OpenZeppelin | security

UERC20 Factory Separation Diff Audit

Uniswap

June 3, 2025

Table of Contents

Summary

Type	DeFi
Timeline	From 2025-05-20 To 2025-05-21
Languages	Solidity

Total Issues	5 (4 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	2 (2 resolved)
Notes & Additional Information	3 (2 resolved)

Scope

OpenZeppelin audited the Uniswap/uerc20-factory repository at commit 5b69661. Specifically, this audit covered the changes shown in the diff between commits 7fbcb61 and 5b69661.

The commits introduced support for Ethereum-only token deployment, and the system was refactored to enhance modularity and deployment flexibility while preserving previously audited behavior. The following changes were observed across most of the in-scope contracts:

- Separation of responsibilities: The non-superchain token functionalities were moved into `BaseUERC20`, an abstract contract with no constructor. Ethereum-only token `UERC20` merely inherits `BaseUERC20` and adds a constructor. Supoer-chain-specific token `UERC20Superchain` inherits `BaseUERC20` and adds the superchain functions to it.
- Dedicated factories: `UERC20Factory` handles Ethereum mainnet deployments, while `UERC20SuperchainFactory` is retained for Superchain-compatible tokens.
- Improved metadata handling: Non-home chain deployments now more clearly signal the absence of metadata. The `displayMetadata` function was updated accordingly.
- Tooling readiness: A `salt` parameter was introduced to improve deployment flexibility and upcoming integration with Uniswap's token launcher.

These changes are primarily organizational and do not introduce new core logic. The core security model remains consistent with the system reviewed during Uniswap's original ERC-20 Token Factory review.

For context on the previous implementation and audit findings, refer to the original report covering commit 7fbcb61.

In scope were the following files:

```
src
├── factories
│   ├── UERC20Factory.sol
│   └── UERC20SuperchainFactory.sol
├── interfaces
│   ├── ITokenFactory.sol
│   ├── IUERC20Factory.sol
│   └── IUERC20SuperchainFactory.sol
├── libraries
```

```
│   └── UERC20MetadataLibrary.sol
└── tokens
    ├── BaseUERC20.sol
    ├── UERC20.sol
    └── UERC20Superchain.sol
```

System Overview

The system under review is a set of smart contracts that allow for permissionless deployment of ERC-20-compliant tokens. There are two varieties, one with the OP Stack's "Superchain transfers" enabled (ERC-7802) and one without.

Superchain transfers allow `UERC20Superchain` tokens to be bridged across chains by being burned on one chain and minted on another. This action must be triggered via Optimism's Superchain Token Bridge precompile. The token contracts must be created and deployed to each Op-Stack chain before being used, but they can safely be deployed in any order. One chain is considered the "home" chain where all tokens are initially minted. On the other hand, UERC20 tokens do not have superchain functionality and are confined to a single chain, which is currently planned to be only the Ethereum mainnet. They retain normal ERC-20 properties.

Both UERC20 and `UERC20Superchain` tokens extend `BaseUERC20`, an abstract contract. They also both use `UERC20MetadataLibrary` to define what metadata is included, such as the token `creator` or `description`. Both tokens are created with their corresponding "factory" contracts, which mostly work the same. Some key differences to note are that the UERC20 tokens must be created by the address labelled `creator`, while `UERC20Superchain` tokens can be created on their non-home chains by anyone. In addition, for `UERC20Superchain` tokens on their non-home chains, all metadata is cleared from the created token contracts. It is also worth noting that all `UERC20Superchain` tokens utilize IERC165, whereas the UERC20 tokens do not.

Security Model and Trust Assumptions

The following trust assumptions were made as part of the audit:

- Anyone may deploy tokens at any time. The token creators have the ability to determine the initial supply of these tokens as well as any metadata associated with them. Thus, anyone using these tokens must trust those deployers. Once the tokens are created, Uniswap retains no special ability to control them.

- Deploying tokens with the same core parameters - `name`, `symbol`, `decimals`, `creator`, and for Superchain tokens, `homeChainId` - is impossible on the same chain. It is assumed that if a deployment error occurs, some core parameter must be changed before re-deployment.
- For every Op-Stack chain to which Uniswap deploys the `UERC20SuperchainFactory`, they will ensure that it is deployed to the same address. Deploying the factory to a different address across chains will result in the created tokens having conflicting addresses across chains.
- The Superchain Token Bridge is responsible for validating cross-chain messages and is the only entity authorized to trigger the mint and burn functions. It is assumed that it will operate safely and normally.
- Any Superchain tokens may be moved to any Op-Stack-enabled chain which has the `UERC20SuperchainFactory` contract deployed on it. It is not possible to limit a token deployed via this system to only a subset of the eligible chains because any user may deploy a token contract on any of these chains at any time.

Low Severity

L-01 Missing Zero-Address Checks for Solady's ERC-20 `_mint()` Invocations

The `BaseUERC20` contract inherits Solady's ERC-20 implementation for default EIP-2612 (Permit) support. Unlike OpenZeppelin's ERC-20 implementation, Solady's `_mint()` function does not validate whether tokens are being minted to the zero address. Currently, the `createToken` [1] [2] and `crosschainMint` functions lack checks for zero address inputs, which could result in token loss and unintended behavior.

Consider implementing validation in the aforementioned functions to prevent issues.

Update: *Resolved in pull request #48 at commit af57aa6.*

L-02 `supportsInterface` Returns `false` for EIP-2612 Support

The `UERC20Superchain` contract uses the Solady ERC-20 implementation for default EIP-2612 (Permit) support but fails to register the `IERC20Permit` interface in its `supportsInterface` function, only declaring the OpenZeppelin `IERC20` interface ID. This leads to external contracts incorrectly detecting a lack of EIP-2612 support.

Consider updating the `supportsInterface` function to include `type(IERC20Permit).interfaceId`, ensuring accurate advertisement of EIP-2612 support.

Update: *Resolved in pull request #47 at commit 00392f5.*

Notes & Additional Information

N-01 Non-Zero `totalSupply` and `recipient` Parameters in Non-Home-Chain Token Creation

The `totalSupply` and `recipient` parameters are irrelevant when the `createToken` function of `UERC20SuperchainFactory` is executed on a non-home chain.

Consider implementing validation checks to ensure that these parameters are set to zero, thereby preventing incorrect assumptions during token deployment.

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

> *Won't change - not too concerned about this since nothing happens if they are filled. Since they are also setting the home chain id, they should expect nothing to be minted.*

N-02 Absence of Validation for `metadata.creator` in the `displayMetadata` Function

When generating a non-home chain superchain token, the metadata is cleared to indicate that it is stored solely on the home chain. However, the `tokenURI` function continues to return token metadata with a zero address in the creator field.

To prevent user confusion, consider implementing a validation check for `metadata.creator` in the `displayMetadata` function to return an empty object or revert if the creator is zero.

Update: *Resolved in pull request #45 at commit 08533b8. The `creator` field is moved out from the metadata struct. Additionally, the team has introduced a salt parameter for the `UERC20Factory` and `UERC20SuperchainFactory` to support their token launcher, which is currently under development.*

N-03 Non-Utilization of Cached Name Hash in Solady `ERC20`

The imported Solady `ERC20` contract declares an overridable `_constantNameHash` function, which, when implemented, enhances the gas efficiency of permit validation and the retrieval of the `DOMAIN_SEPARATOR`. This function is intended for use when the token's `name` field is immutable.

Since the token names for both the `UERC20.sol` and `UERC20Superchain.sol` contracts are immutable, consider overriding the `_constantNameHash` function to optimize performance.

Update: *Resolved in pull request #46 at commit f169a6d.*

Conclusion

The Uniswap UERC20 Token Factory project enables the deployment of `UERC20Superchain` tokens with cross-chain transfer functionality via the Optimism Superchain. This differential audit reviewed refactors for ERC-20 tokens deployed exclusively on the Ethereum mainnet, along with additional updates. It identified minor issues and provided recommendations to enhance code clarity and maintainability. The Uniswap team was highly cooperative and provided clear explanations throughout the audit.