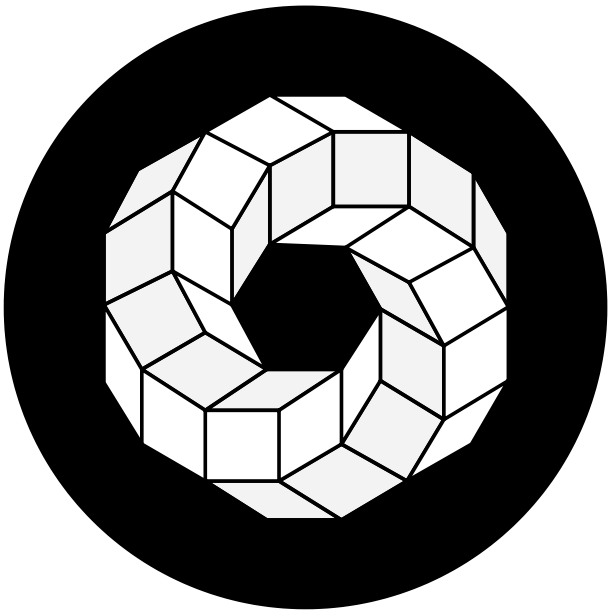

PowerPool Agent Security Analysis

by Pessimistic

This report is public

August 11, 2022

Abstract

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

The initial audit showed two critical issues: Insufficient balance protection and Loss of funds via arbitrary call. We also found several issues of medium severity, including Underflow, ERC20 standard violation, Overpowered role, and many low-severity issues.

The project documentation is private though it is required for integrations.

After the initial audit the codebase was updated. With this update the developers fixed both critical issues, the Underflow issue of medium severity, and all issues of low severity.

General recommendations

We recommend addressing the remaining issues and preparing public documentation.

Project overview

Project description

For the audit, we were provided with PowerPool Agent project on a public GitHub repository, commit 0900d445246033304d2f6cfa4597dda06f411b77.

The scope of the audit includes the whole repository except **PPAgentV2Lens.sol** file.

The documentation for the project includes private specification.

All 126 tests pass successfully. However, the code coverage is not measured.

The total LOC of audited sources is 813.

Codebase update

After the initial audit, the codebase was updated. For the recheck, we were provided with commit 6ad9584dd71951108e16658cdd069293daffb4ec. The developers fixed all critical severity issues, one medium severity issue and all issues of low severity. No new issues were discovered.

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 tool Slither.
  - We manually verify (reject or confirm) all the issues found by the tool.
- 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.

Inter alia, we check whether:

- Keepers' stakes can be stolen or locked.
- Job owners' credit can be stolen or locked.
- Arbitrary calls logic can be exploited.
- `execute` function can be optimized without major refactoring.
- Tightly packed data is processed correctly.
- The codebase includes any standard Solidity issues.
- Assembly blocks are correct and do not break Solidity compiler conventions.

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. Insufficient balance protection (fixed)

Users can select any address as a job owner. As a result, jobs may drain other users' credits. An attacker can utilize `registerJob` or `transferJob` functions of **PPAgentV2** contract.

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

C02. Loss of funds via arbitrary call (fixed)

Jobs require execution of arbitrary calls from `PPAgentV2` address. An attacker can create a job that transfers all CVP tokens from the contract to the chosen address. They also can drain any approved tokens.

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

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.

M01. Underflow (fixed)

The `execute_44g58pv` function of **PPAgentV2** contract includes a low-level call that actively manages sent gas. However, the call will proceed even with insufficient gas since inline assembly does not check for underflows.

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

M02. ERC20 standard violation

ERC-20 standard states:

```
Callers MUST handle false from returns (bool success). Callers MUST
NOT assume that false is never returned!
```

However, returned values from `transfer` and `transferFrom` calls are not checked at lines 962, 1035, and 1056 of **PPAgentV2** contract.

M03. Overpowered role

An admin role can slash keepers stakes and modify minimal stake, withdraw delay, and fee.

In the current implementation, the system depends heavily on the admin role. Thus, some scenarios can lead to undesirable consequences for the project and its users, e.g., if admin's private keys become compromised. We recommend designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig.

M04. No public documentation

The documentation for the project is not publicly available. The contracts require a non-trivial integration from third-party developers, which should not be performed without a detailed description of the system.

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.

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. Types usage (fixed)

Consider using `enum` instead of independent constants at lines 68-70 of **PPAgentV2** contract.

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

L02. Confusing naming (fixed)

`RegisterJob` event includes `maxBaseFee` parameter, while `JobUpdate` event has `maxBaseFeeGwei`. Besides, the code works with basefee in Gwei. We recommend naming similar entities consistently.

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

L03. Code quality (fixed)

PPAgentV2 contract utilizes bitwise XOR at lines 437, 452, 593, 596, 599, 738, 741, and 744 just to update particular bits within an `uint256` value. We recommend using bitwise OR instead since it is a common practice and slightly less error-prone.

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

L04. Assembly usage (fixed)

The **PPAgentV2** contract includes multiple assembly blocks to save gas. However, many of them are ineffective and complicate the code. We recommend using assembly only when it is significantly more efficient than plain Solidity code.

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

L05. Event parameters indexing (fixed)

The `to` parameter of `Slash` event probably should be `indexed` at line 85 in **PPAgentV2.sol**.

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

L06. Short types (fixed)

The **PPAgentV2** contract utilizes shorter uint types for function and event parameters and local variables. In Particular:

- in `withdrawJobCredits` function;
- in `WithdrawJobCredits` and `RegisterJobParams` event.

It increases gas consumption and contract size without providing significant benefits. We recommend using `uint256` instead and reserve shorter types for tightly packed storage.

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

Notes

N01. Slashing logic (fixed)

Agents must provide CVP stake before joining the system. The project owner may slash the stake if the agent misbehaves. However, the stake is tokenized, allowing to circumvent this security measure.

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

N02. Integration caveat

The **PPAgentV2** contract adds an extra layer of security for the integrating projects. Keepers provide stake that may be slashed, only an EoA can execute a job, and the actual transactions should be submitted via flashbots. Therefore, the third party contracts may check that they are being called by the **PPAgentV2** contract. However, an attacker can create a malicious job that interacts with the same protocol but supplies modified data.

N03. Possible revert

The `execute_44g58pv` function of **PPAgentV2** contract may revert after executing the job. If this transaction is mined, the agent (keeper) pays the full price of transaction, but does not get the reward.

N04. UX improvement (fixed)

We recommend implementing "withdraw all" feature to mitigate possible race conditions between the job owner and agents in **PPAgentV2** contract:

1. in `withdrawJobCredits` function;
2. in `withdrawJobOwnerCredits` function.

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

This analysis was performed by Pessimistic:

Evgeny Marchenko, Senior Security Engineer
Ivan Gladkikh, Junior Security Engineer
Nikita Kirillov, Junior Security Engineer
Boris Nikashin, Analyst
Irina Vikhareva, Project Manager

August 11, 2022