Coinbase: EIP7702 proxy

Security Review

Cantina Managed review by:

Eric Wang, Lead Security Researcher

Akshay Srivastav, Security Researcher
RustyRabbit, Security Researcher

April 25, 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

Base is a secure and low-cost Ethereum layer-2 solution built to scale the userbase on-chain.

From Feb 4th to Feb 5th the Cantina team conducted a review of eip-7702-proxy on commit hash 9c0c9e62. The team identified a total of **11** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	1	1	0
Medium Risk	3	1	2
Low Risk	3	2	1
Gas Optimizations	0	0	0
Informational	4	1	3
Total	**11**	**5**	**6**

The reviewers highlight that several issues have not been fully resolved, in particular:

- Implementation contract's `receive` function will never be executed.
- Setting another ERC1967 proxy as a delegate can bypass the EIP1976Proxy.
- Entire 32 bytes of data returned by `isValidSignature` `delegatecall` should be checked.

The reviewers recommend another review of the codebase before production.

3 Findings

3.1 High Risk

3.1.1 `EIP7702Proxy.initialize` can be used to forcefully upgrade the implementation of proxy

Severity: High Risk

Context: EIP7702Proxy.sol#L69-L89

Description: The `EIP7702Proxy.initialize` function can be called repeatedly with the same `args` and `signature` values.

Consider this scenario:

1. EOA sets up the `EIP7702Proxy` with `CoinbaseSmartWallet` as implementation as its delegation designator.
2. A bug is discovered in `CoinbaseSmartWallet` implementation contract.
3. To protect itself the EOA upgrades the proxy (via `upgradeToAndCall`) and migrates to any third party wallet implementation. The EOA also resets the `MultiOwnableStorage` from its storage.
4. An attacker can then call `EIP7702Proxy.initialize` (i.e. `EOA.initialize`) again with the same parameter from step-1 and force the proxy to migrate back to the buggy `CoinbaseSmartWallet` implementation.
5. The attacker exploits the bug discovered in step-2.

While the initialization replayability is an intended feature by Coinbase, it also leads to unintended and critical scenarios like described above which can lead to loss of funds for the users.

Recommendation: Consider adding the `_isInitialized` check in `initialize` function like this:

```
  function initialize(
      bytes calldata args,
      bytes calldata signature
  ) external {
+     if (_isInitialized()) revert ProxyAlreadyInitialized();
      // ...
  }
```

Or implement a signature replay protection for `args`. A deadline for signature validity can also be implemented.

Coinbase: We address this finding with nonce-based replay protection that will invalidate initialization signatures after they are redeemed. Due to the possibility of arbitrarily mutable account storage in the case of 7702 accounts, we implement nonce tracking in a separate contract entirely that tracks nonces based on msg.sender. We recognize that there is a potential "*vector*" here in which an EOA could choose to call the `NonceTracker` contract directly and increment its expected nonce. We chose to avoid putting signature validation in the NonceTracker because a determined EOA could still manipulate its nonces in this design by crafting valid signatures and submitting those along with calls to `verifyAndUseNonce`. We assume that an EOA should have no incentive to "attack" itself, and the worst case outcome of such an attack would be the invalidation of a valid initialize signature that would have to be obtained anew.

We also chose to add an optional chainId to the initialize function to allow chain-specific signatures if desired or chain-agnostic signatures by specifying a `chainId` of 0.

- PR 28 for adding the nonce tracking contract.
- PR 36 for adding optional chainId to initialization.

Cantina Managed: Fix verified.

3.2 Medium Risk

3.2.1 Different behaviour of `isValidSignature` function for implementation contracts

Severity: Medium Risk

Context: EIP7702Proxy.sol#L99

Description: The `EIP7702Proxy.isValidSignature` implements ECDSA based signature validation for EOA along with EIP1271 based signature validation for contracts.

In case the implementation contract also has an `isValidSignature` function marked as `public` then in the implementation code:

- Calling `isValidSignature()` will perform implementation contract specific validation, and won't check for ECDSA based EOA signatures.
- Calling `this.isValidSignature()` will perform `EIP7702Proxy` specific validation and will check for ECDSA based EOA signatures.

These two ways of calling the `isValidSignature` function from implementation contract will lead to two different outcomes and can lead to unintended scenarios. Example implementation contract:

```
contract WalletImplementation {
    function isValidSignature(/*...*/) public returns (bytes4) { /*...*/ }

    function foo(/*...*/) external {
        isValidSignature(/*...*/);          // executes WalletImplementation.isValidSignature
        // ...
        this.isValidSignature(/*...*/)      // executes EIP7702Proxy.isValidSignature
    }
}
```

An EOA signature can be valid when executed with `this.isValidSignature(...)` but will be invalid when executed with `isValidSignature(...)`.

Recommendation: Make sure that the `isValidSignature` function in implementation contract is always marked as `external` so that it can only be called via `this.isValidSignature(...)` (i.e. via proxy).

Coinbase: This fix and the other two findings related to `isValidSignature` are all addressed in PR 30. Contrary to the recommendation, the ERC-1271 `isValidSignature` can't be marked external while still overriding the inherited interface. Instead, we address the possibility of this inconsistency with documentation.

Cantina Managed: Acknowledged.

3.2.2 Use Coinbase specific namespace for `INITIALIZED_SLOT`

Severity: Medium Risk

Context: EIP7702Proxy.sol#L30

Description: The ERC7201 namespace used for the `INITIALIZED_SLOT` is `EIP7702Proxy.initialized` which is generic in nature. This increases the risk of storage slot collision with 3rd party forks of the code as they might be inclined to us the same namespace. If a user has previously used such a wallet as a EIP7702 delegate, designating this `EIP7702Proxy` as delegate would result in `delegatecalls` being allowed before the implementation has been properly initialized with the proper owners.

Recommendation: Consider using a Coinbase specific namespace to avoid storage slot collision with 3rd party contracts used as delegate.

Note that this of course does not protect against malicious intent by the 3rd party where the slot is set by a delegate contract that is seemingly restricted in its use of the EOAs funds, but sets this flag to enable a complete ownership takeover if and when the user would switch to the CBSW.

Coinbase: This finding is remedied by removing the INITIALIZED_SLOT entirely. We noted that the only reason for tracking the proxy's initialized state was to prevent the possibility of making calls via the proxy when its implementation address might be in an undefined state (such as containing a dirty value from a prior delegate). It turns out that this state (unexpected ERC-1967 implementation value) is one that can occur at any point in the proxy's lifecycle, and is therefore something we'll need to address in a different way. Blocking calls pre-initialization is therefore inconsistent with possible future states of the proxy.

Even more importantly, blocking calls pre-initialization also means the EOA wouldn't be able to receive ERC-1155 and -721 tokens sent via safeTransfer, because the `onReceive` hooks would not be able to successfully execute.

PR 25 removes initialization state tracking in the proxy.

An iterative design of this proxy will address safe ways to handle the unavoidable possibility of a mutable ERC-1967 slot value.

Cantina Managed: Fixed.

3.2.3 Implementation contract's `receive` function will never be executed

Severity: Medium Risk

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

Description: The `receive` function added to `EIP7702Proxy` contract in PR 24 is implemented as:

```
receive() external payable {}
```

As `_fallback` function is not executed in the `EIP7702Proxy.receive` function, the `implementation.receive` function will never be executed. So in case there is some custom logic in the implementation's `receive` function that custom logic will not be executed. This can be unexpected for the implementation contracts and can lead to unintended outcomes.

Recommendation: Consider changing the `receive` function to this:

```
receive() external payable {
    _fallback();
}
```

Coinbase: Will be addressed in an iterative version of the `EIP7702Proxy`.

Cantina Managed: Acknowledged.

3.3 Low Risk

3.3.1 Setting another ERC1967 proxy as a delegate can bypass the EIP7702Proxy

Severity: Low Risk

Context: EIP7702Proxy.sol#L145

Description: When a user, after having used the EIP7702Proxy as an EIP7702 delegate they set a new delegate to their EOA that also is an EIP1967 proxy it will still use the `initialImplementation` to delegate calls to. This requires the new proxy to not do any initialization which may happen because of user ignorance or or the new proxy assuming it's already set because it has been initialized before.

Although the primary function of the EIP7702Proxy is to make sure the initialization has been done to assure the owners are set (which still is the case as the owner are set in Coinbase specific namespace), it does mean all other functionality provide by the EIP7702Proxy (eg. making sure the`guardedInitializer` can only be called once and overriding the `isValidSignature`) can also be bypassed.

Recommendation: This is a more general EIP7702 problem, which is already being discussed in other EIPs. One possible workaround in this specific case could be to set a transient storage in the `EIP7702Proxy` to a value specific for the implementation contract before the `delegatecall` so that the target knows it's being called by the correct `EIP7702Proxy` and reverting if necessary.

Coinbase: We realized in working with this finding that there is no onchain way to protect against the possible mutation of the ERC-1967 value onchain without using a completely custom implementation slot. We choose not to pursue this path because:

1. It breaks convention with ERC-1967 and...
2. Is messy to implement, especially without making changes to the core `CoinbaseSmartWallet` v1 contract implementation.

Instead, we propose handling this possibility mostly offchain, wherein the user's wallet software will be responsible for observing and detecting issues with the ERC-1967 slot (comparing it to its most recent expected value). Offchain software will also already have to do this for 7702 accounts that may have a new delegate.

To enable correction of the ERC-1967 slot *regardless of what the current implementation logic is* we introduce a new endpoint on the `EIP7702Proxy` itself, `resetImplementation` that validates a signature from the EOA and can set the implementation value directly.

Addressed in PR 33.

Cantina Managed: Resetting the implementation slot alone is not enough. As you're basically starting from an unknown situation, the owners should be reset as well, so calling the guarded initializer with the needed arguments should be done here as well.

Coinbase: Will be addressed in an iterative version of the EIP7702Proxy.

Cantina Managed: Acknowledged.

3.3.2 Function `EIP7702Proxy.isValidSignature` can revert unexpectedly

Severity: Low Risk

Context: EIP7702Proxy.sol#L122

Description: There can be 3 outcomes of `EIP7702Proxy.isValidSignature` function:

1. Return `ERC1271_MAGIC_VALUE`.
2. Return `ERC1271_FAIL_VALUE`.
3. Revert with any `ECDSA.RecoverError` error when `v`, `r` or `s` components of `signature` are invalid.

The function is only intended to return either `ERC1271_MAGIC_VALUE` and `ERC1271_FAIL_VALUE`, the reverting behaviour in point-3 above is unintended.

Recommendation: Consider using `ECDSA.tryRecover` instead of `ECDSA.recover` to handle signature validation failure gracefully.

Coinbase: This fix and the other two findings related to `isValidSignature` are all addressed in PR 30.

Cantina Managed: Fix verified.

3.3.3 Signing unstructured data could introduce the risk of signed data collision or signature reuse

Severity: Low Risk

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

Description: The EOA signs an initialization hash and passes it to the `initialize()` function to initialize the account after setting its delegation designation to the `EIP7702Proxy`. The signed data hash is calculated as `bytes32 hash = keccak256(abi.encode(proxy, args))`, which is designed not to follow EIP-712 typed messages or EIP-191 Ethereum signed messages to prevent the signature from obtaining through a wallet RPC, according to the protocol team.

However, signing messages without an explicit format could introduce the risk of colliding with the signed data of some other function or application. The EOA could theoretically sign a syntactically valid message for some applications but does not intend to use it to initialize the proxy. However, once the signature becomes public on-chain, anyone could attempt to use that signature for initialization.

It is worth noticing that signing unstructured data would also put the EOA at risk of signing a syntactically valid RLP-encoded transaction. EIP-191 was introduced to mitigate the risk by adding a prefix to the signed data. A similar mitigation can be adopted for `EIP7702Proxy` as well.

Recommendation: Consider encoding the signed data with a prefix string, e.g., `bytes32 hash = keccak256(abi.encode("EIP7702ProxyInitialize", proxy, args))`. The prefix string could also be a type hash, e.g., `keccak256("EIP7702ProxyInitialize(address proxy,bytes args)")`. If, in future versions, the proxy requires another raw signature from the EOA, consider using a different prefix to ensure no collision between signed messages would be possible.

Coinbase: Agreed. Fixed in PR 31 for adding a typehash to the initialization hash scheme.

Cantina Managed: Fix verified.

3.4 Informational

3.4.1 EOA may stil be shown as a proxy to `initialImplementation` in block explorers after setting a new EIP7702 delegate

Severity: Informational

Context: EIP7702Proxy.sol#L145

Description: During the initialization of the EIP7702Proxy the EIP1967 `IMPLEMENTATION_SLOT` is set to the `initialImplementation`set in the constructor. If a user sets a new EIP7702 delegate for their EOA the `IMPLEMENTATION_SLOT` will still be set. As currently it is problematic to get the EOA delegation's contract verified on the block explorer, it is possible they will show the EOA as being a ERC1967 proxy with the `initialImplementation` as implementation contract. This can be misleading for the user.

Recommendation: This is a more generic EIP7702 problem for block explorers and can only be really solved at that level. There's no real way to enforce the removal of EOA storage data intended for the old delegate when a user sets a new delegate. EIP7779 has been discussed as a possible way forward.

Coinbase: Acknowledged, won't fix. We leave this to block explorers and/or to the new delegation that is not an ERC-1967 proxy to clear relevant storage.

Cantina Managed: Acknowledged.

3.4.2 Entire 32 bytes of data returned by `isValidSignature` `delegatecall` should be checked

Severity: Informational

Context: EIP7702Proxy.sol#L112-L118

Description: The `EIP7702Proxy.isValidSignature` decodes the `32` bytes data returned by `implementation.isValidSignature` delegatecall into a `bytes4` datatype.

```
abi.decode(result, (bytes4)) == ERC1271_MAGIC_VALUE
```

Potentially the implementation contract can return extra 28 bytes (`result = magic_value_4_bytes | extra_28_bytes`). In that case it is likely that the decoding to `bytes4` will revert. But still it is better to explicitly check the entire 32 bytes of returned data.

Recommendation: Consider adding these changes:

```
- abi.decode(result, (bytes4)) == ERC1271_MAGIC_VALUE
+ abi.decode(result, (bytes32)) == bytes32(ERC1271_MAGIC_VALUE));
```

Coinbase: This fix and the other two findings related to `isValidSignature` are all addressed in PR 30.

Cantina Managed: This bug is incorrectly fixed. The recommendation was to explicitly check the full 32 bytes of returned data. Change the code to this:

```
- bytes4(result) == ERC1271_MAGIC_VALUE
+ result == bytes32(ERC1271_MAGIC_VALUE)
```

3.4.3 Add `receive` function to silence compiler warning

Severity: Informational

Context: EIP7702Proxy.sol#L13

Description: Compilation of `EIP7702Proxy` contract leads to this warning:

> Warning (3628): This contract has a payable fallback function, but no receive ether function. Consider adding a receive ether function.

Recommendation: Consider adding a `receive` function to silence compiler warning.

```
+ receive() external payable {
+     _fallback();
+ }
```

Coinbase: It turns out this finding was deeper than a mere compiler warning! Without an explicit `receive` function, and when we were blocking calls to the fallback function pre-initialization, it actually meant this address couldn't receive ETH pre-initialization! We've since removed the pre-initialization blocking, but it's also solved by adding a `receive` function. See PR 24.

Cantina Managed: However I see another issue. Since `_fallback` is not executed in the `EIP7702Proxy.receive` function, `implementation.receive` will never be executed. See the issue "Implementation contract's `receive` function will never be executed".

3.4.4 Edge case of `guardedInitializer` when proxy is upgraded to third-party implementation

Severity: Informational

Context: EIP7702Proxy.sol#L135-L139

Description: As per the current logic of `EIP7702Proxy` contract, in case the implementation of `EIP7702Proxy` is upgraded to a third-party contract which also has the `initialize(bytes32[])` function (same as `CoinbaseSmartWallet`), then initializing the implementation via `fallback` won't be possible due to the `if (msg.sig == guardedInitializer) revert()` check. This behaviour could be unexpected for the third-party implementation contract and will prevent them from initialization.

Coinbase: Acknowledged, won't fix. We leave it to any would-be implementations that want to use this proxy to design with the full system in mind.

Cantina Managed: Acknowledged.

4 Appendix

4.1 Questions for auditors by the Coinbase team

The client had some questions to ask the auditors for their opinion.

1. It's been suggested that we introduce a typehash to the signature expected during initialization on the proxy, such as: `EIP7702_TYPEHASH = keccak256("EIP7702Proxy(address proxy,bytes args)");` leading then to an expected signed message of `keccak256(abi.encode(EIP7702_TYPEHASH, proxy, args));`.

   Given that the signature is already tied to a specific proxy address, does this really add much extra security at all?

   Response: We believe it is indeed advised to include a typehash and/or a prefix to reduce the risk of collision of the signed data with other applications.

2. If the implementation storage state is cleared at the EOA but then original implementation logic is re-established, existing signed initialization payloads could be used again to establish original owners. This is an unlikely scenario that would require explicit meddling with state by an EOA, but is it worth trying to introduce some sort of expiry (timestamp?). Another proposal for replay protection from Roger is to build a 7702-specific CBSW that has no initializer but always treats the EOA as an owner and subsequent additions of owners happen via EOA transactions.

   Response: As explained in 3. re-intialization has some possible storage related issues which need to be handled with care.

3. Should we block subsequent initializations in the proxy as well? (Easier now that we have an `INITIALIZED` flag.) Or should it stay agnostic to this?

   Response: The current `initialize()` calls the resets the `IMPLEMENTATION_SLOT` to the `initialImplementation` this can be problematic as the upgraded version could have changed the storage variable in a way that the initial version can no longer operate properly.

   So in it's current form this should be blocked as it can already be abused by an attacker. Given proper handling of the storage issues caused by re-initialization this can be done but requires due diligence.

4. We override `isValidSignature` in the `EIP7702Proxy` to perform a basic ecrecover check in case the `isValidSignature` call to the implementation contract fails. This is suggested behavior by the EF: "*EIP-7702 accounts could have off-chain signatures created prior to adding an account code. To keep those signature working, account SHOULD make sure that ERC-1271 signature are compatible. Again, see the reference implementation*".

   Do we fully agree this is how `isValidSignature` should behave? What types of prior created EOA signatures might be risky if honored in this way?

   Response: Instead of deciding for the user, consider allowing the user to decide this behavior during initialization. There may be valid reasons some users want this behavior while other may explicitly not want this.