Starton Security Analysis

by Pessimistic

This report is public

May 16, 2023

Abstract

In this report, we consider the security of smart contracts of Starton 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 Starton smart contracts. We described the audit process in the section below.

The initial audit showed one critical issue: No role protection. The audit also revealed two issues of medium severity: Unchecked call result, Denial of service. Moreover, several low-severity issues were found.

After the initial audit, the developers provided us with a new version of the code. This version contained fixes for all found issues.

One issue of low severity (L05) was removed from the report as it was identified as a false positive.

The overall code quality of the project is good.

General recommendations

We have no further recommendations.

Project overview

Project description

For the audit, we were provided with Starton project on a public GitHub repository, commit 5226594da4fad1e364401d1af76c58d275a7465f.

The scope of the audit included everything except the `deprecated/` folder.

The documentation for the project included https://docs.starton.io/docs/Smart-contract/understanding-smart-contracts.

All 451 tests pass successfully. The code coverage is 99.45%.

The total LOC of audited sources is 1442.

Codebase update

After the initial audit, the developers provided us with a new version of the code, commit 24cafb961cf568e26dde0ec48ecf1dd5a636002d. In this update, the developers fixed all of the issues.

The tests count increased to 452, and the new code coverage is 97.25%.

Audit process

We started the audit on April 26 and finished on May 4, 2023.

We inspected the materials provided for the audit. Then, we contacted the developers for an introduction to the project.

During the work, we stayed in touch with the developers and discussed confusing or suspicious parts of the code.

We manually analyzed all the contracts within the scope of the audit and checked their logic. Among other, we verified the following properties of the contracts:

- Whether all functions that needed to be protected by roles are protected;
- Is it possible to lock the auction and win it;
- Is it possible to lock functionality by overflowing arrays;
- Whether the vesting calculations are correct;
- Whether the meta-transactions logic works correctly.

We scanned the project with the static analyzer Slither and our plugin Slitherin with an extended set of rules and manually verified the output.

We ran tests and calculated the code coverage.

We combined in a private report all the verified issues we found during the manual audit or discovered by automated tools.

After the initial audit, we discussed the results with the developers. On May 15, 2023, the developers provided us with an updated version of the code. In this update, they fixed all of the issues from our report.

We reviewed the updated codebase and updated 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.

C01. No role protection (fixed)

Anyone can call the `setPrices` function of the **StartonERC1155BaseSale** contract and change the prices of NFTs. If the attacker sets a minimum price, they will be able to mint all `leftSupply` tokens using different addresses.

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

Medium severity issues

Medium severity 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. Unchecked call result (fixed)

The `bid` function in the **StartonERC1155AuctionSale** and **StartonERC721AuctionSale** contracts at lines 87 and 84 do not have a check of the return value of the `call` function.

If the previous winner has no `fallback` function, then the `call` will return `false`, and the user will not receive `ether`. After the end of the auction, their bid will be sent to the `_feeReceiver` address. That is, the previous winner will not be able to receive their bid.

However, if the return value of the call is handled and reverted in case of `false`, then no one else can place a bid. And the winner will be the one who cannot get the `ether` (does not have a `fallback` function).

Consider adding functionality that allows users to withdraw on their own.

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

M02. Denial of service (fixed)

The `_createVesting` and `claimVesting` functions of **StartonVesting** contract iterate over `_beneficiaries` array. A malicious user can create lots of vestings and make the `addVesting` functionality unusable since the gas usage for this function will overflow the block gas limit.

The issue has been fixed and is 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. Gas usage (fixed)

Consider moving `totalAmount <= msg.value` require at line 235 in the **StartonVesting** contract to the line after the `for`-loop to reduce gas consumption.

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

L02. Unused parameter (fixed)

`bytes32[] memory data` parameter is not used in the `mintBatch`, `mint` functions of the **StartonERC721BaseSale** and **StartonERC1155BaseSale** contracts.

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

L03. Gas consumption (fixed)

The expression at line 151 of **StartonERC1155BaseSale** could be moved inside an `unchecked` block since it cannot underflow because of the check at line 149.

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

L04. Usage of msg.sender (fixed)

In the `setUser` function of **StartonERC4907** contract, `msg.sender` is used instead of the `_msgSender()`, given that the contract supports meta-transactions. Also, note that in case the contract is intended to store the tokens, a malicious user can set the user for the contract's tokens by calling the `setUser` function through meta-transactions.

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

This analysis was performed by Pessimistic:

Yhtyyar Sahatov, Junior Security Engineer
Daria Korepanova, Security Engineer
Irina Vikhareva, Project Manager

May 16, 2023