// HALBORN

Astroport.fi Voting Escrow

CosmWasm Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **March 1st, 2022 - March 8th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	03/01/2022	Michal Bazyli
0.2	Document Updates	03/08/2021	Michal Bazyli
0.3	Document Update	03/08/2022	Michal Bazyli
0.4	Draft Review	03/09/2022	Gabi Urrutia
1.0	Remediation Plan	03/18/2021	Michal Bazyli
1.1	Remediation Plan Review	03/21/2021	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

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Astroport.fi engaged Halborn to conduct a security audit on their smart contracts beginning on March 1st, 2022 and ending on March 8th, 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 one week for the engagement and assigned two full-time security engineers 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 impacts of the risks, which were partially addressed by Astroport team. The main ones are:

- Enforce appropriate checks in each contract and package
- Enforce usage of appropriate arithmetical methods

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.

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

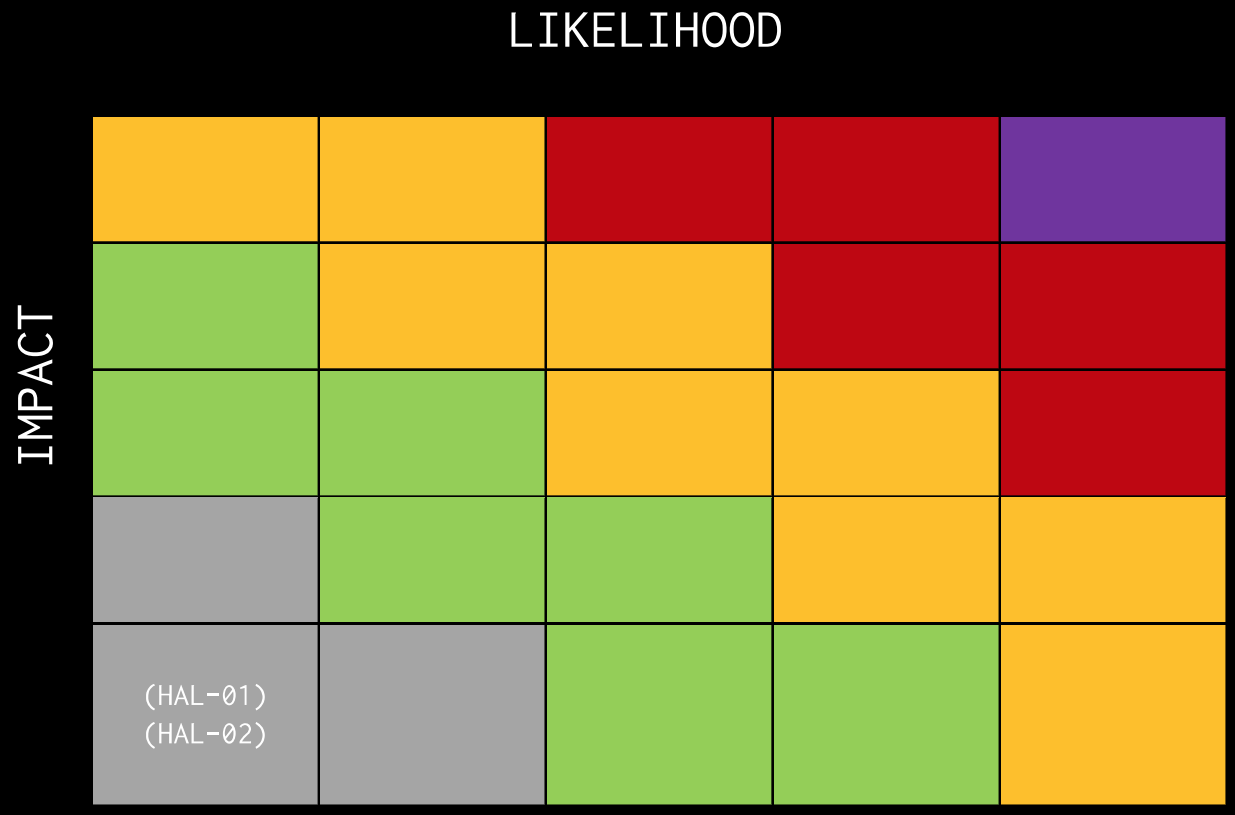
1.4 SCOPE

1. CosmWasm Smart Contracts
   (a) Repository: astroport-governance
   (b) Commit ID: cd374d453c40f18a99f78596d54bfda4036dd5e2
   (c) Contract in scope:
      i. voting_escrow

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

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	0	2



EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) OVERFLOW CHECKS NOT SET FOR PROFILE RELEASE	Informational	ACKNOWLEDGED
(HAL-02) MULTIPLE INSTANCES OF UNCHECKED ARITHMETIC	Informational	SOLVED - 03/18/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) 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 the **Cargo.toml** file for each individual contract and package, which could have unintended consequences if the project is refactored.

Code Location:

Listing 1: Resources affected

```
1 contracts/voting_escrow/Cargo.toml
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

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

Remediation plan:

ACKNOWLEDGED: The Astroport team acknowledged this finding.

3.2 (HAL-02) MULTIPLE INSTANCES OF UNCHECKED ARITHMETIC - INFORMATIONAL

Description:

While many instances of checked arithmetic were observed, several calculations missed these checks. Additional verification performed when using the checked functions ensures that under/overflow states are caught and handled appropriately.

While these instances were not found to be directly exploitable, they should be reviewed to ensure a defence-in-depth approach is achieved.

Code Location:

Listing 2: Resources affected

```
1 voting_escrow contract.rs (#L309, 333)
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider using the `checked_add`, `checked_sub` or `checked_mul` methods instead of addition, subtraction, and multiplication operators respectively, in all instances to handle overflows gracefully.

Remediation plan:

SOLVED: The issue was fixed in commit 4ce58288897668590363ea908260a23005068cbe.

THANK YOU FOR CHOOSING

// HALBORN