CoinFabrik

Security Audit Report

ALEX - Farming Campaign

January 2025

V202411

Executive Summary

CoinFabrik was asked to audit the contracts for the **Farming Campaign** project.

The farming campaign is a blockchain-based initiative aimed at boosting ecosystem engagement through a structured, three-phase process: Registration, Voting/Staking, and Reward Emissions. During registration, anyone can enroll pools for the campaign and add project token rewards. In the voting and staking phase, participants vote on and stake tokens in their preferred pools, using tokens like ALEX and LiALEX to increase their influence. The final phase involves distributing rewards based on voting outcomes

During this audit we found two low issues. Also, an enhancement was proposed.

The findings were acknowledged by the development team.

Scope

The audited files are from the git repository located at https://github.com/alexgo-io/alex-dao-2, in the **`contracts/extensions/`** directory. The audit is based on the commit **`64315266eec389c399a172c8ebeebe84dc29555d`**.

The scope for this audit includes and is limited to the following files:

- **`farming-campaign-v2-02.clar`**: Main contract for the campaign.

No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit.

Findings

In the following table we summarize the security issues we found in this audit. The severity classification criteria and the status meaning are explained below. This table does not include the enhancements we suggest to implement, which are described in a specific section after the security issues.

Each severity label is detailed in the Severity Classification section. Additionally, the statuses are explained in the Issues Status section.

Id	Title	Severity	Status
LO-01	Duplicated Error Code	🟨 Low	**Acknowledged**

Id	Title	Severity	Status
LO-02	Stake Accumulation in Reused Campaigns	🟨 Low	**Acknowledged**

Critical Severity Issues

No issues found.

High Severity Issues

No issues found.

Medium Severity Issues

No issues found.

Low Severity Issues

LO-01 Duplicated Error Code

Location

- `farming-campaign-v2-02.clar: 10, 16`

Classification

- CWE-710: Improper Adherence to Coding Standards[1]

Description

The contract contains instances of duplicated error codes, specifically `err-stake-cutoff-passed` and `err-stake-end-passed`, both assigned the same error code `(err u1005)`. This duplication can lead to issues in error handling, as it obscures the distinction between different errors that occur at distinct points in the lifecycle of a campaign process. When multiple errors share the same code, it becomes challenging to accurately identify and address specific conditions or bugs during debugging and log analysis.

Recommendation

Assign a different error code to one of the errors.

[1] https://cwe.mitre.org/data/definitions/710.html

Status

Acknowledged.

LO-02 Stake Accumulation in Reused Campaigns

Location

- `farming-campaign-v2-02.clar`

Description

The current implementation allows for the possibility of reusing a campaign by updating its details (e.g., `registration-cutoff`, `stake-cutoff`). However, during such a reuse scenario, users who have already staked and claimed rewards in a previous instance of the same campaign can stake additional tokens in the new campaign round. This inadvertently allows users to accumulate stakes over their previously claimed amounts, potentially leading to an incorrect LP and reward token distribution.

Recommendation

Set the staking amount to zero when users unstake.

Status

Acknowledged.

Enhancements

These items do not represent a security risk. They are best practices that we suggest implementing.

Id	Title	Status
EN-01	Remove Unused Code	Not implemented

EN-01 Remove Unused Code

Location

- `farming-campaign-v2-02.clar: 269, 305, 312, 314`

Description

The contracts implement utility and mathematical functions which are not used:

- `max`
- `min`
- `check-err`

Also, `campaign-details` binding in the `update-campaign` function is never used.

Recommendation

Remove unused code.

Status

Not implemented.

Other Considerations

The considerations stated in this section are not right or wrong. We do not suggest any action to fix them. But we consider that they may be of interest to other stakeholders of the project, including users of the audited contracts, token holders or project investors.

Centralization

Various functions require the DAO or an approved extension to be the caller, allowing it to set or update campaigns data (e.g., creating campaigns, whitelisting pools, or setting the total reward amount). The DAO can transfer any token out of the contract.

Upgrades

The contracts do not implement upgradeability mechanisms.

Divergences from Documentation

There is a functional divergence in that the documentation[2] claims voting is limited to once per round, while the code allows repeated voting calls. Practically, the code only restricts total votes by checking new total votes do not exceed the voting power.

[2] https://docs.alexlab.co/~/changes/AqC2XfgksqVO112x20Xr/resources/alex-events/alex-surge

The documentation references a ~27–28 day reward emission, but it is not enforced in the code; the actual times come from `registration-cutoff`, `voting-cutoff`, `stake-cutoff`, and `stake-end` configured by the DAO.

The documentation's mention of a "social leaderboard" for foundation voting is handled off-chain and then applied calling `vote-campaign` with the foundation's 1,000,000 voting power.

About CoinFabrik

CoinFabrik is a research and development company specialized in Web3, with a strong background in cybersecurity. Founded in 2014, we have worked on over 500 decentralization projects, including EVM-based and other platforms like Solana, Algorand, and Polkadot. Beyond development, we offer security audits through a dedicated in-house team of senior cybersecurity professionals, working on code in languages such as Substrate, Solidity, Clarity, Rust, TEAL, and Stellar Soroban.

Our team has an academic background in computer science, software engineering, and mathematics, with accomplishments including academic publications, patents turned into products, and conference presentations. We actively research in collaboration with universities worldwide, such as Cornell, UCLA, and École Polytechnique in Paris, and maintain an ongoing collaboration on knowledge transfer and open-source projects with the University of Buenos Aires, Argentina. Our management and people experience team has extensive expertise in the field.

Methodology

CoinFabrik was provided with the source code, and general documentation about the project. Our auditors spent two weeks auditing the source code provided, which includes understanding the context of use, analyzing the boundaries of the expected behavior of each contract and function, understanding the implementation by the development team (including dependencies beyond the scope to be audited) and identifying possible situations in which the code allows the caller to reach a state that exposes some vulnerability. Without being limited to them, the audit process included the following analyses.

- Arithmetic errors
- Race conditions
- Misuse of block timestamps
- Denial of service attacks
- Excessive runtime usage
- Missing or misused function qualifiers

- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures
- Centralization and upgradeability

After delivering a report with our findings, the development team had the opportunity to comment on every finding and fix the issues they considered convenient. Once fixed and/or commented, our team ran a second review process to verify that the changes to the code effectively solve the issues found and do not unintentionally add new ones. This report includes the final status after the second review.

Severity Classification

Security risks are classified as follows[3]:

🟥 Critical	• Manipulation of governance voting result deviating from voted outcome and resulting in a direct change from intended effect of original results • Direct theft of any user funds, whether at-rest or in-motion, other than unclaimed yield • Direct theft of any user NFTs, whether at-rest or in-motion, other than unclaimed royalties • Permanent freezing of funds • Permanent freezing of NFTs • Unauthorized minting of NFTs • Predictable or manipulable RNG that results in abuse of the principal or NFT • Unintended alteration of what the NFT represents (e.g. token URI, payload, artistic content) • Protocol insolvency
🟧 High	• Theft of unclaimed yield • Theft of unclaimed royalties • Permanent freezing of unclaimed yield • Permanent freezing of unclaimed royalties • Temporary freezing of funds • Temporary freezing NFTs

[3] This classification is based on the smart contract Immunefi severity classification system version 2.3. https://immunefi.com/immunefi-vulnerability-severity-classification-system-v2-3/

Medium	• Smart contract unable to operate due to lack of token funds • Block stuffing • Griefing (e.g. no profit motive for an attacker, but damage to the users or the protocol) • Theft of gas • Unbounded gas consumption • Security best practices not followed
Low	• Contract fails to deliver promised returns, but doesn't lose value • Other security issues with minor impact

Issue Status

An issue detected by this audit has one of the following statuses:

- **Unresolved:** The issue has not been resolved.
- **Resolved:** Adjusted program implementation to eliminate the risk.
- **Partially Resolved:** Adjusted program implementation to eliminate part of the risk. The other part remains in the code, but is a result of an intentional decision.
- **Acknowledged:** The issue remains in the code, but is a result of an intentional decision. The reported risk is accepted by the development team.
- **Mitigated:** Implemented actions to minimize the impact or likelihood of the risk.

Disclaimer

This audit report has been conducted on a **best-effort basis within a tight deadline defined by time and budget constraints**. We reviewed only the specific smart contract code provided by the client at the time of the audit, detailed in the Scope section. We do not review other components that are part of the solution: neither implementation, nor general design, nor business ideas that motivate them.

While we have employed the latest tools, techniques, and methodologies to identify potential vulnerabilities, **this report does not guarantee the absolute security of the contracts, as undiscovered vulnerabilities may still exist.** Our findings and recommendations are suggestions to enhance security and functionality and are not obligations for the client to implement.

The results of this audit are valid solely for the code and configurations reviewed, and any modifications made after the audit are outside the scope of our responsibility. CoinFabrik disclaims all liability for any damages, losses, or legal consequences resulting from the use or misuse of the smart contracts, including those arising from undiscovered vulnerabilities or changes made to the codebase after the audit.

This report is intended exclusively for the **ALEX** team and should not be relied upon by any third party without the explicit consent of CoinFabrik. Blockchain technology and smart contracts are inherently experimental and involve significant risk; users and investors should fully understand these risks before deploying or interacting with the audited contracts.

Changelog

Date	Description
2025-01-03	Initial report based on commit `64315266eec389c399a172c8ebeebe84dc29555d`.
2025-01-06	Final report after development team's feedback.