DIA MULTI SCOPE SECURITY AUDIT REPORT

April 15, 2025

MixBytes()

TABLE OF CONTENTS

1. INTRODUCTION 3

1.1 Disclaimer 3

1.2 Security Assessment Methodology 3

1.3 Project Overview 7

1.4 Project Dashboard 8

1.5 Summary of findings 11

1.6 Conclusion 14

2.FINDINGS REPORT 15

2.1 Critical 15

C-1 Potential Old Data Processing 15

C-2 Message Verification 16

2.2 High 18

H-1 Misconfiguration in the Oracle Removal Process 18

H-2 Incomplete Sorting Mechanism in Median Price Calculation 19

H-3 Issue with Fee Payment During Interchain Callback 20

H-4 Potential DoS of messages in the `postDispatch` function 21

2.3 Medium 22

M-1 Unvalidated Cross-Chain Request Parameters in RequestOracle 22

M-2 Missing `DIAOracleV2Meta.setThreshold()` and `DIAOracleV2Meta.setTimeoutSeconds()` Parameters Validation 23

M-3 Lack of Uniqueness Check in `DIAOracleV2Guardian.addGuardian()` and `DIAOracleV2Meta.addOracle()` 24

M-4 Inconsistent Key Formats 25

M-5 Misleading Return Value in Guardian Value Retrieval 26

M-6 Zero Value Handling in DIAOracleV2Guardian 27

M-7 Missing ISM validation in `RequestOracle.handle()` and `PushOracleReceiver.handle()` 28

2.4 Low 29

L-1 Unnecessary Function 29

L-2 Inefficient Gas Price Derivation 30

L-3 Redundancy in Storage 31

L-4 Unused InterchainSecurityModule in OracleTrigger Contract 32
L-5 Unused `DEFAULT_ADMIN_ROLE` 33
L-6 Lack of Transparency 34
L-7 Insufficient Role Management in Dispatch Function 35
L-8 Inadequate Access Control in `removeOwner` Function 36
L-9 Missing Zero Address Verification 37
L-10 Absence of Event Emission in Oracle Addition 38
L-11 Lack of Revert Reason in `DIAOracleV2` Contract 39
L-12 Missing Check for Allowed Timestamp Range in `setValue` Function 40
L-13 Lack of Value Range Checks in the `setValue` Function 41
L-14 Unnecessary Complexity due to Mod Operation in `setMultipleValues` Function 42
L-15 `oracleUpdater` Address Two-Step Update 43
L-16 Gas Optimization in Guardian Removal Function 44
L-17 Redundancy in Contract Module Importing 45
L-18 Redundant DIAOracleV2metadata Across Repositories 46
L-19 Misleading Payment Hook Naming in RequestOracle 47
L-20 Redundant Event Emission in Ism Contract Methods 48
L-21 Immutable Admin Configuration in DIAOracleV2Meta 49
L-22 Potential Ether Accumulation Risk in Payable Contracts 50
L-23 Unnecessary Import Statements 51
L-24 Potential Missing Interface Implementation 52
L-25 Potential PUSH0 Opcode Limitation 53
L-26 Message Processing in `ProtocolFeeHook` 54
L-27 Input Parameter Validation in DIAOracleV2Guardian 55
L-28 Emergency Pause Mechanism for RequestOracle 56
L-29 Decimal Adjustment Safety in DIAOracleV2Guardian 57
L-30 Limitations in ETH Transfer Prevention Mechanism 59
L-31 Missing Chain Deactivation Functionality in the `OracleTrigger` contract 60
L-32 Missing Chain Existence Check in `addChain` Function 61
L-33 Missing Revert on Non-Existent Oracle Removal 62
L-34 Missing Duplicate Key Check in `setMultipleValues` 63

3. ABOUT MIXBYTES 64

1. INTRODUCTION

1.1 Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of the Client. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 Security Assessment Methodology

A group of auditors are involved in the work on the audit. The security engineers check the provided source code independently of each other in accordance with the methodology described below:

1. Project architecture review:

- Project documentation review.
- General code review.
- Reverse research and study of the project architecture on the source code alone.

Stage goals
- Build an independent view of the project's architecture.
- Identifying logical flaws.

2. Checking the code in accordance with the vulnerabilities checklist:

- Manual code check for vulnerabilities listed on the Contractor's internal checklist. The Contractor's checklist is constantly updated based on the analysis of hacks, research, and audit of the clients' codes.
- Code check with the use of static analyzers (i.e Slither, Mythril, etc).

Stage goal

Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flash loan attacks etc.).

3. Checking the code for compliance with the desired security model:

- Detailed study of the project documentation.
- Examination of contracts tests.
- Examination of comments in code.
- Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit.
- Exploits PoC development with the use of such programs as Brownie and Hardhat.

Stage goal

Detect inconsistencies with the desired model.

4. Consolidation of the auditors' interim reports into one:

- Cross check: each auditor reviews the reports of the others.
- Discussion of the issues found by the auditors.
- Issuance of an interim audit report.

Stage goals

- Double-check all the found issues to make sure they are relevant and the determined threat level is correct.
- Provide the Client with an interim report.

5. Bug fixing & re-audit:

- The Client either fixes the issues or provides comments on the issues found by the auditors. Feedback from the Customer must be received on every issue/bug so that the Contractor can assign them a status (either "fixed" or "acknowledged").
- Upon completion of the bug fixing, the auditors double-check each fix and assign it a specific status, providing a proof link to the fix.
- A re-audited report is issued.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Client with a re-audited report.

6. Final code verification and issuance of a public audit report:

- The Customer deploys the re-audited source code on the mainnet.
- The Contractor verifies the deployed code with the re-audited version and checks them for compliance.
- If the versions of the code match, the Contractor issues a public audit report.

Stage goals

- Conduct the final check of the code deployed on the mainnet.
- Provide the Customer with a public audit report.

Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss of funds.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Low	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

1.3 Project Overview

The DIA Oracle protocol is a cross-chain price feed and data delivery system leveraging Hyperlane's interoperability infrastructure. The system comprises three primary components: the decentral-feeder for price data aggregation, Spectra-interoperability for cross-chain message routing, and Lumina Guardians for multi-source value validation.

1.4 Project Dashboard

Project Summary

Title	Description
Client	DIA
Project name	DIA Multi Scope
Timeline	24.02.2025 - 04.04.2025
Number of Auditors	4

Project Log

Date	Commit Hash	Note
24.02.2025	e2717e8cc403463b9cf511767eb1fa0ab3c68d37	Commit for the audit (decentral-feeder)
24.02.2025	ed9f1e5ff3aa6cfba02d12f0bed1e435aeec24c1	Commit for the audit (Spectra-interoperability)
24.02.2025	472cd6c81b4d9f9f295ae6b56a4d958d8b0c0183	Commit for the audit (lumina-guardians)
19.03.2025	310f12cf35a50179a2eb90228be1ea3c741743a8	Commit for the re-audit (decentral-feeder)
19.03.2025	4699c617cfc90508a184d57e12bb2d2ab8c3ddc6	Commit for the re-audit (Spectra-interoperability)
19.03.2025	8324f1a4e7152f553228af9f93302c2ed4b12308	Commit for the re-audit (lumina-guardians)
26.03.2025	1df0598c2fe19cbf28443cb0a6197b579da3f3a0	Commit for the re-audit 2 (decentral-feeder)
26.03.2025	8bd76d490fa1527d3f5cf1c9bab0ac9fba771e42	Commit for the re-audit 2 (Spectra-interoperability)

Date	Commit Hash	Note
26.03.2025	5bc4a1e7945ded80bcaec167acf35755a75828cf	Commit for the re-audit 2 (lumina-guardians)
26.03.2025	89eb5414aed3d8f6ec3b17940df00e0a4a1e7b5a	Commit with updates (Spectra-interoperability)

Project Scope

The audit covered the following files:

File name	Link
contracts/DIAOracleV2Meta.sol	DIAOracleV2Meta.sol
contracts/DIAOracleV2.sol	DIAOracleV2.sol
contracts/contracts/OracleRequestRecipient.sol	OracleRequestRecipient.sol
contracts/contracts/PushOracleReceiver.sol	PushOracleReceiver.sol
contracts/contracts/libs/TypeCasts.sol	TypeCasts.sol
contracts/contracts/libs/Message.sol	Message.sol
contracts/contracts/ProtocolFeeHook.sol	ProtocolFeeHook.sol
contracts/contracts/RequestOracle.sol	RequestOracle.sol
contracts/contracts/Ism.sol	Ism.sol
contracts/contracts/OracleTrigger.sol	OracleTrigger.sol
contracts/DIAOracleV2Guardian.sol	DIAOracleV2Guardian.sol
contracts/asset_registry.sol	asset_registry.sol

Deployments

File name	Contract deployed on mainnet	Comment
Ism.sol	0x92F0f4...4bBC8cBF	Arbitrum
StaticAggregationIsm.sol	0xB10f35...85d3F675	Arbitrum
StaticMultisigIsm.sol	0x9C4b29...91eA264B	Arbitrum
RequestOracle.sol	0xE74B7D...38266888	Arbitrum
PushOracleReceiver.sol	0xA42217...2D619bb0	Arbitrum
ProtocolFeeHook.sol	0x048050...CF2CbdbB	Arbitrum
Ism.sol	0xE74B7D...38266888	DIA Lasernet
StaticAggregationIsmFactory.sol	0x77494B...8fB3849d	DIA Lasernet (Used to deploy StaticAggregationIsm implementation)
StaticAggregationIsm.sol	0xa8Cb15...B8dB87c4	DIA Lasernet
StaticMessageIdMultisigIsmFactory.sol	0x14b323...bD7209C9	DIA Lasernet (Used to deploy StaticMultisigIsm implementation)
StaticMultisigIsm.sol	0x7bf531...c57AaA5A	DIA Lasernet
OracleTrigger.sol	0x92F0f4...4bBC8cBF	DIA Lasernet
OracleRequestRecipient.sol	0x203a83...8D93Ca4e	DIA Lasernet

1.5 Summary of findings

Severity	# of Findings
Critical	2
High	4
Medium	7
Low	34

ID	Name	Severity	Status
C-1	Potential Old Data Processing	Critical	Fixed
C-2	Message Verification	Critical	Fixed
H-1	Misconfiguration in the Oracle Removal Process	High	Fixed
H-2	Incomplete Sorting Mechanism in Median Price Calculation	High	Fixed
H-3	Issue with Fee Payment During Interchain Callback	High	Fixed
H-4	Potential DoS of messages in the `postDispatch` function	High	Fixed
M-1	Unvalidated Cross-Chain Request Parameters in RequestOracle	Medium	Fixed
M-2	Missing `DIAOracleV2Meta.setThreshold()` and `DIAOracleV2Meta.setTimeoutSeconds()` Parameters Validation	Medium	Fixed
M-3	Lack of Uniqueness Check in `DIAOracleV2Guardian.addGuardian()` and `DIAOracleV2Meta.addOracle()`	Medium	Fixed
M-4	Inconsistent Key Formats	Medium	Fixed

M-5	Misleading Return Value in Guardian Value Retrieval	Medium	Acknowledged
M-6	Zero Value Handling in DIAOracleV2Guardian	Medium	Fixed
M-7	Missing ISM validation in `RequestOracle.handle()` and `PushOracleReceiver.handle()`	Medium	Fixed
L-1	Unnecessary Function	Low	Fixed
L-2	Inefficient Gas Price Derivation	Low	Fixed
L-3	Redundancy in Storage	Low	Fixed
L-4	Unused InterchainSecurityModule in OracleTrigger Contract	Low	Fixed
L-5	Unused `DEFAULT_ADMIN_ROLE`	Low	Fixed
L-6	Lack of Transparency	Low	Fixed
L-7	Insufficient Role Management in Dispatch Function	Low	Fixed
L-8	Inadequate Access Control in `removeOwner` Function	Low	Fixed
L-9	Missing Zero Address Verification	Low	Fixed
L-10	Absence of Event Emission in Oracle Addition	Low	Fixed
L-11	Lack of Revert Reason in `DIAOracleV2` Contract	Low	Fixed
L-12	Missing Check for Allowed Timestamp Range in `setValue` Function	Low	Acknowledged
L-13	Lack of Value Range Checks in the `setValue` Function	Low	Acknowledged
L-14	Unnecessary Complexity due to Mod Operation in `setMultipleValues` Function	Low	Acknowledged
L-15	`oracleUpdater` Address Two-Step Update	Low	Fixed
L-16	Gas Optimization in Guardian Removal Function	Low	Fixed
L-17	Redundancy in Contract Module Importing	Low	Fixed

L-18	Redundant DIAOracleV2metadata Across Repositories	Low	Fixed
L-19	Misleading Payment Hook Naming in RequestOracle	Low	Fixed
L-20	Redundant Event Emission in Ism Contract Methods	Low	Fixed
L-21	Immutable Admin Configuration in DIAOracleV2Meta	Low	Fixed
L-22	Potential Ether Accumulation Risk in Payable Contracts	Low	Fixed
L-23	Unnecessary Import Statements	Low	Fixed
L-24	Potential Missing Interface Implementation	Low	Fixed
L-25	Potential PUSH0 Opcode Limitation	Low	Fixed
L-26	Message Processing in `ProtocolFeeHook`	Low	Fixed
L-27	Input Parameter Validation in DIAOracleV2Guardian	Low	Acknowledged
L-28	Emergency Pause Mechanism for RequestOracle	Low	Fixed
L-29	Decimal Adjustment Safety in DIAOracleV2Guardian	Low	Fixed
L-30	Limitations in ETH Transfer Prevention Mechanism	Low	Fixed
L-31	Missing Chain Deactivation Functionality in the `OracleTrigger` contract	Low	Fixed
L-32	Missing Chain Existence Check in `addChain` Function	Low	Fixed
L-33	Missing Revert on Non-Existent Oracle Removal	Low	Fixed
L-34	Missing Duplicate Key Check in `setMultipleValues`	Low	Acknowledged

1.6 Conclusion

The primary objective of the audit was to ensure that interchain values are successfully delivered to the recipient chain without being tampered with during transmission. Apart from the findings presented in the report, we examined the following attack vectors and potential issues.

We verified the correct flow of message transmission in both scenarios: when the initiator is an actor from the DIA chain and when the initiator is from an external chain. The correctness of fee collection for message delivery was checked, along with the absence of griefing opportunities that could drain funds from the UserWallet when using the `PushOracleReceiver`. We also assessed an attack scenario where a malicious user artificially inflates `tx.gasprice`. However, this attack lacks strong economic incentives, making it an unlikely vector of exploitation. Additionally, we ensured that a malicious actor cannot perform a denial-of-service attack on oracle price delivery.

Protocol resilience and handling of unexpected situations were carefully evaluated. We assessed how delays in price updates (e.g., if DIA L2 stops processing blocks) could affect integrated protocols. Since this could create significant risks for price consumers, they must verify the price state (whether stale or not), which is properly handled in the current oracle implementation.

We thoroughly reviewed the integration with Hyperlane contracts to verify its correctness. We also modeled several interaction scenarios with different chains. The ISM implementation from the scope cannot be relied upon as the sole ISM, which is described in detail in the findings section. Instead, that ISM should be aggregated with an ISM that verifies relayers inside Hyperlane's IAggregationIsm implementation, with a threshold of at least 2.

Overall code correctness: We examined areas involving bitwise operations and type casting to prevent potential silent overflows. We ensured the correct and secure implementation of all methods that process strings.

Despite the overall architectural correctness, the findings section presents a set of recommendations that must be implemented for the protocol to function correctly. Unfortunately, we must acknowledge the current insufficient code quality and inadequate test coverage, which need to be addressed to improve the system's security and reliability.

2. FINDINGS REPORT

2.1 Critical

C-1	Potential Old Data Processing
Severity	Critical
Status	Fixed in 4699c617

Description

An issue has been discovered within the `handle` function of the `PushOracleReceiver` contract. The function is responsible for processing incoming oracle data messages. The issue arises due to the potential for an old message to be processed if it was previously reverted due to an out of gas error. If this happens, the oracle price will be updated, not with the most recent value, but with the old one, which could have potentially significant negative consequences on liquidations and integrated protocols.
This issue is classified as **Critical** severity as it could induce misleading pricing information, which could lead to inaccurate liquidation decisions or manipulation of integrated protocols.

Recommendation

We recommend checking the timestamp of the data in the `handle` function to further verify the recency of the oracle message. This would ensure that the data being processed represents the most current and thereby accurate market conditions, minimizing the chance that old, now-obsolete data will be used.

C-2	Message Verification
Severity	Critical
Status	Fixed in 89eb5414

Description

A critical security vulnerability has been identified within the `verify` function of the `ISM` (Interchain Security Module) contract. The ISM doesn't verify the caller of the `Mailbox` `process` function, which could allow untrusted addresses to pass an arbitrary message to the `Mailbox`, ultimately leading to potential manipulation of the data that will be processed.
There is also a dangerous parameter `allowAll`, which may lead to anyone malicious submitting oracle data if enabled.
This issue is classified as **Critical** because it could lead to manipulation of critical data processed by the `process` function, which could result in unpredictable disruptions or severe losses in the integrated protocols.

Recommendation

We recommend adding caller verification of the `Mailbox` contract within the `verify` function. It should ensure that the caller is indeed a trusted address. We also recommend removing the logic related to the `allowAll` flag as it may affect the protocol security badly.

Client's Commentary

> Client: ISM on DIA chain has allowAll enables, as same contract is used this function has kept here.
>
> On destination chain senderShould be is whitelisted
>
> MixBytes: There is a `process` function in the Hyperlane Mailbox contract, which doesn't have any restrictions for the caller. Anyone can craft the metadata and message values and pass them into the `process` function, which is then passed to the `verify` function of the ISM. So the attacker is able to specify any address, which is present in the `senderShouldBe` mapping. We recommend implementing `trustedRelayer` logic in the ISM as it is done in the Hyperlane example: TrustedRelayerIsm.sol. Keeping `allowAll` flag is also dangerous, as it may lead to unrestricted access to submitting crafted oracle data from the source chain.
>
> Client: We have an Custom ISM that verifies whether a sender is allowed to send messages. These senders are explicitly whitelisted. The Hyperlane protocol, by default, allows anyone to send messages and spoof the sender.
> We were initially recommended to add TrustedRelayerISM, but it turns out that this is not supported by the relayer and is not intended for use.

As a workaround, we implemented MultiSigISM, which exists in the Hyperlane repository: StaticMultisigIsm.sol

This ISM ensures that messages cannot be spoofed as long as they originate from trusted validators. To use both Custom ISM and MultiSigISM together, we implemented AggregateISM, which ensures that both ISMs validate the transaction before it is processed.
https://github.com/hyperlane-xyz/hyperlane-monorepo/tree/main/solidity/contracts/isms/aggregation

MixBytes: This issue was resolved by introducing standard Hyperlane implementations of the Aggregation ISM and Multisig ISM. The Multisig ISM is combined with the custom ISM module presented in the current scope. The Aggregation ISM threshold is expected to be set to 2 to account for both the Multisig ISM and custom ISM responses.

2.2 High

H-1	Misconfiguration in the Oracle Removal Process
Severity	High
Status	Fixed in 1df0598c

Description

During the audit, an issue was identified in the `removeOracle` function of the `DIAOracleV2Meta` contract. This function, intended to remove an oracle from the registry, appears to have an incorrect implementation. Presently, the code tries to replace the address of the oracle to be removed with the address of an oracle that doesn't exist (`oracles[numOracles]`). This has been marked as a **High** severity issue because, in its current form, the oracle removal is flawed, affecting the proper operation of the contract.

Recommendation

It is recommended to replace `oracles[numOracles]` with `oracles[numOracles - 1]`. This will point to the last valid oracle address in the array, ensuring that the action of removing an oracle functions as expected.

Client's Commentary

> Client: This fix was implemted as recommended.
> MixBytes: Oracle removal is partially fixed, as for now `removeOracle` function assigns `oracles[numOracles] = address(0);` to the non-existent element at the index `numOracles`, assuming that it is the last element of an array being moved to the position of the removed element. This line should be changed to `oracles[numOracles - 1] = address(0);`.
> Client: This issue is fixed in commit: PR-80

H-2	Incomplete Sorting Mechanism in Median Price Calculation
Severity	High
Status	Fixed in 1df0598c

Description

A **High** severity issue has been uncovered in the `getValue` function of the `DIAOracleV2Meta` contract. The `getValue` function, which retrieves the median price values for a given asset from the registered oracles, fails to correctly calculate the median due to an incomplete sorting mechanism in its array handling.
The function correctly disregards any values older than the defined threshold but does not properly organize the array of valid values. As a result, it cannot reliably return the median value from the gathered price data as the array may not be sorted in ascending order.

Recommendation

We recommend introducing a sorting mechanism in the `getValue` function. Before calculating the median, the function should ensure that the gathered valid values are properly sorted in some order.

Client's Commentary

> Client: Added a call to a quicksort function
> MixBytes: Median value is calculated incorrectly as the algorithm takes into account uninitialized (default to 0) array values, which were not set in the loop due to the failing check for the timeout threshold. An array of length `numOracles` is sorted and `medianIndex` is calculated based on the `validValues` amount, but uninitialized zeroes are present in the array at the first indices. We recommend changing how the `values` array is sorted - we may pass `left = 0` and `right = validValues - 1` as the boundaries to the `quickSort` function directly. It will produce an array where the first part is sorted and the rest elements are potentially zeroes. In that case the `medianIndex` will be correctly calculated at the line `uint256 medianIndex = validValues / 2;`.
> Also we highly recommend to implement fuzz testing for the Quicksort algorithm used in this contract to ensure its correctness and explore potential corner cases of this implementation. Additionally, this sorting algorithm has not been audited previously, making it even more important to ensure it is thoroughly covered with tests.
> Client: We fixed this issue in this commit: PR-80
> We also did a round of fuzz testing for the Quicksort implementation

H-3	Issue with Fee Payment During Interchain Callback
Severity	High
Status	Fixed in 4699c617

Description

In the `OracleRequestRecipient.handle()` function the `msg.value` parameter is missing OracleRequestRecipient.sol#L76 to `OracleTrigger.dispatch()`. As a result, `IMailbox(mailBox).dispatch()` will always revert, as the passed `msg.value` will be zero.

Also for the `OracleRequestRecipient.handle()` call to be executed with `msg.value > 0`, `Mailbox.process()` must also be called with `msg.value > 0`. This raises the question of how to ensure that relayers operate with an additional `msg.value`. It may be necessary to run custom relayers, but this approach seems suboptimal.

This issue has been assigned a **High** severity level because it breaks the core functionality of the `OracleRequestRecipient` contract, and the only way to fix it would be through a redeployment.

Recommendation

We recommend adding the `msg.value` to the call:

```
IOracleTrigger(oracleTriggerAddress).dispatch{value: msg.value}(
    _origin, sender, key);
```

Additionally, we recommend developing a mechanism to paying the fee when calling `OracleRequestRecipient.handle()`.

Client's Commentary

> Client: this is fixed
> MixBytes: There is an important notice regarding the Hyperlane relayers - the protocol should ensure that the used relayer passes `msg.value` together with the call to the `Mailbox.process` function or run own relayer which will provide the needed functionality.

H-4	Potential DoS of messages in the `postDispatch` function
Severity	High
Status	Fixed in 8bd76d49

Description

The `postDispatch()` function of the `ProtocolFeeHook` contract lacks access control check.

The function updates the `messageValidated` mapping and marks `messageId` as validated to prevent double-processing. But that function is defined as `external` without any checks for the caller address, so that anyone can call it and provide an actual `messageId` before its processing by the protocol, what will lead to not-accepting the original message when it is arrived.

The issue is classified as **High** severity, because it may lead to messages delivery DoS.

Recommendation

We recommend adding the access control check to the `postDispatch` function to ensure that it may be called only by the trusted `Mailbox`.

Client's Commentary

> Logic added to check if message is coming from trustedmailbox

2.3 Medium

M-1	Unvalidated Cross-Chain Request Parameters in RequestOracle
Severity	Medium
Status	Fixed in 8bd76d49

Description

The audit identified multiple security vulnerabilities in the `request` function of the `RequestOracle` contract:

Lack of Whitelisting:

- No checks in place to validate `_destinationDomain` and `receiver`
- Potential for users to send requests to non-existing chains
- Risk of unnecessary network load

Mailbox and Message Body Exposure:

- `_mailbox` and `_messageBody` variables can be controlled by users
- Possibility of calling `dispatch` method on an unofficial contract

The issues are classified as **Medium** severity due to the potential for system manipulation and unintended operations that could lead to data loss.

Recommendation

We recommend implementing the following security measures:

- Add a whitelist function to validate `_destinationDomain` and `receiver`
- Utilize `trustedMailBox` instead of user-provided `_mailbox`
- Implement robust sanity checks for `_messageBody`
- Create a whitelisting mechanism to prevent unauthorized operations
- Ensure only approved domains and receivers can initiate requests

Client's Commentary

> Client: Relayer takes only those chainid which is configured, same for mailbox
> MixBytes: We recommend fixing the mentioned issue as it is possible to have any updates made to the relayer logic, which may not include checks on the used Mailbox and destinationChain addresses. It is better to implement checks on the protocol side in order not to depend on the third-party protocol security measures.
> Client: Whitelist for receiver and origin is added and trustedMailbox is used instead of mailbox from user

M-2	Missing `DIAOracleV2Meta.setThreshold()` and `DIAOracleV2Meta.setTimeoutSeconds()` Parameters Validation
Severity	Medium
Status	Fixed in 310f12cf

Description

The issue has been located in the `setThreshold` and `setTimeoutSeconds` functions of the `DIAOracleV2Meta` contract. More specifically, it was found that the functions don't check for zero values in the `newThreshold` and `newTimeoutSeconds` parameters. As a result, if zero is assigned as a value, it causes the `getValue` function to revert and disrupts normal operation.

There also needs to be an upper bound on `newTimeoutSeconds`. In case of abnormally large `newTimeoutSeconds`, `(0, 0)` can bypass DIAOracleV2Meta.sol#L111-L113 so it would affect the median value.

Recommendation

We advise implementing a validation check in the `setThreshold` and `setTimeoutSeconds` functions to ensure that zero is not accepted as an assigned value and also `newTimeoutSeconds` is less than some reasonable value.

Client's Commentary

> These checks were added as recommended

M-3	Lack of Uniqueness Check in `DIAOracleV2Guardian.addGuardian()` and `DIAOracleV2Meta.addOracle()`
Severity	Medium
Status	Fixed in 8324f1a4

Description

The audit discovered an issue in the `addGuardian` function of the `DIAOracleV2Guardian` contract. The issue pertains to the absence of a check for the uniqueness of the `newGuardianAddress` for adding a new guardian. As a result, the contract function provides room for adding a guardian with the same address more than once, leading to the same address's results being counted multiple times. This oversight can potentially distort the contract's intended function and reduce its overall accuracy.

A similar issue was found in the `addOracle` function of the `DIAOracleV2Meta` contract. If the error results in multiple oracles with the same address being added, the median will be distorted.

Recommendation

It is recommended to incorporate a uniqueness check for the `DIAOracleV2Guardian.addGuardian()` and `DIAOracleV2Meta.addOracle()` functions. Such an addition ensures that each address can only be added once, preventing potential double-counting of its results and maintaining the integrity of the contract's intended function.

Client's Commentary

> This check has been added in both contracts.

M-4	Inconsistent Key Formats
Severity	Medium
Status	Fixed in 5bc4a1e7

Description

An inconsistency was found in the `getGuardedValue` function of the `DIAOracleV2Guardian` contract. The problem lies in the format of the key that is being passed to the `assetRegistry.getValue(...)` function. The format doesn't match the information set in the `asset_registry` constructor, leading to potential malfunctions. The keys set in the `asset_registry` constructor also contain addresses with capital letters, while `toHexString` function returns the string in lowercase format, containing the address, which is then used during the registry key construction. This may lead to the inability to find the corresponding guardian key.

Recommendation

We strongly recommend ensuring the consistency of key formats throughout all related functions and in the `asset_registry` constructor. Having a consistent data format could prevent possible deployment issues and ensure the smooth operation.

Client's Commentary

> This issue has been fixed by removing the constructor which introduced keys in the inconsistent format. This was used for development and not in production: PR-5

M-5	Misleading Return Value in Guardian Value Retrieval
Severity	Medium
Status	Acknowledged

Description

The `getGuardedValue` method returns `(0, 0)` when the minimum guardian threshold is not reached, which can be critically misinterpreted by other protocols:

- Returns a seemingly valid zero value instead of explicitly indicating a failure
- Other systems might process this as a legitimate oracle response
- Potential for incorrect financial calculations or decision-making
- Masks the actual state of insufficient guardian consensus

The issue is classified as **Medium** severity due to the potential for unintended protocol interactions and the risk of systems treating an invalid response as a valid data point.

Recommendation

We recommend modifying the `getGuardedValue` method to:

- Explicitly revert the transaction when the minimum guardian threshold is not met
- Consider adding an event or error code to signal insufficient guardian participation

Client's Commentary

> We specify the return value of (0,0) in the documentation comment above the function. The reason why we don't want to revert is that it could happen that the call to this function is only one of many in a more complex setup, with other instances of the contract returning correct data. In that case, a "revert" in any place would revert the entire transaction, whereas a return value of (0,0) can be handled by the caller.

M-6	Zero Value Handling in DIAOracleV2Guardian
Severity	Medium
Status	Fixed in 5bc4a1e7

Description

The `guardianDiffBips()` function of the `DIAOracleV2Guardian` contract lacks handling of zero value scenarios.

The function calculates the difference between two values in basis points (1/1000th of a percent) but fails to check if `value` is zero before performing division. If `value` is zero, the division operation `/ value` will cause the transaction to revert due to division by zero.

The issue is classified as **Medium** severity, focusing on proactive numerical robustness improvements.

Recommendation

We recommend implementing comprehensive zero value handling:

- Add explicit checks for zero input values
- Implement safe division and comparison mechanisms

Client's Commentary

> Client: Added a check and handling for the case of 0 value.
> MixBytes: There is a possibility where 0 is returned from the `guardianDiffBips` function and is accepted as the value passed the `maxDeviationBips` check, but that may not be true as that value should be also compared to the `currGuardianValue` and accepted only if there is no significant deviation. It is recommended to enhance the implemented check for zero value inside the `guardianDiffBips` function and compare it to the `currGuardianValue`.
> Client: Fixed by adding a boolean return to indicate success: PR-5

M-7	Missing ISM validation in `RequestOracle.handle()` and `PushOracleReceiver.handle()`
Severity	Medium
Status	Fixed in 8bd76d49

Description

This issue has been identified within the `handle()` function of `RequestOracle` and `PushOracleReceiver` contracts.

The `handle()` function only validates that the message sender is the trusted mailbox but does not verify if the Interchain Security Module has been properly configured. If the ISM is not set via `setInterchainSecurityModule()`, Hyperlane will use default ISM which does not validate the origin of messages.

This vulnerability could allow an attacker to send unauthorized messages from DIA chain, potentially leading to manipulation of oracle data.

The issue is classified as Medium severity because it requires specific conditions when ISM wasn't set.

Recommendation

We recommend adding a check in the `handle()` function to ensure ISM is not `address(0)`.

Client's Commentary

> Reverts if ISM is set as Zero Address

2.4 Low

L-1	Unnecessary Function
Severity	Low
Status	Fixed in 4699c617

Description

A non-utilized `request` function was identified in the `PushOracleReceiver` contract. The function is expected to dispatch an interchain message via a provided mailbox, but it isn't called or used anywhere else in the code. Having unused functions in the contract increases the contract's complexity and can potentially open up security vulnerabilities.

Recommendation

We recommend removing the unused `request` function from the contract code. This will reduce unnecessary complexity, thus making the contract easier to maintain and less prone to future vulnerabilities.

L-2	Inefficient Gas Price Derivation
Severity	Low
Status	Fixed in 4699c617

Description

The issue was identified in the `handle` function of the `PushOracleReceiver` contract. The `gasUsedPerTx` should be fetched from the `ProtocolFeeHook` contract to reduce possible problems in the future updates.

Recommendation

We recommend fetching the `gasUsedPerTx` value from the `ProtocolFeeHook` contract

L-3	Redundancy in Storage
Severity	Low
Status	Fixed in 4699c617

Description

During the audit of the `RequestOracle` smart contract, it was identified that the contract maintains two unused parameters - `lastCaller` and `lastCallMessage`.

Recommendation

We recommend removing these unused parameters to optimize the storage space of the contract.

L-4	Unused InterchainSecurityModule in OracleTrigger Contract
Severity	Low
Status	Fixed in 4699c617

Description

An issue of an unused variable was identified in the `OracleTrigger` contract. The `interchainSecurityModule` variable is supposed to be used in integration with the `MailBox` contract, but it is not used.

Recommendation

We recommend removing the `interchainSecurityModule` variable from the storage.

L-5	Unused `DEFAULT_ADMIN_ROLE`
Severity	Low
Status	Fixed in 4699c617

Description

During the audit of the `OracleTrigger` contract, an issue was identified within the constructor function. The issue pertains to an unused role, `DEFAULT_ADMIN_ROLE`, that is granted during the contract initialization process but is never used afterwards in the contract. Additionally, as the default admin role is not utilized, it's not removed even in the case of an owner's removal.

Recommendation

To enhance the clarity of the contract and avoid potential authorization confusion, we recommend not granting roles that are not used within the contract. If `DEFAULT_ADMIN_ROLE` is intended to be used for administrative purposes like owner removal, modifications should be made accordingly in the contract to use this role.

Client's Commentary

> this role has been added to allow owner to add/remove Dispatcher roles too

L-6	Lack of Transparency
Severity	Low
Status	Fixed in 4699c617

Description

A transparency-related issue has been located within the `updateChain` function of the `OracleTrigger` contract. Currently, when the recipient address for a certain chain is updated, only the new recipient address is emitted in the event `ChainUpdated`. The existing code does not emit or log the previous address, which can result in a lack of transparency in the event.

Recommendation

We recommend modifying the `ChainUpdated` event to include the old recipient address in addition to the new one.

L-7	Insufficient Role Management in Dispatch Function
Severity	Low
Status	Fixed in 4699c617

Description

The issue has been detected within the `dispatch` function of the `OracleTrigger` contract. Currently, the function is called from the `OracleRequestRecipient`, making it necessary to assign a more specific role to it.

Recommendation

We recommend setting up more stringent role management policies. Specifically, it would be advisable to assign a separate role only to the `OracleRequestRecipient` contract. This separation can aid in mending potential security loopholes.

L-8	Inadequate Access Control in `removeOwner` Function
Severity	Low
Status	Fixed in 4699c617

Description

Currently, all owners can remove any other owner in the `removeOwner` function of the `OracleTrigger` contract. This could potentially lead to unauthorized changes in ownership or unexpected loss of owner privileges if one owner malignantly removes others.

Recommendation

The recommendation is to implement stricter access controls and permissions in the `removeOwner` function. It would be better to allow only the `DEFAULT_ADMIN_ROLE` role to add and remove owners to prevent potential misuse of owner privileges.

L-9	Missing Zero Address Verification
Severity	Low
Status	Fixed in 310f12cf

Description

The issue is associated with the `addOracle` function in the `DIAOracleV2Meta` contract. Specifically, the function lacks a check for zero addresses when adding a new oracle. If an oracle with a zero address is added, it will trigger the `getValue` function to revert abruptly.

Recommendation

We recommend implementing a validation check specifically for zero addresses in the `addOracle` function.

Client's Commentary

> This check has been added.

L-10	Absence of Event Emission in Oracle Addition
Severity	Low
Status	Fixed in 310f12cf

Description

Our audit identified an issue within the `addOracle` function of the `DIAOracleV2Meta` contract. In the current implementation, when a new oracle is added to the registry, the function does not emit an event to signify this change. The lack of event emission makes it difficult to track updates to the oracle registry and weakens the transparency of contract operations.

Recommendation

We recommend the implementation of event emission upon the addition of a new oracle.

Client's Commentary

> The event was added

L-11	Lack of Revert Reason in `DIAOracleV2` Contract
Severity	Low
Status	Fixed in 310f12cf

Description

Our team identified an issue within the `setValue`, `setMultipleValues` and `updateOracleUpdaterAddress` functions of the `DIAOracleV2` contract. Specifically, these three functions use a `require` statement to validate that the message sender is the `oracleUpdater`. However, the `require` statement is utilized without associated revert messages. This absence can cause difficulties when analyzing a transaction that has been reverted due to this requirement not being met.

Recommendation

We recommend appending informative revert messages to all `require` statements.

Client's Commentary

> The messages were added in all three places.

L-12	Missing Check for Allowed Timestamp Range in `setValue` Function
Severity	Low
Status	Acknowledged

Description

An issue was identified in the `setValue` function of the `DIAOracleV2` contract. Notably, the function does not validate the permissible range of the timestamp parameter. This oversight could lead to a situation where the timestamp could be placed far into the future or the past, which is not the correct or expected behavior.

Recommendation

We recommend adding a check in the `setValue` function that validates the timestamp parameter's range. Specifically, it should be ensured that the timestamp is neither far into the future nor in the past.

Client's Commentary

> This was intentionally left unchecked, as this affects the business logic of the payload. There are scenarios where backfilling can be needed and/or clocks of feeder nodes might not run in sync.
>
> Also, the guardian system and later checks by the consumer contracts can be used to prevent unwanted consumption of too old/too new data.

L-13	Lack of Value Range Checks in the `setValue` Function
Severity	Low
Status	Acknowledged

Description

An issue regarding inadequate constraints has been discovered within the `setValue` function of the `DIAOracleV2` contract. Specifically, the function does not have any controls in place to limit or regulate the range of the `value` parameter. Consequently, there can be instances where a zero or highly volatile `value` might be set, leading to potentially misleading or nonsensical price updates.

Recommendation

We recommend adding appropriate checks to regulate the acceptable range for the `value` parameter within the `setValue` function. It would be pertinent to disallow zero values and to create a mechanism for the admin to configure the maximum acceptable price deviation.

Client's Commentary

> The value parameter depends on the business logic of the data hosted within, and we can't predict all possible scenarios. So a plausibility check needs to be done on the consumer side, which is also one of the reasons why the guardian system exists.

L-14	Unnecessary Complexity due to Mod Operation in `setMultipleValues` Function
Severity	Low
Status	Acknowledged

Description

The issue is discovered within the `setMultipleValues` function of the `DIAOracleV2` contract. In the code, values are retrieved from the `compressedValues` array, and are split into two parts using the mod operation `% 2**128`, which is used for timestamp calculation. This operation is unnecessary and adds unnecessary complexity without providing any significant benefits.

Recommendation

In an effort to reduce the complexity of the code, we advise rewriting the line `(uint128)(currentCvalue % 2**128)` to a simpler version: `(uint128)(currentCvalue)`.

Client's Commentary

> We would like to keep this expression in the code for better readability and a more intuitive understanding of how the packing works without relying on (undefined?) behaviour of the casting call. We could not find a definitive definition of what happens when a uint256 is casted to a uint128 (i.e. which bits "survive") so that might change in future versions of the compiler and thus we'd like to keep the explicit modulo call.

L-15	`oracleUpdater` Address Two-Step Update
Severity	Low
Status	Fixed in 310f12cf

Description

The issue has been detected within the `updateOracleUpdaterAddress` function of the `DIAOracleV2` contract.
The problem arises because the function lacks a two-step address transfer mechanism, making it risk-prone. Currently, there's no safeguard in place to confirm that a new updater address has indeed acknowledged and accepted the role.

Recommendation

We recommend updating the `oracleUpdater` in a two-step process. The new address should explicitly accept the role to ensure that the updated address is valid and authorized to carry out the updater duties.

Client's Commentary

> This is intentional to be able to also update to the 0 address in order to "archive" an oracle feeder. Our system has later points where it can handle such errors (e.g. by removing the "blackholed" oracle from the Meta contract).

L-16	Gas Optimization in Guardian Removal Function
Severity	Low
Status	Fixed in 8324f1a4

Description

During the audit of the `DIAOracleV2Guardian` contract, a coding inefficiency was discovered in the `removeGuardian` function. Specifically, the function continues to execute even after the necessary address has been removed. This redundant operation leads to unnecessary gas usage, which is thereby making the contract operations less cost efficient.

Recommendation

For optimization purposes and efficient gas utilization, we recommend rewriting the function to exit as soon as the desired address has been removed.

Client's Commentary

> A return has been added.

L-17	Redundancy in Contract Module Importing
Severity	Low
Status	Fixed in 8324f1a4

Description

The local versions of the `Ownable` and `Context` contracts in the `StringKeyValueStore` contract is unnecessary as can be seamlessly imported from the OpenZeppelin library.

Recommendation

It is advised to remove the local implementations of the `Ownable` and `Context` contracts and instead import them directly from the OpenZeppelin library.

Client's Commentary

> The recommended imports were added

L-18	Redundant DIAOracleV2metadata Across Repositories
Severity	Low
Status	Fixed in 4699c617

Description

The `DIAOracleV2metadata` is currently present in two different repositories: Spectra-interoperability and decentral-feeder. This redundancy can lead to potential inconsistencies and maintenance challenges.

Recommendation

We recommend implementing a minimal contract interface for `DIAOracleV2metadata` in the Spectra interoperability module, instead of duplicating the full source code.

L-19	Misleading Payment Hook Naming in RequestOracle
Severity	Low
Status	Fixed in 4699c617

Description

The payment hook in the RequestOracle contract is named `merkleTree`, which is semantically incorrect and misleading. Currently, there is no actual implementation or logic related to Merkle trees associated with this hook.

Recommendation

We recommend renaming the payment hook to a more descriptive and accurate name that reflects its actual functionality.

L-20	Redundant Event Emission in Ism Contract Methods
Severity	Low
Status	Fixed in 4699c617

Description

The `setSenderShouldBe` and `setAllowAll` methods in the Ism contract lack checks to prevent setting the same value multiple times. This oversight can result in unnecessary event emissions and increased gas consumption.

When these methods are called with a value identical to the existing one, they will still trigger events and modify contract state, even though no meaningful change occurs.

The issue is classified as **Low** severity because it does not directly impact the contract's core functionality or security but represents an inefficiency in contract design.

Recommendation

We recommend adding value comparison checks before state modification:

- Implement a pre-check to compare the new value with the existing one
- Only emit events and modify state if the new value differs from the current value

L-21	Immutable Admin Configuration in DIAOracleV2Meta
Severity	Low
Status	Fixed in 310f12cf

Description

The current implementation of `DIAOracleV2Meta` sets the admin during contract initialization with no mechanism for future admin changes. Specifically:

- The admin is set to `msg.sender` during contract deployment
- There is no function to transfer or change the admin role
- This creates a potential long-term governance inflexibility

The issue is classified as **Low** severity because it does not immediately compromise system security but restricts future contract governance and upgradability.

Recommendation

We recommend adopting the OpenZeppelin `Ownable` implementation to:

- Provide a standardized, secure method for admin/ownership transfer
- Implement `transferOwnership` and `renounceOwnership` functions
- Improve contract flexibility and maintainability

Client's Commentary

> openzeppelin "Ownable" was implemented and the custom admin functions were removed.

L-22	Potential Ether Accumulation Risk in Payable Contracts
Severity	Low
Status	Fixed in 4699c617

Description

Multiple contracts in the protocol have payable functions with no explicit mechanism to handle accumulated Ether:

- Contracts can receive Ether without a clear withdrawal strategy
- Potential risk of unintended Ether accumulation
- Impacts contracts including:
  - OracleRequestRecipient
  - OracleTrigger
  - RequestOracle

While no immediate Ether locking is present, the current design creates a potential risk for future fund management complications.

The issue is classified as **Low** severity because there is no immediate fund loss, but it represents a potential future operational risk.

Recommendation

We recommend implementing a robust Ether management strategy that:

- Provides a clear mechanism for handling received Ether
- Includes authorized withdrawal functions

L-23	Unnecessary Import Statements
Severity	Low
Status	Fixed in 4699c617

Description

Redundant import statements were identified in the `OracleTrigger.sol` contract:

- Imported interfaces `IInterchainGasPaymaster` and `IPostDispatchHook` are not used in the contract

The issue is classified as **Low** severity because it does not impact contract functionality or security, but represents a minor code organization concern.

Recommendation

We recommend:

- Removing unused import statements
- Only importing interfaces and libraries that are actively used in the contract

L-24	Potential Missing Interface Implementation
Severity	Low
Status	Fixed in 310f12cf

Description

The `DIAOracleV2` contract appears to lack explicit implementation of its corresponding interface:

- No inheritance from the `IDIAOracleV2` interface

The issue is classified as **Low** severity because while it does not immediately compromise contract functionality, it represents a potential design oversight that could lead to future maintainability challenges.

Recommendation

We recommend:

- Explicitly implementing the `IDIAOracleV2` interface
- Ensuring all required interface methods are correctly implemented

Client's Commentary

> The interface has been added.

L-25	Potential PUSH0 Opcode Limitation
Severity	Low
Status	Fixed in 4699c617

Description

The contract uses Solidity compiler version 0.8.0, which may generate bytecode with PUSH0 opcodes:

- Solc 0.8.20 defaults to Shanghai EVM version
- PUSH0 opcode not supported on all blockchain networks
- Risk of deployment failures on certain L2 chains or alternative networks

The issue is classified as **Low** severity because it does not affect contract functionality but could prevent successful deployment on some blockchain environments.

Recommendation

We recommend:

- Explicitly specify an EVM version compatible with target deployment chains
- Test deployment across intended blockchain networks
- Verify deployment compatibility before mainnet launch

Client's Commentary

> Client: solc is updated to 0.8.29, Can you share chain for which deployment may be issue, currently deployment is tested in
> Unichain, Sepolia, Optimism Sepolia,Arbitrum Sepolia
> MixBytes: There are some EVM-compatible networks like EOS Network (https://docs.eosnetwork.com/evm/miscellaneous/evm-compatibility/) and IOTA EVM which don't support PUSH0 opcode yet. It is important to take into account that there may be new networks without support for the mentioned opcode.

L-26	Message Processing in `ProtocolFeeHook`
Severity	Low
Status	Fixed in 8bd76d49

Description

The `ProtocolFeeHook.postDispatch()` function could benefit from improved message processing practices.

According to Hyperlane's documentation (https://docs.hyperlane.xyz/docs/reference/hooks/overview), post-dispatch hooks can be "replayable," meaning the same message could be processed multiple times. The contract should implement a mechanism to prevent message replay, typically using a message ID tracking system with a modifier like `validateMessageOnce`.

Without this protection, an attacker could potentially replay the same message multiple times, leading to unintended consequences such as repeated fee payments or other side effects dependent on message processing.

The issue is classified as **Low** severity, focusing on proactive design improvements rather than an immediate security threat.

Recommendation

We recommend implementing a robust message processing mechanism:

- Add a message ID tracking system
- Create a modifier to validate message uniqueness
- Implement a mapping to track processed message IDs

Client's Commentary

> Every message is processed only once, modifier validateMessageOnce is added

L-27	Input Parameter Validation in DIAOracleV2Guardian
Severity	Low
Status	Acknowledged

Description

The `DIAOracleV2Guardian` contract could enhance its input validation for critical parameters:

- Lack of comprehensive bounds checking for `maxDeviationBips`
- Potential for setting extreme or invalid parameter values

Key considerations:

- Ensuring parameter values remain within logical ranges
- Preventing potential system manipulation
- Enhancing overall contract reliability

The issue is classified as **Low** severity, focusing on design improvements and best practices.

Recommendation

We recommend implementing robust input validation:

- Add range checks for `maxDeviationBips`, `maxTimestampAge`, and `numMinGuardianMatches`
- Ensure parameters fall within logically defined boundaries
- Provide clear error messages for invalid inputs

L-28	Emergency Pause Mechanism for RequestOracle
Severity	Low
Status	Fixed in 4699c617

Description

The `RequestOracle` contract lacks an emergency pause mechanism for user-triggered function `request`:

- Limited administrative control during unexpected scenarios

Key considerations:

- Providing administrators with rapid incident response capabilities
- Mitigating potential risks from arbitrary user interactions

The issue is classified as **Low** severity, focusing on proactive system design improvements.

Recommendation

We recommend implementing a targeted pause mechanism:

- Integrate OpenZeppelin's `Pausable` pattern
- Apply pause functionality specifically to the `request` function
- Create admin-controlled pause and unpause methods
- Ensure only user-triggered functions can be paused

L-29	Decimal Adjustment Safety in DIAOracleV2Guardian
Severity	Low
Status	Fixed in 8324f1a4

Description

The `guardianDiffBips()` function could benefit from enhanced decimal adjustment safety:

- Potential for arithmetic overflow during decimal normalization
- Concerns with large decimal place differences

Key considerations:

- Preventing potential numerical instability
- Ensuring robust decimal place conversion

```
function guardianDiffBips(
    uint128 value,
    uint128 guardianValue,
    uint8 oracleDecimals,
    uint8 guardianDecimals
) private pure returns (uint256) {
    // Get synchronised decimals for guardian and oracle value
    if (oracleDecimals > guardianDecimals) {
       // Oracle has more decimals
       guardianValue *=
        uint128(10 ** (oracleDecimals - guardianDecimals));
    } else if (oracleDecimals < guardianDecimals) {
       // Guardian has more decimals
       value *=
        uint128(10 ** (guardianDecimals - oracleDecimals));
    }

    // Check deviation and return in bps unit...
}
```

The function adjusts values to normalize decimal places by multiplying by powers of 10. However, there's no check to ensure that the resulting value after multiplication doesn't exceed the maximum value representable in a `uint128`. If the difference between `oracleDecimals` and `guardianDecimals` is large enough, the multiplication by `10 ** (difference)` can overflow the `uint128` type.

The issue is classified as **Low** severity, focusing on proactive numerical safety improvements.

Recommendation

We recommend implementing robust decimal adjustment techniques:

- Add overflow protection mechanisms
- Implement limits on decimal place differences

Client's Commentary

> Client: Decimals were confined to 0 <= x < 30

L-30	Limitations in ETH Transfer Prevention Mechanism
Severity	Low
Status	Fixed in 4699c617

Description

The `OracleRequestRecipient` contract's `receive()` function attempts to prevent direct ETH transfers through a revert mechanism:

- Current implementation can be bypassed using `selfdestruct`
- Ineffective protection against forcible ETH sending
- Potential for unintended ETH accumulation

The issue is classified as **Low** severity, focusing on the inherent challenges of preventing ETH transfers in Solidity.

Recommendation

We recommend adding a function to withdraw any ETH that might have been forcibly sent via `selfdestruct`.

L-31	Missing Chain Deactivation Functionality in the `OracleTrigger` contract
Severity	Low
Status	Fixed in 8bd76d49

Description

This issue has been identified within the `OracleTrigger` contract. While the contract provides functions to add and update chains (`addChain` and `updateChain`), there is no function to deactivate or remove a chain configuration. Without a deactivation mechanism, obsolete or compromised chains cannot be disabled, posing a potential security risk and reducing contract flexibility.

The issue is classified as **Low** severity because it does not immediately compromise the system's integrity but may introduce operational inefficiencies or vulnerabilities in the long term.

Recommendation

We recommend implementing a `removeChain` function that allows authorized addresses to deactivate a chain.

Client's Commentary

> Added chain delete function

L-32	Missing Chain Existence Check in `addChain` Function
Severity	Low
Status	Fixed in 4699c617

Description

This issue has been identified within the `addChain` function of the `OracleTrigger` contract. Currently, there is no validation to check whether a chain configuration already exists before adding a new one. This could unintentionally overwrite an existing configuration, leading to potential disruptions in interchain messaging.

The issue is classified as **Low** severity because it does not immediately compromise security but could cause operational inefficiencies or misconfigurations.

Recommendation

We recommend adding a check to ensure the chain ID does not already exist in the `chains` mapping. If the chain is already configured, the function should revert.

L-33	Missing Revert on Non-Existent Oracle Removal
Severity	Low
Status	Fixed in 310f12cf

Description

This issue has been identified within the `removeOracle` function of the `DIAOracleV2Meta` contract.

Currently, if an oracle address to be removed does not exist in the registry, the function silently returns without providing any feedback. This behavior can cause confusion and hinder troubleshooting.

The issue is classified as **Low** severity because it doesn't directly affect security but impacts contract usability and error detection.

Recommendation

We recommend adding a revert statement when the oracle address is not found in the registry. This would provide immediate feedback and prevent silent failures, improving the overall robustness of the contract.

Client's Commentary

> A revert was added

L-34	Missing Duplicate Key Check in `setMultipleValues`
Severity	Low
Status	Acknowledged

Description

This issue has been identified within the `setMultipleValues` function of the `DIAOracleV2` contract.

Currently, the function does not check for duplicate keys in the `keys` array. If duplicate keys are passed, the later values will overwrite the earlier ones without any warning. This could lead to unintended data loss or inconsistencies in the stored asset values.

The issue is classified as **Low** severity because it affects data integrity but does not pose an immediate security risk.

Recommendation

We recommend adding a check to revert the transaction if duplicates are found.

Client's Commentary

> As this fix would result in higher gas costs for each call, we will not implement this check in the oracle contract

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build opensource solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

Contacts

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://twitter.com/mixbytes