OpenZeppelin | security

OpenZeppelin Cross-Chain Messaging Audit

December 4, 2024

Table of Contents

Summary

Type	Library
Timeline	From 2024-11-07 To 2024-11-13
Languages	Solidity

Total Issues	12 (11 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	2 (2 resolved)
Low Severity Issues	2 (2 resolved)
Notes & Additional Information	8 (7 resolved)

Scope

We audited the OpenZeppelin/openzeppelin-community-contracts repository at commit 12dd1d5. In scope were the following files:

```
contracts
└── crosschain
    ├── axelar
    │   ├── AxelarGatewayBase.sol
    │   ├── AxelarGatewayDestination.sol
    │   └── AxelarGatewaySource.sol
    ├── interfaces
    │   └── draft-IERC7786.sol
    └── utils
        └── draft-ERC7786Receiver.sol
```

System Overview

This audit focuses on the ERC-7786 standard and the Axelar Network adapters that implement it.

ERC-7786

ERC-7786 proposes a standard interface and workflow for smart contracts to send arbitrary data through cross-chain messaging protocols. The goal is to enable interoperability between the various existing cross-chain communication protocols, which currently have proprietary interfaces.

The key aspects of the standard include:

- **Message encoding:** Messages consist of a sender, receiver, payload, and attributes. The sender and receiver are represented using CAIP-10 account identifiers, the payload is opaque bytes, and attributes are extensible key-value pairs encoded as Solidity function calls.
- **Sending procedure:** A "Source Gateway" contract implements the sending interface, allowing contracts to initiate message transfers through the `sendMessage` function. Gateways may require post-processing steps before the message is effectively sent (e.g., to cover gas costs).
- **Receiving procedure:** The "Destination Gateway" is responsible for validating and delivering messages to the receiver contract, which implements a standard `executeMessage` function. For this purpose, Gateways can operate in either active or passive mode. In active mode, the Gateway calls the receiver contract directly, while in passive mode, anyone can call the receiver contract. In the latter case, the message is validated through the standard `setMessageExecuted` function of the Gateway.
- **Properties:** The underlying cross-chain protocols are expected to provide safety (messages are only delivered if sent), liveness (messages are eventually delivered), and potentially timeliness (bounded delivery time) guarantees in a trustless manner.

The standard aims to improve the composability and interoperability of cross-chain applications by providing a common interface, while still allowing Gateways to expose their unique features through extensible attributes. This should reduce vendor lock-in and enable new cross-chain-native applications.

Axelar Adapters

Axelar is a proof-of-stake blockchain network that allows for secure cross-chain communication between separate blockchains. This could involve making cross-chain token transfers, calling a smart contract on another chain, or passing general, arbitrary messages across chains.

Axelar Gateways enable communication between Axelar and its connected chains, and they have two main functions. On the source chain where the message originates, the Gateway enables initiating cross-chain message requests. On the destination chain where the message is received, the Gateway enables the message to be executed and completes the cross-chain protocol. On EVM chains, Gateways are smart contracts.

To make Axelar Gateways compatible with ERC-7786, a set of adapter contracts was developed and reviewed during this audit:

- The `AxelarGatewaySource` contract implements the `IERC7786GatewaySource` interface, providing functionality for sending messages to a remote chain through the Axelar Network via the `sendMessage` function.
- The `AxelarGatewayDestination` contract implements both the `IAxelarExecutable` and `IERC7786GatewayDestinationPassive` interfaces to support both active and passive reception modes, as defined in ERC-7786.
- The `AxelarGatewayBase` contract serves as a base contract for the other two, offering utility functions for mapping CAIP-2 chain identifiers to Axelar chain identifiers and remote Gateway addresses.

Security Model and Trust Assumptions

Auditing libraries requires a shift in focus due to their composability within blockchain protocols. While the scope of an audit is typically limited to the code itself, the scope expands when it comes to libraries because of their potential internal and external integrations. Libraries act as foundational components for many protocols. This means that their security is influenced not just by their internal robustness, but also by how they are utilized by integrators. As a result, ensuring a library's security involves reviewing the code as well as anticipating its various use cases and integration scenarios.

In addition to these general library-related security considerations, the following potential risks for integrators should also be taken into account:

- The `ERC7786Receiver` contract relies on a trust assumption regarding the Gateways, enforced through `_isKnownGateway`. Gateways control which messages are processed, so it is critical to ensure that the Gateway is trusted and that the underlying cross-chain protocol is secure.
- Attributes defined in ERC-7786 are encoded as Solidity functions, with the attribute key represented by only 4 bytes. This design can lead to potential collisions, so developers should remain cautious when handling attributes to mitigate this risk.
- The Security Considerations section of ERC-7786 outlines the ambiguities related to CAIP-2 and CAIP-10 identifiers. Developers should be aware that these IDs are not inherently unique and must be validated before use to avoid inconsistencies.
- Developers should review and adhere to the repository's security guidelines.

Privileged Roles

There is only one privileged role in the codebase, which is the owner of the `AxelarGatewayBase` contract. This role has the authority to register chain equivalences between CAIP-2 and Axelar chain identifier strings, as well as to assign the remote Gateway address for each CAIP-2 string. These mappings are critical as they are used by the extended `AxelarGatewaySource` and `AxelarGatewayDestination` contracts to ensure interoperability between Axelar gateways and ERC-7786. We assume that the account in charge of this role always acts in the intended way. Hence, any attacks or vulnerabilities targeting this part of the system were not considered throughout this audit.

Medium Severity

M-01 Function and Event Names Do Not Match the Latest Specification

The `setExecutedMessage` function in the `IERC7786GatewayDestinationPassive` interface does not align with the current or previous naming conventions specified in ERC-7786. Additionally, in the current standard, the `receiveMessage` function is specified as `executeMessage`, while the `MessageCreated` and `MessageSent` events have been merged into a single `MessagePosted` event. These mismatches prevent contracts implementing these interfaces from being ERC-7786-compliant, thereby impacting interoperability.

Consider renaming the `setExecutedMessage` and `receiveMessage` functions to align with the latest ERC-7786 specification. Similarly, consider replacing the `MessageCreated` and `MessageSent` events with the unified `MessagePosted` event, and ensure that all references to outdated names in docstrings [1, 2, 3] are updated for consistency.

Update: *Resolved in pull request #22 at commit 5edcfcb.*

M-02 Address Format Does Not Match the Specification

The `AxelarGatewaySource` and `AxelarGatewayDestination` contracts implement ERC-7786 Gateway adapters for the Axelar Network. However, these contracts deviate from the standard in their handling of account addresses:

- The `sendMessage` function currently takes a raw `receiver` address instead of the expected CAIP-10 identifier.
- The `setExecutedMessage` and `receiveMessage` functions take a `sender` argument that is expected to be in CAIP-10 format according to the standard [1, 2]. However, the sender information is encoded as an address string by the source Gateway. Consequently, in both active and passive modes, users must provide the `sender` in an incorrect format to pass validation, which violates the standard.

To align with ERC-7786 and ensure interoperability, consider updating the `sendMessage`, `setExecutedMessage`, and `_execute` functions to work with a CAIP-10 identifier for `receiver` addresses. In addition, within `sendMessage`, consider including the CAIP-10-formatted `sender` address in the message payload rather than the raw address.

Update: *Resolved in pull request #22 at commit 5edcfcb. The intention of the function arguments was clarified with additional documentation. The Contracts team stated:*

> *CAIP-10 string contains two components, the `chain_id` and the `account_address`. When joined together, you get the `account_id`*
>
> `account_id: chain_id + ":" + account_address chain_id: [-a-z0-9]{3,8}:[-_a-zA-Z0-9]{1,32} (See [CAIP-2][]) account_address: [-.%a-zA-Z0-9]{1,128}`
>
> *in the sendMessage function, we take the destinationChain and receiver as two strings, that correspond to these two subcomponents. The alternative would be to use a single-input parameter that if the joined CAIP-10 identifier.*
>
> *It is documented in the ERC document that the second field is the `CAIP-10 account address`. Having the full CAIP-10 account id in the second field would imply duplication of the `chain_id` part, with possible inconsistency.*

Low Severity

L-01 `_processMessage` May Not Receive the Verified Gateway Address

The `_processMessage` function of the `ERC7786Receiver` contract handles the logic for executing cross-chain messages and takes `gateway` as its first argument. This `gateway` address should correspond to the contract validated as a known Gateway in `receiveMessage`. However, if the Gateway operates in active mode, `gateway` could be set to `address(0)`. Consequently, `_processMessage` would lack access to this address, which may be required for its operations.

To ensure that `_processMessage` has the necessary Gateway context, consider setting `gateway` to `msg.sender` when the Gateway is in active mode.

Update: *Resolved, this is not an issue. The gateway address shall be retrieved as* `msg.sender` *when the* `gateway` *argument indicates active mode through* `address(0)`*. The Contracts team stated:*

> *This is by design of the ERC. In active mode, it is said that:* `The arguments` `` `gateway` and `gatewayMessageKey` are unused in active mode and `` `SHOULD be zero and empty respectively.`

L-02 Missing Error Messages in `require` Statements

Within `AxelarGatewayBase.sol`, there are two `require` statements that lack error messages:

- The `require` statement in line 50
- The `require` statement in line 58

Consider throwing custom errors in `require` statements to improve the overall code clarity and facilitate troubleshooting whenever a requirement is not satisfied.

Update: *Resolved in pull request #22 at commit c07671e.*

Notes & Additional Information

N-01 Unused Imports

Having unused imports can negatively affect the readability and clarity of the codebase. Within `AxelarGatewayDestination.sol`, the `CAIP2` and `CAIP10` imports are unused.

Consider removing any unused imports to improve code clarity and maintainability.

Update: *Resolved in pull request #22 at commit 00c5054.*

N-02 Inexplicit Error

The `sendMessage` function of the `AxelarGatewaySource` contract is not intended to support any attributes as defined in ERC-7786. Thus, if a `bytes` type attribute is provided, the call reverts with the `UnsupportedAttribute` custom error, logging the first four bytes of the first attribute. However, in case the first attribute is shorter than 4 bytes, attempting to slice the first four bytes will cause a generic revert.

Consider adding a check for the attribute's length before processing the byte slice and reverting with a specific error if the attribute length is insufficient.

Update: *Resolved in pull request #22 at commit b3d5990.*

N-03 Code Clarity

Throughout the codebase, multiple opportunities for improving code clarity were identified:

- The `_remoteGateways` mapping maps from `string caip2` to `string remoteGateway`. However, the expected format for `remoteGateway` is unclear. According to the `getRemoteGateway` function, it should be an address string.
- The `UnsupportedAttribute` error and `supportsAttribute` function take the `bytes4 selector` argument, while the standard refers to the argument as `bytes4 signature`. Given that the attributes are encoded in the manner of a Solidity function, "selector" seems to be more accurate. Hence, consider updating the standard for consistency.
- The `sendMessage` function currently returns a hardcoded 0 as the `outboxId` and uses it when emitting the `MessageCreated` event. For improved clarity, consider introducing an `outboxId` named return variable, explicitly initializing it to zero, and using it as an argument for the event.

Consider applying the above recommendations to ensure a clearer and more maintainable codebase that adheres to the implemented standard.

Update: *Resolved in pull request #22 at commit 5edcfcb. The Contracts team stated:*

> *The ERC was updated to mention bytes4 selector in the `supportsAttribute` function and the `UnsupportedAttribute` error.*

N-04 Misleading Documentation

The documentation for the `_execute` function is misleading regarding the `remoteAccount` parameter. It first describes `remoteAccount` as the sender of the cross-chain message, but later contradicts this by suggesting the opposite.

For improved clarity, consider avoiding the term "sender" when describing both the Gateway on the source chain and the user who requested the message to be sent. This would help distinguish between the different entities involved and prevent confusion.

Update: *Resolved in pull request #22 at commit c31ff36.*

N-05 Lack of Post-Processing Support for Gas Payments

In ERC-7786, after a sender invokes `sendMessage`, additional steps may be required by the Gateways to make the message effective. This process, called post-processing, typically includes actions such as processing a payment to cover the gas costs for executing the message on the destination chain.

Currently, post-processing is not implemented in the `AxelarGatewaySource` contract. This means that developers must interact with Axelar contracts directly to pay for the gas costs on the source chain. This is suboptimal, as it requires developers to navigate the adapter's internal logic and replicate the arguments used in the `callContract` function to pay for gas.

To improve the developer experience, consider implementing a post-processing function to handle gas payments on the source chain. Alternatively, consider providing clear documentation on how developers can use existing Axelar contracts directly to cover gas costs.

Update: *Acknowledged, will resolve. Additional documentation around the adapter contracts and gas payments is being prepared outside of the code. Furthermore, the Contracts team stated:*

> *The Axelar bridge is designed in such a way that the gas payment is handled by a third-party contract (not the IAxelarGateway), and can be performed by anyone. The `AxelarGatewaySource` only passes the message details (intended by the sender) in the `IAxelarGateway.callContract`*

> *Gas payment is a separate operation that can be performed by anyone, either on-chain or off-chain.*
>
> *Please note that ERC-7786 does not include any details about this post processing step, other than leaving it open to the implementation. That is because many bridges have very different ways to deal with that, and the ERC doesn't want to assume anything about how this is handled by the underlying protocols.*

N-06 Version Conflict With Custom Errors in `require` Statements

In the `AxelarGatewayBase` contract, the `getEquivalentChain` and `getRemoteGateway` functions make use of a `require` statement that reverts with a custom error on failure. However, this feature (using custom errors with `require` statements) has only been available since Solidity version `0.8.26`, while the file's version has been specified as `pragma solidity ^0.8.0`.

To be able to use custom errors with `require` statements, consider updating the pragma statement to `^0.8.26`.

Update: *Resolved in pull request #22 at commit c07671e. The Contracts team stated:*

> *The pragma was updated to 0.8.27.*

N-07 Typographical Errors

Throughout the codebase, multiple instances of typographical errors were identified:

- In `draft-IERC7786.sol`:
  - In lines 13-14, "non standardized" should be "non-standardized".
  - In line 25, "non zero" should be "non-zero".
  - In line 65, "no-one" should be "no one".
  - In line 81, "contracts" should be "contract".
- In `draft-ERC7786Receiver.sol`:
  - In line 11, "two function" should be "two functions".
  - In line 51, "in" should be "is".
- In `AxelarGatewaySource.sol`:
  - In line 15, "using" should be "via" to avoid repetition.

- In `AxelarGatewayDestination.sol`:
  - In line 15, one occurrence of "implements" should be removed.

Consider fixing the above-listed typographical errors to improve the readability of the codebase.

Update: *Resolved in pull request #22 at commit 2501575.*

N-08 Usage of `require` Messages Is Outdated

Since Solidity version 0.8.4, custom errors provide a cleaner and more cost-efficient way to explain to users why an operation failed. As such, it is generally cheaper to use custom errors instead of `require` or `revert` strings. Throughout the codebase, multiple instances of `require` strings were identified:

- In `AxelarGatewayDestination.sol`, the `require` statement with the "Invalid origin gateway" string.
- In `AxelarGatewaySource.sol`, the `require` statement with the "Value not supported" string.

For conciseness and gas savings, consider replacing `require` and `revert` strings with custom errors.

Update: *Resolved in pull request #22 at commit a67fa52.*

Conclusion

ERC-7786 defines a standard interface and workflow for cross-chain communication, extending beyond the Ethereum/EVM ecosystem. The adapter Gateway contracts for Axelar were introduced to enable interoperability between existing Axelar Gateways and the new standard.

During the audit, special attention was given to identifying potential risks for integrators to ensure safe interaction with these contracts. These efforts aim to strengthen the codebase, recognizing its importance as a foundational component of the blockchain ecosystem. The Contracts team has shown a strong commitment to enhancing the library's utility and security, and we appreciate the opportunity to collaborate with them on this milestone.