Security Audit Report

Nudge Campaigns

v1.1

March 7, 2025

Table of Contents

License

Disclaimer

This audit has been performed by

Oak Security GmbH

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security GmbH has been engaged by DeFi Labs GmbH to perform a security audit of the Nudge Campaign smart contracts.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following targets:

Repository	https://github.com/violetprotocol/nudge-demo
Commit	`1f264013adb17d1d794533b00024365ddc99ba6d`
Scope	Only the Solidity contracts in the `contracts/src/campaign` directory were in the scope of this audit.
Fixes verified at commit	`2c05a4ac17530de1cfb0fcc990ca56d1fdea2cef` Note that only fixes to the issues described in this report have been reviewed at this commit. Any further changes such as additional features have not been reviewed.

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line-by-line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

Nudge is a Reallocation Marketplace where protocols incentivize users to move their assets across blockchains and ecosystems. Campaigns can be created to reward users for reallocating funds with either native tokens (ETH), ERC-20 tokens, or off-chain points.

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending, Acknowledged**, **Partially Resolved**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Code Quality Criteria

The auditor team assesses the codebase's code quality criteria as follows:

Criteria	Status	Comment
Code complexity	**Low-Medium**	-
Code readability and clarity	**High**	-
Level of documentation	**High**	The client provided detailed documentation outlining the specifications of the intended contract behavior.
Test coverage	**High**	The contracts in the `nudge-demo` repository have a test line coverage of `99.2%`. Notably, a few tests in the repository failed and had to be fixed before determining the test coverage.

Summary of Findings

No	Description	Severity	Status
1	Noneffective increment of the participation ID leads to overwriting participation details	Major	Resolved
2	Deploying and funding a new campaign with `msg.value` that is greater than `initialRewardAmount` locks native tokens in the factory contract	Minor	Resolved
3	Accidental ETH transfers are possible when ERC-20 tokens are expected to be transferred	Minor	Resolved
4	An existing points campaign can be overwritten by the admin	Minor	Resolved
5	Campaign participation can be invalidated multiple times resulting in incorrect reward allocation	Minor	Resolved
6	The `CAMPAIGN_ADMIN_ROLE` role cannot be managed due to missing role admin	Minor	Acknowledged
7	Inconsistent usage of `rewardFactor` and `rewardBps`	Minor	Resolved
8	Permission conflict regarding campaign activation	Minor	Acknowledged
9	The campaign reward fee can be set to 100%	Informational	Acknowledged
10	`NudgeCampaign` start timestamp set to the current block timestamp still requires manual activation by the admin	Informational	Resolved
11	Updating the protocol fee lacks emitting an event for off-chain monitoring	Informational	Resolved
12	Different campaign IDs can be used with the same campaign	Informational	Resolved
13	Miscellaneous comments	Informational	Acknowledged

Detailed Findings

1. Noneffective increment of the participation ID leads to overwriting participation details

Severity: Major

Within the `handleReallocation` function of the `NudgePointsCampaigns` contract, a campaign's participation counter `pID` is incremented in `src/campaign/NudgePointsCampaigns.sol:152-154` to facilitate consecutive IDs for campaign participations.

However, the campaign's `pID` is reset to its previous value by the `newpID` variable after incremenation effectively causing the `pID` to always remain `0` and subsequently overwriting a campaign's participation at index 0.

Recommendation

We recommend replacing the post-increment of `pID` with a pre-increment such that the first participation starts at index `1`, which is consistent with the `NudgeCampaign` contract. Furthermore, the overwrite of the `pID` by the `newID` variable is recommended to be removed.

Status: Resolved

2. Deploying and funding a new campaign with `msg.value` that is greater than `initialRewardAmount` locks native tokens in the factory contract

Severity: Minor

In `src/campaign/NudgeCampaignFactory.sol:132-176`, the `deployAndFundCampaign` function deploys and funds a new campaign with either native tokens, i.e., ETH or ERC-20 tokens. In the case of native tokens, `initialRewardAmount` is sent to the campaign contract via a low-level `call`.

However, `msg.value` can be larger than `initialRewardAmount`, as the function only validates that `msg.value` is equal to or greater than `initialRewardAmount`. This can lead to locking native tokens in the factory contract as there is no recovery mechanism.

Recommendation

We recommend transferring `msg.value` instead of `initialRewardAmount` to avoid locking funds in the factory contract.

Status: Resolved

3. Accidental ETH transfers are possible when ERC-20 tokens are expected to be transferred

Severity: Minor

Both native tokens, i.e., ETH and ERC-20 tokens, can be used with the contracts in the scope of the review. However, in all instances listed below, it is not verified that `msg.value` is 0 when ERC-20 tokens are involved. This can lead to accidental ETH transfers, resulting in the native tokens being locked in the contracts.

- `src/campaign/NudgeCampaignFactory.sol:164-172`
- `src/campaign/NudgePointsCampaigns.sol:135-141`
- `src/campaign/NudgeCampaign.sol:175-181`

Recommendation

We recommend verifying that `msg.value` is 0 when ERC-20 tokens are expected to be transferred.

Status: Resolved

4. An existing points campaign can be overwritten by the admin

Severity: Minor

The Nudge admin can create multiple points campaigns in a single transaction using the `createPointsCampaigns` function in `src/campaign/NudgePointsCampaigns.sol:82-109`.

However, the function lacks a check to prevent overwriting an existing campaign with an already existing `campaignId`. As a result, the admin can accidentally overwrite an existing campaign and its values, causing issues with the off-chain points rewarding system and potentially calculating incorrect rewards.

Recommendation

We recommend adding a check in the `createPointsCampaigns` function to prevent overwriting an existing campaign, similar to the check in lines `59-61` of the `createPointsCampaign` function.

Status: Resolved

5. Campaign participation can be invalidated multiple times resulting in incorrect reward allocation

Severity: Minor

The `invalidateParticipations` function in `src/campaign/NudgeCampaign.sol:272-281` allows the Nudge operator to invalidate specific campaign participations. For example, if a user has not held the required tokens for the required period.

However, the function does not check whether the participation status is already set to `INVALIDATED`. This would allow the operator to repeatedly invalidate the same participation, which leads to incorrectly decreasing the `pendingRewards` value. Consequently, it allows the withdrawal of seemingly unallocated rewards, which are supposed to be allocated to the user.

Recommendation

We recommend verifying that the participation status is set to `PARTICIPATING` before invalidating the participation to prevent repeated invalidations.

Status: Resolved

6. The `CAMPAIGN_ADMIN_ROLE` role cannot be managed due to missing role admin

Severity: Minor

In `src/campaign/NudgeCampaign.sol:94`, the custom `CAMPAIGN_ADMIN_ROLE` role is granted to the `campaignAdmin` address, which is then permitted to withdraw unallocated campaign rewards from the contract.

However, to actively manage this custom role, for example, to grant or revoke it, either an explicitly set role admin or the `DEFAULT_ADMIN_ROLE` is required. As neither is set, the `CAMPAIGN_ADMIN_ROLE` role cannot be managed.

Recommendation

We recommend granting the `DEFAULT_ADMIN_ROLE` role to a trusted address to manage the `CAMPAIGN_ADMIN_ROLE` role.

Status: Acknowledged

7. Inconsistent usage of `rewardFactor` and `rewardBps`

Severity: Minor

In the constructor of the `NudgeCampaign` contract, the `rewardBps` parameter in `src/campaign/NudgeCampaign.sol:67` is expected to be denominated in base points.

However, the `NudgeCampaignFactory` contract provides the `rewardFactor` parameter instead in `src/campaign/NudgeCampaignFactory.sol:102`, which is denominated in units of `targetToken` for `100` units of `targetToken` reallocated.

This inconsistency can lead to severe misconfiguration of a campaign in terms of reward issuance.

Recommendation

We recommend consistent usage of `rewardBps` in accordance with the `NudgeCampaign` contract.

Status: Resolved

8. Permission conflict regarding campaign activation

Severity: Minor

Anyone can create a new campaign using the `NudgeCampaignFactory` contract, which is immediately activated in case `startTimestamp == 0`.

However, if a custom non-zero `startTimestamp` is specified, the campaign has to be manually activated using the `setIsCampaignActive` function defined in `src/campaign/NudgeCampaign.sol:313-325`, which can only be called by an account that was granted the `NUDGE_ADMIN_ROLE`.

Recommendation

We recommend allowing permissionless activation of a campaign as soon as the `startTimestamp` is reached. At least, campaign activation should be possible by the campaign admin, see `CAMPAIGN_ADMIN_ROLE`.

Status: Acknowledged

9. The campaign reward fee can be set to 100%

Severity: Informational

The `updateFeeSetting` function in `src/campaign/NudgeCampaignFactory.sol:246-250` allows the admin to update the fee in basis points for future campaigns created by the `NudgeCampaignFactory`.

However, the upper bound of 100%, which restricts the new fee `newFeeBps` value, would allow a fee of 100% to be set, effectively transferring all of the rewards to the treasury. This might lead to trust and reputation issues.

Recommendation

We recommend adding a reasonable upper bound for the fee in basis points. For example, a cap of 10% might be suitable.

Status: Acknowledged

10. `NudgeCampaign` start timestamp set to the current block timestamp still requires manual activation by the admin

Severity: Informational

The constructor of the `NudgeCampaign` contract verifies in `src/campaign/NudgeCampaign.sol:77-79` that the given `startTimestamp_` is set to a future timestamp if it is non-zero. Otherwise, it errors. In this case, the campaign requires manual activation by the Nudge admin.

However, `startTimestamp_` can be set to the current block timestamp, suggesting the campaign can start immediately. Counterintuitively, it still requires manual activation.

Recommendation

We recommend erroring if `startTimestamp_ <= block.timestamp` to prevent such a configuration.

Status: Resolved

11. Updating the protocol fee lacks emitting an event for off-chain monitoring

Severity: Informational

The `updateFeeSetting` function in `src/campaign/NudgeCampaignFactory.sol:246-250` allows the admin to update the fee in basis points for future campaigns created by the `NudgeCampaignFactory`.

However, the function does not emit an event after updating the fee, preventing off-chain monitoring of the fee changes.

Recommendation

We recommend emitting an event with the new fee value.

Status: Resolved

12. Different campaign IDs can be used with the same campaign

Severity: Informational

The `handleReallocation` function of the `NudeCampaign` contract accepts a `campaignId` parameter in `src/campaign/NudgeCampaign.sol:153`, which is not subject to any subsequent validation and is currently utilized for event emission.

However, this allows to (accidentally) use different campaign IDs with the same campaign, leading to potential issues during off-chain event handling.

Recommendation

We recommend assigning an individual campaign ID to each `NudeCampaign` instance during construction and storing it in the contract for subsequent usage and validation purposes.

Status: Resolved

13. Miscellaneous comments

Severity: Informational

Miscellaneous recommendations can be found below.

Recommendation

The following are some recommendations to improve the overall code quality and functionality:

1. The `NudgeCampaign` constructor calculates the scaling factors in `src/campaign/NudgeCampaign.sol:87-88` to normalize token amounts to 18 decimals by subtracting the token decimals from 18. However, this would lead to an underflow error for ERC-20 tokens with more than 18 decimals, preventing the use of such tokens.
2. Campaign IDs can (accidentally) collide among regular and points-only campaigns. It is recommended to centrally handle the allocation of individual IDs for both campaign types, e.g., using the `NudgeCampaignFactory` contract.
3. When obtaining the `availableBalance` for reward claiming in `src/campaign/NudgeCampaign.sol:225`, it is recommended to subtract the `accumulatedFees` storage variable.

Status: Acknowledged