Security Audit Report

Push Protocol – Comm Cairo

v1.0

November 29, 2024

Table of Contents

License

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

THIS AUDIT REPORT WAS PREPARED EXCLUSIVELY FOR AND IN THE INTEREST OF THE CLIENT AND SHALL NOT CONSTRUE ANY LEGAL RELATIONSHIP TOWARDS THIRD PARTIES. IN PARTICULAR, THE AUTHOR AND HIS EMPLOYER UNDERTAKE NO LIABILITY OR RESPONSIBILITY TOWARDS THIRD PARTIES AND PROVIDE NO WARRANTIES REGARDING THE FACTUAL ACCURACY OR COMPLETENESS OF THE AUDIT REPORT.

FOR THE AVOIDANCE OF DOUBT, NOTHING CONTAINED IN THIS AUDIT REPORT SHALL BE CONSTRUED TO IMPOSE ADDITIONAL OBLIGATIONS ON COMPANY, INCLUDING WITHOUT LIMITATION WARRANTIES OR LIABILITIES.

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 Push Comm Ltd to perform a security audit of Push Protocol – Comm Cairo.

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 target:

Repository	https://github.com/push-protocol/push-comm-cairo
Commit	`e5ff0181da0aa4b2a844fca8a2e8826ab976bff7`
Scope	All contracts were in scope.
Fixes verified at commit	`b997970bc395cde591668572ada434c410be76f5` 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

Push Protocol is a Web3 communication protocol that enables any dApps, smart contracts, backends, or protocols to communicate both on-chain and off-chain via user wallet addresses in an open, gasless, multichain, and platform-agnostic fashion.

The Push Comm Cairo contract includes features that allow users to subscribe to a channel and unsubscribe from a channel as well as sending notifications as a channel's delegate.

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	Medium	The code is mostly self-explanatory but lacks code comments providing function documentation and explaining business logic.
Level of documentation	Medium-High	The documentation describes the fundamental flow and functionality of the program with some technical details.
Test coverage	Medium-High	Comprehensive unit test coverage, but lack of integration testing with off-chain components.

Summary of Findings

No	Description	Severity	Status
1	Incorrect value emitted in `UserNotifcationSettingsAdded` event	Minor	Resolved
2	Notifications settings are not cleared when a user unsubscribes from a channel	Minor	Acknowledged
3	Delegates are not automatically subscribed to a channel upon designation	Minor	Acknowledged
4	Channel alias verification will not be possible in case of hard fork	Minor	Resolved
5	Inconsistency in authorization logic between EVM and Cairo contracts	Minor	Acknowledged
6	Inconsistency in return values between EVM and Cairo contracts	Minor	Acknowledged
7	Usage of experimental OpenZeppelin library version	Minor	Resolved
8	Usage of outdated and experimental Scarb toolchain version	Minor	Resolved
9	Lack of delegate existence verification before removal	Informational	Resolved
10	Potential transaction reverts due to event data size limit	Informational	Resolved
11	Limited upgrade flexibility	Informational	Resolved
12	Miscellaneous comments	Informational	Resolved

Detailed Findings

1. Incorrect value emitted in `UserNotifcationSettingsAdded` event

Severity: Minor

In `src/lib.cairo:331`, the `change_user_channel_settings` function emits a `UserNotifcationSettingsAdded` event. This event includes a `notif_settings` parameter that is intended to provide the notification settings for a specific channel and user.

However, the function incorrectly emits the `notif_settings` value instead of the updated `modified_notif_settings` value, which incorporates the `notif_id`. As a consequence, the event does not accurately reflect the changes made to the user's notification settings.

Recommendation

We recommend modifying the `change_user_channel_settings` function to emit the `modified_notif_settings` value in the `UserNotifcationSettingsAdded` event. This will ensure that the event accurately reflects the updated notification settings.

Status: Resolved

2. Notifications settings are not cleared when a user unsubscribes from a channel

Severity: Minor

In `src/lib.cairo:320-323`, the `user_to_channel_notifs` mapping is written within the `change_user_channel_settings` method according to the user's specified settings.

However, when a user unsubscribes from a communication channel, the corresponding notification settings are never cleared from the `user_to_channel_notifs` mapping.

Consequently, when a user unsubscribes from a channel and re-subscribes at a later point in time, the prior notification settings still persist which might be unexpected.

Recommendation

We recommend clearing the corresponding notification settings from the `user_to_channel_notifs` mapping when a user unsubscribes, i.e. restore the defaults.

Status: Acknowledged

3. Delegates are not automatically subscribed to a channel upon designation

Severity: Minor

In `src/lib.cairo:349-353`, the `add_delegate` method enables channel owners to add delegates who are granted the authority to send notifications on behalf of the channel.

According to the documentation:
"Delegates are automatically subscribed to the channel upon designation but remain subscribed if their delegate status is removed."

However, the corresponding functionality is not implemented and delegates are not automatically subscribed.

Recommendation

We recommend implementing the missing auto-subscription functionality according to the documentation.

Status: Acknowledged

4. Channel alias verification will not be possible in case of hard fork

Severity: Minor

The `verify_channel_alias` function emits a `ChannelAlias` event that includes the `chain_id`, which is hardcoded in the constructor. In the event of a hard fork, this hardcoded value will then differ from the actual chain ID, causing issues for off-chain systems relying on this event.

Consequently, any off-chain system relying on this event to link an Ethereum channel address to a Starknet channel address will break, as it will associate the channel with the incorrect chain.

Recommendation

We recommend removing the hardcoded `chain_id` variable and modifying the `verify_channel_alias` function to fetch the current `chain_id` using `get_execution_info` to be used in the `ChannelAlias` event. This ensures the event always contains the correct chain ID, even after a hard fork.

Status: Resolved

5. Inconsistency in authorization logic between EVM and Cairo contracts

Severity: Minor

The Cairo version of the `_checkNotifReq` function is missing the `pushChannelAdmin` authorization logic present in the EVM version. This discrepancy could lead to inconsistent behavior, as the Solidity version grants the `pushChannelAdmin` special privileges to send notifications, while the Cairo version does not.

Recommendation

We recommend implementing one of the following:

- Align the Cairo version's functionality by adding the `pushChannelAdmin` authorization logic. This might involve introducing a `push_channel_admin` variable in the contract's state and updating the `_check_notif_req` function accordingly.
- Document the intentional difference in authorization logic between the EVM and Cairo versions, clearly outlining the specific privileges of the `pushChannelAdmin` in each.

Status: Acknowledged

6. Inconsistency in return values between EVM and Cairo contracts

Severity: Minor

The EVM and Cairo contracts have inconsistent return values for the `_add_user`, `batch_subscribe`, and `batch_unsubscribe` functions. The EVM versions return `true`, while the Cairo versions do not return any value. While not a security concern currently, this discrepancy could lead to confusion or unexpected behavior when interacting with the contracts.

Recommendation

We recommend either aligning the Cairo contract with the EVM contract by returning `bool` for these functions or documenting the reasons for the differences in return values.

Status: Acknowledged

7. Usage of experimental OpenZeppelin library version

Severity: Minor

The project depends on an experimental release candidate version of OpenZeppelin's `cairo-contracts` library, i.e. `v0.15.0-rc.0`, which contains the following security warning:
"This project is still in a very early and experimental phase. It has never been audited nor thoroughly reviewed for security vulnerabilities. Do not use in production."

Consequently, there might be unknown security as well as functionality implications when relying on this version.

Recommendation

We recommend relying on a more recent and stable release of OpenZeppelin's `cairo-contracts` library.

Status: Resolved

8. Usage of outdated and experimental Scarb toolchain version

Severity: Minor

The project currently uses an outdated and experimental release candidate version of the `Scarb` toolchain, specifically version `2.7.0-rc.4` which subsequently relies on `Starkware Cairo` version `2.7.0-rc.3`.

This version may not include the latest bug fixes, security improvements, and performance enhancements provided in the current stable release, `2.8.4`. Furthermore, as a release candidate, `2.7.0-rc.4` may present unknown stability issues, potentially impacting project reliability and security.

Recommendation

We recommend relying on a more recent and stable release of the `Scarb` toolchain.

Status: Resolved

9. Lack of delegate existence verification before removal

Severity: Informational

The `remove_delegate` function currently does not verify whether the specified delegate exists before attempting to remove it. This can lead to scenarios where a delegate removal event is triggered even though the delegate was never assigned.

As a result, off-chain indexers and systems that rely on event data could misinterpret the removal action, potentially logging errors or misrepresenting the delegate state.

Recommendation

We recommend introducing verification in the `remove_delegate` function to confirm the delegate's existence before removing them.

Status: Resolved

10. Potential transaction reverts due to event data size limit

Severity: Informational

The event data field has a size limit of `300 felts`, see Starknet limits, equivalent to approximately `9500 bytes`. However, the `send_notification` and `change_user_channel_settings` functions can accept `ByteArray` payloads that may exceed this limit. If a payload larger than `300 felts` is passed, the event will fail to log, causing a non-specific error (similar to a panic) that may lead to a transaction revert.

Recommendation

We recommend implementing a check with a custom actionable error in both the `send_notification` and the `change_user_channel_settings` functions to verify the length of the `ByteArray` before emitting the event.

Status: Resolved

11. Limited upgrade flexibility

Severity: Informational

In `src/lib.cairo:172`, the contract implementation is upgraded to a new class hash using the `upgrade` method.

Recommendation

We recommend reconsidering potential future use cases of the `upgrade_and_call` method which would allow for more flexibility during the upgrade process.

Status: Resolved

12. Miscellaneous comments

Severity: Informational

Miscellaneous recommendations can be found below.

Recommendation

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

- In `src/lib.cairo:251`, the `recipient` parameter of the `_check_notif_req` function is unused. We recommend removing this unused parameter.
- We recommend adding in-line documentation/Natspec to all functions, especially complex ones like `change_user_channel_settings`, to improve readability.
- We recommend removing unused variables, such as `push_core_address` in `src/lib.cairo:59`, to reduce code complexity and potential confusion.
- In `src/lib.cairo:89`, there is a typo. We recommend replacing `UserNotifcationSettingsAdded` with `UserNotificationSettingsAdded`.
- We recommend removing unused imports:
  - `ContractAddress` in `src/lib.cairo:3` (imported twice)
  - `Clone` in `src/lib.cairo:13`
  - `Serde` in `src/lib.cairo:11`
  - `Zero` in `src/lib.cairo:14`
  - `TryInto` in `src/lib.cairo:10`
  - `IPushComm` in `src/lib.cairo:9`
  - `StorageMapReadAccess` in `src/lib.cairo:20`
  - `OwnableABI` in `src/lib.cairo:24`
  - `MutableStorageNode` in `src/lib.cairo:16`

Status: Resolved