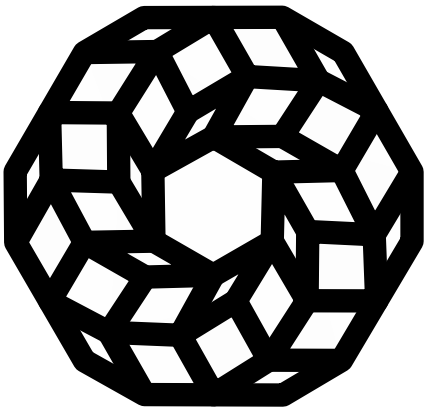

PowerPool Lending Security Analysis

by Pessimistic

This report is public.

Published: September 25, 2020

Abstract

In this report, we consider the security of smart contracts of the PowerPool project. Our task is to find and describe security issues in 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 smart contracts of PowerPool project. We performed our audit according to the procedure described below.

The initial audit showed no vulnerabilities. There were roles that could perform certain changes to the project that can affect the system operation. Also, there were some inconsistencies in error handling.

After initial audit, developers fixed some issues and provided comments to the rest of them.

General recommendations

All the issues were either fixed or commented. Thus we do not have any further recommendations.

Procedure

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

1. Whether code logic corresponds to the specification.
2. Whether the code is secure.
3. Whether the code meets best practices.

We perform our audit according to the following procedure:

- Manual audit
  - We inspect the specification and check whether the logic of smart contracts is consistent with it.
  - 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.

Project overview

Project description

In our analysis we consider smart contracts of PowerPool project on GitHub repository, commit b8c52d3144cf7320de36d326627a715b309cef14.

Latest version of the code

After initial audit, some fixes were applied and the code was updated, commit 21d8ab339366e51b70ccace9fa042ba587e1ec54.

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 smart contracts security. We highly recommend fixing them.

The audit showed no critical issues.

Medium severity issues

Medium issues can influence project operation in current implementation. We highly recommend addressing them.

Code style

In the original Compound project, **ErrorReporter** contract is used for handling errors. However, in PowerPool project this behavior is different in most cases of modifications of the original project:

- **CToken.sol**
  - Modifications of `mintFresh()` and `borrowFresh()` functions.

    > *Comment from developers: These changes do not affect **cToken** contract's storage data and are disabled by setting `tokenRestrictions` variable to zero address by the owner of **CTokenRestrictions** contract. Restrictions allow testing new markets by activating the maximum supply and borrowing limits and the whitelist of participants. For markets that have passed the testing stage, this option will be disabled.*

  - `_setTokenRestrictions()` function: `require` is used instead of **ErrorReporter** contract's functions and `return 0` is used instead of `return uint(Error.NO_ERROR)`.

    > *Comment from developers: We intentionally avoided error codes incompatibility with future versions of Compound protocol contracts as they are upgradeable. Compound Protocol itself uses both `requires()` and checks with error code outputs.*

  - At lines 501 and 502, the first component of `mulScalarTruncate` return value is not checked to be equal to `MathError.NO_ERROR`.

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

- `_setPendingAdmin` and `_acceptAdmin` functions use `return 0` instead of return `uint(Error.NO_ERROR)`, same issue also applies to 5 more files.

  > *Comment from developers: We intentionally followed the format of Compound responses but did not want to add unnecessary custom error codes.*

- **CTokenRestrictions.sol**
  - `OnlyAdmin` modifier in **CTokenRestrictions** contract uses `require`.

    > *Comment from developers: We intentionally avoided error code incompatibility with future versions of Compound protocol contracts as they are upgradeable.*

  - `getUserRestrictionsAndValidateWhitelist()` calls `validateWhitelistedUser()` function that fails if a user is not whitelisted.

We recommend following the code style of the original project.

Overpowered roles

- The owner can change `comp` address without any restrictions. This change affects the whole system and may result in unpredictable consequences.

  We recommend disabling the ability to change `comp` address after initialization.

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

- In Compound's **PriceOracle** contract there are some restrictions apply when setting a new price. In PowerPool, a manager can set any price without any restrictions.

  We recommend implementing restrictions for the price changes.

  > *Comment from developers: It is a mock contract which will be replaced with new **PowerPool Open Price Oracle**.*

Low severity issues

Low severity issues can influence project operation in future versions of code. We recommend taking them into account.

Misleading comment

The comment in **Reservoir.sol**, line 22, states that `dripRate` variable is immutable. However, the functionality for changing this variable is implemented.

We recommend either fixing the comment or removing the corresponding functionality.

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

Misleading names

- The name of `addUserToWhiteList()` function in **CTokenRestrictions.sol** file is misleading: the function not only adds a user to the whitelist but also calls `_setUserRestrictions()` function and thus, sets restrictions for that user.

  We recommend either removing `_setUserRestrictions()` call from `addUserToWhiteList()` function or renaming it to describe its functionality.

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

- There are functions that have `Comp` instead of `Cvp` in their names (e.g. `transferComp`, `ClaimComp`, etc.).
  We recommend using `Cvp` instead of `Comp` to avoid confusion.

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

Code logic

AdministratedPriceOracle contract consists of two parts: first part adds/removes managers and admin, the second part overrides public functions of **SimplePriceOracle** contract.

We recommend splitting this functionality into 3 separate contracts:

- contract that adds/removes admin
- contract that adds/removes managers
- contract that overrides functions.

> *Comment from developers: It is a mock contract which will be replaced with new **PowerPool Open Price Oracle**. These changes are not required.*

Gas optimization

`setUnderlyingPrice()` and `setDirectPrice()` functions read `prices[asset]` value twice. Consider using local variables for these values.

> *Comment from developers: **SimplePriceOracle.sol** is a mock contract of Compound Protocol. We will not use it in production deployments.*

Code style

- In **PriceOracleProxy.sol** at lines 54–60 indentation violates Solidity Style Guide.

  > *Comment from developers: It is a mock contract which will be replaced with new **PowerPool Open Price Oracle**.*

- In **CToken.sol** at line 509, brackets are unnecessary.

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

Typo

In **InterestRateModel.sol** at lines 15 and 24, there is `amnount` instead of `amount`.

> *Comment from developers:* *There was a typo in Compound Protocol source code.*
>
> *The issue was fixed and is not present in the latest version of the code.*

Code duplication

There is a code block that is repeated in several files. It includes `admin` and `pendingAdmin` variables, `onlyAdmin` modifier, `_setPendingAdmin` and `_acceptAdmin` functions.

We recommend moving this functionality into a separate file in inherit it.

> *Comment from developers:* *This is Compound Protocol approach, which is already used in* ***CToken****, and* ***Unitroller*** *contracts with code duplication.*

Redundant code

Using `TotalRestrictions` struct that consists of a single value is excessive.

We recommend using regular `uint256` value instead to simplify project structure.

> *Comment from developers:* *The list of restrictions may be changed in the future and* ***CTokenRestrictions*** *contract will be replaced with a new version. The structure allows us to add new variables to total restrictions, which we will use in the future by their getter functions for each structure field.*
>
> *The issue was fixed and is not present in the latest version of the code.*

This analysis was performed by Pessimistic:

Igor Sobolev, Security Engineer
Boris Nikashin, Analyst
Alexander Seleznev, Founder

September 25, 2020