Security Audit Report

Axelar evm-native-unwrapper
Ethereum smart contract

by ARDA

on July 5, 2026

Table of Contents

Disclaimer **3**

Terminology **4**

Objective **4**

Summary **5**

Code Issues & Recommendations **6**

C1: executeWithInterchainToken ignores token argument and WETH balance can be drained 6

C2: “receive” accepts ETH from any sender but should only accept from WETH 8

C3: Solidity linter cannot resolve the IInterchainTokenExecutable import 9

Disclaimer

The report makes no statements or warranties, either expressed or implied, regarding the security of the code, the information herein or its usage. It also cannot be considered as a sufficient assessment regarding the utility, safety and bugfree status of the code, or any other statements.

This report does not constitute legal or investment advice. It is for informational purposes only and is provided on an "as-is" basis. You acknowledge that any use of this report and the information contained herein is at your own risk. The authors of this report shall not be liable to you or any third parties for any acts or omissions undertaken by you or any third parties based on the information contained herein.

Terminology

Code: The code with which users interact.

Inherent risk: A risk for users that comes from a behavior inherent to the code's design.

Inherent risks only represent the risks inherent to the code's design, which are a subset of all the possible risks. **No inherent risk doesn't mean no risk.** It only means that no risk inherent to the code's design has been identified. Other kind of risks could still be present. For example, the issues not fixed incur risks for the users, or the upgradability of the code might also incur risks for the users.

Issue: A behavior unexpected by the users or by the project, or a practice that increases the chances of unexpected behaviors to appear.

Critical issue: An issue intolerable for the users or the project, that must be addressed.

Major issue: An issue undesirable for the users or the project, that we strongly recommend to address.

Medium issue: An issue uncomfortable for the users or the project, that we recommend to address.

Minor issue: An issue imperceptible for the users or the project, that we advise to address for the overall project security.

Objective

Our objective is to share everything we have found that would help assessing and improving the safety of the code:

1. The **inherent risks** of the code, labelled R1, R2, etc.
2. The **issues** in the **code**, labelled C1, C2, etc.
3. The **issues** in the **testing** of the code, labelled T1, T2, etc.
4. The **issues** in the **other** parts related to the code, labelled O1, O2, etc.
5. The **recommendations** to address each issue.

Summary

Blockchain: Ethereum

Initial scope

- **Repository:** https://github.com/axelarnetwork/axelar-app
- **Commit:** `fce510ca2d02752be45449397091c4855874b4bb`
- **Path(s):** `./apps/contracts/contracts/NativeUnwrapper.sol`

Final scope

- **Repository:** https://github.com/axelarnetwork/axelar-app
- **Commit:** `62b87d4cfbe976b0e04ec4fdb43d249e42f94aae`
- **Path(s):** `./apps/contracts/contracts/NativeUnwrapper.sol`

0 inherent risk in the final scope

0 issue in the final scope

3 issues reported in the initial scope and 0 remaining in the final scope.

Severity	Reported			Remaining		
	Code	Test	Other	Code	Test	Other
Critical	0	0	0	0	0	0
Major	0	0	0	0	0	0
Medium	1	0	0	0	0	0
Minor	2	0	0	0	0	0

Code Issues & Recommendations

C1: executeWithInterchainToken ignores token argument and WETH balance can be drained

Severity: Medium **Status:** Fixed

Location

```
apps/contracts/contracts/NativeUnwrapper.sol
   executeWithInterchainToken
```

Description

Current behavior: During an inbound interchain transfer with data, Axelar `InterchainTokenService` calls the destination's endpoint `executeWithInterchainToken`, and provides as arguments the `token` (and `tokenId`) that have been transferred.

However, the function `executeWithInterchainToken` of `NativeUnwrapper` ignores `token` (and `tokenId`). It calls `weth.withdraw(amount)` against the hardcoded `weth`, then forwards the unwrapped ETH to destination.

Because the function never asserts that `token` matches WETH, the only on-chain precondition for a successful withdraw is `weth.balanceOf(address(this)) >= amount`. Any inbound ITS message that satisfies this balance precondition causes a transfer of the contract's WETH to a recipient encoded entirely by the message sender.

Expected behavior: The function should reject any call whose `tokenId` is not associated with WETH in ITS.

Worst consequence: An attacker can drain any WETH ever held by `NativeUnwrapper`. For this, he would simply make an interchain transfer to the `NativeUnwrapper` with a fake custom `tokenId`.

Note however that in practice, `NativeUnwrapper` should have no WETH balance. It would hold WETH only if someone transferred WETH with a direct ERC-20 `transfer`, e.g. in a donation or simply by mistake.

Recommendation

In `executeWithInterchainToken`, we recommend verifying that the argument `token` equals WETH.

We would further add a unit test witnessing that `executeWithInterchainToken` reverts when `token` is not WETH.

C2: “receive” accepts ETH from any sender but should only accept from WETH

Severity: Minor **Status:** Fixed

Location

`apps/contracts/contracts/NativeUnwrapper.sol`
`receive`

Description

The `receive` function is `external` and `payable` with an empty body — it does not restrict `msg.sender` (NativeUnwrapper.sol:41).

However, the only legitimate sender is the WETH wrapper, which sends back native tokens in `weth.withdraw(amount)`.

In turn, any native token arriving from a path other than the `weth.withdraw` callback is permanently stuck in the contract.

Recommendation

In `receive`, we recommend requiring that the sender is the WETH address.

C3: Solidity linter cannot resolve the IInterchainTokenExecutable import

Severity: Minor **Status:** Fixed

Location

`apps/contracts/contracts/NativeUnwrapper.sol`

Description

The import `IInterchainTokenExecutable` (`NativeUnwrapper.sol:4`) is incorrect, as the Solidity linter complains:

```
@axelar-network/interchain-token-
service/interfaces/IInterchainTokenExecutable.sol
```

It should rather be:

```
@axelar-network/@axelar-network/interchain-token-
service/contracts/interfaces/IInterchainTokenExecutable.sol
```

Recommendation

We recommend correcting the import.