Astroport.fi Astral Assembly

CosmWasm Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **March 7th, 2022 - March 18th, 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 6
RISK METHODOLOGY 6
1.4 SCOPE 8
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW 9
3 FINDINGS & TECH DETAILS 10
3.1 (HAL-01) LACK OF PROPOSAL SANITIZATION COULD HARM USERS - MEDIUM 12
Description 12
Code Location 12
Risk Level 13
Recommendation 13
Remediation plan 13
3.2 (HAL-02) INCONSISTENT XASTRO BLOCK HEIGHT SELECTION - MEDIUM 15
Description 15
Code Location 16
Risk Level 16
Recommendation 17
Remediation plan 17
3.3 (HAL-03) NO MINIMUM THRESHOLD FOR EFFECTIVE DELAY PERIOD - LOW 18

Description 18

Code Location 18

Risk Level 19

Recommendation 19

Remediation plan 19

3.4 (HAL-04) UNCHECKED ARITHMETIC - INFORMATIONAL 20

Description 20

Code Location 20

Risk Level 20

Recommendation 20

Remediation plan 21

4 AUTOMATED TESTING 22

4.1 AUTOMATED ANALYSIS 23

Description 23

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	03/07/2022	Jakub Heba
0.2	Document Update	03/08/2022	Jose C. Ramirez
0.3	Draft Version	03/17/2022	Jose C. Ramirez
0.4	Draft Review	03/17/2022	Gabi Urrutia
1.0	Remediation Plan	04/01/2022	Jose C. Ramirez
1.1	Remediation Plan Review	04/01/2022	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
Jakub Heba	Halborn	Jakub.Heba@halborn.com
Jose Ramirez	Halborn	Jose.Ramirez@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Astroport.fi engaged Halborn to conduct a security audit on their smart contracts beginning on March 7th, 2022 and ending on March 18th, 2022. The security assessment was scoped to the smart contracts provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided two weeks for the engagement and assigned two full-time security engineers to audit the security of the smart contract. The security engineers are a 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 multiple risks, which has been mostly addressed by Astroport.fi. The main ones are the following:

- Sanitize proposal information that will be displayed on the web frontend.
- Select a consistent block height when requesting token information to avoid potential quorum bypasses.
- Enforce thresholds on proposal delay periods.

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: astroport-governance

1. CosmWasm Smart Contracts
   (a) Commit ID: 7918a7c9d1b93a224e91d955da6e360456df8f81
   (b) Contract within scope:
      i. assembly

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

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	2	1	1

LIKELIHOOD

IMPACT

🟨	🟨	🟥	🟥	🟪
🟩 (HAL-03)	🟨	🟨	🟥	🟥
🟩	🟩	🟨 (HAL-01) (HAL-02)	🟨	🟥
⬜	🟩	🟩	🟨	🟨
⬜ (HAL-04)	⬜	🟩	🟩	🟨

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) LACK OF PROPOSAL SANITIZATION COULD HARM USERS	Medium	PARTIALLY SOLVED
(HAL-02) INCONSISTENT XASTRO BLOCK HEIGHT SELECTION	Medium	NOT APPLICABLE
(HAL-03) NO MINIMUM THRESHOLD FOR EFFECTIVE DELAY PERIOD	Low	SOLVED - 03/25/2022
(HAL-04) UNCHECKED ARITHMETIC	Informational	SOLVED - 03/25/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) LACK OF PROPOSAL SANITIZATION COULD HARM USERS - MEDIUM

Description:

When a new proposal is submitted via the submit_proposal function in the **assembly** contract, the submitter can introduce a malicious external URL that tricks users into being redirected to a phishing DApp that could steal their funds.

It is worth noting that URLs will appear in the official Astroport frontend, so legitimate users will not be aware if those URLs are malicious or not. In addition, unsanitized title and description fields could lead to other web application exploits such as Cross-site scripting.

Code Location:

Listing 1: contracts/assembly/src/contract.rs (Lines 200,206,211,216,224,227)

```
199     // Validate title
200     if title.len() < MIN_TITLE_LENGTH {
201         return Err(ContractError::InvalidProposal(
202             "Title too short".to_string(),
203         ));
204     }
205
206     if title.len() > MAX_TITLE_LENGTH {
207         return Err(ContractError::InvalidProposal("Title too long"
  ↳ .to_string()));
208     }
209
210     // Validate the description
211     if description.len() < MIN_DESC_LENGTH {
212         return Err(ContractError::InvalidProposal(
213             "Description too short".to_string(),
        ));
    }
    if description.len() > MAX_DESC_LENGTH {
        return Err(ContractError::InvalidProposal(
            "Description too long".to_string(),
        ));
    }

    // Validate Link
    if let Some(link) = &link {
        if link.len() < MIN_LINK_LENGTH {
            return Err(ContractError::InvalidProposal("Link too short".to_string()));
        }
        if link.len() > MAX_LINK_LENGTH {
            return Err(ContractError::InvalidProposal("Link too long".to_string()));
        }
    }
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

If possible, restrict that URLs are submitted to a whitelist of domains. As this may not be feasible due to the nature of the feature, it is recommended to limit the link, title and description fields to alphanumeric characters plus a subset of symbols, or perform XSS validation using Rust libraries such as libinjection.

Reference: LibInjection's XSS validation function

Remediation plan:

PARTIALLY SOLVED: URL and character validation measures were implemented in commit 6910dbf80aba668b4788cc7e73efcf16108cef33. However, Astroport.

fi stated that the content of the proposal will be carefully HTML-encoded in a future release of the front-end to mitigate any potential XSS vectors not covered by the smart contract sanitization steps. As the front-end is out of scope for this audit scope and therefore the remediation cannot be reviewed, this issue has been marked as "Partially solved".

3.2 (HAL-02) INCONSISTENT XASTRO BLOCK HEIGHT SELECTION - MEDIUM

Description:

The **assembly** contract uses different block heights when casting a vote and when calculating the total voting power available for a proposal. This situation could allow proposals to bypass the quorum required for approval.

The proposal_quorum is defined as the ratio of total votes to total voting power. However, the calculation of each vote cast and thus total_votes queries **XAstro token** in proposal.start_block but total_voting_power does so in proposal.start_block - 1, effectively retrieving different states of the XAstro supply. On the other hand, the **VxAstro token** consistently uses proposal.start_time - 1 in both cases.

To illustrate this issue, take the following example attack scenario which, for simplicity, only considers the XAstro token:

1. Suppose at block height 100 the XAstro's total supply is 1000, proposal_required_quorum set to 10% and proposal_required_threshold set to 20%.
2. In the next block (101), more XAstro tokens are minted, taking the total supply to 1500. At this point, a proposal is submitted.
3. By the end of the voting period, the active proposal will have received 100 votes, 21 of which will be "for" votes.
4. Finally, end_proposal is called on the aforementioned proposal. proposal_quorum will be 100 / 1000 since proposal.start_block - 1 is requested for the denominator instead of 100 / 1500.

This will unfairly mark the proposal as "Passed", since the quorum check requires a smaller than effective total_voting_power to cast votes.

Code Location:

Listing 2: contracts/assembly/src/contract.rs (Line 700)

```
689 pub fn calc_voting_power(
690     deps: &DepsMut,
691     sender: String,
692     proposal: &Proposal,
693 ) -> StdResult<Uint128> {
694     let config = CONFIG.load(deps.storage)?;
695
696     let xastro_amount: BalanceResponse = deps.querier.
  ↳ query_wasm_smart(
697         config.xastro_token_addr,
698         &XAstroTokenQueryMsg::BalanceAt {
699             address: sender.clone(),
700             block: proposal.start_block,
701         },
```

Listing 3: contracts/assembly/src/contract.rs (Line 746)

```
739 pub fn calc_total_voting_power_at(deps: &DepsMut, proposal: &
  ↳ Proposal) -> StdResult<Uint128> {
740     let config = CONFIG.load(deps.storage)?;
741
742     // Total xASTRO supply at a specified block
743     let mut total: Uint128 = deps.querier.query_wasm_smart(
744         config.xastro_token_addr,
745         &XAstroTokenQueryMsg::TotalSupplyAt {
746             block: proposal.start_block - 1,
747         },
748     )?;
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

A consistent block height and time should be used when retrieving information about total supplies and total balances that are expected to reflect the same state of a token.

Remediation plan:

NOT APPLICABLE: The underlying logic in storing SnapshotMap effectively reflected the user's voting power (calculated at calc_voting_power) in proposal.start_block - 1. Causing the block height to be selected consistently in all the relevant places.

3.3 (HAL-03) NO MINIMUM THRESHOLD FOR EFFECTIVE DELAY PERIOD - LOW

Description:

Timelocks are defined in the **assembly** contract to allow users of the protocol to react in time if a change made is bad faith or is not in the best interest of the protocol and its users.

The instantiate and update_config functions of the **assembly** contract do not restrict timelocks (execution_delay_period and proposal_expiration_period) from being greater than or equal to a **minimum threshold**. Therefore, malicious changes proposed through voting could even be executed immediately if execution_delay_period is not set appropriately.

Code Location:

Listing 4: contracts/assembly/src/contract.rs (Lines 67,68)

```
62 let config = Config {
63     xastro_token_addr: addr_validate_to_lower(deps.api, &msg.
↳ xastro_token_addr)?,
64     vxastro_token_addr: addr_validate_to_lower(deps.api, &msg.
↳ vxastro_token_addr)?,
65     builder_unlock_addr: addr_validate_to_lower(deps.api, &msg.
↳ builder_unlock_addr)?,
66     proposal_voting_period: msg.proposal_voting_period,
67     proposal_effective_delay: msg.proposal_effective_delay,
68     proposal_expiration_period: msg.proposal_expiration_period,
69     proposal_required_deposit: msg.proposal_required_deposit,
70     proposal_required_quorum: Decimal::from_str(&msg.
↳ proposal_required_quorum)?,
71     proposal_required_threshold: Decimal::from_str(&msg.
↳ proposal_required_threshold)?,
72 };
```

Listing 5: contracts/assembly/src/contract.rs (Lines 557,561)

```
556 if let Some(proposal_effective_delay) = updated_config.proposal_effective_delay {
557     config.proposal_effective_delay = proposal_effective_delay;
558 }
559
560 if let Some(proposal_expiration_period) = updated_config.proposal_expiration_period {
561     config.proposal_expiration_period = proposal_expiration_period;
562 }
```

Risk Level:

Likelihood - 1
Impact - 4

Recommendation:

Add a validation routine inside the instantiate and update_config functions to ensure that timelock (execution_delay_period and proposal_expiration_period) is greater than or equal to a **minimum threshold** that allows users of Astroport will act promptly against any issue the protocol could have when changes are made. The following are some examples of timelocks used in other protocols:

- Uniswap: 48-hours timelock
- Compound: 48-hours timelock
- Aave: 24-hours timelock (Short time lock)

Remediation plan:

SOLVED: The issue was fixed in commit 6dcfefe59514a85495a4e34a2ec50ea41f0d10d2.

3.4 (HAL-04) UNCHECKED ARITHMETIC - INFORMATIONAL

Description:

In computer programming, an overflow occurs when an arithmetic operation attempts to create a numeric value that is outside the range that can be represented by a given number of bits -- either greater than the maximum value or less than the minimum representable value.

This issue has been raised as informational only, since it was not possible to define a clear exploitation scenario for the affected case.

Code Location:

Listing 6: Affected resources

```
contracts/assembly/src/contract.rs:715:            .checked_add(
↳ locked_amount.params.amount - locked_amount.status.astro_withdrawn
↳ )?;
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

In “release” mode Rust does not panic on overflows and overflown values just “wrap” without any explicit feedback to the user. It is recommended then to use vetted safe math libraries for arithmetic operations consistently throughout the smart contract system. Consider replacing the addition operator with Rust’s checked_add method, the subtraction operator with Rust’s checked_subs method, and so on.

Remediation plan:

SOLVED: The issue was fixed in commit 6dcfefe59514a85495a4e34a2ec50ea41f0d10d2.

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.

No unmaintained or yanked library was found.

THANK YOU FOR CHOOSING

// HALBORN