StakeAll Campaign Feature Audit Report

September 28, 2022

Shard Labs

Disclaimer

The audit makes no statements or warranties about the 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 the 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 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.

Severity Level Reference

Findings discovered during the audit are classified as follows: Every issue in this report was assigned a severity level from the following:

CRITICAL:

A bug leading to assets theft, fund access locking, or any other loss of funds due to transfer to unauthorized parties.

MAJOR:

A bug that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.

WARNING:

A bug that can break the intended contract logic or expose it to DDoS attacks.

INFO:

Minor issue or recommendation reported to / acknowledged by the client's team.

Security Assessment Methodology

A group of auditors is involved in the work on this audit. Each of them checks the provided source code independently of each other in accordance with the security assessment methodology described below:

1. Project architecture review:

Manually code study of the architecture of the code based on the source code only to find out the errors and bugs.

2. Check the code against the list of known vulnerabilities

The verification process of the code against the constantly updated list of already known vulnerabilities maintained by the company.

3. Architecture and structure check of the security model

Study project documentation and its comparison against the code, including the study of the comments and other technical papers.

4. Result's cross-check by different auditors

Normally the research of the project is made by more than two auditors. After that, there is a step of the mutual cross-check process of audit results between different task performers.

5. Report consolidation

Consolidation of the audited report from multiple auditors.

Status Level Reference

Based on the feedback received from the client's team regarding the list of findings discovered by the contractor, the following statuses were assigned to the findings:

NEW:

Waiting for the project team's feedback.

FIXED:

Recommended fixes have been made to the project code, and the identified issue no longer affects the project's security.

ACKNOWLEDGED:

The project team is aware of this finding. Recommended fixes for this finding are planned to be made. This finding does not affect the overall security of the project.

NO ISSUE:

Finding does not affect the overall security of the project and does not violate the logic of its work.

DISMISSED:

The issue or recommendation was dismissed by the client.

Project overview

More detail about the project here

Audit Scope

https://github.com/stakeall/cross-chain-staking-shuttle/pull/30 Settings Refresh

Audit Comments for hash: 18c2738b5373440dc49fcd68f2779302d20d149c

Report

CRITICAL:

None

MAJOR

None

WARNING

None

INFO

1. **Use of the SafeMath** `Campaign.sol`

The use of the SafeMath library on line 15 is not necessary as of solidity v0.8.7 as the runtime checks for overflows and underflows during the execution of arithmetic operations.

Issue: New

Status: Fixed

2. Reloading storage `Campaign.sol`

Avoid reloading storage variables at 70, 71, and 82 and cache currentCampaign value in memory instead.

Issue: New

Status: Fixed

3. Use custom errors `Campaign.sol`

Use custom errors in place of `require` on lines 102, 106, to reduce gas consumption https://blog.soliditylang.org/2021/04/21/custom-errors/

Issue: New

Status: Fixed

4. Extract onlyRole modifier `Campaign.sol`

Extract `onlyRole` modifier on lines 66, 121, 135, 150, 209 into an internal function to avoid duplicating modifier in all functions it's called in during compile time.

Issue: New

Status: Fixed

5. Use interface keyword `ICampaign.sol`

On line 6 declare as `interface` instead of `contract`

Issue: New

Status: Fixed

6. Index event parameters `ICampaign.sol`

In ICampaign.sol Index the most relevant event parameters for easy log filtering

Issue: New

Status: Acknowledge

7. Use custom errors `ChildPool.sol`

Use custom errors in place of `require` to reduce gas consumption at 484.
https://blog.soliditylang.org/2021/04/21/custom-errors/

Issue: New

Status: Fixed

8. Reloading storage `ChildPool.sol`

Avoid reloading storage variables and reuse function parameter in event at 488.

Issue: New

Status: Fixed

9. Index event parameters `IChildPool.sol`

Index event parameter at line 42.

Issue: New

Status: Acknowledge

Conclusion

The following table contains the total number of issues that were found during audit:

Level	Amount
CRITICAL	0
MAJOR	0
WARNING	0

INFO	**9**
Total	**9**