Work X Reward Security Analysis

by Pessimistic

This report is public

February 9, 2024

Abstract

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

The initial audit showed the following issues of medium severity: Exceeding the total level rewards limit, Privileged roles and No documentation. Also, several low-severity issues were found.

All tests passed. The project did not have the documentation.

The logic of the code interacts very closely with the NFT part of the codebase, which is very complex and intricate. We believe that a detailed specification is needed against which the code of this project can be compared. Otherwise, it is impossible to guarantee that every edge case works as intended.

After the initial audit, the codebase was updated. All issues from the initial audit were fixed. The developers provided sufficient documentation.

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 Work X Reward project on a public repository, commit 8d28cfe13b12e3a34d4d2db8051c75ae409e3041.

The scope of the audit included the **IRewarder** interface and the `reward` folder included the following contracts: **RewardShares**, **RewardTokens**, and **RewardWrapper**.

The project has no documentation.

609 tests out of 663 pass successfully, 54 tests have pending status. The code coverage of the current scope is 96.37%.

The total LOC of audited sources is 404.

Codebase update

After the initial audit, the developers applied fixes to Work X Reward codebase. We reviewed the updated code at commit 2a68ac35e55950276447d10a231dbc782549c616.

In the updated code, the developers fixed all present issues. In addition to this, they thoroughly described staking and rewards logic in README and a blog post. We reviewed the changes and found no new issues.

613 tests out of 667 pass successfully, 54 tests have pending status. The code coverage of the current scope is 93%.

Audit process

We started the audit on January 29 and finished on January 31, 2024.

We manually analyzed all the contracts within the scope of the audit and checked their logic. During the work, we stayed in touch with the developers and discussed confusing or suspicious parts of the code. Among other, we verified the following properties of the contracts:

- The interaction with the NFT part of the project;
- Whether the sum of rewards does not exceed the limit;
- Whether the values in the hardcoded arrays match the descriptions;
- Whether it is not possible to claim someone else's rewards.

We scanned the project with the following tools and manually verified the output:

- Static analyzer Slither;
- Our plugin Slitherin with an extended set of rules for Slither;
- Semgrep rules for smart contracts. We also sent the results to the developers in the text file.

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.

On February 6, 2024, we received an updated version of the codebase with all fixes and new documentation for the staking and rewards logic.

We checked all changes since the previous iteration. No issues were discovered during the review. We updated the report to reflect current status.

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 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. Exceeding the total level rewards limit (fixed)

The `claim` function of the **RewardShares** contract checks the `totalLevelClaimed` variable inside the `getClaimable` call at line 296 before updating it. This means that the user can call `claim` and exceed the `TOTAL_LEVEL_REWARDS` limit by a maximum of 640 tokens. If this happens, these rewards will be deducted from the share's rewards balance. As a result, users may not be able to get their shares rewards in the last or even penultimate month.

Consider letting the last user to claim the remaining level reward.

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

M02. Privileged roles (addressed)

The owner of the project can:

- **(fixed)** withdraw all reward tokens from the contract balance in the `withdrawTokens` function of the **RewardTokens** and **RewardShares** contracts;
- add a malicious rewarder in the `RewardWrapper.setRewarders` function who can revert the `claim` function of this contract, preventing the user from picking up the rewards. It is not so essential as the user can pick up the rewards individually.

> *Comment from the developers: The setRewarders owner function is required in order to clean up reward programs that have concluded as well as add new reward programs in the future. Since the owner will be transfered to the Work X multisig wallet the governance of the rewarders is secure.*

We recommend designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig.

M03. No documentation (fixed)

The project has no documentation. The documentation is a critical part that helps to improve security and reduce risks. It should explicitly explain the purpose and behavior of the contracts, their interactions, and key design choices.

This code interacts with the NFT part of the project and depends heavily on it. However, as there is no detailed documentation, we had to communicate a lot with developers during the audit process. We consider that a complete description of the whole project is needed to avoid missing any edge case.

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. Simplified version of the function (fixed)

The `getRewarders` function of the **RewardWrapper** contract can be simplified to one line of code `{return rewarder;}`.

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

L02. Immutable (fixed)

The following variables are set during contract deployment and never change later. Declare them as `immutable` to reduce gas consumption.

- at line 11 of the **RewardWrapper** contract;
- at lines 20-21 of the **RewardShares** contract;
- at lines 18-19 of the **RewardTokens** contract.

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

L03. No inheritance (fixed)

The **RewardTokens** contract is not inherited from the **IRewarder** interface. But it seems that it should since it has similar functionality to the **RewardShares** contract.

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

L04. No event in the setter (fixed)

The `setRewarders` function of the **RewardWrapper** contract does not emit an event. Consider adding an appropriate event to make off-chain interaction with the contract easier.

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

L05. ERC-20 standard deviation (fixed)

The `withdrawTokens` function of the **RewardTokens** and **RewardShares** contracts does not check the returned value of the `transfer` function and does not follow the ERC-20 standard state:

```
Callers MUST handle false from returns (bool success). Callers MUST
NOT assume that false is never returned!
```

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

L06. Payable status (fixed)

The `withdrawTokens` function of the **RewardTokens** and **RewardShares** is `payable`. However, it does not have functionality with `ether`. On the contrary, it allows the owner to withdraw tokens from the contract balance.

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

L07. The discrepancy with NatSpec comments (fixed)

The rewards sum (it is equal to `1386516`) at lines 33-72 of the **RewardTokens** contract does not correspond to the `1386522` value from the comment at lines 14-15:

```
This reward is capped by 1,386,522 $WORK tokens, and spread out over
40 months, giving a predetermined total reward portions per month,
that are shared proportionally by all NFTs based on the amount of
tokens staked in them.
```

Perhaps this error can be due to rounding during level rewards calculations.

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

This analysis was performed by Pessimistic:

Evgeny Marchenko, Senior Security Engineer
Daria Korepanova, Senior Security Engineer
Irina Vikhareva, Project Manager
Konstantin Zherebtsov, Business Development Lead

February 9, 2024