CYBRO × PESSIMISTIC

SECURITY ANALYSIS

by Pessimistic

This report is public

July 31, 2024

ABSTRACT

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

The audit showed one critical issue: Inflation attack. The audit also revealed two issues of medium severity: Insufficient documentation, Project roles. Moreover, several low-severity issues were found. All the tests passed.

After the initial audit, the developers commented on all issues.

The project has the integration with Juice Finance protocol. We cannot assume that this integration is safe as this project does not have detailed technical documentation and is out of the current scope.

GENERAL RECOMMENDATIONS

We recommend fixing the mentioned issues, improving the documentation and NatSpec coverage. We also 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 CYBRO project on a public GitHub repository, commit c4971aa677277957c8a6f1780e16c8d6d152b6ed.

The scope of the audit included:

- AaveVault.sol;
- BaseVault.sol;
- CompoundVaultErc20.sol;
- CompoundVaultEth.sol;
- JuiceVault.sol.

The documentation for the project included https://docs.cybro.io.

All 8 tests pass successfully. The code coverage is 65,59%.

The total LOC of audited sources is 433.

AUDIT PROCESS

We started the audit on July 26 and finished on July 30, 2024.

Before the audit, we contacted the developers for an introduction to the project. During the work, we stayed in touch with them and discussed confusing or suspicious parts of the code.

We manually analyzed all the contracts within the scope of the audit and checked their logic. Among other, we verified the following properties of the contracts:

- The integrations with Aave, Compound, and Juice Finance (not in detail, since it is out of the scope) protocols;
- Whether the shares and their price calculations are correct;
- Whether the owner's influence does not affect the system much;
- Standard Solidity checks.

We scanned the project with the following tools:

- Static analyzer Slither;
- Our plugin Slitherin with an extended set of rules;
- 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.

The developers provided comments for all issues after the initial audit. That is why we updated the report on July 31, 2024.

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.

C01. **Inflation attack** (commented)

The inflation attack can be applied to line 65 of the `BaseVault.deposit` function.

The example:

- The first user calls the `deposit` function and sends 1 underlying token. As a result, `totalSupply = 1`, `totalAssetsBefore = 1`;
- Then, the second user wants to deposit 3 underlying tokens;
- The malicious can front-run the transaction of the second user and send 6 cTokens or aTokens (it depends on the vault's version) to the vault;
- The amount of shares for the second user is equal to 0, as `totalSupply = 1`, `totalAssetsBefore = 1 + 6`, `increase = 3`, and `totalSupply() * increase / totalAssetsBefore < 1`;
- It can be repeat multiple times;
- Malicious user calls the `redeem` function, burns their 1 share, and withdraw all underlying tokens from the vault.

> `Comment from the developers:`
> **To ensure the safety of our users' deposits, we implement a mitigation strategy known as "dead shares." Before publishing the vault, we always make a small deposit ourselves and are ready to lose it. For the rest of the users depositing will always be safe.**
>
> **While we recognize that this approach doesn't fully resolve the underlying issue without complicating the vault logic significantly, we decided to leave it simple.**

> `Pessimistic's comment:`
>
> At the moment, 2 vaults (can be found here) are deployed. Each of them has its first deposits, which makes the described attack impossible. But all the corresponding shares are on EOA addresses. This leaves the possibility of using the above attack, as all deposits can be withdrawn. It will lead to returning `totalSupply` and `totalAssetsBefore` to 0.

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. **Insufficient documentation** (commented)

The project has insufficient documentation and no NatSpecs. That is why, it is unclear how users can claim additional project rewards, such as `COMP` tokens in Compound protocol. In addition, some project features are not used in the current code version, such as `type(uint256).max` when withdrawing funds in the Juice Finance protocol.

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. It is also essential for any further integrations.

> `Comment from the developers:`
>
> Acknowledged, won't fix in current version.

M02. **Project roles** (commented)

The admin role can change the vaults' implementations as they are upgradeable. Consider designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig to avoid scenarios that can lead to undesirable consequences for the project and its users, e.g., if admin's private keys become compromised.

> `Comment from the developers:`
>
> Acknowledged, won't fix in current version.

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. Unchecked return values (commented)

The returned value of the `pool.withdraw(assets)` call is not checked in the `JuiceVault._redeem` function. Consider checking it; it is not critical, but further calculations in the **BaseVault** contract rely on the value of the `assets`. And as the Juice Finance protocol was not included in the current scope, the actual number of withdrawn tokens may not equal this value.

> `Comment from the developers:`
> Acknowledged, won't fix in current version.

L02. Unnecessary check (commented)

The `BaseVault._validateTokenToRecover(token, address(pool))` check prevents the owner from withdrawing a token of the integrated project (`aToken`, `cToken` or `liquidityToken`) from the vault contract.

In the `JuiceVault.withdrawFunds` function, the address of the token being withdrawn is compared to the address of the `pool`, not the address of the `liquidityToken`. And this token always reverts during `transfer`. Thus, this case can never occur.

> `Comment from the developers:`
> Acknowledged, won't fix in current version.

Notes

N01. Integrations (commented)

The project has integrations with pools from Aave, Compound, and Juice Finance protocols. Any restrictions that may be placed on these pools, such as pause, limited deposits, deprecated pools, or any other, can influence CYBRO vaults' functionality.

> `Comment from the developers:`
> Acknowledged, won't fix in current version.

This analysis was performed by Pessimistic:

Daria Korepanova, Senior Security Engineer
Evgeny Bokarev, Junior Security Engineer
Konstantin Zherebtsov, Business Development Lead
Irina Vikhareva, Project Manager
Alexander Seleznev, CEO

July 31, 2024