BURRA SEC

Centrifuge(LayerZeroAdapter) Security Review

Reviewed by: windhustler, Goran Vladika

21st August, 2025

Centrifuge(LayerZeroAdapter) Security Review Report

Burra Security

August 21, 2025

Introduction

A time-boxed security review of the **LayerZeroAdapter** built by the Centrifuge team was done by **Burra Security** team, focusing on the security aspects of the smart contracts.

Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource, and expertise-bound effort where we try to find as many vulnerabilities as possible. We can not guarantee 100% security after the review or even if the review will find any vulnerabilities. Subsequent security reviews, bug bounty programs, and on-chain monitoring are recommended.

About Burra Security

Burra Sec offers security auditing and advisory services with a special focus on cross-chain and inter-operability protocols and their integrations.

About LayerZeroAdapter

The focus on the security review was the `LayerZeroAdapter` contract that enables cross-chain communication between different blockchains using LayerZeroV2.

Severity classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Impact - The technical, economic, and reputation damage from a successful attack

Likelihood - The chance that a particular vulnerability gets discovered and exploited

Severity - The overall criticality of the risk

Informational - Findings in this category are recommended changes for improving the structure, usability, and overall effectiveness of the system.

Security Assessment Summary

***review commit hash* - ba0e8c5e56b1aa4faf57f07cb044532205bb1909**

Scope

The following smart contracts were in the scope of the audit:

- src/adapters/LayerZeroAdapter.sol

Findings Summary

ID	Title	Severity	Status
L-01	LayerZeroAdapter is not compatible with Endpoint that uses Lz tokens	Low	Ack
L-02	LayerZero messages can be forced to enter the failed state	Low	Ack
I-01	Missing zero address checks for delegate and endpoint addresses	Info	Resolved
I-02	Swapped natspec in LZ adapter interface	Info	Resolved
I-03	Default max message size of 10000 bytes could limit the batch size	Info	Ack

Detailed Findings

[L-01] LayerZeroAdapter is not compatible with Endpoint that uses Lz tokens

Target

- LayerZeroAdapter.sol

Severity

- Impact: High
- Likelihood: Low

Description

The `LayerZeroAdapter` contract is designed to work exclusively with LayerZero's standard `EndpointV2` contract, which accepts native gas tokens as payment for cross-chain messaging. However, on chains where the native gas token has no value, LayerZero deploys an alternative endpoint called `EndpointV2Alt` that only accepts LZ tokens as payment instead of native tokens.

Recommendation

If you're planning to deploy on chains that only have the alternative `EndpointV2Alt`, the send logic should handle ERC20 token transfers from the sender's address to pay for message bridging.

Centrifuge

Acknowledged, we are not planning to deploy on any chains where this is the case.

BurraSec

The issue has been acknowledged.

[L-02] LayerZero messages can be forced to enter the failed state

Target

- LayerZeroAdapter.sol#L66-L75

Severity

- Impact: Medium
- Likelihood: Low

Description

Description

There is a possibility of DoS with the current function calls where you have:

`LayerZeroAdapter::lzReceive` → `Gateway::handle` → `MessageProcessor::handle`

Since the `LayerZeroAdapter::lzReceive` function only validates that the `msg.sender` is the `Endpoint` contract but doesn't check the executor of `Endpoint::lzReceive`, once lz messages get verified, anyone can execute them via `EndpointV2::lzReceive` with any gas limit.

Due to EIP-150, only 63/64 of gas is forwarded to external calls. Gateway uses **`try`**/**`catch`** for message processing:

```
try processor_.handle(centrifugeId, message) {
    emit ExecuteMessage(centrifugeId, message);
} catch (bytes memory err) {
    bytes32 messageHash = keccak256(message);
    failedMessages[centrifugeId][messageHash]++;
    emit FailMessage(centrifugeId, message, err);
}
```

An attacker can execute the following:

1. A legitimate cross-chain message is verified and ready for execution.
2. An attacker calls `lzReceive`, which calls `processor_.handle` with just enough gas to:
    - Use the 63/64 gas and cause out-of-gas revert in the try block, although the execution should be successful
    - The remaining 1/64 gas is sufficient to execute the catch block, which forces the message into the failed state

Recommendation

Use a library from Optimism and Liquity, which you can place before the external call, which ensures that the `gasleft` is sufficient to execute the external call and if the external call fails due to some logical issue there is still enough gas to execute the catch block.

This approach requires analyzing and setting the appropriate gas amounts required for various message types that are processed inside the `MessageProcessor`. The gas amount should be sufficient to cover the most gas intensive execution path for each message type.

Centrifuge

Seems unlikely but in theory possible! We will fix this by adding the gas library you suggested in the next bigger protocol upgrade.

BurraSec

The issue has been acknowledged.

[I-01] Missing zero address checks for delegate and endpoint addresses

Target

- LayerZeroAdapter.sol#L39-L42

Severity

Informational

Description

The `LayerZeroAdapter::setDelegate` function lacks zero-address validation for the `newDelegate` parameter.

```
## LayerZeroAdapter.sol

function setDelegate(address newDelegate) external auth {
    endpoint.setDelegate(newDelegate); // @audit No zero address
        validation
    emit SetDelegate(newDelegate);
}
```

The LayerZeroV2 OAppCore reference implementation sets the delegate address during contract deployment, which also implicitly checks if the endpoint address is correct.

```
// LayerZero V2 OAppCore reference
constructor(address _endpoint, address _delegate) {
    endpoint = ILayerZeroEndpointV2(_endpoint);

    if (_delegate == address(0)) revert InvalidDelegate();
    endpoint.setDelegate(_delegate);
}
```

Recommendation

Add zero address validation to the `setDelegate` function:

```
function setDelegate(address newDelegate) external auth {
+   require(newDelegate != address(0), InvalidDelegate());
    endpoint.setDelegate(newDelegate);
    emit SetDelegate(newDelegate);
}
```

Additionally, consider setting the delegate in the constructor, following the LayerZero V2 OAppCore implementation.

Centrifuge

Fix for setting delegate on constructor is 956ee809312321cccdbb09e5722a8166d1c1fbef

BurraSec

Fix looks good.

[I-02] Swapped natspec in LZ adapter interface

Target

- ILayerZeroAdapter.sol#L46..L54

Severity

INFO

Description

Natspec comments for `allowInitializePath` and `nextNonce` are swapped.

Recommendation

Adjust the natspec.

Centrifuge

Fixed with 43abc7f80ef20152329f215b97e6a4ec5453aaf2

BurraSec

Fix looks good.

[I-03] Default max message size of 10000 bytes could limit the batch size

Target

- LayerZeroAdapter.sol#L103

Severity

INFO

Description

The default value of executor's config for max message size is 10_000 bytes:

```
cast call -r $ARB $ARB_SEND_LIB "executorConfigs(address,uint32) ((
    uint32,address))" 0x0000000000000000000000000000000000000000
    $BASE_EID
(10000 [1e4], 0x31CAe3B7fB82d847621859fb1585353c5720660D)
```

If message sent is bigger than that TX will revert on the source chain. Just something to be aware of, in case some future batched payload would grow in size and if limiting the gas on the centrifuge side would not be enough, additional guardrails for batch size could be added.

Recommendation

Test the system with the biggest realistic batch sizes to see if additional limit enforcement is needed.

Centrifuge

Acknowledged, not a feasible limit to hit with the Centrifuge protocol.

BurraSec

Acknowledged