OpenZeppelin | security

# Uniswap Labs Phoenix Fees Diff Audit


November 24, 2025

# Table of Contents

# Summary

| | |
|---|---|
| Type | DeFi |
| Timeline | From 2025-11-20<br>To 2025-11-21 |

## Languages

| | |
|---|---|
| Total Issues | 4 (3 resolved) |
| Critical Severity Issues | 0 (0 resolved) |
| High Severity Issues | 0 (0 resolved) |
| Medium Severity Issues | 0 (0 resolved) |
| Low Severity Issues | 2 (1 resolved) |
| Notes & Additional Information | 2 (2 resolved) |
| Client Reported Issues | 0 (0 resolved) |

# Scope

OpenZeppelin audited the diff of the Uniswap/protocol-fees repository between commits ac23ddf and b8d5546.

In scope were the following files:

```
script
├── 01_DeployMainnet.s.sol
├── 02_DeployUnichain.s.sol
├── 03_UnificationProposal.s.sol
└── deployers
    ├── MainnetDeployer.sol
    └── UnichainDeployer.sol

src
├── interfaces
│   ├── IReleaser.sol
│   └── external
│       └── IL2StandardBridge.sol
└── releasers
    ├── ExchangeReleaser.sol
    ├── Firepit.sol
    └── OptimismBridgedResourceFirepit.sol
```

# System Overview

The update to the Uniswap Fee Collection system consists of two changes: the renaming of the `AssetSink` contract to `TokenJar` and the addition of the `OptimismBridgedResourceFirepit` contract. The `OptimismBridgedResourceFirepit` contract extends the existing `ExchangeReleaser` by adding logic that withdraws the UNI tokens, paid by the caller to release the fees, from Unichain to the burn address on Ethereum.

Additionally, we reviewed the scripts for deploying the Fee Collection system to both Ethereum and Unichain.

# Low Severity

## L-01 Unused Imports

Within `OptimismBridgedResourceFirepit.sol`, there are two imports that are unused and could be removed.

- The `ERC20` import from `solmate/src/tokens/ERC20.sol`
- The `SafeTransferLib` import from `solmate/src/utils/SafeTransferLib.sol`

Consider removing unused imports to improve the overall clarity and readability of the codebase.

***Update:*** *Resolved in pull request #101 at commit 3b433a6.*

## L-02 Multiple Open TODOs in `MainnetDeployer`

In the `MainnetDeployer` script contract there are multiple open TODOs. The `LABS_UNI_RECIPIENT` address and `INITIAL_MERKLE_ROOT` are supposed to be updated.

Consider updating the values or removing the TODOs if they do not apply anymore to improve the clarity of the scripts.

***Update:*** *Acknowledged. The Uniswap team will resolve this before deployment.*

# Notes & Additional Information

## N-01 Lack of Event Emission in `release` Function

The `ExchangeReleaser` abstract contract facilitates the release of assets from a `TokenJar`. Its primary external function, `release`, orchestrates the transfer of a required

resource token before instructing the `TokenJar` to release a specified list of assets to a designated recipient. This function is critical for the core operation of any inheriting contract.

Currently, the `release` function does not emit any event upon successful completion. The absence of an on-chain event log for this activity significantly hinders the ability of off-chain services to monitor and react to asset releases. This lack of visibility complicates tracking, indexing, and real-time alerting based on the contract's operations.

To enhance the observability and auditability of the system, consider introducing a `Released` event in the `ExchangeReleaser` contract. This event should be emitted at the end of the `release` function and include essential parameters such as the `nonce`, the `assets` array, and the `recipient` address. Doing so will create a verifiable on-chain history of all release operations.

***Update:*** *Resolved in pull request #103.*

## N-02 README Links to Wrong Proposal

The `README` links to the wrong proposal when pointing the reader to further documentation regarding the implemented system.

Consider updating it to point to the UNIfication Proposal on the forum instead.

***Update:*** *Resolved in pull request #103.*

# Conclusion

The update prepares the Fee Collection system for deployment on Ethereum by adding the relevant deployment scripts. It also introduces the `OptimismBridgedResourceFirepit` contract to allow collecting the burned UNI tokens in the same place. The audit identified no substantial issues besides suggestions to improve the clarity and off-chain trackability of the codebase.

The Uniswap Labs team is appreciated for being highly cooperative and providing clear explanations throughout the audit process.

# Appendix

## Issue Classification

OpenZeppelin classifies smart contract vulnerabilities on a 5-level scale:

- Critical
- High
- Medium
- Low
- Note/Information

### Critical Severity

This classification is applied when the issue's impact is catastrophic, threatening extensive damage to the client's reputation and/or causing severe financial loss to the client or users. The likelihood of exploitation can be high, warranting a swift response. Critical issues typically involve significant risks such as the permanent loss or locking of a large volume of users' sensitive assets or the failure of core system functionalities without viable mitigations. These issues demand immediate attention due to their potential to compromise system integrity or user trust significantly.

### High Severity

These issues are characterized by the potential to substantially impact the client's reputation and/or result in considerable financial losses. The likelihood of exploitation is significant, warranting a swift response. Such issues might include temporary loss or locking of a significant number of users' sensitive assets or disruptions to critical system functionalities, albeit with potential, yet limited, mitigations available. The emphasis is on the significant but not always catastrophic effects on system operation or asset security, necessitating prompt and effective remediation.

### Medium Severity

Issues classified as being of medium severity can lead to a noticeable negative impact on the client's reputation and/or moderate financial losses. Such issues, if left unattended, have a moderate likelihood of being exploited or may cause unwanted side effects in the system.

These issues are typically confined to a smaller subset of users' sensitive assets or might involve deviations from the specified system design that, while not directly financial in nature, compromise system integrity or user experience. The focus here is on issues that pose a real but contained risk, warranting timely attention to prevent escalation.

### Low Severity

Low-severity issues are those that have a low impact on the client's operations and/or reputation. These issues may represent minor risks or inefficiencies to the client's specific business model. They are identified as areas for improvement that, while not urgent, could enhance the security and quality of the codebase if addressed.

### Notes & Additional Information Severity

This category is reserved for issues that, despite having a minimal impact, are still important to resolve. Addressing these issues contributes to the overall security posture and code quality improvement but does not require immediate action. It reflects a commitment to maintaining high standards and continuous improvement, even in areas that do not pose immediate risks.