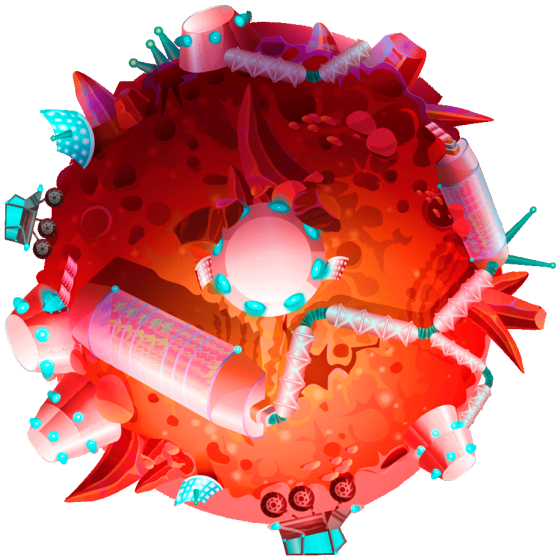

Mars Colony Security Analysis

by Pessimistic

This report is public

March 23, 2022

Abstract

In this report, we consider the security of smart contracts of Mars Colony project. Our task is to find and describe security issues in the smart contracts of the platform.

Disclaimer

The audit does not give any warranties on the security of the code. A single audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, a security audit is not investment advice.

Summary

In this report, we considered the security of Mars Colony smart contracts. We performed our audit according to the procedure described below.

The audit showed an issue with tests and several issues of low severity.

After the initial audit, another codebase update was performed. This update included fixes of Test isses and low severity issues.

General recommendations

We recommend implementing CI to run tests, calculate code coverage, and analyze code with linters and security tools.

Project overview

Project description

For the audit, we were provided with Mars Colony project on a public GitHub repository, commit d180a9617d135124bf565faaab2e5fefedad65c4.

The project has a **README.md** file with a short description of the project. The codebase has useful comments. However, there is no detailed documentation available.

Seven tests out of 51 do not pass. The code coverage is not measured.

The total LOC of audited sources is 447.

Codebase update

After the initial audit, we were provided with commit f00216f70112c43cd4e1376cf886cd6070315f7f.

In this update, the developers fixed all the issues from initial report and added new tests. All 61 tests pass, the code coverage is 100%.

Procedure

In our audit, we consider the following crucial features of the code:

1. Whether the code is secure.
2. Whether the code corresponds to the documentation (including whitepaper).
3. Whether the code meets best practices.

We perform our audit according to the following procedure:

- Automated analysis
  - We scan the project's codebase with the automated tool Slither.
  - We manually verify (reject or confirm) all the issues found by the tool.
- Manual audit
  - We manually analyze the codebase for security vulnerabilities.
  - We assess the overall project structure and quality.
- Report
  - We reflect all the gathered information in the report.

Manual analysis

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

The audit showed no critical issues.

Medium severity issues

Medium issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

M01. Tests issues (fixed)

Seven tests out of 51 do not pass. The overall code coverage is not measured.

Testing is crucial for code security. An audit does not replace tests in any way. We highly recommend covering the codebase with tests and ensuring that all tests pass and the code coverage is sufficient.

The issues have been fixed and are not present in the latest version of the code.

Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

L01. Code quality (fixed)

In **GameManager** contract, consider declaring `getEarningData` function as `external` instead of `public` to improve code readability and optimize gas consumption.

The issue has been fixed and is not present in the latest version of the code.

L02. Code quality (fixed)

In **GameManager** contract, an admin can change important parameters of the project. Consider emitting proper events when these parameters change:

- In `setMaxTokenId` function.
- In `setPrice` function.

The issues have been fixed and are not present in the latest version of the code.

L03. Code quality (fixed)

In **GameManager** contract, the following functions violate the CEI pattern:

1. `mintNFT` function.
2. `buildBaseStation` function.
3. `buildAndPlaceBaseStation` function.
4. `placeBaseStation` function.
5. `buildTransport` function.
6. `buildAndPlaceTransport` function.
7. `placeTransport` function.
8. `buildRobotAssembly` function.
9. `buildAndPlaceRobotAssembly` function.
10. `placeRobotAssembly` function.
11. `buildPowerProduction` function.
12. `buildAndPlacePowerProduction` function.

13. `placePowerProduction` function.
14. `claimEarned` function.

We highly recommend following the CEI pattern to increase the predictability of the code execution and to protect the contracts from some types of re-entrancy attacks.

The issues have been fixed and are not present in the latest version of the code.

L04. Project management (fixed)

1. Tools are located in the `dependencies` section of **package.json** file. Consider installing them as `devDependencies`.

2. Optimization is switched off in **truffle.js** config file. The `runs` value is not specified.

The issues have been fixed and are not present in the latest version of the code.

Notes

N01. Unsafe ERC721 (fixed)

In **MC** contract, the `mint` function makes an external call to `IERC721ReceiverUpgradeable(to).onERC721Received()` through a `_safeMint` call. Note that the receiving contract can have any logic.

The issue has been fixed and is not present in the latest version of the code.

N02. Overpowered role

The project has a `DAO` role. This role can perform the following actions (also stated in the documentation):

- Transfer itself to another address.
- Pause/unpause the whole system.
- Airdrop tokens.
- Change treasury and liquidity addresses.
- Change the fee of NFT.
- Increase maximum number of NFT tokens.
- Withdraw ETH (or Harmony or whatever depending on blockchain), but only to DAO address.
- Change the base URI of the metadata of NFTs.
- Change GameManager address in NFT and CLNY tokens to reorganise the system.

> *Comment from documentation: DAO is the owner of all contracts - actually can be EOA or a contract (multisig for example).*

This analysis was performed by Pessimistic:

Daria Korepanova, Security Engineer
Evgeny Marchenko, Senior Security Engineer
Nikita Kirillov, Junior Security Engineer
Boris Nikashin, Analyst
Irina Vikhareva, Project Manager

March 23, 2022