function only verifies if state.are_claims_allowed is set to **true** before allowing users claim their rewards.

Listing 1: contracts/lockdrop/src/contract.rs (Lines 892)

```
pub fn handle_claim_rewards_and_unlock_for_lockup(
    mut deps: DepsMut,
    env: Env,
    info: MessageInfo,
    terraswap_lp_token: String,
    duration: u64,
    withdraw_lp_stake: bool,
) -> StdResult<Response> {
    let state = STATE.load(deps.storage)?;

    if !state.are_claims_allowed {
        return Err(StdError::generic_err("Reward claim not allowed
            "));
    }

    let config = CONFIG.load(deps.storage)?;
    let user_address = info.sender;

    let terraswap_lp_token = deps.api.addr_validate(&
       terraswap_lp_token)?;
```

Risk Level:

Likelihood - 3
Impact - 5

Recommendations:

Add a restriction in handle_claim_rewards_and_unlock_for_lockup function to throw an error message if claiming in **lockdrop** contract is made before **deposit** and **withdrawal** windows.

Remediation plan:

SOLVED: The issue was fixed in commit f20d8471b8ef5325d56a556f3a1953185ddc4145.

3.2 (HAL-02) PRIVILEGED ADDRESS CAN BE TRANSFERRED WITHOUT CONFIRMATION - MEDIUM

Description:

An incorrect use of the handle_update_config function in contracts can set owner to an invalid address and inadvertently lose control of the contracts, which cannot be undone in any way. Currently, the owner of the contracts can change **owner address** using the aforementioned function in a single transaction and without confirmation from the new address.

The affected smart contracts are the following:

- airdrop
- auction
- lockdrop

Code Location:

Listing 2: contracts/airdrop/src/contract.rs (Lines 143)

```
if let Some(owner) = owner {
    config.owner = deps.api.addr_validate(&owner)?;
    attributes.push(attr("new_owner", owner.as_str()))
}
```

Listing 3: contracts/auction/src/contract.rs (Lines 178)

```
// UPDATE :: ADDRESSES IF PROVIDED
if let Some(owner) = new_config.owner {
    config.owner = deps.api.addr_validate(&owner)?;
    attributes.push(attr("owner", config.owner.to_string()));
}
```

Listing 4: contracts/lockdrop/src/contract.rs (Lines 249)

```
if let Some(owner) = new_config.owner {
    config.owner = deps.api.addr_validate(&owner)?;
    attributes.push(attr("new_owner", owner.as_str()))
};
```

Risk Level:

Likelihood - 2
Impact - 4

Recommendations:

It is recommended to split **owner transfer** functionality into set_owner and accept_ownership functions. The latter function allows the transfer to be completed by the recipient.

Remediation plan::

RISK ACCEPTED: The Astroport team accepted the risk for this finding.

3.3 (HAL-03) INITIAL TIMESTAMPS DO NOT HAVE A MINIMUM THRESHOLD - LOW

Description:

instantiate function in contracts does not verify that initial timestamps (from_timestamp, init_timestamp) are greater than current timestamp. As a consequence, if contracts are deployed with inaccurate initial timestamps could generate unexpected situations, e.g.: possibility to withdraw Terraswap LP tokens after claiming rewards, see HAL-01 for more details.

The affected smart contracts are the following:

- airdrop
- auction

Code Location:

Listing 5: contracts/airdrop/src/contract.rs (Lines 50)

```
46 let config = Config {
47     owner,
48     astro_token_address: deps.api.addr_validate(&msg.
           astro_token_address)?,
49     merkle_roots: msg.merkle_roots.unwrap_or_default(),
50     from_timestamp,
51     to_timestamp: msg.to_timestamp,
52     auction_contract_address: None,
53     are_claims_enabled: false,
54 };
```

Listing 6: contracts/auction/src/contract.rs (Lines 52)

```
39 let config = Config {
40     owner: msg
41         .owner
42         .map(|v| deps.api.addr_validate(&v))
43         .transpose()?
```

```
            .unwrap_or(info.sender),
        astro_token_address: deps.api.addr_validate(&msg.
            astro_token_address)?,
        airdrop_contract_address: deps.api.addr_validate(&msg.
            airdrop_contract_address)?,
        lockdrop_contract_address: deps.api.addr_validate(&msg.
            lockdrop_contract_address)?,
        pool_info: None,
        generator_contract: None,
        astro_incentive_amount: None,
        lp_tokens_vesting_duration: msg.lp_tokens_vesting_duration,
        init_timestamp: msg.init_timestamp,
        deposit_window: msg.deposit_window,
        withdrawal_window: msg.withdrawal_window,
};
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended to update the logic of instantiate function in contracts mentioned above to validate that initial timestamps (from_timestamp, init_timestamp) are greater than current timestamp.

Remediation plan::

RISK ACCEPTED: The Astroport team accepted the risk for this finding.

3.4 (HAL-04) OWNER CAN INITIALIZE REPEATED POOLS IN LOCKDROP CONTRACT - LOW

Description:

handle_initialize_pool function in **contracts/lockdrop/src/contract.rs** allows the possibility that owner mistakenly initializes repeated pools in **lockdrop** contract. This issue happens because **may_load** function will consider that two terraswap_lp_token addresses are different if they differ just in their upper / lower cases.

The situation described above could generates unexpected situations, e.g.: a user could increase size of an unintended **lockup position**.

Code Location:

Listing 7: contracts/lockdrop/src/contract.rs (Lines 344,345)

```
341 let terraswap_lp_token = deps.api.addr_validate(&
        terraswap_lp_token)?;
342
343 // CHECK ::: Is LP Token Pool already initialized
344  if ASSET_POOLS
345      .may_load(deps.storage, &terraswap_lp_token)?
346      .is_some()
347  {
348      return Err(StdError::generic_err("Already supported"));
349  }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendations:

Update the logic of handle_initialize_pool to turn terraswap_lp_token address into lowercase before calling **may_load** function.

Remediation plan::

RISK ACCEPTED: The Astroport team accepted the risk for this finding.

3.5 (HAL-05) INCENTIVES SHARE CAN BE CHANGED UNRESTRICTEDLY - LOW

Description:

handle_update_pool function in **contracts/lockdrop/src/contract.rs** allows owner to change incentives share (incentives_share) unrestrictedly in **lockdrop** contract. If this value is changed to a lower one after init_timestamp, it could reduce rewards for users that have already deposited Terraswap LP tokens.

Code Location:

Listing 8: contracts/lockdrop/src/contract.rs (Lines 402,418)

```
if env.block.time.seconds() >= config.init_timestamp + config.
    deposit_window {
    return Err(StdError::generic_err(
        "Pools cannot be updated post deposit window closure",
    ));
}

let terraswap_lp_token = deps.api.addr_validate(&
    terraswap_lp_token)?;

// CHECK ::: Is LP Token Pool initialized
let mut pool_info = ASSET_POOLS.load(deps.storage, &
    terraswap_lp_token)?;

// update total incentives
state.total_incentives_share =
    state.total_incentives_share - pool_info.incentives_share +
        incentives_share;

// Update Pool Incentives
pool_info.incentives_share = incentives_share;

ASSET_POOLS.save(deps.storage, &terraswap_lp_token, &pool_info)?;
STATE.save(deps.storage, &state)?;
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendations:

It is recommended to not allow changes to incentives_share after init_timestamp. Otherwise, do not allow lower values if someone has already deposited.

Remediation plan:

SOLVED: The issue was fixed in commit f20d8471b8ef5325d56a556f3a1953185ddc4145.

3.6 (HAL-06) MAXIMUM THRESHOLD FOR SLIPPAGE IS NOT ENFORCED WHEN ADDING LIQUIDITY - INFORMATIONAL

Description:

When owner calls handle_init_pool function in **contracts/auction/src/contract.rs** to add liquidity and do not specify slippage tolerance in the operation, Astroport AMM protocol does not enforce a default maximum threshold, which could affect the amount of tokens received in return.

However, this is an unlikely scenario because the operation is executed before users other than owner are able to claim their ASTRO tokens to provide liquidity to ASTRO-UST pool.

Code Location:

The slippage argument was directly provided to the build_provide_liquidity_to_lp_pool_msg function without any previous validation.

Listing 9: contracts/auction/src/contract.rs (Lines 478)

```
472 msgs.push(build_provide_liquidity_to_lp_pool_msg(
473     deps.as_ref(),
474     config.astro_token_address,
475     astro_ust_pool_address,
476     ust_coin.amount,
477     state.total_astro_delegated,
478     slippage,
479 )?);
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendation:

It is recommended to enforce the use of a **default maximum threshold** for slippage in handle_init_pool function. As a reference, **max slippage** for Uniswap Pool is 50%.

Remediation plan::

ACKNOWLEDGED: The Astroport team acknowledged this finding.

3.7 (HAL-07) AUCTION ADDRESS CANNOT BE CHANGED IF SET WITH AN ERRONEOUS VALUE - INFORMATIONAL

Description:

handle_update_config function in contracts does not allow owner to change **auction address** if set with an erroneous value. As a consequence, users won't be able to delegate their ASTRO tokens and operations will always throw error messages.

The affected smart contracts are the following:

- airdrop
- lockdrop

Code Location:

Listing 10: contracts/airdrop/src/contract.rs

```
if let Some(auction_contract_address) = auction_contract_address {
    match config.auction_contract_address {
        Some(_) => {
            return Err(StdError::generic_err("Auction contract
                already set."));
        }
        None => {
            config.auction_contract_address =
                Some(deps.api.addr_validate(&
                    auction_contract_address)?);
            attributes.push(attr("auction_contract",
                auction_contract_address))
        }
    }
}
```

Listing 11: contracts/lockdrop/src/contract.rs

```
if let Some(auction) = new_config.auction_contract_address {
    match config.auction_contract {
        Some(_) => {
            return Err(StdError::generic_err("Auction contract
                already set."));
        }
        None => {
            config.auction_contract = Some(deps.api.addr_validate
                (&auction)?);
            attributes.push(attr("auction_contract", auction))
        }
    }
};
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendations:

It is recommended to allow changes to **auction** address in contracts before initial timestamps (from_timestamp, init_timestamp).

Remediation plan::

ACKNOWLEDGED: The Astroport team acknowledged this finding.

3.8 (HAL-08) ASSETS WITHDRAWN FROM TERRASWAP SHOULD MATCH ASSETS IN ASTROPORT POOL - INFORMATIONAL

Description:

handle_migrate_liquidity function in **contracts/lockdrop/src/contract.rs** does not ensure that assets to withdraw from Terraswap match assets in Astroport pool. This security issue is considered as informational because the validation is done in Astroport AMM protocol and operations will always throw error messages.

However, from a security in depth perspective, validation should not rely only in external components (AMM protocol in this case) and should be enforced in **lockdrop** contract, too.

Code Location:

Listing 12: contracts/lockdrop/src/contract.rs (Lines 541,543)

```
540 // COSMOS MSG :: CALLBACK AFTER LIQUIDITY WITHDRAWAL
541 let update_state_msg = CallbackMsg::
      WithdrawLiquidityFromTerraswapCallback {
542      terraswap_lp_token: terraswap_lp_token.clone(),
543      astroport_pool: astroport_pool.clone(),
544      prev_assets: assets.try_into().unwrap(),
545 }
546 .to_cosmos_msg(&env)?;
547 cosmos_msgs.push(update_state_msg);
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendations:

Update the logic of handle_migrate_liquidity function to ensure that assets to withdraw from Terraswap match assets in Astroport pool.

Remediation plan::

ACKNOWLEDGED: The Astroport team acknowledged this finding.

3.9 (HAL-09) POTENTIAL INCONSISTENCY IN TOTAL AIRDROP SIZE - INFORMATIONAL

Description:

instantiate function in **contracts/airdrop/src/contract.rs** does not validate that total_airdrop_size has the same value than amount of ASTRO tokens transferred to **aidrop** contract, which could create an inconsistency in case an inaccurate amount of ASTRO tokens are transferred.

Code Location:

Listing 13: contracts/airdrop/src/contract.rs (Lines 57)

```
let state = State {
    total_airdrop_size: msg.total_airdrop_size,
    total_delegated_amount: Uint128::zero(),
    unclaimed_tokens: msg.total_airdrop_size,
};
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendations:

Create a separate function in contract to update total_airdrop_size when called by ASTRO token contract through a Cw20ReceiveMsg message.

Remediation plan::

SOLVED: The issue was fixed in commit c61cc86c9ce86b2175ff70d1070ba9ae17013a53.

3.10 (HAL-10) POTENTIAL DIVISION BY ZERO - INFORMATIONAL

Description:

Several functions in contracts do not check if denominators in divisions are different than zero. This situation could cause multiple instances of **division by zero** on different parts of the code, potentially causing the Rust code to panic. The affected smart contracts are the following:

- auction
- lockdrop

Code Location:

Listing 14: Resources affected

```
auction: contract.rs (L729,940,979,1065)
lockdrop: contract.rs (L1075,1091,1560,1572,1666,1680)
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to handle situations where denominator is zero in divisions and return appropriate values or throw descriptive error messages.

Remediation plan::

ACKNOWLEDGED: The Astroport team acknowledged this finding.

3.11 (HAL-11) INACCURATE CONDITIONAL WHEN CLAIMING REWARDS IN AUCTION CONTRACT - INFORMATIONAL

Description:

Conditional expression in handle_claim_rewards_and_withdraw_lp_shares function from **contracts/auction/src/contract.rs** verifies if the value of pending_rewards.pending_on_proxy (related to **proxy_reward_token**) is different to zero.

However, in the logic of the contract, **proxy_reward_token** is never used, thus the conditional statement is inaccurate and could generates unexpected situations, e.g.: claiming non-existent rewards and incurring in additional gas fees consumption.

Code Location:

Listing 15: contracts/auction/src/contract.rs (Lines 653,654)

```
652 if !pending_rewards.pending.is_zero()
653     || (pending_rewards.pending_on_proxy.is_some()
654         && !pending_rewards.pending_on_proxy.unwrap().is_zero())
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendations:

Remove from conditional expression shown above the statement related to pending_rewards.pending_on_proxy.

Remediation plan::

ACKNOWLEDGED: The Astroport team 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

THANK YOU FOR CHOOSING

HALBORN