Audit Report

Fields of Mars

v1.0

February 17, 2022

Table of Contents

License

THIS WORK IS LICENSED UNDER A CREATIVE COMMONS ATTRIBUTION-NODERIVATIVES 4.0 INTERNATIONAL LICENSE.

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

COPYRIGHT OF THIS REPORT REMAINS WITH THE AUTHOR.

This audit has been performed by

Oak Security

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security has been engaged by Delphi Labs to perform a security audit of the Fields of Mars smart contracts

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following GitHub repository:

https://github.com/mars-protocol/fields-of-mars

Commit hash: `386ffc7c0aebdac0c7402aca01af34466c6bee9b`

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line by line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

The submitted code implements the smart contract of Martian Field, a leveraged yield farming strategy utilizing contract-to-contract (C2C) lending from Mars Protocol.

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending,** **Acknowledged** or **Resolved**.

Note that audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than a security audit and vice versa.

Summary of Findings

No	Description	Severity	Status
1	Harvest message can be sandwiched to skim rewards	Major	Resolved
2	High spread might cause swapped primary asset not enough to cover user's debt	Minor	Resolved
3	Missing validation of initialization parameters may cause errors or unexpected behavior	Minor	Resolved
4	Extra denoms sent are lost in contract	Minor	Resolved
5	Tax deduction logic is not implemented	Minor	Acknowledged
6	Typos in response attributes may degrade the user experience	Informational	Resolved
7	Remove unused code in the codebase	Informational	Acknowledged
8	Outstanding TODOS are present in the codebase	Informational	Resolved

Code Quality Criteria

Criteria	Status	Comment
Code complexity	Medium-High	-
Code readability and clarity	Medium-High	-
Level of documentation	Medium-High	-
Test coverage	Low	There are no unit tests in the codebase. We recommend augmenting the test coverage.

Detailed Findings

1. Harvest message can be sandwiched to skim rewards

Severity: Major

The `Harvest` operation in `contracts/martian-field/src/contract.rs:23-26` claims rewards from the Astroport generator and then, after treasury fees have been applied, sells ASTRO to rebalance between the primary and secondary assets (in two steps).

Since the Harvest message is permissionless and can be called by anyone, there is an opportunity to skim the operation, reducing the total amount bonded in `contracts/martian-field/src/execute.rs:210`.

An example of the arbitrage attack to skim rewards would be the following, executed as a single transaction:

1. User swaps ASTRO for the primary asset.
2. User calls `Harvest` message, effectively adding selling pressure to ASTRO and buying pressure to the primary asset.
3. User swaps back the primary asset for ASTRO.
4. Go to step 1 when ASTRO rewards have accrued.

Bots could perpetually exploit this arbitrage if it were economically viable, hindering the overall financial performance of the `Field of Mars` protocol.

Recommendation

We recommend making the `Harvest` operation permissioned, to reduce the opportunity for arbitrage. Alternatively, the team/community could aim to call the `Harvest` function with enough frequency so the arbitrage attack is not economically feasible due to transaction fees, but this is also dependent on the ASTRO rewards claimed.

Status: Resolved

2. High spread might cause swapped primary asset not enough to cover user's debt

Severity: Minor

In `contracts/martian-field/src/execute_callbacks.rs:517`, the primary asset of the borrower would be swapped to the secondary asset with a `max_spread` of 50%. The huge spread might cause the returned amount to be lower than the reversed simulated amount queried from `L502-505`. This might cause the swapped primary asset not enough to fully pay off the borrower's debt, leading to a failed liquidation attempt.

Recommendation

We recommend lowering the `max_spread` value to 5%.

Status: Resolved

3. Missing validation of initialization parameters may cause errors or unexpected behavior

Severity: Minor

In `packages/field-of-mars/src/martian_field.rs:86`, the method check performs validation on some of the Config parameters, namely validating the input addresses. This method is used during the contract initialization in `contracts/martian-field/src/execute.rs:16` and on Config updates in `contracts/martian-field/src/contract.rs:32`. However, the Decimal fields `max_ltv`, `fee_rate`, and `bonus_rate` are left unvalidated. This may cause errors and unexpected behavior, such as setting up a `fee_rate` greater than `Decimal:one`.

The same issue also applies to the `UpdateConfig` message where the supplied configuration is directly overwritten without any user input validation.

Recommendation

In order to avoid human errors, we recommend validating also the numerical parameters in Config, providing bounds to their allowed values.

Status: Resolved

4. Extra denoms sent are lost in contract

Severity: Minor

In the `Deposit` function, users are able to deposit CW20 or native tokens to the contract. The code logic in `contracts/martian-field/src/execute.rs:128-135` does not account for whether there are extra unused funds sent to the contract. This would cause the user's funds to be lost in the contract.

Recommendation

We recommend reverting with an error if there are any `info.funds` sent when `asset` specified is `CW20` and making sure extra funds cause a revert when `asset` specified is `Native`.

Status: Resolved

5. Tax deduction logic is not implemented

Severity: Minor

The Field of Mars contract uses the `cw-asset` library as a dependency, to facilitate the handling of both CW20 and native assets. The `cw-asset` library does not account for Terra taxes logic, which may cause forward-compatibility issues in the future. Even though the tax rate is currently set to zero, the tax mechanism itself has not been removed. Whilst unlikely, a future governance vote might decide to make use of the mechanism again and augment the tax rate, which would affect functions such as `provide_liquidity()` in `martian-field/src/execute_callbacks.rs:18`, potentially causing the loss of native contract funds due to accounting errors, and/or panics.

Recommendation

We recommend maintaining the tax deduction logic and tax queries while the tax rate mechanism has not been completely removed from Terra Core. It should be noted that we deem the probability of a vote in favour of the increase of the tax rate as low in the foreseeable future, and hence we classify this issue as Minor.

Status: Acknowledged

6. Typos in response attributes may degrade the user experience

Severity: Informational

In the `UpdatePosition` msg, in `contracts/martian-field/src/execute.rs:109`, there is a typo in the attribute of the response `"martian_field :: excute :: update_position"`. This may degrade the user or development experience when using this contract.

Recommendation

We recommend fixing the typo to ensure a more clear user experience when calling `UpdatePosition`.

Status: Resolved

7. Remove unused code in the codebase

Severity: Informational

The Snapshot feature in the codebase is not used anywhere in the codebase. There's no entry point for users or contract admin to execute that function. Additionally, there's a TODO comment in `contracts/martian-field/src/state.rs:12` hinting that the code should be deleted.

Recommendation

We recommend removing the snapshot feature from the codebase.

Status: Acknowledged

The Mars team states that the snapshot feature was used by the frontend to calculate a user's profit-and-loss (PnL). This is temporary and will be removed through a contract upgrade once their web dev team builds an off-chain transaction indexer that can calculate PnL without using snapshots saved on-chain.

8. Outstanding TODOS are present in the codebase

Severity: Informational

During the audit engagement, several TODO comments were found in the following code lines:

- `contracts/martian-field/src/execute.rs:186-187`
- `contracts/martian-field/src/execute.rs:266`
- `contracts/martian-field/src/execute_callbacks.rs:517`
- `contracts/martian-field/src/state.rs:12`

This implies that the contract might still be under development and not yet ready for mainnet deployment.

Recommendation

We recommend resolving the TODO issues and/or removing them from the codebase.

Status: Resolved