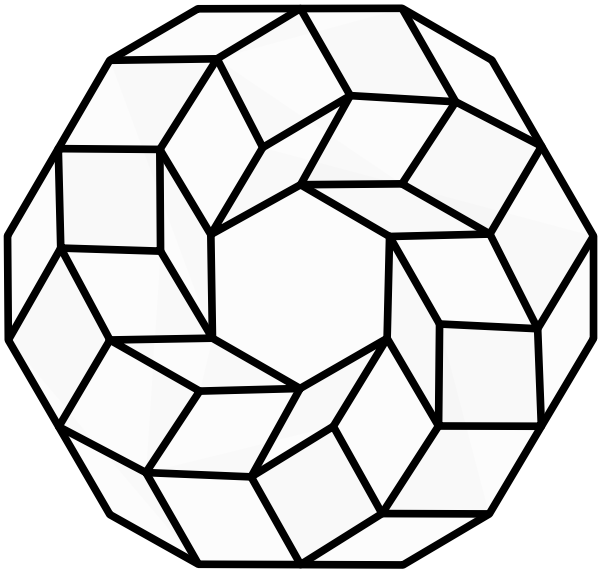

PowerPool Tornado Connector & Router Security Analysis

by Pessimistic

This report is public

April 25, 2022

Abstract

In this report, we consider the security of smart contracts of PowerPool Tornado Connector & Router 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 PowerPool smart contracts. We performed our audit according to the procedure described below.

The initial audit showed several issues of low severity. They do not endanger project security.

The code within the scope is of good quality and is decently covered by NatSpec comments. However, it is somewhat complex and lacks proper documentation. Besides, reward logic might become vulnerable to exploitation if the contracts are improperly initialized.

Later the codebase was updated. The developers fixed or commented all previously discovered issues.

General recommendations

We recommend improving documentation of the project.

Project overview

Project description

For the audit, we were provided with PowerPool Tornado Connector & Router project on a public GitHub repository, commit feecb42530898af36e6f6d3ec5b06be556336237.

The scope of the audit included only the pull request #7.

The project does not have any documentation. The developers provided a description for the project during communication. The codebase has detailed NatSpec comments.

All 102 tests pass successfully. The overall code coverage is 76.51%.

Codebase update

After the initial audit, the codebase was updated. For the recheck, we were provided with commit f0c5355df73aad935a16b768d5e738cd59a31259.

The developers either fixed or commented all issues. The update includes minimal required changes.

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 tools: Slither and SmartCheck.
  - We manually verify (reject or confirm) all the issues found by the tools.
- 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.

The audit showed no issues of medium severity.

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. Missing interface (fixed)

WrappedPiErc20 contract should implement Permit interface, we recommend the one from OpenZeppelin.

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

L02. Complicated code structure (fixed)

The codebase includes multiple very short public functions, which complicates code structure. Most of these functions are only called once, consider inlining them to improve code readability.

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

L03. CEI pattern violation

`_pokeFrom` function of **PowerIndexRouter** contract violates CEI (checks-effects-interactions) pattern. We highly recommend following CEI pattern to increase the predictability of the code execution and protect from some types of re-entrancy attacks.

> *Comment from the developers: It's not possible in current state of project to do all necessary checks in the first step of _pokeFrom function due to complexity of logic. Also external interactions are separated to two stages to avoid a situation where there may not be enough balance, and there's a check between them to optimize gas usage.*

L04. Code quality (fixed)

Consider declaring functions as external instead of public when possible, e.g., `getPendingRewards`, `accumulatedRewardRateDiffForLastUpdate`, and `packClaimParams` functions of **TornPowerIndexConnector** contract. It improves code readability and optimizes gas consumption.

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

Notes

N01. Possible reward manipulations

TornPowerIndexConnector contract allows to claim rewards only when the expected compound reward exceeds transaction fee of the call. However, this function does not take the poker's reward into account. Besides, the current criteria is susceptible to manipulation, especially for very short poke intervals.

This analysis was performed by Pessimistic:

Evgeny Marchenko, Senior Security Engineer
Nikita Kirillov, Junior Security Engineer
Irina Vikhareva, Project Manager

April 25, 2022