OpenZeppelin | security

OpenZeppelin Confidential Contracts Vulnerability Fix Review

January 7, 2026

Table of Contents

Summary

Type	DeFi
Timeline	From 2025-12-15 To 2025-12-26
Languages	Solidity

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

Scope

OpenZeppelin reviewed commit c620d43 in the OpenZeppelin/openzeppelin-confidential-contracts repository.

In scope was the `ERC7984ERC20Wrapper` contract.

System Overview

The `ERC7984ERC20Wrapper` contract is a wrapper contract built on top of ERC-7984 that allows for wrapping an ERC-20 token into an ERC-7984 token. An ERC-7984 token is a fungible token where balances and transfers are encrypted using the Zama fhEVM, where *fh* stands for fully homomorphic encryption.

On-chain, Zama uses hash-based pointers for encrypted values, while the actual FHE computation occurs on the Zama coprocessor. This architecture makes it impossible to have Solidity code that natively does branching operations when doing operations involving Zama pointers. As such, it requires special handling of overflows and custom value limits.

Vulnerability Disclosure

Due to the silent overflow behavior described in the previous section, in `ERC7984ERC20Wrapper`, users who call `wrap` and cause the ERC-7984 `totalSupply` function to overflow would not be receiving any confidential tokens while still locking the underlying tokens.

Fix Description

The code now infers the confidential total supply based on the ERC-20 balance of the wrapper contract itself, and reverts in the `wrap` function if it is possible for the confidential total supply to overflow. This fix includes the ERC-20 donations when inferring the confidential total supply, which could lead to the inferred value being greater than the actual value, causing reverts even when there is no actual overflow. The Zama team deemed this behavior as acceptable.

Notes & Additional Information

N-01 Missing Docstring

The internal `_update` function does not have docstring.

Consider explicitly documenting the `_update` function following the Ethereum Natural Specification Format (NatSpec). If the comments are identical to the base contract, the `@inheritdoc` tag can be used.

Update: *Resolved in pull request #280.*

N-02 Naming and Comment Suggestions

The name `totalSupply` does not refer to the actual total supply of the contract, and its only purpose is to prevent minting tokens above the limit. Using a standard ERC-20 function name for this purpose could result in misuse and confusion. Consider naming it to `inferredTotalSupply` or something similar.

The name `_checkConfidentialTotalSupply` does not refer to the actual sum of confidential user balances, it refers to the aforementioned derived total supply. Consider naming it to `_checkInferredTotalSupply` or something similar.

The comment above `_checkConfidentialTotalSupply` states: "This function must revert if the new {confidentialTotalSupply} is invalid (overflow occurred)". However, the function could revert sooner than that due to the donations leading to a greater value of inferred total supply than the actual confidential total supply. Consider updating the comment to clarify this.

Consider implementing the above suggestions to improve code clarity and maintainability.

Update: *Resolved in pull request #284. The team stated:*

> Note that we prefer the function to remain named `_checkConfidentialTotalSupply` as its role is to validate the

> `confidentialTotalSupply`. The method it uses to validate the confidential value is via the inferred total supply, but that's not inherent to the function.

Conclusion

The fix under review adequately addressed the disclosed overflow issue in the `wrap` function of the `ERC7984ERC20Wrapper` contract.

No issues were identified in the fix commit, and only simple documentation and naming suggestions were made.

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.