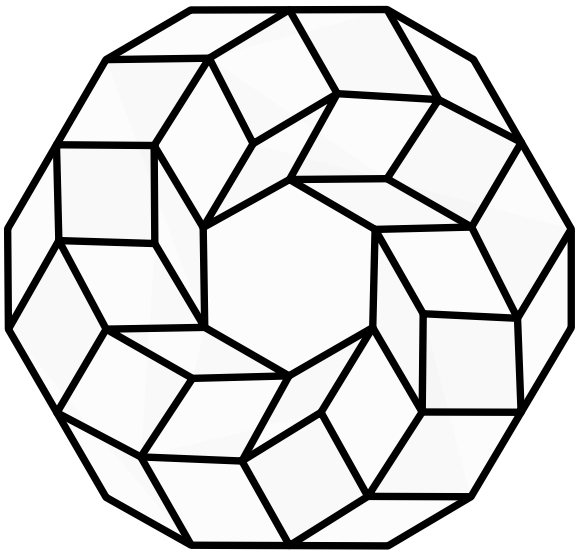

Powerpool Vaults Security Analysis

by Pessimistic

This report is public

December 24, 2021

Abstract

In this report, we consider the security of smart contracts of Powerpool Vaults 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. One 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, security audit is not an investment advice.

Summary

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

The initial audit showed two issues of medium severity: insufficient documentation and a bug. Also, several low-severity issues were found.

The code base is well-covered with tests.

After the initial audit, the code base was updated. The bug was fixed, also, the majority of the low severity issues were fixed.

General recommendations

We recommend adding public documentation and addressing the rest of the issues.

Project overview

Project description

For the audit, we were provided with Powerpool Vaults project on a public GitHub repository, commit 3b6a41d132c649e452171d71bcadf11b87e608b4.

The project has no documentation, the code base lacks NatSpec comments.

All 101 tests pass, t code coverage is 85.31%.

The total LOC of audited sources is 1666.

Code base update

After the initial audit, the code base was updated. For the recheck, we were provided with commit 748fa2f8a172ef26172a4e43a0dc8fe4db91f7ac.

In this update, the majority of issues were fixed. No new functionality was added to the project. Some code was removed. One test was added to the project and code coverage reached 88.97%. Numerous NatSpec comments were provided.

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 project's code base with automated tools: Slither and SmartCheck.
    - We manually verify (reject or confirm) all the issues found by tools.
- Manual audit
    - We manually analyze code base for security vulnerabilities.
    - We assess 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 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.

Bug (fixed)

`resultPerformanceFeeDebt` return value of `_distributePerformanceFee` function of **AbstractConnector** contract is calculated incorrectly.

However, this value is not used within the scope of the audit. Nonetheless, we recommend properly updating it or removing it completely, since otherwise it might affect other components of the system or future versions of the code.

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

Insufficient documentation

The project has almost no documentation, and the code base lacks NatSpec comments. The documentation is a critical part that helps to improve security and reduce risks. Proper documentation should explicitly describe the purpose and behavior of the contracts, their interactions, and main design choices. It is also essential for any further integrations.

Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in the future versions of the code. We recommend taking them into account.

Uncertain payability (fixed)

The project utilizes `callExternal` function of **WrappedPiErc20** contract, which is not declared as payable. However, it accepts transferred value as an argument. Thus, current version of code is contradictive: either it supports eth transfer or it does not.

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

Refactoring Issues

The codebase includes traces of earlier refactoring. We recommend removing them to simplify the code and improve readability.

- The function name `_callStaking` in **AbstractConnector** is misleading, since it accepts arbitrary selector as an argument and in fact is used for redeeming as well as for staking. Consider renaming, re-implementing, or adding comments.

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

- `onlyPiToken` modifier is never utilized within the project.

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

- File **Imports.sol** is not used.

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

- **Interfaces** folder contains code irrelevant to the project.

  > *Comment from developers: Some interfaces that are not used by the contracts are used for tests.*

- Function `callExternalMultiple` of **WrappedPiErc20** is never called within the project.

  > *Comment from developers: This function is implemented for future use.*

- The `initRouterByConnector` method of **PowerIndexRouter** relies on `initRouter` function, which is not present in the project.

Code quality

- Return value of `claimRewards` function is not explicitly specified in the case of zero reward in **AbstractMasterChefIndexConnector**.

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

- Consider declaring `redeem` and `stake` functions of **PowerIndexRouter** as `external` instead of `public`.

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

- Divide before multiply in stake calculation may lead to unnecessary precision loss at line 617 in **PowerIndexRouter**.

> *Comment from developers: We're ok with this precision loss.*

Gas consumption

The source code of **PowerIndexRouter** (at lines 468, 477 and 491) includes redundant checks. Note, that `connectors` are always checked in the setter function, thus `require`'s in getter functions can be safely changed for `assert`'s or removed.

> *Comment from developers: We're ok with the extra checks*

This analysis was performed by Pessimistic:

Evgeny Marchenko, Senior Security Engineer
Vladimir Tarasov, Security Engineer
Boris Nikashin, Analyst
Irina Vikhareva, Project Manager
Alexander Seleznev, Founder

December 24, 2021