Push Protocol Smart Contracts Review

By: ChainSafe Systems

May 2023

Push Protocol Smart Contracts Review

Auditors: Anderson Lee, Tanya Bushenyova, Oleksii Matiiasevych

WARRANTY

This Code Review is provided on an “as is” basis, without warranty of any kind, express or implied. It is not intended to provide legal advice, and any information, assessments, summaries, or recommendations are provided only for convenience (each, and collectively a “recommendation”). Recommendations are not intended to be comprehensive or applicable in all situations. ChainSafe Systems does not guarantee that the Code Review will identify all instances of security vulnerabilities or other related issues.

Introduction

Push Protocol requested ChainSafe Systems to perform a review of the EPNS Protocol 2.0 smart contracts. The contracts can be identified by the following git commit hashes:

`0c62b7a377dbb010cd19288f6d8974773e51326f`

There are 3 smart contracts in scope, specifically EPNSCoreV2, EPNSCoreStorageV2, EPNSCommV2. After the initial review, Push Protocol team applied a number of updates which can be identified by the following git commit hashes:

`993130c48211a83fd2b19fb082f2e25f91fa04cc`

Additional verification was performed after that.

Disclaimer

The review makes no statements or warranties about the utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about the fitness of the contracts for any specific purpose, or their bug free status.

Executive Summary

There are **no** known compiler bugs for the specified compiler version (0.6.11), that might affect the contracts' logic.

There were **3 critical**, 0 major, 0 minor, 11 informational/optimizational issues identified in the initial version of the contracts. All issues found in the contracts were not present in the final verified version of the contracts. They are described below for historical purposes. The majority of changes are related to the introduction of staking functionality in the EPNS Core contract. The distribution is split into the epochs during which users could stake and rewards could be accumulated. There are some special properties of staking mechanics, such as no matter if the user stakes in the beginning or the end of the epoch, they would be treated as if they had staked in the beginning. Users are not allowed to unstake (withdraw) until the whole epoch has passed since their previous stake.

Critical Bugs and Vulnerabilities

Three critical issues were found in the EPNSCoreV2 contract that could allow draining of all rewards from the contract (5.6, 5.8, 5.9). The issues were addressed and are not present in the final verified version of the contract.

Line by Line Review. Fixed Issues

1. EPNSCommV2, line 791. Optimization, the `createIncentivizeChatRequest()` function will overwrite `chatData.requestSender` variable with the same value on every call after the first one.

2. EPNSCommV2, line 792. Note, the `createIncentivizeChatRequest()` function will overwrite `chatData.requestReceiver`.

3. EPNSCoreV2, line 988. Note, the `lastEpochRelative()` function has a require condition reason typo, should be `Blocknumber` instead of `Blocnumber`.

4. EPNSCoreV2, line 1064. Optimization, the `_stake()` function could rewrite the `userFeesInfo[_staker].lastClaimedBlock` with the same value if it is already set.

5. EPNSCoreV2, line 1123. Note, the `harvestTill()` logic is severely different from `harvestPaginated` and `daoHarvestPaginated` even on the full range of epochs.

6. EPNSCoreV2, line 1129. **Critical**, the `harvestTill()` function allows setting `lastClaimedBlock` to value that would allow draining of all the rewards.

7. EPNSCoreV2, line 1162. Note, the `harvestPaginated()` function has a misleading comment about the `lastClaimedBlock` value update. It sets it at the beginning of an `_endepoch`, not the end.

8. EPNSCoreV2, line 1166. **Critical**, the `harvestPaginated()` function allows setting `lastClaimedBlock` to value that would allow draining of all the rewards.

9. EPNSCoreV2, line 1216. **Critical**, the `daoHarvestPaginated()` function allows setting `lastClaimedBlock` to value that would allow draining of all the rewards.

10. EPNSCoreV2, line 1247. Optimization, `_adjustUserAndTotalStake()` function reads `userFeesInfo[_user].stakedWeight` multiple times from storage.

11. EPNSCoreV2, line 1269. Note, the `_adjustUserAndTotalStake()` function has duplicated codes in 'if' and 'else' case. Could only make the if statement checking `'i == currentEpoch - 1'` and update `'userFeesInfo[_user].stakedWeight'` in the if statement.

12. EPNSCoreV2, line 1358. Note, the `setRelayerAddress()` function could emit an event.

13. EPNSCoreV2, line 1362. Note, the `setBridgeAddress()` function could emit an event.

14. EPNSCoreStorageV2, line 31. Optimization, the `epochDuration` could be made constant as it is set only once to a fixed value.

Anderson Lee

Tanya Bushenyova

Oleksii Matiiasevych