OpenZeppelin | security

The Uniswap ERC-20 Token Factory Audit

Uniswap

March 14, 2025

Table of Contents

Summary

Type	DeFi
Timeline	From 2025-03-03 To 2025-03-05
Languages	Solidity

Total Issues	6 (5 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	1 (1 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	2 (2 resolved)
Notes & Additional Information	3 (2 resolved)

Scope

We audited the Uniswap/uniswapERC20-factory repository at commit 7fbcb61.

In scope were the following files:

```
src
├── UniswapERC20.sol
├── UniswapERC20Factory.sol
├── base
│   └── SuperchainERC20.sol
├── interfaces
│   ├── IUniswapERC20.sol
│   └── IUniswapERC20Factory.sol
└── libraries
    └── UniswapERC20Metadata.sol
```

System Overview

The `UniswapERC20` contract is an extended ERC-20 implementation designed for cross-chain compatibility within the Optimism Superchain ecosystem.

Cross-chain transfer hooks (`crosschainBurn` and `crosschainMint`) allow Uniswap ERC-20 tokens to interface with the Superchain Token Bridge. By deploying the same token contract on each network and coordinating via the bridge, the `UniswapERC20` contract ensures that tokens exist on only one chain at a time, mitigating liquidity fragmentation.

`UniswapERC20` tokens are created and deployed via the `UniswapERC20Factory` contract rather than by arbitrary deployers. The `UniswapERC20Factory` determines the token address based on the home chain ID, metadata creator, name, symbol, and token decimals. This deterministic deployment ensures that each token maintains a consistent address across all supported chains when initialized with identical parameters.

Security Model and Privileged Roles

This audit covers the Uniswap implementation of ERC-20 Superchain tokens but not the Optimism Superchain implementation itself. Any vulnerabilities present in the Superchain may affect the security of Uniswap ERC-20 tokens.

The Superchain Token Bridge contract is the only entity allowed to call `crosschainMint`, and typically the only one to invoke `crosschainBurn`. Thus, the bridge contract is a privileged role in this system, responsible for validating cross-chain messages and then minting/burning accordingly.

`UniswapERC20` has no other administrative or owner-only functions. There is no pausability switch, blocklist, or upgrade hook in the token contract.

High Severity

H-01 JSON Injection Allows Overwriting of Creator Metadata and Enables Phishing Attacks

The metadata library creates a JSON object based on the provided metadata. The `displayMetadata` function generates a JSON object but does not sufficiently escape the parameters. This makes it possible to overwrite the Creator value in the JSON by injecting additional data into one of the parameters (`description`, `website`, or `image`), thereby altering the underlying JSON data structure.

In most JSON parsers, if the same key appears multiple times within the same object, only the last occurrence of the key will be valid, while the previous ones will be overwritten. This means an attacker could deploy a malicious token that appears to have been created by a legitimate creator, enabling further phishing attacks.

Consider using the `String` library's `escapeJSON` function to properly escape special characters in the `description`, `website`, and `image` metadata.

Update: *Resolved in pull request #29.*

Low Severity

L-01 Floating Pragma

Pragma directives should be fixed to clearly identify the Solidity version with which the contracts will be compiled.

Throughout the codebase, multiple instances of floating pragma directives were identified:

- `IUniswapERC20.sol` has the `solidity ^0.8.0` floating pragma directive.
- `IUniswapERC20Factory.sol` has the `solidity ^0.8.0` floating pragma directive.
- `SuperchainERC20.sol` has the `solidity ^0.8.28` floating pragma directive.
- `UniswapERC20.sol` has the `solidity ^0.8.28` floating pragma directive.
- `UniswapERC20Factory.sol` has the `solidity ^0.8.28` floating pragma directive.

- `UniswapERC20Metadata.sol` has the `solidity ^0.8.28` floating pragma directive.

Consider using fixed pragma directives.

Update: *Resolved in pull request #30 at commit 41bf73c. The Uniswap Labs team stated:*

> *We made the pragmas fixed on our core logic contracts, but kept our pragmas floating on ^0.8.0 on libraries/interfaces that our integrators use.*

L-02 Different Pragma Directives Across Files

In order to clearly identify the Solidity version with which the contracts will be compiled, pragma directives should be fixed and consistent across file imports.

`IUniswapERC20Factory.sol` and `IUniswapERC20.sol` have the `pragma solidity ^0.8.0;` pragma directive, while the other contracts use `pragma solidity ^0.8.28;`

Consider using the same fixed pragma directive across all the files.

Update: *Resolved in pull request #30 at commit 41bf73c. The Uniswap Labs team stated:*

> *As it related to L-01, we made the pragmas fixed on our core logic contracts, but kept our pragmas floating on ^0.8.0 on libraries/interfaces that our integrators use.*

Notes & Additional Information

N-01 Redundant Getter Function

Within the `UniswapERC20` contract in `UniswapERC20.sol`, the `getMetadata` function is redundant because the `metadata` state variable already has a getter given that it is a `public` variable.

To improve the overall clarity and readability of the codebase, consider removing the redundant getter function.

Update: *Resolved in pull request #28 and pull request #31.*

N-02 Function Visibility Overly Permissive

The `name`, `symbol`, and `decimals` functions in `UniswapERC20.sol` with `public` visibility could be limited to `external`.

To better convey the intended use of functions and to potentially realize some additional gas savings, consider changing a function's visibility to be only as permissive as required.

Update: *Acknowledged, not resolved. The team stated:*

> *We cannot fix it because UniswapERC20 inherits from Solady's ERC20 which has those functions as public, not external.*

N-03 Incorrect Docstrings

The docstrings of the `create` function in the `IUniswapERC20Factory` interface first describe the `totalSupply` parameter and then the `recipient` parameter, whereas, the function itself first accepts `recipient` and then `totalSupply`.

Consider swapping the order of the parameters in the docstrings to align with the implementation.

Update: *Resolved in pull request #27 at commit c5a2609.*

Conclusion

The Uniswap ERC-20 Token Factory project enables the creation of ERC-20 tokens with seamless cross-chain transfers within the Optimism Superchain. The audit identified one high-severity issue, and various recommendations aimed at improving code consistency and readability were made. The Uniswap team was highly cooperative and provided clear explanations throughout the audit process.