POWERPOOL CVP VESTING SMART CONTRACT AUDIT

December 28, 2020

PowerPool

MixBytes()

CONTENTS

1. INTRODUCTION

1.1 DISCLAIMER

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of PowerPool (name of Client). If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 PROJECT OVERVIEW

PowerPool.Finance is a meta-governance protocol aimed at accumulating voting power in decentralized protocols and using it based on decisions of PowerPool (CVP) token holders.

1.3 SECURITY ASSESSMENT METHODOLOGY

At least 2 auditors are involved in the work on the audit who check the provided source code independently of each other in accordance with the methodology described below:

01 "Blind" audit includes:
> Manual code study
> "Reverse" research and study of the architecture of the code based on the source code only

Stage goal:
Building an independent view of the project's architecture
Finding logical flaws

02 Checking the code against the checklist of known vulnerabilities includes:
> Manual code check for vulnerabilities from the company's internal checklist
> The company's checklist is constantly updated based on the analysis of hacks, research and audit of the clients' code

Stage goal:
Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flashloan attacks, etc.)

03 Checking the logic, architecture of the security model for compliance with the desired model, which includes:
> Detailed study of the project documentation
> Examining contracts tests
> Examining comments in code
> Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of the reports from all auditors into one common interim report document
> Cross check: each auditor reviews the reports of the others
> Discussion of the found issues by the auditors
> Formation of a general (merged) report

Stage goal:
Re-check all the problems for relevance and correctness of the threat level
Provide the client with an interim report

05 Bug fixing & re-check.
> Client fixes or comments on every issue
> Upon completion of the bug fixing, the auditors double-check each fix and set the statuses with a link to the fix

Stage goal:
Preparation of the final code version with all the fixes

06 Preparation of the final audit report and delivery to the customer.

Findings discovered during the audit are classified as follows:

FINDINGS SEVERITY BREAKDOWN

Level	Description	Required action
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party	Immediate action to fix issue
Major	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.	Implement fix as soon as possible
Warning	Bugs that can break the intended contract logic or expose it to DoS attacks	Take into consideration and implement fix in certain period
Comment	Other issues and recommendations reported to/acknowledged by the team	Take into consideration

Based on the feedback received from the Customer's team regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The project team is aware of this finding. Recommendations for this finding are planned to be resolved in the future. This finding does not affect the overall safety of the project.
No issue	Finding does not affect the overall safety of the project and does not violate the logic of its work.

1.4 EXECUTIVE SUMMARY

The purpose of PowerPool Timed Vesting Contract is to provide a temporary lock and to follow it vesting for the CVP ERC20 tokens received by the participants in the Beta/Gamma testing rounds of PowerPool. The contract has a special function, which allows one to vote using tokens locked in the contract with an additional ability to delegate votes to any address. The audited scope's business logic was matched to the code reference located in here: https://github.com/powerpool-finance/powerpool-docs/blob/master/specifications/PPTimedVesting.md, documentation located in here: https://github.com/powerpool-finance/powerpool-docs and some application logic explanations found in here: https://gov.powerpool.finance and in here: https://medium.com/@powerpoolcvp.

1.5 PROJECT DASHBOARD

Client	PowerPool
Audit name	CVP Vesting
Initial version	254816cde016c060db4fec8968859213b59ec268
Final version	782e12f76dd67e201e5c1313284cb7d892728c47
SLOC	377
Date	2020-12-10 - 2020-12-28
Auditors engaged	2 auditors

FILES LISTING

PPTimedVesting.sol	PPTimedVesting.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	1
Warning	1
Comment	1

CONCLUSION

The audited scope revealed itself to contain several issues, one of them can formally be considered a major one, but it can be taken into action only after 1779 years, so fixing it is not a crucial thing. One more issue revealed itself to be classified as a warning leading to the potentially incorrect contract initialization. Other issues do not bring any harm. During the work on the issues all problems have been resolved and smart contract is assumed as safe to use according our security criteria.

2. FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

MJR-1	Block number type gets narrowed to uint32
File	PPTimedVesting.sol
Severity	Major
Status	Acknowledged

DESCRIPTION

This warning is about `block.number` value which is narrowed to `uint32` in here: PPTimedVesting.sol#L78 without any application logic necessity while the actual `block.number` type is `uint256`. This, first of all, leads to the time limitation of this particular contract which will remain valid. Secondly, this will lead to the contract failure right after the `block.number` exceeds the variable validity range according to the following statement: PPTimedVesting.sol#L560. This issue also makes several checks and comparisons useless because of the type used for `fromBlock`: PPTimedVesting.sol#L258, PPTimedVesting.sol#L270. These checks were initially introduced to bring additional insurance that the application logic works well, but since the `fromBlock` type is `uint32` and `block.number` is `uint256`, they will become useless after some time. Some statements, according to the types sizes differences, will become unreachable: PPTimedVesting.sol#L273

RECOMMENDATION

It is recommended to perform the application logic refactoring to avoid the contract failure earlier than it could take place.

CLIENT'S COMMENTARY

The use of the `uint32` for `block.number` value has initially been due to the Compound Protocol code, which we took as a basis for the `CVP` and `PPGovernorL1` contracts. The maximum value that the `fromBlock` variable can accept is 4,294,967,295. At the current block time, it will take 1,779 years to reach this value.

2.3 WARNING

WRN-1	Initialization potential overflow
File	PPTimedVesting.sol
Severity	Warning
Status	Fixed at PR-12

DESCRIPTION

This warning is about the initialization of vesting end timestamp in the constructor in here:
PPTimedVesting.sol#L156
and in here:
PPTimedVesting.sol#L152.

Despite the contract is `Ownable`, the `_startT`, `_startV`, `_durationT`, `_durationV` parameters are passed incorrectly. It can lead to the overflowed `endT` and `endV` result values which can only be fixed with re-initialization. Overflowed `endT` and `endV` values will lead to the overall application logic failure.

RECOMMENDATION

It is recommended to use `SafeMath`-based functions for for `_startV + _durationV` and `_startT + _durationT`-like statements in the constructor to avoid faulty initialization.

2.4 COMMENTS

CMT-1	Debug-only functions in the production code
File	PPTimedVesting.sol
Severity	Comment
Status	Fixed at PR-12

DESCRIPTION

This comment is about debug-only functions still present inside the production release in here: PPTimedVesting.sol#L222

RECOMMENDATION

Despite this particular issue, the function brings no previously unknown data to an attacker, it is recommended to remove all the debug-related functionality before the production release.

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build open-source solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

BLOCKCHAINS

 Ethereum

 Cosmos

 EOS

 Substrate

TECH STACK

 Python

 Solidity

 Rust

 C++

CONTACTS

 https://github.com/mixbytes/audits_public

 https://mixbytes.io/

 hello@mixbytes.io

 https://t.me/MixBytes

 https://twitter.com/mixbytes