Coinbase EIP 7702 proxy

Security Review

Cantina Managed review by:

Optimum, Lead Security Researcher
RustyRabbit, Security Researcher

March 11, 2025

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Coinbase is a secure online platform for buying, selling, transferring, and storing cryptocurrency.

From Mar 6th to Mar 7th the Cantina team conducted a review of eip-7702-proxy on commit hash 358ab76f. The team identified a total of **4** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	1	1	0
Gas Optimizations	0	0	0
Informational	3	1	2
Total	**4**	**2**	**2**

3 Findings

3.1 Low Risk

3.1.1 `validateAccountState()` may be called on an incorrect `validator`

Severity: Low Risk

Context: EIP7702Proxy.sol#L97

Description: When setting the implementation contract a validator contract is specified as input parameter . However there is no mechanism to verify that the validator performs appropriate verification for the specified implementation contract. This could potentially result in inadequate validation for the implementation contract being deployed.

Recommendation: Consider enhancing the validator contract to accept a parameter indicating the implementation being set. It can then ensure it references the correct implementation contract for which the validation is intended. This approach would necessitate dedicated validator contracts for each implementation contract instance, rather than the current model where validators can be reused across different versions with identical validation requirements.

It should be noted that this approach still permits users to specify non-Coinbase validator contract addresses, which would succeed if the call does not revert. To mitigate this risk, consider implementing a magic return value mechanism that would catch instances where incorrect validator addresses are accidentally used.

Coinbase: Acknowledged, accepting recommendation. Fix PRs: PR 46 PR 47, PR 49, PR 50.

We've implemented an expected/supported implementation on the CoinbaseSmartWalletValidator, and an additional parameter to the validateAccountState function on IAccountStateValidator. This method should revert if the provided implementation being validated does not match what the validator was deployed to expect. Note that a validator could support multiple implementations, as there is no restriction on what logic a validator uses to determine whether an implementation is supported, but it should make this determination.

Additionally, we've implemented the return of a magic value to indicate validation success to avoid silent false-positives or incorrectly implemented validators.

Cantina Managed: Fixed by the combination of the above PRs.

3.2 Informational

3.2.1 Potential function selector collision between proxy and implementation for `receiver`

Severity: Informational

Context: EIP7702Proxy.sol#L34

Description: The `immutable` `receiver` is specified as `public`. There doesn't seem to be a reason and as it's a rather generic name it could lead to a function selector collision if the implementation contract would implement the same function.

Recommendation: Set `receiver` to `internal`.

Coinbase: Acknowledged, accepting recommendation. Fixed in PR 45.

Cantina Managed: Fixed.

3.2.2 Notes about cross-chain support

Severity: Informational

Context: EIP7702Proxy.sol#L79

Description: The `setImplementation` function allows a user to upgrade the implementation of their EOA to a new one. It is designed to support cross-chain signatures, enabling an upgrade with a single signature. However, there are specific requirements, observations, and limitations in the current implementation:

1. Nonce Consistency & Execution Order: The same nonce is used for both single and multi-chain upgrades, while the signature also includes the current implementation address (`ERC1967Utils.getImplementation()`). This requires all signatures to be executed across all chains in the same order. A potential issue arises if a version contains a security flaw—any newly introduced chain will momentarily use the vulnerable version by default. A workaround involves crafting a chain-specific signature to bypass the vulnerable version, but this requires artificially incrementing the nonce by calling `setImplementation` multiple times with the target version as the current version to reach the latest nonce.
2. Address Uniformity: The `_proxy`, `validator`, and `newImplementation` addresses must be identical across all chains for the function to operate seamlessly.

Coinbase: Acknowledged, choosing not to fix. After some debate with our team, we've decided to leave the option for cross-chain signatures in place. The most compelling reason to do this is that in the case of a user who may eventually want to destroy their EOA private key and operate permanently in smart wallet mode, the only way to upgrade on new chains that may not exist yet is with a combination of both a cross-chain 7702 auth signature (presumably with account nonce 0) and a cross-chain account initialization signature (with nonce 0 in the NonceTracker sequence). We will recommend using chain-specific signatures in initial offchain integration, but are uncomfortable completely eliminating this possibility for future needs. In this scenario, mismatched nonces will not be a problem because the EOA private key would have been destroyed and would no longer be available to increment the nonce series managed in the `NonceTracker`. We acknowledge that there are other risks associated with destroying the private key (for example, crosschain 7702 authorizations for a different `non-EIP7702Proxy` delegate that may consume nonce 0 on new chains and render an existing crosschain auth signature invalid), but this is beyond the scope of this finding.

Cantina Managed: Acknowledged.

3.2.3 Inconsistent initial implementation value

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: During the first call to `setImplementation` on a given chain, the signed value of `ERC1967Utils.getImplementation()` will be `0`, while the actual implementation (`_implementation()`) will be set to the `receiver` address. This discrepancy may cause confusion when verifying the initial implementation state.

Recommendation: While this does not present a security risk, it could lead to misunderstandings. We are filing this issue to raise awareness and suggest reviewing the implementation logic for clarity.

Coinbase: Acknowledged, will handle with example integration for generating correct A $\rightarrow$ B signatures for `setImplemenation` using the actual ERC1967 value.

Cantina Managed: Acknowledged.