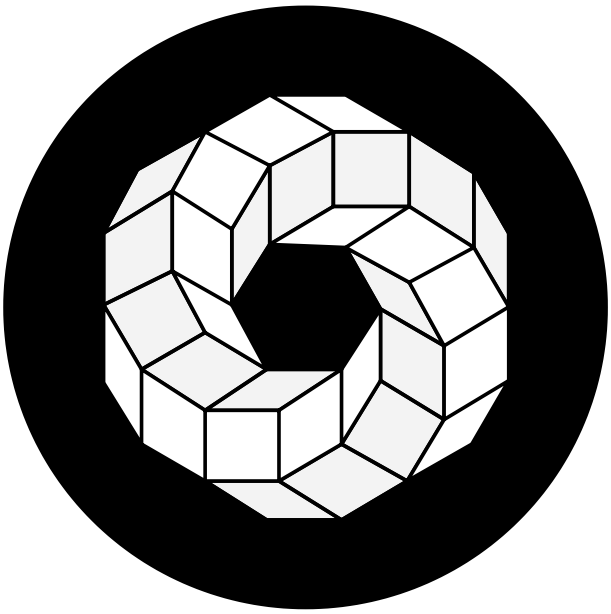

PowerPool CVP Bridge Security Analysis

by Pessimistic

This report is public

May 11, 2023

Abstract

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

The initial audit showed two issues of medium severity: Owner role and Lack of documentation. Also, several low-severity issues were found.

The overall quality of the initial code was mediocre. The code did not contain any serious issues. However, it had several code style flaws, described in the Low severity issues section.

After the initial audit, the developers provided us with a new version of the code. This version added NatSpec comments to the code and contained several fixes for the low severity issues.

During the recheck process, we identified one additional issue of low severity. This issue was not previously identified in the initial review and was only discovered as a result of updated DeBridge documentation.

After the first recheck, the developers updated the codebase. In this update, they fixed the remaining issues and commented on the M01 issue.

General recommendations

We do not have any additional recommendations.

Project overview

Project description

For the audit, we were provided with PowerPool CVP Bridge project on a public GitHub repository, commit bb65991dccc0298d9a5eec315e8a1991a1852f70.

The scope of the audit includes the whole repository.

The developers did not provide any documentation for the project.

The project has 1 test that passes successfully. This test covers 94.87% of statements, however only 57.14% of branches are covered.

The total LOC of audited sources is 311.

Codebase update #1

After the initial audit, the developers provided us with a new version of the code, commit 2dfd053a6e401a3db4f98cef6fe8a0c6bf496dee. In this update, the developers fixed most of the issues. We added one more issue due to DeBridge documentation update.

The update does not contain any new tests.

Codebase update #2

After the first codebase update, the developers provided us with a commit 49f8e04a6bd57d96145d53e64396f1784867b995. In this update, the developers fixed the remaining low severity issues, commented on the M01 issue and implemented several optimizations.

The update does not contain any new tests.

Audit process

We started the audit on February 13, 2023 and finished on February 17, 2023.

We inspected the materials provided for the audit. Then we investigated the used bridging solution: DeBridge. After the preliminary research, we specified the parts of the code and logic that require additional attention during the audit:

- The correctness of the DeBridge integration;
- The correctness of unlock transaction checks.

We manually analyzed all the contracts within the scope of the audit and checked their logic.

We scanned the project with the automated tool slither and manually verified all the occurrences found by it.

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 April 18, 2023, the developers provided us with an updated version of the code. In this update, they fixed most of the issues from our report, improved code quality, and added NatSpec comments.

We reviewed the updated codebase and updated the report.

After the first update, the developers provided a new version of the code. In this update, they fixed issue L05 which was identified during the previous codebase update. In addition to this, the developers contacted DeBridge team and verified that the new function from N02 note can be used. As a result of this, L04 issue was also fixed. Also, the developers optimized the code and started using `chainid()` function for obtaining chain id instead of providing it as a constructor argument.

We reviewed the 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.

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. Owner role (commented)

The system relies heavily on the `Owner` role. It can change crucial parameters of the system:

- `deBridgeGate` address;
- Addresses of bridge contracts on other chains;
- Token transfer limits.

Moreover, this role is able to withdraw tokens from the bridge contract at any time.

There are scenarios that can lead to undesirable consequences for the project and its users, e.g., if owner private key becomes compromised. We recommend designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig.

> *Comment from the developers: Thanks for highlighting concerns. We will use multisig for the contract's Owner role, enhancing security, and later transfer it to decentralized governance.*

M02. Lack of documentation (fixed)

The project has no documentation. We recommend writing it since it not only helps to avoid issues during the project development, but also makes code maintenance easier.

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. Constructor visibility (fixed)

CvpBridgeLocker contract has a constructor with explicit visibility. Note, that this visibility is ignored.

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

L02. Misleading comments (fixed)

The code contains misleading comments left from the DeBridge tutorial at lines 108-109. We recommend removing them.

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

L03. Misleading variable names (fixed)

The code contains limits for token transfers. These limits are stored in the following variables: `chainLimitPerDay` and `transfersPerDay`. However, according to line 50, these variables limit transfer volume per a week. We recommend changing variable names.

The developers changed the code logic instead of variable names. In the latest version of the code, they represent a limit transfer volume per day.

L04. Redundant code (fixed)

In **CvpBridgeLocker** contract, `_send` function accepts `_executionFee` argument and contains some calculations with it at lines 96-98. However, this variable is always constant and equals to zero. We recommend removing redundant code.

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

L05. Asset fee argument value (fixed)

According to the updated DeBridge documentation, `_useAssetFee` argument:

> *Should also be set to False. (Reserved for future use by governance to accept fees in the form of the transferred token.)*

Due to this, we recommend setting that argument value to `False` in **CvpBridgeLocker.sol** file at line 163. Disabling asset fee will help make the code behavior more predictable when the expected functionality is implemented by the DeBridge protocol.

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

Notes

N01. DeBridge development is in progress

Currently, DeBridge protocol is actively developing and hence may have some instabilities. Moreover, some features are not yet implemented, for example, submission ids retrieval via IPFS.

N02. Sending messages in the future versions of DeBridge (fixed)

DeBridge protocol allows sending custom messages with calldata between blockchains. Currently, it is done via `send` function. However, in the future version of the code, `DeBridge` will have a more convenient function for that purpose: `sendMessage`. We recommend using that one once DeBridge will roll out an update.

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

N03. Referral code

We recommend setting DeBridge referral code at line 133 in **CvpBridgeLocker.sol** file.

After the initial audit, the developers added the ability for the user to provide their referral code.

N04. Token limits

CvpBridgeLocker contract contains limits for token transfers. Transfer values are checked both when the tokens are sent and when they are received. Since the receiving transaction is mined after the sending one, these checks might check the different periods in `_checkChainLimits` function, i.e. when the user receives tokens the next week after sending the bridging transaction.

N05. Protocol fee check (fixed)

In **CvpBridgeLocker** contract, the line 88 checks `msg.value >= (protocolFee + _executionFee)`. Note that the transaction with the `msg.value` greater than the required will succeed. This might be useful in the case when DeBridge changes protocol fee while `send` transaction has not been mined. In other cases, `msg.value` should be equal to `protocolFee`.

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

This analysis was performed by Pessimistic:

Pavel Kondratenkov, Security Engineer
Yhtyyar Sahatov, Junior Security Engineer
Irina Vikhareva, Project Manager
Alexander Seleznev, Founder

May 11, 2023