// HALBORN

Mars Protocol - Periphery Vesting

CosmWasm Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **September 12th, 2022 - October 28th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	09/12/2022	Connor Taylor
0.2	Draft Version	09/16/2022	Connor Taylor
0.3	Draft Review	09/16/2022	Gabi Urrutia
1.0	Remediation Plan	09/21/2022	Connor Taylor
1.1	Remediation Plan Review	09/22/2022	Gabi Urrutia
1.2	Document Update	10/28/2022	Elena Maranon
1.3	Document Update Review	11/02/2022	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

Luis Quispe Gonzales	Halborn	Luis.QuispeGonzales@halborn.com
Connor Taylor	Halborn	Connor.Taylor@halborn.com
Elena Maranon	Halborn	Elena.Maranon@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Mars Protocol engaged Halborn to conduct a security audit on their smart contracts beginning on September 12th, 2022 and ending on October 28th, 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 smart contract. 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 to:

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

In summary, Halborn found the contract to follow secure development best practices in many areas; however, the following additional measures were highlighted during the assessment:

- Update ownership transfer functionality to require two-step validation before updating the contract owner.
- Implement thresholds on the vesting schedule to ensure the vesting window is not set in the past.

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

First round of testing (Sep 12th - Sep 16th):

1. CosmWasm Smart Contracts
    (a) Repository: periphery
    (b) Commit ID: cd5914b25072060d4cb423766ea00ac992a59844
    (c) Contracts in scope:
        i. vesting

Second round of testing (Oct 26th - Oct 28th):

1. CosmWasm Smart Contracts
    (a) Repository: periphery
    (b) Commit ID: 0bbfbb11bc29560a936c32dedf92df3dd28c48e5
    (c) Contracts in scope:
        i. vesting

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

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	2	2

LIKELIHOOD

IMPACT

(HAL-01)				
	(HAL-02)			
(HAL-03) (HAL-04)				

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) PRIVILEGED ADDRESS TRANSFERRED WITHOUT CONFIRMATION	Low	RISK ACCEPTED
(HAL-02) INITIAL TIMESTAMPS DO NOT HAVE A MINIMUM THRESHOLD	Low	RISK ACCEPTED
(HAL-03) OUTDATED RUST EDITION IN USE	Informational	SOLVED - 09/20/2022
(HAL-04) OVERFLOW CHECKS NOT SET FOR PROFILE RELEASE	Informational	SOLVED - 09/20/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) PRIVILEGED ADDRESS TRANSFERRED WITHOUT CONFIRMATION - LOW

Description:

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

Code Location:

Ownership transfer function:

Listing 1: contracts/vesting/src/contract.rs (Line 156)

```
146 pub fn transfer_ownership(
147     deps: DepsMut,
148     sender_addr: Addr,
149     new_owner_addr: Addr,
150 ) -> StdResult<Response> {
151     let owner_addr = OWNER.load(deps.storage)?;
152     if sender_addr != owner_addr {
153         return Err(StdError::generic_err("only owner can transfer
 ↳ ownership"));
154     }
155 
156     OWNER.save(deps.storage, &new_owner_addr)?;
157 
158     Ok(Response::new()
159         .add_attribute("action", "mars/vesting/transfer_ownership"
 ↳ )
160         .add_attribute("previous_owner", owner_addr)
161         .add_attribute("new_owner", new_owner_addr))
162 }
```

Risk Level:

Likelihood - 1
Impact - 4

Recommendation:

It is recommended to split the **owner transfer** functionality into the set_owner and accept_ownership functions. This last function allows the recipient to complete the transfer. In governance contract it might be worth considering to setting the contract to be administrated itself, so this way users might be able to manage it using governance and no central administrative entity is needed.

Remediation Plan:

RISK ACCEPTED: Since the Mars Protocol team is deploying this contract in their bespoke app-chain, they have decided to accept this risk. In case an error occurs during the call to the transfer_ownership function, the Mars Protocol team will perform a chain upgrade to resolve it.

3.2 (HAL-02) INITIAL TIMESTAMPS DO NOT HAVE A MINIMUM THRESHOLD - LOW

Description:

The schedule struct used to define the length of the vesting period within the instantiate function does not verify that initial timestamps (start-time, duration) are greater than current timestamp. As a consequence, if contracts are deployed with inaccurate initial timestamps could generate unexpected situations, such as immediate withdrawal of Mars tokens.

Code Location:

Listing 2: contracts/vesting/src/contract.rs (Line 37)

```
28 pub fn instantiate(
29     deps: DepsMut,
30     _env: Env,
31     _info: MessageInfo,
32     msg: InstantiateMsg,
33 ) -> StdResult<Response> {
34
35     set_contract_version(deps.storage, CONTRACT_NAME,
↳ CONTRACT_VERSION)?;
36
37     OWNER.save(deps.storage, &deps.api.addr_validate(&msg.owner)?)
↳ ?;
38     UNLOCK_SCHEDULE.save(deps.storage, &msg.unlock_schedule)?;
39
40     Ok(Response::new())
41
42 }
```

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 plus duration are greater than current timestamp.

Remediation Plan:

RISK ACCEPTED: As the Mars Protocol team is deploying this contract in their bespoke app-chain, they have decided to accept this risk. In case of a misconfiguration happens during the call to the instantiate function, the Mars Protocol team will perform a chain upgrade to resolve it.

3.3 (HAL-03) OUTDATED RUST EDITION IN USE - INFORMATIONAL

Description:

A review of the Edition parameter within the contracts **Cargo.toml** file showed that the outdated **2018** Rust version had been included. The Rust team released the latest implementation of the Rust language in October 2021, known as the **2021 edition**. The latest iteration of the language includes many stability and feature improvements which may allow developers to implement more consistent, readable and secure code.

Code Location:

Listing 3: Resources affected

```
1 contracts/vesting/Cargo.toml
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to review functionality introduced within the **2021 edition** of the Rust language and, where compatible, upgrade to the latest edition.

Remediation Plan:

SOLVED: The Mars Protocol team fixed the issue in commit b206f007a2ba7eeae5c99ce11df62f3aeebe1030.

3.4 (HAL-04) OVERFLOW CHECKS NOT SET FOR PROFILE RELEASE - INFORMATIONAL

Description:

Although the overflow-checks parameter is set to **true** in profile.release and 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 4: Resources affected

```
1 contracts/vesting/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:

SOLVED: The Mars Protocol team fixed the issue in commit b206f007a2ba7eeae5c99ce11df62f3aeebe1030.

THANK YOU FOR CHOOSING

// HALBORN