MixBytes()

Eywa CDP Security Audit Report

DECEMBER 15, 2025

Table of Contents

1. Introduction

1.1 Disclaimer

The audit makes no statements or warranties regarding the utility, safety, or security of the code, the suitability of the business model, investment advice, endorsement of the platform or its products, the regulatory regime for the business model, or any other claims about the fitness of the contracts for a particular purpose or their bug-free status.

1.2 Executive Summary

Eywa CDP (Cross-Chain Data Protocol) is a messaging protocol designed for cross-chain data transfers, utilizing multiple projects like LayerZero, Axelar Bridge and Eywa Bridge. This security audit covers the latest updates to the protocol logic, including the integration of Router Protocol as an additional cross-chain messaging layer.

The audit was conducted over 2 days by 3 auditors, involving an in-depth manual code review and automated analysis within the scope.

During the audit, in addition to verifying standard attack vectors and our internal checklist, we conducted an in-depth review of the following areas:

- **Cross-Chain Message Replay Protection.** We verified that cross-chain messages in the `Receiver` contract can be executed only once and no replay attacks are possible. The contract uses unique `requestIds` and maintains execution state to prevent duplicate message processing.
- **Cross-Chain Data Decoding Consistency.** We confirmed that `data` decoding in `RouterReceiver.iReceive()` function works consistently with other receiver contracts (`ReceiverLZ`, `ReceiverAxelar`) and no data corruption or cross-chain message breaks due to wrong decoding are possible. All receiver contracts follow the same data structure and decoding patterns.
- **Treasury Fund Protection.** We verified that funds from treasury cannot be stolen, only authorized callers can transfer funds from the contract. The treasury implements proper access controls and role-based permissions to prevent unauthorized withdrawals.
- **Bridge State Enforcement.** We verified that the `BridgeRouter` contract can send messages only when in the `active` state; there is no way to get around this restriction.
- **Threshold-Based Message Validation.** We confirmed that only senders with non-zero threshold can send messages through `GateKeeper.sendData()`. The contract validates threshold requirements before processing any cross-chain messages, preventing unauthorized message transmission.
- **Threshold and Validation Enforcement.** We verified that all cross-chain messages undergo proper threshold checks and validation through `Receiver` contracts before execution,

preventing unauthorized or malicious message processing.

- **Multi-Bridge Priority System.** We verified that the bridge priority system in GateKeeper correctly selects and validates bridges before sending messages, ensuring proper fallback mechanisms and preventing message delivery failures.

- **State Consistency.** We confirmed that bridge state management is consistent across all operations and state changes are properly validated and enforced.

- **Request ID Uniqueness.** We confirmed that request ID generation and validation mechanisms prevent duplicate message processing and ensure proper message tracking.

- **Correctness of the integration with Router.** The main improvement in the project compared to the previously audited versions is the integration with the Router bridge. We verified whether the integration works correctly, including protection against relayer spoofing, sufficient authorization of the peer, and correctness of fee payments.

- **Verification of the fee compensation module.** Another improvement in the project was the introduction of a fee compensation module. We assessed how well this system is protected against manipulation, theft, and unjustified payouts, as well as the overall correctness of its implementation.

In addition to the technical areas above, we recommend placing special emphasis on sustaining and improving overall code quality. Specifically, the project should:

- Use automated tooling to catch errors, typos, and anti-patterns early (e.g., linters, static analysis, and AI-assisted code review agents).
- Implement more effective and comprehensive testing (unit, integration, and fuzzing tests) to increase confidence in behavior under edge cases.
- Subject changes to internal peer review by other developers to surface design concerns, ensure consistency, and share domain knowledge.
- Conduct thorough end-to-end testing on public and private testnets, including cross-chain message flows.

The audited files from the GitLab repository at commit 3173d41cc28dc7f8b5b2dc23175ce4839b2afbc3 (as defined in the **Scope of Audit**) fully match the corresponding files in the GitHub repository at commit bf7b7fce8e1862914ca9df7202066782b6c3c397: https://github.com/eywa-protocol/eywa-cdp/tree/bf7b7fce8e1862914ca9df7202066782b6c3c397

The issues we identified during the audit are documented in the **Findings Report** section.

1.3 Project Overview

Summary

Title	Description
Client Name	Eywa
Project Name	CDP
Type	Solidity
Platform	EVM
Timeline	29.07.2025 - 24.10.2025

Scope of Audit

File	Link
contracts/bridge/DVN/EywaDVN.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/04dbad0012b2d47321670ba7af8a737cdf073145/contracts/bridge/DVN/EywaDVN.sol
contracts/bridge/receive/ReceiverRouter.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/04dbad0012b2d47321670ba7af8a737cdf073145/contracts/bridge/receive/ReceiverRouter.sol
contracts/bridge/send/BridgeRouter.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/04dbad0012b2d47321670ba7af8a737cdf073145/contracts/bridge/send/BridgeRouter.sol
contracts/bridge/ExecutorFeeManager.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/04dbad0012b2d47321670ba7af8a737cdf073145/contracts/bridge/ExecutorFeeManager.sol
contracts/bridge/GateKeeper.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/04dbad0012b2d47321670ba7af8a737cdf073145/contracts/bridge/GateKeeper.sol

File	Link
contracts/bridge/Receiver.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/04dbad0012b2d47321670ba7af8a737cdf073145/contracts/bridge/Receiver.sol
contracts/bridge/NativeTreasury.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/04dbad0012b2d47321670ba7af8a737cdf073145/contracts/bridge/NativeTreasury.sol
contracts/bridge/oracles/ChainIdAdapter.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/bridge/oracles/ChainIdAdapter.sol
contracts/bridge/oracles/GasPriceOracle.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/bridge/oracles/GasPriceOracle.sol
contracts/bridge/oracles/LayerZeroOracle.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/bridge/oracles/LayerZeroOracle.sol
contracts/bridge/receive/ReceiverAxelar.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/bridge/receive/ReceiverAxelar.sol
contracts/bridge/receive/ReceiverLZ.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/bridge/receive/ReceiverLZ.sol
contracts/bridge/send/BridgeAxelar.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/bridge/send/BridgeAxelar.sol
contracts/bridge/send/BridgeLZ.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/bridge/send/BridgeLZ.sol

File	Link
contracts/bridge/BridgeV3.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/bridge/BridgeV3.sol
contracts/bridge/EPOA.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/bridge/EPOA.sol
contracts/bridge/GovBridgeV2.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/bridge/GovBridgeV2.sol
contracts/bridge/NodeRegistryV2.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/bridge/NodeRegistryV2.sol
contracts/utils/Block.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/utils/Block.sol
contracts/utils/Bls.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/utils/Bls.sol
contracts/utils/Merkle.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/utils/Merkle.sol
contracts/utils/ModUtils.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/utils/ModUtils.sol
contracts/utils/RequestIdLib.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/utils/RequestIdLib.sol

File	Link
contracts/utils/Typecast.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/utils/Typecast.sol
contracts/utils/Utils.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/utils/Utils.sol
contracts/utils/ZeroCopySource.sol	https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/blob/9a497e2a7c4949371854fb2b53a8d206c07b4064/contracts/utils/ZeroCopySource.sol

Versions Log

Date	Commit Hash	Note
29.07.2025	04dbad0012b2d47321670ba7af8a737cdf073145	Initial commit
11.08.2025	2af615e8aa1f6ea1803dfacfc93e4692f6ffe54b	Re-audit commit
29.09.2025	9a497e2a7c4949371854fb2b53a8d206c07b4064	Second re-audit commit
20.10.2025	52720bf2d46914436f10053522181ff20d4af02d	Third re-audit commit
24.10.2025	3173d41cc28dc7f8b5b2dc23175ce4839b2afbc3	Fourth re-audit commit

Mainnet Deployments

File	Address	Blockchain
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Arbitrum
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Arbitrum
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Arbitrum

File	Address	Blockchain
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Arbitrum
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Arbitrum
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Arbitrum
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Arbitrum
ReceiverLZ.sol	0xf3E769a5965a89904741bF887542EeA81096169c	Arbitrum
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Arbitrum
BridgeAxelar.sol	0x5eEdDcE72530e4fC96d43E3d70Fe09aD0D037175	Arbitrum
ReceiverAxelar.sol	0xB2185950F5A0A46687ac331916508aadA202e063	Arbitrum
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Arbitrum
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Arbitrum
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Avalanche
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Avalanche
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Avalanche
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Avalanche
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Avalanche
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Avalanche
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Avalanche
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Avalanche
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Avalanche
BridgeAxelar.sol	0x5eEdDcE72530e4fC96d43E3d70Fe09aD0D037175	Avalanche
ReceiverAxelar.sol	0xB2185950F5A0A46687ac331916508aadA202e063	Avalanche
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Avalanche
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Avalanche

File	Address	Blockchain
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Base
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Base
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Base
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Base
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Base
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Base
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Base
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Base
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Base
BridgeAxelar.sol	0x5eEdDcE72530e4fC96d43E3d70Fe09aD0D037175	Base
ReceiverAxelar.sol	0xB2185950F5A0A46687ac331916508aadA202e063	Base
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Base
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Base
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Berachain
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Berachain
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Berachain
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Berachain
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Berachain
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Berachain
BridgeAxelar.sol	0x5eEdDcE72530e4fC96d43E3d70Fe09aD0D037175	Berachain
ReceiverAxelar.sol	0xB2185950F5A0A46687ac331916508aadA202e063	Berachain
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Berachain
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Berachain

File	Address	Blockchain
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Blast
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Blast
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Blast
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Blast
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Blast
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Blast
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Blast
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Blast
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Blast
BridgeAxelar.sol	0x5eEdDcE72530e4fC96d43E3d70Fe09aD0D037175	Blast
ReceiverAxelar.sol	0xB2185950F5A0A46687ac331916508aadA202e063	Blast
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Blast
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Blast
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	BSC
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	BSC
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	BSC
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	BSC
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	BSC
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	BSC
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	BSC
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	BSC
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	BSC
BridgeAxelar.sol	0x5eEdDcE72530e4fC96d43E3d70Fe09aD0D037175	BSC

File	Address	Blockchain
ReceiverAxelar.sol	0xB2185950F5A0A46687ac331916508aadA202e063	BSC
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	BSC
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	BSC
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Celo
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Celo
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Celo
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Celo
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Celo
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Celo
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Celo
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Celo
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Celo
BridgeAxelar.sol	0x5eEdDcE72530e4fC96d43E3d70Fe09aD0D037175	Celo
ReceiverAxelar.sol	0xB2185950F5A0A46687ac331916508aadA202e063	Celo
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Core
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Core
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Core
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Core
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Core
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Core
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Cronos
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Cronos
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Cronos

File	Address	Blockchain
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Cronos
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Cronos
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Cronos
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Ethereum
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Ethereum
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Ethereum
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Ethereum
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Ethereum
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Ethereum
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Ethereum
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Ethereum
LayerZeroOracle.sol	0xA71CE928934CD996907589FcDd4083124509E8fb	Ethereum
BridgeAxelar.sol	0x5eEdDcE72530e4fC96d43E3d70Fe09aD0D037175	Ethereum
ReceiverAxelar.sol	0xB2185950F5A0A46687ac331916508aadA202e063	Ethereum
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Ethereum
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Ethereum
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Fantom
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Fantom
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Fantom
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Fantom
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Fantom
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Fantom
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Fantom

File	Address	Blockchain
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Fantom
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Fantom
BridgeAxelar.sol	0x5eEdDcE72530e4fC96d43E3d70Fe09aD0D037175	Fantom
ReceiverAxelar.sol	0xB2185950F5A0A46687ac331916508aadA202e063	Fantom
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Fantom
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Fantom
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Fraxtal
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Fraxtal
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Fraxtal
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Fraxtal
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Fraxtal
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Fraxtal
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Fraxtal
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Fraxtal
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Fraxtal
BridgeAxelar.sol	0x5eEdDcE72530e4fC96d43E3d70Fe09aD0D037175	Fraxtal
ReceiverAxelar.sol	0xB2185950F5A0A46687ac331916508aadA202e063	Fraxtal
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Gnosis
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Gnosis
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Gnosis
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Gnosis
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Gnosis
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Gnosis

File	Address	Blockchain
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Gnosis
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Gnosis
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Gnosis
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Gnosis
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Gnosis
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Haust
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Haust
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Haust
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Haust
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Haust
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Haust
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Kava
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Kava
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Kava
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Kava
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Kava
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Kava
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Kava
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Kava
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Kava
BridgeAxelar.sol	0x5eEdDcE72530e4fC96d43E3d70Fe09aD0D037175	Kava
ReceiverAxelar.sol	0xB2185950F5A0A46687ac331916508aadA202e063	Kava
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Linea

File	Address	Blockchain
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Linea
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Linea
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Linea
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Linea
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Linea
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Linea
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Linea
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Linea
BridgeAxelar.sol	0x5eEdDcE72530e4fC96d43E3d70Fe09aD0D037175	Linea
ReceiverAxelar.sol	0xB2185950F5A0A46687ac331916508aadA202e063	Linea
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Linea
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Linea
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Manta
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Manta
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Manta
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Manta
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Manta
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Manta
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Manta
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Manta
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Manta
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Manta
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Manta

File	Address	Blockchain
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Mantle
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Mantle
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Mantle
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Mantle
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Mantle
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Mantle
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Mantle
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Mantle
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Mantle
BridgeAxelar.sol	0x5eEdDcE72530e4fC96d43E3d70Fe09aD0D037175	Mantle
ReceiverAxelar.sol	0xB2185950F5A0A46687ac331916508aadA202e063	Mantle
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Mantle
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Mantle
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Metis
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Metis
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Metis
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Metis
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Metis
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Metis
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Metis
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Metis
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Metis
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Metis

File	Address	Blockchain
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Metis
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Mode
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Mode
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Mode
BridgeV3.sol	0x9291c5586217348542720C5FB1028c9C976C7b50	Mode
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Mode
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Mode
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Mode
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Mode
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Mode
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Mode
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Mode
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Moonbeam
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Moonbeam
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Moonbeam
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Moonbeam
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Moonbeam
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Moonbeam
BridgeAxelar.sol	0x5eEdDcE72530e4fC96d43E3d70Fe09aD0D037175	Moonbeam
ReceiverAxelar.sol	0xB2185950F5A0A46687ac331916508aadA202e063	Moonbeam
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Optimism
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Optimism
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Optimism

File	Address	Blockchain
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Optimism
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Optimism
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Optimism
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Optimism
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Optimism
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Optimism
BridgeAxelar.sol	0x5eEdDcE72530e4fC96d43E3d70Fe09aD0D037175	Optimism
ReceiverAxelar.sol	0xB2185950F5A0A46687ac331916508aadA202e063	Optimism
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Optimism
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Optimism
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Polygon
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Polygon
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Polygon
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Polygon
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Polygon
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Polygon
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Polygon
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Polygon
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Polygon
BridgeAxelar.sol	0x5eEdDcE72530e4fC96d43E3d70Fe09aD0D037175	Polygon
ReceiverAxelar.sol	0xB2185950F5A0A46687ac331916508aadA202e063	Polygon
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Polygon
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Polygon

File	Address	Blockchain
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Sonic
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Sonic
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Sonic
GovBridgeV2.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Sonic
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Sonic
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Sonic
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Sonic
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Sonic
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Sonic
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Sonic
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Sonic
NodeRegistryV2.sol	0xE442fFC9FFEF2588B4620C29FDeBD03971eb4319	Sonic
EPOA.sol	0xab5a107009E25e6A9ee507ef18EC63878188BAb8	Sonic
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Taiko
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Taiko
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Taiko
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Taiko
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Taiko
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Taiko
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Taiko
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Taiko
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Taiko
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Taiko

File	Address	Blockchain
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Taiko
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Unichain
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Unichain
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Unichain
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Unichain
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Unichain
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Unichain
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	Unichain
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	Unichain
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Unit0
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Unit0
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Unit0
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Unit0
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Unit0
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Unit0
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	X Layer
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	X Layer
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	X Layer
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	X Layer
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	X Layer
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	X Layer
BridgeRouter.sol	0x6D52E271d444B9b23CD24Cb235c7cca8c060fD44	X Layer
ReceiverRouter.sol	0xfD3856DD50A6f18beF86DA76aF11eD646d374803	X Layer

File	Address	Blockchain
Receiver.sol	0x0F00F1a6A32e644815C5686aD7dc305A54B11200	Aurora
GateKeeper.sol	0x1cFBd51e558782bD24268Eaed312631cc7F2a99F	Aurora
ExecutorFeeManager.sol	0x0Ad89b3e12a4ad21E52F521bEB0C005bbb1a4Ea9	Aurora
BridgeV3.sol	0x9291c55862173485427 20C5FB1028c9C976C7b50	Aurora
ChainIdAdapter.sol	0xD2bc3186655442e321eF00e82259005557015975	Aurora
GasPriceOracle.sol	0x6f09fe05b2B9Ac652942b1A4E94598C8c6695fF6	Aurora
BridgeLZ.sol	0xCbc87906558036C6d0b7fA6B0Fc460B7Bc545C4a	Aurora
ReceiverLZ.sol	0x76407E610a258bF86B110C6A09FB716922B1CFBb	Aurora
LayerZeroOracle.sol	0xb0ED75BC5A32a3cbD23F132D040fD74A4be77222	Aurora

1.4 Security Assessment Methodology

Project Flow

Stage	Scope of Work
Interim audit	**Project Architecture Review:** · Review project documentation · Conduct a general code review · Perform reverse engineering to analyze the project's architecture based solely on the source code · Develop an independent perspective on the project's architecture · Identify any logical flaws in the design OBJECTIVE: UNDERSTAND THE OVERALL STRUCTURE OF THE PROJECT AND IDENTIFY POTENTIAL SECURITY RISKS.
	Code Review with a Hacker Mindset: · Each team member independently conducts a manual code review, focusing on identifying unique vulnerabilities. · Perform collaborative audits (pair auditing) of the most complex code sections, supervised by the Team Lead. · Develop Proof-of-Concepts (PoCs) and conduct fuzzing tests using tools like Foundry, Hardhat, and BOA to uncover intricate logical flaws. · Review test cases and in-code comments to identify potential weaknesses. OBJECTIVE: IDENTIFY AND ELIMINATE THE MAJORITY OF VULNERABILITIES, INCLUDING THOSE UNIQUE TO THE INDUSTRY.
	Code Review with a Nerd Mindset: · Conduct a manual code review using an internally maintained checklist, regularly updated with insights from past hacks, research, and client audits. · Utilize static analysis tools (e.g., Slither, Mythril) and vulnerability databases (e.g., Solodit) to uncover potential undetected attack vectors. OBJECTIVE: ENSURE COMPREHENSIVE COVERAGE OF ALL KNOWN ATTACK VECTORS DURING THE REVIEW PROCESS.

Stage	Scope of Work
	Consolidation of Auditors' Reports: · Cross-check findings among auditors · Discuss identified issues · Issue an interim audit report for client review OBJECTIVE: COMBINE INTERIM REPORTS FROM ALL AUDITORS INTO A SINGLE COMPREHENSIVE DOCUMENT.
Re-audit	**Bug Fixing & Re-Audit:** · The client addresses the identified issues and provides feedback · Auditors verify the fixes and update their statuses with supporting evidence · A re-audit report is generated and shared with the client OBJECTIVE: VALIDATE THE FIXES AND REASSESS THE CODE TO ENSURE ALL VULNERABILITIES ARE RESOLVED AND NO NEW VULNERABILITIES ARE ADDED.
Final audit	**Final Code Verification & Public Audit Report:** · Verify the final code version against recommendations and their statuses · Check deployed contracts for correct initialization parameters · Confirm that the deployed code matches the audited version · Issue a public audit report, published on our official GitHub repository · Announce the successful audit on our official X account OBJECTIVE: PERFORM A FINAL REVIEW AND ISSUE A PUBLIC REPORT DOCUMENTING THE AUDIT.

1.5 Risk Classification

Severity Level Matrix

Severity	Impact: High	Impact: Medium	Impact: Low
Likehood: High	Critical	High	Medium
Likehood: Medium	High	Medium	Low
Likehood: Low	Medium	Low	Low

Impact

- **High** – Theft from 0.5% OR partial/full blocking of funds (>0.5%) on the contract without the possibility of withdrawal OR loss of user funds (>1%) who interacted with the protocol.
- **Medium** – Contract lock that can only be fixed through a contract upgrade OR one-time theft of rewards or an amount up to 0.5% of the protocol's TVL OR funds lock with the possibility of withdrawal by an admin.
- **Low** – One-time contract lock that can be fixed by the administrator without a contract upgrade.

Likelihood

- **High** – The event has a 50–60% probability of occurring within a year and can be triggered by any actor (e.g., due to a likely market condition that the actor cannot influence).
- **Medium** – An unlikely event (10–20% probability of occurring) that can be triggered by a trusted actor.
- **Low** – A highly unlikely event that can only be triggered by the owner.

Action Required

- **Critical** – Must be fixed as soon as possible.
- **High** – Strongly advised to be fixed to minimize potential risks.
- **Medium** – Recommended to be fixed to enhance security and stability.
- **Low** – Recommended to be fixed to improve overall robustness and effectiveness.

Finding Status

- **Fixed** – The recommended fixes have been implemented in the project code and no longer impact its security.
- **Partially Fixed** – The recommended fixes have been partially implemented, reducing the impact of the finding, but it has not been fully resolved.
- **Acknowledged** – The recommended fixes have not yet been implemented, and the finding remains unresolved or does not require code changes.

1.6 Summary of Findings

Findings Count

Severity	Count
Critical	1
High	1
Medium	4
Low	10

Findings Statuses

ID	Finding	Severity	Status
C-1	Missing Source Authenticity Verification in ReceiverRouter	Critical	Fixed
H-1	Refund Can Be Lost Due To payExecutorGasFee() Front-run	High	Fixed
M-1	BridgeRouter Cannot Set FeePayer When Fees Are Required	Medium	Fixed
M-2	Incorrect Decoding of options_ Causes Gas Fee Misestimation in estimateGasFee()	Medium	Fixed
M-3	Incorrect Use of to Address in _send() Function Breaks Message Routing	Medium	Fixed
M-4	Cross-Chain Transfer May Revert Due to Missing Fee in _send() Function	Medium	Fixed
L-1	getRequestMetadata() Helper Not Implemented in BridgeRouter Contract	Low	Fixed
L-2	Missing iAck Handling in BridgeRouter Contract	Low	Fixed
L-3	estimateGasFee() Reverts When Executor Is Not Configured	Low	Fixed

L-4	Use of Revert Strings Instead of Custom Errors	Low	Acknowledged
L-5	Centralization Risks	Low	Acknowledged
L-6	Typographical Errors in Comments of GateKeeper and Receiver Contracts	Low	Fixed
L-7	Unused and Misapplied Imports in ReceiverRouter and BridgeRouter	Low	Fixed
L-8	Fee Estimation Requires Dummy Nonce in Options Configuration	Low	Fixed
L-9	Missing Zero-Address Check in Public Withdraw Function	Low	Fixed
L-10	Misspelled Interface Name and Missing Inheritance	Low	Fixed

2. Findings Report

2.1 Critical

C-1	Missing Source Authenticity Verification in ReceiverRouter		
Severity	Critical	**Status**	Fixed in https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/commit/2af615e8aa1f6ea1803dfacfc93e4692f6ffe54b

Description

The ReceiverRouter.iReceive() function assumes its input parameters originate from a trusted source (the Router bridge); however, it lacks any verification to confirm that the caller is actually legitimate.

A malicious user with no privileges could impersonate the bridge and fake the receipt of a cross-chain message from a trusted source, using arbitrary parameters.

This effectively compromises the Router bridge integration. If the threshold in the Receiver contract is greater than 1, it provides no meaningful protection; and if the threshold is set to 1, the system becomes vulnerable to unauthorized message execution.

This issue is classified as **Critical** severity because ReceiverRouter may execute unverified and potentially harmful data, leading to unauthorized actions and a high risk of financial loss.

Recommendation

We recommend restricting the iReceive() function so that it can only be called by the legitimate Router bridge.

Do not rely solely on the requestSender string for source authenticity. Instead, enforce that the caller is an approved Gateway contract and validate the message origin using the protocol's official delivery verification mechanisms.

Client's Commentary:

Fixed at af8da87b9bfb7b89971492c19bf615dfaaa8d9f7

2.2 High

H-1	Refund Can Be Lost Due To payExecutorGasFee() Front-run		
Severity	High	**Status**	Fixed in https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/commit/2af615e8aa1f6ea1803dfacfc93e4692f6ffe54b

Description

The ExecutorFeeManager.payExecutorGasFee() function can be front-run to reroute gas-fee refunds.

An attacker may call the function with the same requestId and set their malicious refundTarget. When an executor calls ExecutorFeeManager.refund() to receive the refund, it will be transferred to the attacker.

This issue is classified as **High** severity because it leads to executor's funds being lost.

Recommendation

We recommend restricting access to the ExecutorFeeManager.payExecutorGasFee() function by applying a proper access control modifier.

If the function is intended to be callable by external parties (not only the GateKeeper contract), we recommend adding a validation step to ensure that refundTarget is properly initialized before execution and that only the GateKeeper contract can set the refundTarget.

Client's Commentary:

Fixed at 4fe8b7709b38b0b184770ed8855e11005f0e7ce8

2.3 Medium

M-1	BridgeRouter Cannot Set FeePayer When Fees Are Required		
Severity	Medium	**Status**	Fixed in https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/commit/2af615e8aa1f6ea1803dfacfc93e4692f6ffe54b

Description

The BridgeRouter.setDappMetadata() function does not forward received native value to the Gateway contract. As a result, when gateway.iSendDefaultFee() is non-zero, an address with OPERATOR_ROLE cannot successfully set the FeePayer identifier because the required fee is not transferred to the Gateway.

The fee associated with any cross-chain request initiated by a dApp is paid by the dApp's corresponding fee payer account on the Router Chain. To cover the relayer incentives and validation costs, the Router Chain deducts the estimated fee and incentive amount upfront from the fee payer address.

Because the protocol cannot register the fee payer identifier, the Router Chain is unable to deduct the required fees from the dApp's fee payer account, preventing cross-chain requests from being executed. This issue is therefore classified as a **Medium** severity.

Recommendation

We recommend checking the current iSendDefaultFee value and forwarding the appropriate amount of native funds to the Gateway within the BridgeRouter.setDappMetadata() function.

Client's Commentary:

Fixed at 37fcddbea88f7e9539a71c8855d8c56194d63426

M-2	Incorrect Decoding of options_ Causes Gas Fee Misestimation in estimateGasFee()		
Severity	Medium	**Status**	Fixed in https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/commit/2af615e8aa1f6ea1803dfacfc93e4692f6ffe54b

Description

The BridgeRouter.estimateGasFee() function decodes the options_ bytes array by reading the first 32 bytes. According to Router Protocol documentation, the options_ parameter should contain a requestMetadata struct with the following fields:

```
uint64 destGasLimit;
uint64 destGasPrice;
uint64 ackGasLimit;
uint64 ackGasPrice;
uint128 relayerFees;
uint8 ackType;
bool isReadCall;
string asmAddress;
```

https://docs.routerprotocol.com/develop/message-transfer-via-crosstalk/evm-guides/iDapp-functions/iSend/#5-requestmetadata

However, the estimateGasFee() function incorrectly assumes that the first 32 bytes represent a single gas fee value. In reality, the first 32 bytes contain multiple parameters, leading to incorrect decoding of the options_ data.

This improper decoding may cause inaccurate gas fee estimations or failures when interacting with the Router Protocol.

Recommendation

We recommend fetching the current iSendDefaultFee directly from the Gateway contract in the BridgeRouter.estimateGasFee() function and including this fee when forwarding cross-chain messages.

Client's Commentary:

Fixed at 5b8d54c0479e4c2f0d422fab02ce705d94b193be

M-3	Incorrect Use of `to` Address in `_send()` Function Breaks Message Routing		
Severity	Medium	**Status**	Fixed in https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/commit/2af615e8aa1f6ea1803dfacfc93e4692f6ffe54b

Description

The `BridgeRouter._send()` function incorrectly uses the `to` address specified by the caller of `GateKeeper.sendData()` as the destination address:

```
string memory destinationAddress =
    Strings.toHexString(uint160(uint256(params.to)), 20);

// Encode destination address and data for Router Protocol
bytes memory payload = abi.encode(destinationAddress, params.data);
```

However, the actual destination should be the contract address stored in the `receivers` mapping (set via `BridgeRouter.setReceiver()`), which points to the `ReceiverRouter` contract on the destination chain that implements the `iReceive()` method and handles the cross-chain message.
Using the caller-provided `to` address bypasses this logic and may result in messages not being delivered or processed correctly.

Recommendation

We recommend using the receiver addresses stored in the `receivers` mapping when sending messages.

Client's Commentary:

Fixed at b0a6776be7de3991321ad8d3b4bb18e510c0e3eb

M-4	Cross-Chain Transfer May Revert Due to Missing Fee in _send() Function		
Severity	Medium	**Status**	Fixed in https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/commit/2af615e8aa1f6ea1803dfacfc93e4692f6ffe54b

Description

The BridgeRouter._send() function may fail when attempting to initiate a cross-chain transfer through the Router Protocol Gateway because the required fee is not forwarded. When the Gateway's iSendDefaultFee is non-zero, it enforces a minimum native funds payment:

```
if (msg.value < iSendDefaultFee) revert Utils.C03();
```

However, BridgeRouter._send() calls IGateway(gateway).iSend() without attaching any value, which leads to a revert on all cross-chain messages when a fee is required.
This issue is classified as a **Medium** severity issue because it prevents cross-chain transfers from being sent when a non-zero iSendDefaultFee is set on the Gateway.

Recommendation

We recommend forwarding the iSendDefaultFee to the iSend() call during BridgeRouter._send() execution:

```
--  IGateway(gateway).iSend(
++  IGateway(gateway).iSend{value: iSendDefaultFee}(
        routerVersion,
        routerRouteAmount,
        routerRouteRecipient,
        destChainId,
        options_,
        payload
);
```

Client's Commentary:

Fixed at 924941a4c9305d78e2c34e612be3891882c171c2

2.4 Low

L-1	getRequestMetadata() Helper Not Implemented in BridgeRouter Contract		
Severity	Low	**Status**	Fixed in https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/commit/2af615e8aa1f6ea1803dfacfc93e4692f6ffe54b

Description

According to the Router Protocol's documentation, the requestMetadata struct can be encoded both on-chain and off-chain. Currently, this struct is encoded within the options parameter, which is passed by the caller of the GateKeeper.sendData() function. However, it is clearer and more maintainable to implement a dedicated getter function and use the resulting data directly within the BridgeRouter._send() function.
The implementation can be found here:
https://docs.routerprotocol.com/develop/message-transfer-via-crosstalk/evm-guides/iDapp-functions/iSend/#5-requestmetadata

Recommendation

We recommend implementing a getter function that returns the requestMetadata struct. This will enable on-chain encoding in a standardized and transparent manner.

Client's Commentary:

Fixed at b8598642fb20a36d577440ecf7dedd1510252fca

L-2	Missing iAck Handling in BridgeRouter Contract		
Severity	Low	**Status**	Fixed in https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/commit/2af615e8aa1f6ea1803dfacfc93e4692f6ffe54b

Description

When contract calls are executed on the destination chain, the Gateway contract on that chain sends an acknowledgment back to the Router Chain. iDapps have the option to receive this acknowledgment on the source chain and execute logic based on it.
The Router Protocol documentation states:

> To process the acknowledgment of their requests on the source chain, developers must implement an iAck function in their source chain contracts.

Currently, callers of the GateKeeper.sendData() function are allowed to provide arbitrary requestMetadata (currentOptions array). As a result, they can specify a non-zero ackType, implying that the iDapp intends to process acknowledgments - but the BridgeRouter contract does not implement the iAck function to handle such acknowledgments.
More information related to the requestMetadata can be found here: https://docs.routerprotocol.com/develop/message-transfer-via-crosstalk/evm-guides/iDapp-functions/iSend/#5-requestmetadata

Recommendation

We recommend either implementing the iAck function in the BridgeRouter contract to handle acknowledgments properly, or restricting the caller to only specify a zero ackType in the requestMetadata.

Client's Commentary:

Fixed at a7502f44f57a62852abc9ca5b2506306685996c8

L-3	estimateGasFee() Reverts When Executor Is Not Configured		
Severity	Low	**Status**	Fixed in https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/commit/2af615e8aa1f6ea1803dfacfc93e4692f6ffe54b

Description

The GateKeeper.estimateGasFee() function does not validate whether the caller is associated with a configured executor, nor does it check the isAutoExecutable flag before estimating the execution fee:

```
uint256 executeFee = IExecutorFeeManager(
    executors[msg.sender]
).estimateExecutorGasFee(
    chainIdTo,
    currentOptions[currentOptions.length - 1]
);
```

In contrast, the GateKeeper.sendData() function includes a proper check:

```
if (isAutoExecutable[msg.sender]) {
    executeFee = _payForExecute(
        requestId,
        chainIdTo,
        currentOptions[currentOptions.length - 1]
    );
}
```

Due to the missing validation, calls to GateKeeper.estimateGasFee() - such as from EywaDVN.getFee() - may revert if no executor has been configured for the caller, breaking gas estimation for otherwise valid requests.

Recommendation

We recommend adding proper validation in GateKeeper.estimateGasFee() to match the logic in GateKeeper.sendData().

Client's Commentary:

Fixed at b6d313e7439508a9b979adacf1518c0b339d35bf

L-4	Use of Revert Strings Instead of Custom Errors		
Severity	Low	**Status**	Acknowledged

Description

All contracts within the audit scope use hardcoded revert strings for error handling. Since Solidity version 0.8.4, custom errors can be used as a more efficient alternative.
Using custom errors reduces bytecode size and significantly saves gas, especially in cases where error strings are long or frequently used.

Recommendation

We recommend replacing revert strings with custom errors to improve gas efficiency and maintain cleaner code.

Client's Commentary:

We initially adopted revert strings before custom errors became widely used, prioritizing clarity and ease of debugging. To maintain a consistent coding style across the codebase, we continued with this approach.

L-5	Centralization Risks		
Severity	Low	**Status**	Acknowledged

Description

The `OPERATOR_ROLE` holds excessive privileges and control over critical protocol parameters, creating significant centralization risks. They can modify bridge configurations, fee structures, threshold settings, and system states across all bridge implementations.
The current architecture relies heavily on operator trust, which contradicts the decentralized principles expected from cross-chain protocols. A malicious or compromised operator could severely affect the protocol's functionality, security, and overall user experience.

Recommendation

We recommend implementing a DAO-based governance model or using a multi-signature wallet to distribute control among multiple trusted parties.

Client's Commentary:

We're aware of the centralization risks associated with the current OPERATOR_ROLE. As part of our planned improvements, we intend to transition operator responsibilities to a multisig setup to reduce trust assumptions and improve security.

L-6	Typographical Errors in Comments of GateKeeper and Receiver Contracts		
Severity	Low	**Status**	Fixed in https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/commit/2af615e8aa1f6ea1803dfacfc93e4692f6ffe54b

Description

There are a couple of typos in comments/parameter names in the scope of the project:

- GateKeeper.sol, line 45: comment uses conract instead of contract.
- Receiver.sol, line 144: the parameter is written as sende instead of sender.

This issue is classified as **Low** severity because the misspellings exist only in comments and parameter documentation, not in executable logic.

Recommendation

We recommend fixing the listed typos to improve code clarity and maintainability. Optionally, add a linter or spell-check step for comments/documentation to catch similar issues automatically in future reviews.

Client's Commentary:

Fixed at b26808739dccc32acf008038617ebb5a8e85b8d5

L-7	Unused and Misapplied Imports in ReceiverRouter and BridgeRouter		
Severity	Low	**Status**	Fixed in https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/commit/2af615e8aa1f6ea1803dfacfc93e4692f6ffe54b

Description

The code contains imports that are either mistakenly added and unused or improperly applied, leading to code clutter and, in the case of BridgeRouter, a mismatch between expected API/interface usage and actual implementation relative to the Router Protocol integration. In detail:

- **contracts/bridge/receive/ReceiverRouter.sol**
  - AxelarExpressExecutable – mistakenly added and unused (no Axelar logic in this contract).
  - AddressToString – imported alongside StringToAddress but not used.
- **contracts/bridge/send/BridgeRouter.sol**
  - Utils – mistakenly added and unused.
  - IDapp – imported but not used. The contract is intended to work via Router Protocol and **should inherit from** IDapp, but currently does not, risking divergence from integration specifications.

Recommendation

We recommend:

1. Removing the unused imports AxelarExpressExecutable and AddressToString from ReceiverRouter.sol.
2. Removing the unused Utils import from BridgeRouter.sol.
3. Ensuring that BridgeRouter.sol inherits from IDapp as required by the Router Protocol integration.

Client's Commentary:

Fixed at 2af615e8aa1f6ea1803dfacfc93e4692f6ffe54b

L-8	Fee Estimation Requires Dummy Nonce in Options Configuration		
Severity	Low	**Status**	Fixed in https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/commit/3173d41cc28dc7f8b5b2dc23175ce4839b2afbc3

Description

`GateKeeper.estimateGasFee` requires the last element of `currentOptions` to be an external nonce for decoding. Currently, `EywaDVN.getFee` passes `options` directly without appending a nonce, meaning operators must pre-configure `options` with a trailing dummy nonce element for estimation to work. While this currently functions because the DVN uses only EywaBridge (whose `estimateGasFees` expects `uint32` options), future changes to bridge usage could introduce incompatibilities. The code could be refactored so that if `options` already includes a dummy nonce element, both `assignJob` and `getFee` can reuse it.

Recommendation

We recommend ensuring operators configure `EywaDVN.options` to include a trailing dummy nonce slot. Additionally, `EywaDVN.assignJob` could be refactored to copy `options` into `currentOptions` and replace the last element with the actual nonce, rather than allocating a new array with `optionsLength + 1` elements.

Client's Commentary

Fixed in https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/commit/0009916dbc4a6405e977cc000c31d7cbd662c5e0

L-9	Missing Zero-Address Check in Public Withdraw Function		
Severity	Low	**Status**	Fixed in https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/commit/3173d41cc28dc7f8b5b2dc23175ce4839b2afbc3

Description

The `withdrawFees` function is publicly accessible and transfers assets to the `treasury` address. If `treasury` is not set when fees have already accumulated, calling this function will send ETH to the zero address or attempt an ERC20 transfer to `address(0)`, resulting in permanently lost funds.

Recommendation

We recommend adding `require(treasury != address(0), "GateKeeper: treasury not set");` at the start of the `withdrawFees` function to prevent accidental fund loss.

Client's Commentary

Fixed in https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/commit/41062ef4dc273ec7655913af84833adbb647c0a2

L-10	Misspelled Interface Name and Missing Inheritance		
Severity	Low	**Status**	Fixed in https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/commit/3173d41cc28dc7f8b5b2dc23175ce4839b2afbc3

Description

The interface `IValidatedDataReciever` is named with an incorrect spelling of "Receiver". Additionally, `EywaDVN` implements the `receiveValidatedData` function but does not explicitly inherit from `IValidatedDataReciever`, which reduces code clarity and integration safety.

Recommendation

We recommend renaming the interface to `IValidatedDataReceiver` to fix the typo and updating `EywaDVN` to explicitly inherit from the corrected interface.

Client's Commentary

Fixed in https://gitlab.ubertech.dev/blockchainlaboratory/eywa-cdp/-/commit/3173d41cc28dc7f8b5b2dc23175ce4839b2afbc3

3. About MixBytes

MixBytes is a leading provider of smart contract audit and research services, helping blockchain projects enhance security and reliability. Since its inception, MixBytes has been committed to safeguarding the Web3 ecosystem by delivering rigorous security assessments and cutting-edge research tailored to DeFi projects.

Our team comprises highly skilled engineers, security experts, and blockchain researchers with deep expertise in formal verification, smart contract auditing, and protocol research. With proven experience in Web3, MixBytes combines in-depth technical knowledge with a proactive security-first approach.

Why MixBytes

- **Proven Track Record:** Trusted by top-tier blockchain projects like Lido, Aave, Curve, and others, MixBytes has successfully audited and secured billions in digital assets.
- **Technical Expertise:** Our auditors and researchers hold advanced degrees in cryptography, cybersecurity, and distributed systems.
- **Innovative Research:** Our team actively contributes to blockchain security research, sharing knowledge with the community.

Our Services

- **Smart Contract Audits:** A meticulous security assessment of DeFi protocols to prevent vulnerabilities before deployment.
- **Blockchain Research:** In-depth technical research and security modeling for Web3 projects.
- **Custom Security Solutions:** Tailored security frameworks for complex decentralized applications and blockchain ecosystems.

MixBytes is dedicated to securing the future of blockchain technology by delivering unparalleled security expertise and research-driven solutions. Whether you are launching a DeFi protocol or developing an innovative dApp, we are your trusted security partner.

Contact Information

 https://mixbytes.io/

 https://github.com/mixbytes/audits_public

 hello@mixbytes.io

 https://x.com/mixbytes