AstraSec

TownSquare LayerZeroAdapter Security Audit Report

September 16, 2025

Contents

1 Introduction

1.1 About LayerZeroAdapter

The **LayerZeroAdapter** is a cross-chain bridge adapter that enables communication between blockchain networks using LayerZero's omnichain protocol. It serves as a bridge component within the cross-chain infrastructure, allowing the system to send and receive messages across supported chains.

1.2 Source Code

The following source code was reviewed during the audit:

- https://github.com/TowneSquare/ts-crosschain-contracts.git
- CommitID: 3441e91

And this is the final version representing all fixes implemented for the issues identified in the audit:

- https://github.com/TowneSquare/ts-crosschain-contracts.git
- CommitID: d005561

Note this audit only covers the LayerZeroAdapter contract itself, the composability risks arising from its interactions with BridgeRouter and other contracts are not within the scope of this audit.

1.3 Revision History

Version	Date	Description
v1.0	September 13, 2025	Initial Audit

2 Overall Assessment

This report has been compiled to identify issues and vulnerabilities within the LayerZeroAdapter contract. Throughout this audit, we identified a total of 2 issues spanning various severity levels. By employing auxiliary tool techniques to supplement our thorough manual code review, we have discovered the following findings.

Severity	Count	Acknowledged	Won't Do	Addressed
Critical	1	–	–	1
High	–	–	–	–
Medium	1	1	–	–
Low	–	–	–	–
Informational	–	–	–	–
Total	2	1	–	1

3 Vulnerability Summary

3.1 Overview

Click on an issue to jump to it, or scroll down to see them all.

~~C-1~~ Lack of Access Control in lzReceive()

M-1 Potential Risks Associated with Centralization

3.2 Security Level Reference

In web3 smart contract audits, vulnerabilities are typically classified into different severity levels based on the potential impact they can have on the security and functionality of the contract. Here are the definitions for critical-severity, high-severity, medium-severity, and low-severity vulnerabilities:

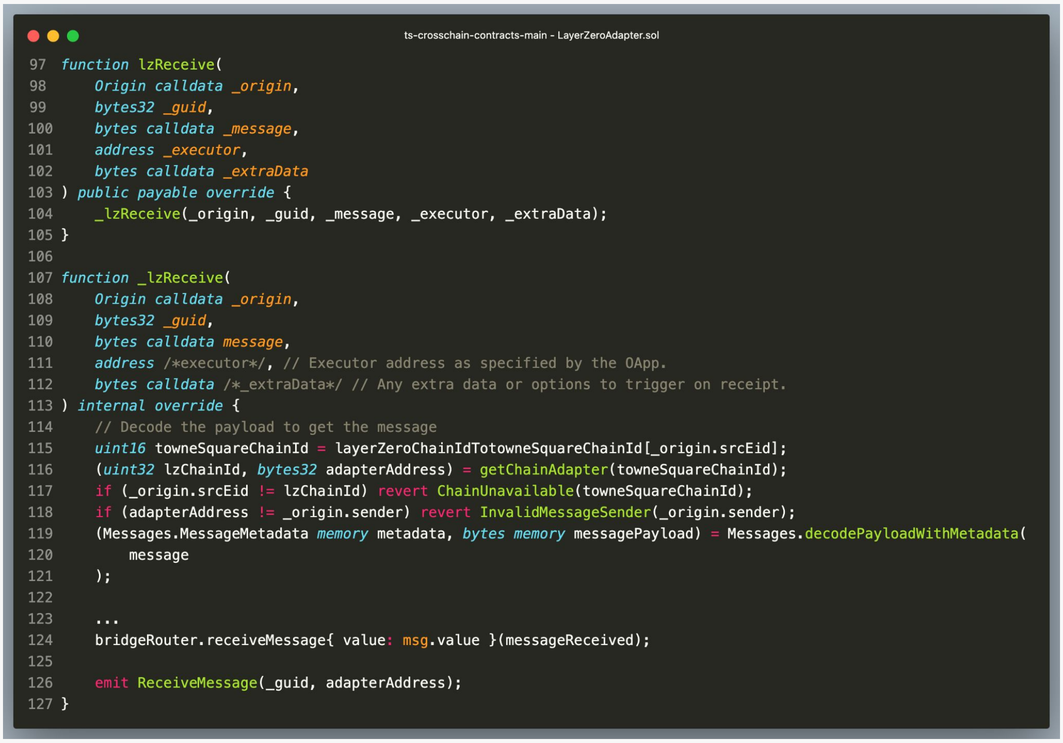
Severity	Acknowledged
C-X (Critical)	A severe security flaw with immediate and significant negative consequences. It poses high risks, such as unauthorized access, financial losses, or complete disruption of functionality. Requires immediate attention and remediation.
H-X (High)	Significant security issues that can lead to substantial risks. Although not as severe as critical vulnerabilities, they can still result in unauthorized access, manipulation of contract state, or financial losses. Prompt remediation is necessary.
M-X (Medium)	Moderately impactful security weaknesses that require attention and remediation. They may lead to limited unauthorized access, minor financial losses, or potential disruptions to functionality.
L-X (Low)	Minor security issues with limited impact. While they may not pose significant risks, it is still recommended to address them to maintain a robust and secure smart contract.
I-X (Informational)	Warnings and things to keep in mind when operating the protocol. No immediate action required.
U-X (Undetermined)	Identified security flaw requiring further investigation. Severity and impact need to be determined. Additional assessment and analysis are necessary.

3.3 Vulnerability Details

3.3.1 [C-1] Lack of Access Control in lzReceive()

TARGET	CATEGORY	IMPACT	LIKELIHOOD	STATUS
LayerZeroAdapter.sol	Business Logic	High	High	Addressed

The LayerZeroAdapter contract exposes lzReceive() as public and forwards directly to _lzReceive() without verifying that msg.sender is the official LayerZero Endpoint. As the Origin fields and message are caller-controlled, an attacker can invoke lzReceive() locally, craft _origin values that pass the internal chain/sender checks, and force bridgeRouter.receiveMessage(...) (line 124) to process arbitrary messages. This enables forged cross-chain messages, unauthorized handler execution, and potential loss of funds.



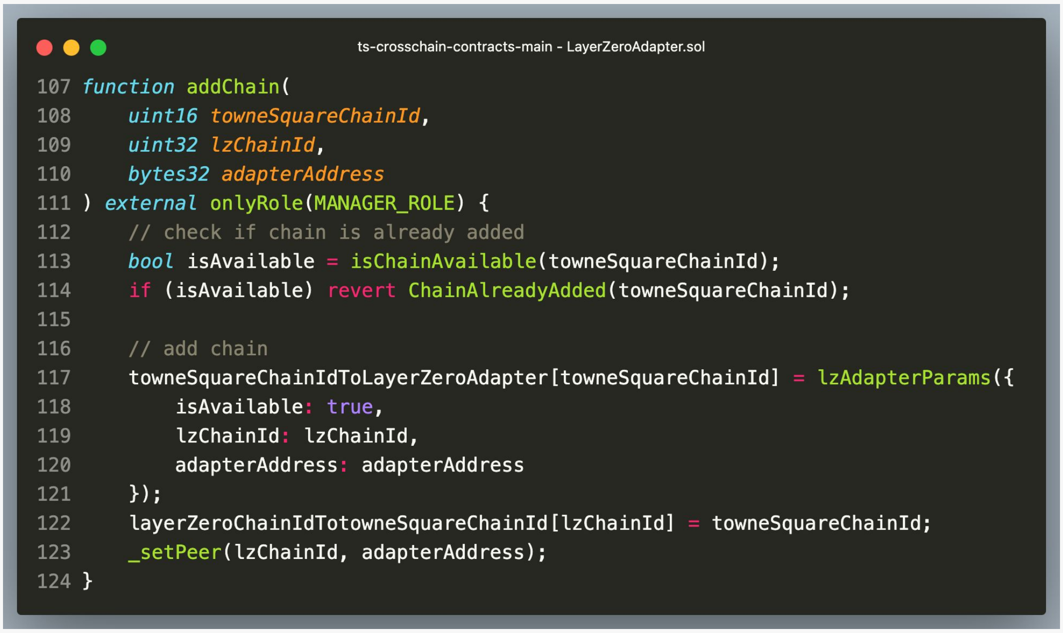
```
97  function lzReceive(
98      Origin calldata _origin,
99      bytes32 _guid,
100     bytes calldata _message,
101     address _executor,
102     bytes calldata _extraData
103 ) public payable override {
104     _lzReceive(_origin, _guid, _message, _executor, _extraData);
105 }
106
107 function _lzReceive(
108     Origin calldata _origin,
109     bytes32 _guid,
110     bytes calldata message,
111     address /*executor*/, // Executor address as specified by the OApp.
112     bytes calldata /*_extraData*/ // Any extra data or options to trigger on receipt.
113 ) internal override {
114     // Decode the payload to get the message
115     uint16 towneSquareChainId = layerZeroChainIdTotowneSquareChainId[_origin.srcEid];
116     (uint32 lzChainId, bytes32 adapterAddress) = getChainAdapter(towneSquareChainId);
117     if (_origin.srcEid != lzChainId) revert ChainUnavailable(towneSquareChainId);
118     if (adapterAddress != _origin.sender) revert InvalidMessageSender(_origin.sender);
119     (Messages.MessageMetadata memory metadata, bytes memory messagePayload) = Messages.decodePayloadWithMetadata(
120         message
121     );
122
123     ...
124     bridgeRouter.receiveMessage{ value: msg.value }(messageReceived);
125
126     emit ReceiveMessage(_guid, adapterAddress);
127 }
```

Remediation Limit the lzReceive() function to be invoked solely by the authorized endpoint.

3.3.2 [M-1] Potential Risks Associated with Centralization

TARGET	CATEGORY	IMPACT	LIKELIHOOD	STATUS
LayerZeroAdapter.sol	Security	High	Low	Acknowledged

In the LayerZeroAdapter contract, the existence of a series of privileged accounts introduces centralization risks, as they hold significant control and authority over critical operations governing the protocol. In the following, we show the representative function potentially affected by the privileges associated with the privileged accounts.


ts-crosschain-contracts-main - LayerZeroAdapter.sol

```
function addChain(
    uint16 towneSquareChainId,
    uint32 lzChainId,
    bytes32 adapterAddress
) external onlyRole(MANAGER_ROLE) {
    // check if chain is already added
    bool isAvailable = isChainAvailable(towneSquareChainId);
    if (isAvailable) revert ChainAlreadyAdded(towneSquareChainId);

    // add chain
    towneSquareChainIdToLayerZeroAdapter[towneSquareChainId] = lzAdapterParams({
        isAvailable: true,
        lzChainId: lzChainId,
        adapterAddress: adapterAddress
    });
    layerZeroChainIdTotowneSquareChainId[lzChainId] = towneSquareChainId;
    _setPeer(lzChainId, adapterAddress);
}
```

Remediation To mitigate the identified issue, it is recommended to introduce multi-sig mechanism to undertake the role of the privileged accounts. Moreover, it is advisable to implement timelocks to govern all modifications to the privileged operations.

Response By Team This issue has been acknowledged by the team. The team plans to introduce a Multisig mechanism to mitigate it.

4 Appendix

4.1 About AstraSec

AstraSec is a blockchain security company that serves to provide high-quality auditing services for blockchain-based protocols. With a team of blockchain specialists, AstraSec maintains a strong commitment to excellence and client satisfaction. The audit team members have extensive audit experience for various famous DeFi projects. AstraSec's comprehensive approach and deep blockchain understanding make it a trusted partner for the clients.

4.2 Disclaimer

4.3 Contact

Phone	+86 156 0639 2692
Email	contact@astrasec.ai
Twitter	https://x.com/AstraSecAI