PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For LayerZero Aptos Endpoint V2

14 Jan 2025

paladinsec.co   info@paladinsec.co

Table of Contents

Table of Contents 2

Disclaimer 10

1 Overview 11

- 1.1 Summary 11
- 1.2 Contracts Assessed 12
- 1.3 Findings Summary 16
  - 1.3.1 Global 17
  - 1.3.2 endpoint_v2::channels 17
  - 1.3.3 endpoint_v2::store 17
  - 1.3.4 endpoint_v2::timeout 18
  - 1.3.5 endpoint_v2::messaging_receipt 18
  - 1.3.6 endpoint_v2::msglib_manager 18
  - 1.3.7 endpoint_v2::endpoint 18
  - 1.3.8 endpoint_v2::admin 19
  - 1.3.9 endpoint_v2::registration 19
  - 1.3.10 endpoint_v2::messaging_composer 19
  - 1.3.11 endpoint_v2_common::assert_no_duplicates 19
  - 1.3.12 endpoint_v2_common::bytes32 19
  - 1.3.13 endpoint_v2_common::config_eid_tagged 20
  - 1.3.14 endpoint_v2_common::contract_identity 20
  - 1.3.15 endpoint_v2_common::guid 20
  - 1.3.16 endpoint_v2_common::packet_raw 20
  - 1.3.17 endpoint_v2_common::packet_v1_codec 20
  - 1.3.18 endpoint_v2_common::send_packet 20
  - 1.3.19 endpoint_v2_common::serde 21
  - 1.3.20 router_node_0::router_node 21
  - 1.3.21 router_node_1::router_node 21
  - 1.3.22 blocked_msglib::router_calls 21
  - 1.3.23 simple_msglib::router_calls 21
  - 1.3.24 simple_msglib::msglib 21

1.3.25 msglib_3::router_calls 22
1.3.26 msglib_4::router_calls 22
1.3.27 uln_302::msglib 22
1.3.28 uln_302::router_calls 22
1.3.29 uln_302::admin 22
1.3.30 uln_302::verification 22
1.3.31 uln_302::sending 23
1.3.32 uln_302::uln_302_store 23
1.3.33 uln_302::configuration 23
1.3.34 uln_302::for_each_dvn 23
1.3.35 uln_302::assert_valid_default_uln_config 23
1.3.36 uln_302::assert_valid_uln_config 24
1.3.37 msglib_types::configs_executor 24
1.3.38 msglib_types::configs_uln 24
1.3.39 endpoint_v2_common::dvn_verify_params 24
1.3.40 msglib_types::worker_options 24
1.3.41 treasury::treasury 25
1.3.42 dvn_fee_lib_router_0::dvn_fee_lib_router 25
1.3.43 dvn_fee_lib_router_1::dvn_fee_lib_router 25
1.3.44 executor_fee_lib_router_0::executor_fee_lib_router 25
1.3.45 executor_fee_lib_router_1::executor_fee_lib_router 25
1.3.46 executor_fee_lib_0::executor_fee_lib 26
1.3.47 executor_fee_lib_0::executor_option 26
1.3.48 executor_fee_lib_1::executor_fee_lib 26
1.3.49 dvn_fee_lib_0::dvn_fee_lib 26
1.3.50 dvn_fee_lib_1::dvn_fee_lib 26
1.3.51 price_feed_module_0::feeds 27
1.3.52 price_feed_module_0::price 27
1.3.53 price_feed_module_0::eid_model_pair 27
1.3.54 price_feed_module_1::feeds 27
1.3.55 price_feed_module_2::feeds 27
1.3.56 price_feed_router_0::router 28
1.3.57 price_feed_router_1::router 28

1.3.58 worker_common::worker_config 28
1.3.59 worker_common::worker_config_store 28
1.3.60 worker_common::multisig 29
1.3.61 worker_common::multisig_store 29
2 Findings 30
2.1 Global 30
2.1.2 Issues & Recommendations 30
2.2 endpoint_v2::channels 36
2.2.1 Privileged Functions 36
2.2.2 Issues & Recommendations 37
2.3 endpoint_v2::store 40
2.3.1 Privileged Functions 40
2.3.2 Issues & Recommendations 41
2.4 endpoint_v2::timeout 43
2.4.1 Privileged Functions 43
2.4.2 Issues & Recommendations 44
2.5 endpoint_v2::messaging_receipt 46
2.5.1 Privileged Functions 46
2.5.2 Issues & Recommendations 46
2.6 endpoint_v2::msglib_manager 47
2.6.1 Privileged Functions 47
2.6.2 Issues & Recommendations 48
2.7 endpoint_v2::endpoint 52
2.7.1 Privileged Functions 52
2.7.2 Issues & Recommendations 53
2.8 endpoint_v2::admin 56
2.8.1 Privileged Functions 56
2.8.2 Issues & Recommendations 57
2.9 endpoint_v2::registration 59
2.9.1 Privileged Functions 59
2.9.2 Issues & Recommendations 60
2.10 endpoint_v2::messaging_composer 61
2.10.1 Privileged Functions 61

2.10.2 Issues & Recommendations 62
2.11 endpoint_v2_common::assert_no_duplicates 63
2.11.1 Privileged Functions 63
2.11.2 Issues & Recommendations 63
2.12 endpoint_v2_common::bytes32 64
2.12.1 Privileged Functions 64
2.12.2 Issues & Recommendations 65
2.13 endpoint_v2_common::config_eid_tagged 66
2.13.1 Privileged Functions 66
2.13.2 Issues & Recommendations 66
2.14 endpoint_v2_common::contract_identity 67
2.14.1 Privileged Functions 67
2.14.2 Issues & Recommendations 68
2.15 endpoint_v2_common::guid 70
2.15.1 Privileged Functions 70
2.15.2 Issues & Recommendations 70
2.16 endpoint_v2_common::packet_raw 71
2.16.1 Privileged Functions 71
2.16.2 Issues & Recommendations 72
2.17 endpoint_v2_common::packet_v1_codec 73
2.17.1 Privileged Functions 73
2.17.2 Issues & Recommendations 74
2.18 endpoint_v2_common::send_packet 75
2.18.1 Privileged Functions 75
2.18.2 Issues & Recommendations 76
2.19 endpoint_v2_common::serde 77
2.19.1 Privileged Functions 77
2.19.2 Issues & Recommendations 78
2.20 router_node_0::router_node 81
2.20.1 Privileged Functions 81
2.20.2 Issues & Recommendations 82
2.21 router_node_1::router_node 84
2.21.1 Privileged Functions 84

2.21.2 Issues & Recommendations 84
2.22 blocked_msglib::router_calls 85
2.22.1 Privileged Functions 85
2.22.2 Issues & Recommendations 85
2.23 simple_msglib::router_calls 86
2.23.1 Privileged Functions 86
2.23.2 Issues & Recommendations 87
2.24 simple_msglib::msglib 88
2.24.1 Privileged Functions 88
2.24.2 Issues & Recommendations 89
2.25 msglib_3::router_calls 90
2.25.1 Privileged Functions 90
2.25.2 Issues & Recommendations 90
2.26 msglib_4::router_calls 91
2.26.1 Privileged Functions 91
2.26.2 Issues & Recommendations 92
2.27 uln_302::msglib 93
2.27.1 Privileged Functions 93
2.27.2 Issues & Recommendations 94
2.28 uln_302::router_calls 95
2.28.1 Privileged Functions 95
2.28.2 Issues & Recommendations 96
2.29 uln_302::admin 97
2.29.1 Privileged Functions 97
2.29.2 Issues & Recommendations 97
2.30 uln_302::verification 98
2.30.1 Privileged Functions 98
2.30.2 Issues & Recommendations 99
2.31 uln_302::sending 100
2.31.1 Privileged Functions 100
2.31.2 Issues & Recommendations 101
2.32 uln_302::uln_302_store 105
2.32.1 Privileged Functions 105

2.32.2 Issues & Recommendations 106

2.33 uln_302::configuration 109

2.33.1 Privileged Functions 109

2.33.2 Issues & Recommendations 110

2.34 uln_302::for_each_dvn 112

2.34.1 Privileged Functions 112

2.34.2 Issues & Recommendations 113

2.35 uln_302::assert_valid_default_uln_config 114

2.35.1 Privileged Functions 114

2.35.2 Issues & Recommendations 115

2.36 uln_302::assert_valid_uln_config 116

2.36.1 Privileged Functions 116

2.36.2 Issues & Recommendations 117

2.37 msglib_types::configs_executor 119

2.37.1 Privileged Functions 119

2.37.2 Issues & Recommendations 119

2.38 msglib_types::configs_uln 120

2.38.1 Privileged Functions 120

2.38.2 Issues & Recommendations 121

2.39 endpoint_v2_common::dvn_verify_params 123

2.39.1 Privileged Functions 123

2.39.2 Issues & Recommendations 124

2.40 msglib_types::worker_options 125

2.40.1 Privileged Functions 125

2.40.2 Issues & Recommendations 126

2.41 treasury::treasury 129

2.41.1 Privileged Functions 129

2.41.2 Issues & Recommendations 130

2.42 dvn_fee_lib_router_0::dvn_fee_lib_router 133

2.42.1 Privileged Functions 133

2.42.2 Issues & Recommendations 133

2.43 dvn_fee_lib_router_1::dvn_fee_lib_router 134

2.43.1 Privileged Functions 134

2.43.2 Issues & Recommendations 134
2.44 executor_fee_lib_router_0::executor_fee_lib_router 135
2.44.1 Privileged Functions 135
2.44.2 Issues & Recommendations 135
2.45 executor_fee_lib_router_1::executor_fee_lib_router 136
2.45.1 Privileged Functions 136
2.45.2 Issues & Recommendations 136
2.46 executor_fee_lib_0::executor_fee_lib 137
2.46.1 Privileged Functions 137
2.46.2 Issues & Recommendations 138
2.47 executor_fee_lib_0::executor_option 143
2.47.1 Privileged Functions 143
2.47.2 Issues & Recommendations 144
2.48 executor_fee_lib_1::executor_fee_lib 145
2.48.1 Privileged Functions 145
2.48.2 Issues & Recommendations 145
2.49 dvn_fee_lib_0::dvn_fee_lib 146
2.49.1 Privileged Functions 146
2.49.2 Issues & Recommendations 147
2.50 dvn_fee_lib_1::dvn_fee_lib 150
2.50.1 Privileged Functions 150
2.50.2 Issues & Recommendations 150
2.51 price_feed_module_0::feeds 151
2.51.1 Privileged Functions 152
2.51.2 Issues & Recommendations 154
2.52 price_feed_module_0::price 163
2.52.1 Privileged Functions 163
2.52.2 Issues & Recommendations 164
2.53 price_feed_module_0::eid_model_pair 165
2.53.1 Privileged Functions 165
2.53.2 Issues & Recommendations 166
2.54 price_feed_module_1::feeds 167
2.54.1 Privileged Functions 167

2.54.2 Issues & Recommendations 167
2.55 price_feed_module_2::feeds 168
2.55.1 Privileged Functions 168
2.55.2 Issues & Recommendations 168
2.56 price_feed_router_0::router 169
2.56.1 Privileged Functions 169
2.56.2 Issues & Recommendations 169
2.57 price_feed_router_1::router 170
2.57.1 Privileged Functions 170
2.57.2 Issues & Recommendations 170
2.58 worker_common::worker_config 171
2.58.1 Privileged Functions 171
2.58.2 Issues & Recommendations 172
2.59 worker_common::worker_config_store 175
2.59.1 Privileged Functions 175
2.59.2 Issues & Recommendations 176
2.60 worker_common::multisig 184
2.60.1 Privileged Functions 184
2.60.2 Issues & Recommendations 185
2.61 worker_common::multisig_store 187
2.61.1 Privileged Functions 187
2.61.2 Issues & Recommendations 188

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team. Paladin retains the right to re-use any and all knowledge and expertise gained during the audit process, including, but not limited to, vulnerabilities, bugs, or new attack vectors. Paladin is therefore allowed and expected to use this knowledge in subsequent audits and to inform any third party, who may or may not be our past or current clients, whose projects have similar vulnerabilities. Paladin is furthermore allowed to claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for LayerZero Aptos Endpoint V2 on the Aptos network. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Project Name	LayerZero Aptos Endpoint V2
URL	https://www.layerzero.foundation/
Platform	Aptos
Language	Move
Preliminary	https://github.com/LayerZero-Labs/monorepo/tree/56e51be3e6232-9925b8cf4fe8302c981cc60edc4/packages/layerzero-v2/aptos/contracts → https://github.com/LayerZero-Labs/monorepo/tree/677ef898c6405f-24968366820dc477fbd6e90365/packages/layerzero-v2/aptos/contracts for OFT (TODO)
Resolution	https://github.com/LayerZero-Labs/monorepo/commit/6fd7d061f3afe512d44abf2feaffe7d662dafa69
Resolution 2	https://github.com/LayerZero-Labs/monorepo/tree/6fd7d061f3afe512d44abf2feaffe7d662dafa69/packages/layerzero-v2/aptos/contracts

1.2 Contracts Assessed

Name	Contract	Live Code Match
endpoint_v2::chann-els		PENDING
endpoint_v2::store		PENDING
endpoint_v2::timeo-ut		PENDING
endpoint_v2::messa-ging_receipt		PENDING
endpoint_v2::msgli-b_manager		PENDING
endpoint_v2::endpo-int		PENDING
endpoint_v2::admin		PENDING
endpoint_v2::regis-tration		PENDING
endpoint_v2::messa-ging_composer		PENDING
endpoint_v2_common-::assert_no_duplic-ates		PENDING
endpoint_v2_common-::bytes32		PENDING
endpoint_v2_common-::config_eid_tagge-d		PENDING
endpoint_v2_common-::contract_identit-y		PENDING
endpoint_v2_common-::guid		PENDING
endpoint_v2_common-::packet_raw		PENDING
endpoint_v2_common-::packet_v1_codec		PENDING
endpoint_v2_common-::send_packet		PENDING
endpoint_v2_common-::serde		PENDING

router_node_0::router_node	PENDING
router_node_1::router_node	PENDING
blocked_msglib::router_calls	PENDING
simple_msglib::router_calls	PENDING
simple_msglib::msglib	PENDING
msglib_3::router_calls	PENDING
msglib_4::router_calls	PENDING
uln_302::msglib	PENDING
uln_302::router_calls	PENDING
uln_302::admin	PENDING
uln_302::verification	PENDING
uln_302::sending	PENDING
uln_302::uln_302_store	PENDING
uln_302::configuration	PENDING
uln_302::for_each_dvn	PENDING
uln_302::assert_valid_default_uln_config	PENDING
uln_302::assert_valid_uln_config	PENDING
msglib_types::configs_executor	PENDING
msglib_types::configs_uln	PENDING
endpoint_v2_common::dvn_verify_params	PENDING
msglib_types::worker_options	PENDING
	PENDING

treasury::treasury	
dvn_fee_lib_router_0::dvn_fee_lib_router	PENDING
dvn_fee_lib_router_1::dvn_fee_lib_router	PENDING
executor_fee_lib_router_0::executor_fee_lib_router	PENDING
executor_fee_lib_router_1::executor_fee_lib_router	PENDING
executor_fee_lib_0::executor_fee_lib	PENDING
executor_fee_lib_0::executor_option	PENDING
executor_fee_lib_1::executor_fee_lib	PENDING
dvn_fee_lib_0::dvn_fee_lib	PENDING
dvn_fee_lib_1::dvn_fee_lib	PENDING
price_feed_module_0::feeds	PENDING
price_feed_module_0::price	PENDING
price_feed_module_0::eid_model_pair	PENDING
price_feed_module_1::feeds	PENDING
price_feed_module_2::feeds	PENDING
price_feed_router_0::router	PENDING
price_feed_router_1::router	PENDING
worker_common::worker_config	PENDING
worker_common::worker_config_store	PENDING
	PENDING

`worker_common::multisig`	
`worker_common::multisig_store`	PENDING

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
● Governance	2	-	-	2
● High	5	2	-	3
● Medium	7	6	-	1
● Low	29	9	-	20
● Informational	56	46	3	7
Total	**99**	**63**	**3**	**33**

Classification of Issues

Severity	Description
● Governance	Issues under this category are where the governance or owners of the protocol have certain privileges that users need to be aware of, some of which can result in the loss of user funds if the governance's private keys are lost or if they turn malicious, for example.
● High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
● Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
● Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
● Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 Global

ID	Severity	Summary	Status
01	GOV	OApps need to opt-out of LayerZero governance	ACKNOWLEDGED
02	HIGH	A compromised LayerZero governance can DoS all OApps due to a low-level imperfection in the extendable router pattern	ACKNOWLEDGED
03	INFO	Library re-configurations can cause race conditions for messages which are already in flight	ACKNOWLEDGED
04	INFO	Aptos endpoint V2 has a near identical design to the LayerZero endpoint and therefore suffers from some of the same design trade-offs	ACKNOWLEDGED
05	INFO	Typographical issues	✓ RESOLVED

1.3.2 endpoint_v2::channels

ID	Severity	Summary	Status
06	LOW	Nillified packets can still be burned even though comments state otherwise	ACKNOWLEDGED
07	INFO	Duplicate checks can waste gas	✓ RESOLVED
08	INFO	Inconsistent checking of whether payload hash exists prior to accessing it	✓ RESOLVED
09	INFO	Typographical issues	✓ RESOLVED

1.3.3 endpoint_v2::store

ID	Severity	Summary	Status
10	HIGH	A compromised LayerZero governance can DoS all OApps due to an imperfection in the global storage architecture	✓ RESOLVED
11	INFO	Typographical issues	✓ RESOLVED

1.3.4 endpoint_v2::timeout

ID	Severity	Summary	Status
12	LOW	Use of block height instead of timestamp can lead to inconsistencies in Timeout expiries	ACKNOWLEDGED
13	INFO	Typographical issues	✓ RESOLVED

1.3.5 endpoint_v2::messaging_receipt

No issues found.

1.3.6 endpoint_v2::msglib_manager

ID	Severity	Summary	Status
14	LOW	A positive grace_period can be provided in set_default_receive_library without it actually being used	ACKNOWLEDGED
15	LOW	get_registered_libraries() response can be too large and run out of gas	✓ RESOLVED
16	INFO	The timeout setting functions emit a zero address old library	✓ RESOLVED
17	INFO	Unnecessary validation that timeout is active prior to removing	ACKNOWLEDGED
18	INFO	Typographical issues	✓ RESOLVED

1.3.7 endpoint_v2::endpoint

ID	Severity	Summary	Status
19	MEDIUM	Lack of enforcement that messages get cleared during execution makes things difficult for executors, potentially causing them to accidentally double spend	✓ RESOLVED
20	LOW	Lack of validation when creating alerts can result in spam	ACKNOWLEDGED
21	INFO	Typographical issues	✓ RESOLVED

1.3.8 endpoint_v2::admin

ID	Severity	Summary	Status
22	INFO	admin not explicitly dropped for set_eid and set_zro	✓ RESOLVED
23	INFO	Typographical issues	✓ RESOLVED

1.3.9 endpoint_v2::registration

ID	Severity	Summary	Status
24	LOW	Module string names have no cap on length, which can impact gas consumption	✓ RESOLVED
25	INFO	Typographical issues	✓ RESOLVED

1.3.10 endpoint_v2::messaging_composer

ID	Severity	Summary	Status
26	LOW	Lack of public view function to inspect whether a compose message hash exists	✓ RESOLVED
27	INFO	Typographical issues	✓ RESOLVED

1.3.11 endpoint_v2_common::assert_no_duplicates

No issues found.

1.3.12 endpoint_v2_common::bytes32

ID	Severity	Summary	Status
28	INFO	Typographical issues	ACKNOWLEDGED

1.3.13 endpoint_v2_common::config_eid_tagged

No issues found.

1.3.14 endpoint_v2_common::contract_identity

ID	Severity	Summary	Status
29	LOW	Multiple identity generation can cause contract identities to be compromised if the underlying module address signer is compromised	✓ RESOLVED
30	INFO	Typographical issues	ACKNOWLEDGED

1.3.15 endpoint_v2_common::guid

No issues found.

1.3.16 endpoint_v2_common::packet_raw

ID	Severity	Summary	Status
31	INFO	Typographical issues	✓ RESOLVED

1.3.17 endpoint_v2_common::packet_v1_codec

ID	Severity	Summary	Status
32	INFO	Typographical issues	✓ RESOLVED

1.3.18 endpoint_v2_common::send_packet

ID	Severity	Summary	Status
33	INFO	Typographical issues	✓ RESOLVED

1.3.19 endpoint_v2_common::serde

ID	Severity	Summary	Status
34	INFO	extract_uint is insecure for bytes equal to zero.	✓ RESOLVED
35	INFO	Typographical issues	✓ RESOLVED

1.3.20 router_node_0::router_node

ID	Severity	Summary	Status
36	HIGH	Asymmetric library configurations can cause privilege escalation by LayerZero governance	✓ RESOLVED
37	INFO	Typographical issues and gas optimizations	✓ RESOLVED

1.3.21 router_node_1::router_node

No issues found.

1.3.22 blocked_msglib::router_calls

No issues found.

1.3.23 simple_msglib::router_calls

ID	Severity	Summary	Status
38	INFO	The token inputs for the send function are not actually reduced	✓ RESOLVED

1.3.24 simple_msglib::msglib

ID	Severity	Summary	Status
39	INFO	Typographical issues	✓ RESOLVED

1.3.25 msglib_3::router_calls

No issues found.

1.3.26 msglib_4::router_calls

ID	Severity	Summary	Status
40	INFO	Typographical issues	✓ RESOLVED

1.3.27 uln_302::msglib

ID	Severity	Summary	Status
41	LOW	verifiable may return true even if the actual commit_verification flow reverts	✓ RESOLVED

1.3.28 uln_302::router_calls

ID	Severity	Summary	Status
42	INFO	Typographical issues	✓ RESOLVED

1.3.29 uln_302::admin

No issues found.

1.3.30 uln_302::verification

ID	Severity	Summary	Status
43	MEDIUM	check_verifiable has exponential execution complexity on the number of optional DVNs, potentially bricking receipts	✓ RESOLVED
44	INFO	Typographical issues	✓ RESOLVED

1.3.31 uln_302::sending

ID	Severity	Summary	Status
45	LOW	Worker payment events used by the worker may contain insufficient information for the worker to easily verify payment	ACKNOWLEDGED
46	LOW	The use of indexing rather than DVN lookup for assigning DVN options can break when a default or OApp-specific ULN configuration is updated prior to the message being sent	ACKNOWLEDGED
47	INFO	Typographical issues	✓ RESOLVED

1.3.32 uln_302::uln_302_store

ID	Severity	Summary	Status
48	LOW	Store eid is only set to non-zero potentially periods after deployment	✓ RESOLVED
49	INFO	Typographical issues	✓ RESOLVED

1.3.33 uln_302::configuration

ID	Severity	Summary	Status
50	INFO	Unnecessary validation in get_executor_configcan lead to wasted gas	✓ RESOLVED
51	INFO	Typographical issues	✓ RESOLVED

1.3.34 uln_302::for_each_dvn

ID	Severity	Summary	Status
52	INFO	Typographical issues	ACKNOWLEDGED

1.3.35 uln_302::assert_valid_default_uln_config

ID	Severity	Summary	Status
53	INFO	Typographical issues	✓ RESOLVED

1.3.36 uln_302::assert_valid_uln_config

ID	Severity	Summary	Status
54	LOW	The non-duplicate check fails to detect if a DVN is both part of the required and optional DVNs, thus permitting a DVN to potentially contribute in both, doubling its confirmation power	ACKNOWLEDGED
55	INFO	Typographical issues	✓ RESOLVED

1.3.37 msglib_types::configs_executor

No issues found.

1.3.38 msglib_types::configs_uln

ID	Severity	Summary	Status
56	LOW	Lack of validation during ULN configuration construction	ACKNOWLEDGED
57	INFO	Typographical issues	✓ RESOLVED

1.3.39 endpoint_v2_common::dvn_verify_params

ID	Severity	Summary	Status
58	INFO	Typographical issues	PARTIAL

1.3.40 msglib_types::worker_options

ID	Severity	Summary	Status
59	LOW	Executional complexity of option grouping is exponential which could lead to gas usage concerns	✓ RESOLVED
60	INFO	extract_type_3_options and extract_legacy_options both function even if an empty array is provided	ACKNOWLEDGED
61	INFO	Typographical issues	✓ RESOLVED

1.3.41 treasury::treasury

ID	Severity	Summary	Status
62	INFO	get_fee rounds against the favor of the protocol	✓ RESOLVED
63	INFO	Unnecessary additional validation of ensure_primary_store_exists wastes gas	✓ RESOLVED
64	INFO	Typographical issues	✓ RESOLVED

1.3.42 dvn_fee_lib_router_0::dvn_fee_lib_router

No issues found.

1.3.43 dvn_fee_lib_router_1::dvn_fee_lib_router

No issues found.

1.3.44 executor_fee_lib_router_0::executor_fee_lib_router

No issues found.

1.3.45 executor_fee_lib_router_1::executor_fee_lib_router

No issues found.

1.3.46 executor_fee_lib_0::executor_fee_lib

ID	Severity	Summary	Status
65	HIGH	Fee with margin calculation can be in incorrect magnitude due to inconsistent denominations	ACKNOWLEDGED
66	MEDIUM	get_executor_fee does not use the effective price feed	✓ RESOLVED
67	LOW	The native_cap worker safeguard can be circumvented through many smaller transactions	ACKNOWLEDGED
68	LOW	get_executor_fee lacks validation of the executor options	ACKNOWLEDGED
69	INFO	Typographical issues	✓ RESOLVED

1.3.47 executor_fee_lib_0::executor_option

ID	Severity	Summary	Status
70	INFO	Typographical issues	✓ RESOLVED

1.3.48 executor_fee_lib_1::executor_fee_lib

No issues found.

1.3.49 dvn_fee_lib_0::dvn_fee_lib

ID	Severity	Summary	Status
71	HIGH	Fee with margin calculation can be in incorrect magnitude due to inconsistent denominations	ACKNOWLEDGED
72	MEDIUM	get_dvn_fee does not use the effective price feed	✓ RESOLVED
73	LOW	Calldata size is based on local multi-signature configuration instead of the remote one	✓ RESOLVED
74	INFO	Typographical issues	PARTIAL

1.3.50 dvn_fee_lib_1::dvn_fee_lib

No issues found.

1.3.51 price_feed_module_0::feeds

ID	Severity	Summary	Status
75	LOW	Gas models appear to be approximative and do not reflect current L2 models	ACKNOWLEDGED
76	LOW	Gas model selection contains testnet evaluations even on mainnet	ACKNOWLEDGED
77	LOW	Feed ownership cannot be directly modified	ACKNOWLEDGED
78	LOW	EID prices cannot be explicitly discontinued by a feed provider	ACKNOWLEDGED
79	LOW	Arbitrum traits cannot be configured per chain	ACKNOWLEDGED
80	LOW	estimate_fee_on_send lacks details on the timestamp of the data	ACKNOWLEDGED
81	INFO	Fees round in favor of the user	✓ RESOLVED
82	INFO	Typographical issues	PARTIAL

1.3.52 price_feed_module_0::price

ID	Severity	Summary	Status
83	INFO	Typographical issues	✓ RESOLVED

1.3.53 price_feed_module_0::eid_model_pair

ID	Severity	Summary	Status
84	INFO	The EidModelPair should have the storeability to be future proof	✓ RESOLVED

1.3.54 price_feed_module_1::feeds

No issues found.

1.3.55 price_feed_module_2::feeds

No issues found.

1.3.56 price_feed_router_0::router

No issues found.

1.3.57 price_feed_router_1::router

No issues found.

1.3.58 worker_common::worker_config

ID	Severity	Summary	Status
85	MEDIUM	Unsupported option types do not appear to be rejected	ACKNOWLEDGED
86	LOW	Fee library configuration lacks explicit validation that the fee library is designed for the designated worker type	ACKNOWLEDGED
87	INFO	Typographical issues	✓ RESOLVED

1.3.59 worker_common::worker_config_store

ID	Severity	Summary	Status
88	MEDIUM	Executor and DVN configurations cannot be reconfigured after they are set once	✓ RESOLVED
89	LOW	A worker can set itself as the price feed delegate	ACKNOWLEDGED
90	LOW	Configuration setters lack validation that the worker id is the correct type	ACKNOWLEDGED
91	LOW	Workers get initialized in the unpaused state, even though they cannot be fully configured yet	✓ RESOLVED
92	INFO	An intermediary configuration state exists where native_decimals_r-ate is unset	✓ RESOLVED
93	LOW	Significant usage of vector types which have an unbound gas specifi-cation	ACKNOWLEDGED
94	INFO	Typographical issues	✓ RESOLVED

1.3.60 worker_common::multisig

ID	Severity	Summary	Status
95	MEDIUM	The signature assertion functions could be vulnerable to hash mining	✓ RESOLVED
96	INFO	Typographical issues	✓ RESOLVED

1.3.61 worker_common::multisig_store

ID	Severity	Summary	Status
97	GOV	Worker owner (signer) has full control over multisig and can adjust its quorum and all verification properties without going through the multisig	ACKNOWLEDGED
98	INFO	Lacking check that the number of signers >= quorum after callingse-t_multisig_signers	✓ RESOLVED
99	INFO	Typographical issues	✓ RESOLVED

2 Findings

2.1 Global

The issues in this section apply to the protocol as a whole, and may involve more than one contract.

2.1.2 Issues & Recommendations

Issue #01	OApps need to opt-out of LayerZero governance
Severity	● GOVERNANCE
Description	As discussed in previous EVM-specific audits, LayerZero's "decentralization" is opt-out for OApps. This means that unless explicitly configured otherwise, OApps will rely on many of the default configurations of LayerZero, such as their default message libraries, DVNs and executors. This is beneficial as LayerZero is probably the best entitity to quickly change a configuration and patch a vulnerability, if for example a DVN signer is compromised. The flip side of this is that if the LayerZero governance is compromised, all OApps which have the defaults enabled will be compromised and can be drained through fake messages as well. It should be noted that OApps can at many levels fully opt out of this. Next, it should be noted that Aptos modules are upgradeable. LayerZero has indicated that they will mark all modules except for the router leafs as immutable. **We recommend users validate this for themselves on the deployments, as such validation of future deployments is out of scope of this audit.** Finally, it should be noted that we assume that the `@endpoint_v2` ownership is fully revoked and no further signers can be generated. If they can, this would seriously compromise the validity of the security of the endpoint. The client has already directly indicated that it will be revoked.
Recommendation	Consider writing a careful and comprehensive guide on how to opt-out of all defaults fully.

	Consider securing `@layerzero_admin` and all upgrade admins with absolute extreme care. The degree of control these governance roles possess is extreme and should not be discounted from a security perspective. We recommend a very carefully constructed multisig of independent parties which are unlikely to be compromised at the same time.
Resolution	● ACKNOWLEDGED The client will document this.

Issue #02	**A compromised LayerZero governance can DoS all OApps due to a low-level imperfection in the extendable router pattern**
Severity	● HIGH SEVERITY

Description

Throughout this audit, the router pattern is explained and used. It is a pattern introduced by the LayerZero team to address a shortcoming of all Move blockchains: *There's no dynamic dispatch*.

The router pattern addresses this by having immutable routers reference each other in a chain-like manner, with the last router remaining upgradeable. This way, new routers can be added over time by configuring the leaf router and subsequently making it immutable. Whenever such a leaf router is configured it will reference a new upgradeable leaf router.

This seemed like an amazingly clever pattern to get around the shortcomings of the Move language, but through careful gas analysis we realize it comes with a flip-side: *The cost of any Aptos transaction is proportional to the total bytecode size of all modules it touches.*

This means that as more routers get added and more modules are referenced from these routers, the cost of interacting with the original top-level router is expected to increase as well.

In normal usage, this does not appear to be a problem, since only a handful of routers and modules are expected to be added. However, when an exploiter compromises the LayerZero keys, they can drive this to the extreme.

In particular, it appears that an upgrade strategy exists which can increase the gas costs of ALL LayerZero sends and receives without any limit. We'll describe this strategy below:

1. Deploy a new upgradeable module at a new unused address.
2. Take the currently upgradeable module, upgrade it with a reference to the new upgradeable module. This new module has significant bytecode but does not reference any other modules.

3. Repeat 1 with the new module.

It does not appear that this loop has a limit or increases in cost over time. This means that the loop can be repeated forever and eventually top level calls will fully abort due to the dependency tree growing too large.

By doing this, an exploiter who manages to get a hold of the governance keys of LayerZero can effectively fully block sending and receiving for *all* Aptos OApps forever as long as they make everything immutable. Not a single app would be useable anymore, even those that do not seem to be dependant on LayerZero governance by setting all defaults to fixed configurations.

It should finally be noted that this issue applies to all dependencies of the endpoint or message libraries which are upgradeable. We recommend users ensure that all other deployed modules referenced in the endpoint (directly or indirectly) are therefore immutable as well.

Recommendation

This may be an inherent shortcoming of the router pattern, which makes this issue extremely tough to resolve. The only "clean" solution we see is to put the pattern upside-down by going for a hot-potato dynamic dispatch. However, this may have significant UX drawbacks, alongside with it just being an absolutely significant redesign which we would not be able to check under a normal resolution round.

If those drawbacks are too large, we recommend reaching out to the Aptos team to see if there is a clever way around this issue, as we do not see an immediate way to make the router pattern robust to this shortcoming.

Resolution

● ACKNOWLEDGED

The client plans to address this once Aptos introduces proper support for a solution. Given the risk of introducing significantly intrusive changes at this point, waiting for a clean, Aptos team supported solution does sound like the way to go.

Issue #03	Library re-configurations can cause race conditions for messages which are already in flight
Severity	● INFORMATIONAL
Description	The Aptos LayerZero V2 stack allows for OApps to reconfigure themselves across many layers of the stack: they can change their message libraries, change their workers, change their configurations and so forth. However, all configuration changes are by nature asynchronous in a multi-chain setting. When changes are made, certain messages may still be in flight while they were created with a previous version of the configuration. Or perhaps with a new version while the destination chain still employs an old version as its update has not propagated yet. If executors and DVNs are not able to resolve such changes, this may cause for such transactions to be stuck.
Recommendation	Consider carefully evaluating that all off-chain components are able to handle these race-conditions. Consider documenting best practices for such configuration changes and how to organize them.
Resolution	● ACKNOWLEDGED

Issue #04	Aptos endpoint V2 has a near identical design to the LayerZero endpoint and therefore suffers from some of the same design trade-offs
Severity	● INFORMATIONAL
Description	Several months ago, our team audited the EVM LayerZero endpoint V2 in three sections: • V2 • MessageLib • VerifierNetwork During this audit, we raised many trade-offs with the design of LayerZero V2. Through discussion, it was clear that many of these trade-offs were desired by the LayerZero team. This is reflected in these three audits as those issues got either `acknowledged` or `resolved` without any change being made. Example trade-offs relate to the channel design, the availability decisions (sending can become unavailable as soon as a single worker becomes malicious) and so forth. For brevity purposes, these acknowledged considerations will not be repeated within this report. We recommend the reader to navigate to these reports and read up on the issues which were not fix there to get a sense of the LayerZero V2 design considerations.
Recommendation	Consider reading through the original EVM report and consider whether any of the originally acknowledged design issues warrant fixing.
Resolution	● ACKNOWLEDGED The client has read through the considerations from previous reports and confirms that they are still comfortable with their high level design decisions.

Issue #05	Typographical issues
Severity	● INFORMATIONAL
Description	Throughout the codebase, `vector::empty()` and `vector[]` are both used. Consider consistently using one. They compile to slightly different implementations though the end result is the same. ——— Throughout the codebase, error codes are left unused, consider removing the unused ones. We have done our best to highlight them. ——— Throughout the codebase, certain variables get initialized after the module is deployed. We ask here and in most of the relevant sections in this report for the client to minimize this unitialized state and ideally fully remove it. It increases the attack surface without much benefit.
Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED Everything but the re-initialization comment have been addressed. The latter was not addressed due to limitations within Aptos Move.

2.2 endpoint_v2::channels

The `channels` module defines the logic when interacting with the router to calculate the cost of sending messages with specific params, interacting with the `msglib_router` to send messages, registering whitelisted pathways for oapps, setting the verification state (valid, invalid, nilify) for packets, along with other helper functions for checking the state of packets and other similar logic.

All functions are protected by the `friend` modifier so that only the `endpoint` module is able to call functions in this module.

2.2.1 Privileged Functions

None.

2.2.2 Issues & Recommendations

Issue #06	Nillified packets can still be burned even though comments state otherwise
Severity	● LOW SEVERITY
Description	Line 162 `/// Nillified packets cannot be burnt.` The comment above `nilify` indicates that after `nilify` is called, `burn` can no longer be called. However, looking at the actual implementation of these two functions, this path of action still appears possible.
Recommendation	In case this callpath is undesired, consider strengthening the checks within `burn` so that the `has_payload_hash` check is followed by a check that the payload hash is not nil.
Resolution	● ACKNOWLEDGED

Issue #07	Duplicate checks can waste gas
Severity	● INFORMATIONAL

Description	Within the `verify` function call, there are duplicate checks for whether a pathway is registered that resolve to calling `store::receive_pathway_registered` which waste gas.
	
	These are on line 257-261
	
	`store::assert_registered_pathway(`
	`    receiver,`
	`    src_eid,`
	`    sender`
	`);`
	
	and 264
	
	`assert!(verifiable(receiver, src_eid, sender, nonce), ENOT_VERIFIABLE);`

Recommendation	Consider removing the duplicate checks.
Resolution	✓ RESOLVED

Issue #08	Inconsistent checking of whether payload hash exists prior to accessing it
Severity	● INFORMATIONAL

Description	There is an inconsistent use of checking if a payload hash exists prior to accessing it (borrowing or removing) in `nilify` and `burn`.
	
	Line 175
	
	`assert!(inbound_payload_hash == payload_hash, EPAYLOAD_HASH_DOES_NOT_MATCH);`
	
	Line 188-189
	`assert!(store::has_payload_hash(receiver, src_eid, sender, nonce), ENO_PAYLOAD_HASH);`
	`let inbound_payload_hash = store::remove_payload_hash(receiver, src_eid, sender, nonce);`

Recommendation	Consider always first checking if the payload hash exists to keep the abort message consistent.
Resolution	✓ RESOLVED

Issue #09	Typographical issues
Severity	● INFORMATIONAL

Description	Line 15
	
	`use endpoint_v2_common::guid;`
	
	The only place where this is used is for a function which is already imported by itself.
	
	Lines 63 and 108
	
	`let latest_nonce = increment_outbound_nonce(sender, dst_eid, receiver);`
	`let latest_nonce = store::outbound_nonce(sender, dst_eid, receiver) + 1;`
	
	This variable name is semantically incorrect as it is actually the new nonce.
	
	Line 91
	
	`let guid = guid::compute_guid(latest_nonce, store::eid(), bytes32::from_address(sender), dst_eid, receiver);`
	
	This does not need to be recomputed if it can just be fetched from the previously existing `packet`. Consider copying its `guid` before it is moved into `msglib_router::send`.
	
	Line 210
	
	`while (current_nonce <= nonce) {`
	
	It is not immediately apparent why a `for` loop was not used here.
	
	Line 506
	
	`const EINVALID_MSGLIB_AUTH: u64 = 3;`
	
	This constant appears unused.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.3 endpoint_v2::store

The store module defines the GlobalStore struct which stores the global state referenced by the modules within endpoint_v2, and is stored at @endpoint_v2. The global state includes this endpoint's eid, instance of the ContractSigner from the contract_identity module for performing authentication, tables for registering oapps and composers, each with their respective states, reference to the zro token, and receive / send msg libraries.

There are separate send and receive msg libraries which are set for each eid in separate tables, including both default (msglibs_default_send_libs / msglibs_default_receive_libs) and OApp-specific set send and receive msg libraries (msglibs_send_libs / msglibs_receive_libs). These are all stored in tables with the key being the eid (u32) and the value being the address of the msg library. For both the default (msglibs_grace_period_default_receive_lib_config) and OApp-specific (msglibs_grace_period_receive_lib_config) receive msg libraries, there is an additional table which stores msg libs which can be used up until an expiry which is set using the Timeout struct from the timeout module.

This module exposes all the functionality required to read and set variables inside the GlobalStore.

It should be noted that after discussion with the client, the ability for the GlobalStore to parallelize has been moved out of scope of this audit. This is because the client said that an employee at Aptos indicated that the GlobalStore was fine from that perspective, and we found it extremely difficult to find documentation on the requirements to write parallelizable Aptos code.

2.3.1 Privileged Functions

None.

2.3.2 Issues & Recommendations

Issue #10	**A compromised LayerZero governance can DoS all OApps due to an imperfection in the global storage architecture**
Severity	● HIGH SEVERITY

Description

Lines 33-44

```
struct GlobalStore has key {
  eid: u32,
  contract_signer: ContractSigner,
  zro: Option<Object<Metadata>>,
  oapps: Table<address, OAppState>,
  composers: Table<address, ComposerState>,
  msglibs: vector<address>,
  msglibs_registered: Table<address, bool>,
  msglibs_default_send_libs: Table<u32, address>,
  msglibs_default_receive_libs: Table<u32, address>,
  msglibs_grace_period_default_receive_lib_config: Table-
<u32, Timeout>,
}
```

The `GlobalStore` struct appears quite safe at first glance, with the only notable concerns being parallelization as all apps access the same resource. However, upon closer inspection, a critical flaw can be detected within this `GlobalStore` singleton resource — its size is unbound. This is because the `msglibs` element has a variable size as it is of the `vector` type.

Although none of the OApps ever need to directly use this `msglibs` element, all OApps need to use the `GlobalStore` struct.

Even though nearly all access instructions within Move have a fixed gas cost, Aptos has a special gas schedule which charges separately for disk reads and writes. When fields of the `GlobalStore` are read and written to, this cost is levied and multiplied with the full size of the `GlobalStore` struct.

This specifically means that even if transactions never interact with the `msglibs` vector, these transactions will become more expensive if the LayerZero governance adds more and more `msglibs` over time.

In the worst case scenario, a compromised governance will simply add addresses to `msglibs` until the transaction cost to add more becomes unbearable (or too large for a single block/transaction).

If this happens, we expect that all transactions interacting with `GlobalStore` will have become prohibitavely expensive.

This issue has been rated as high as LayerZero is designed to allow OApps to opt-out of all "governed" components such as defaults. It should be noted that apps which do not opt-out of default configurations have much more to worry about when the LayerZero governance gets compromised. But since this is by design, that is raised as a governance concern while this, which is not by design, is raised as a high severity issue.

Recommendation

Consider moving the `msglibs` vector to a separate resource. Alternatively, a `TableWithLength` can be used, which is indexed with indices, essentially turning it into a vector.

Resolution

✓ RESOLVED

`msglibs` is now a table indexed by the array index.

Issue #11	**Typographical issues**
Severity	● INFORMATIONAL

Description

Lines 171 and 176

```
let channel_key = ChannelKey { remote_eid, remote_oapp:
remote_address };
```

There is an inline function which is supposed to be used for creating channel keys. It not consistently used on these lines.

Lines 523, 524 and 527

```
const EALREADY_REGISTERED: u64 = 1;
const EEID_ALREADY_SET: u64 = 2;
const EUNREGISTERED_MODULE: u64 = 5;
```

All of these error codes appear unused.

It is safer to initialize the `eid` immediately, as is thoroughly discussed in multiple sections of this report.

Recommendation

Consider fixing the typographical issues.

Resolution

✓ RESOLVED

2.4 endpoint_v2::timeout

The `timeout` module is a small submodule which defines the `Timeout` struct which represents the grace period given to an old receive msg library. It exposes functionality to create, modify, and examine `Timeout` structs. The expiry date for this struct is based on the block number rather than timestamp.

2.4.1 Privileged Functions

None.

2.4.2 Issues & Recommendations

Issue #12	**Use of block height instead of timestamp can lead to inconsistencies in Timeout expiries**
Severity	● LOW SEVERITY
Description	Line 38 `let expiry = get_current_block_height() + timeout_in_blocks;` The `timeout` module uses the block height to set the expiration for `Timeout` structs. This can lead to inconsistencies, as blocks are not always produced in the same amount of time.
Recommendation	Consider using the block timestamp for setting the expiry rather than the block number.
Resolution	● ACKNOWLEDGED The client has indicated that block intervals are sufficiently consistent on the chains they will deploy to.

Issue #13	Typographical issues
Severity	● INFORMATIONAL

Description	Line 9
	
	`/// The Grace Period conifguration is used when migrating the Receive message library, and it defines the window`
	
	"conifguration" should be "configuration".
	
	Line 42
	
	`public(friend) fun new_null_timeout(): Timeout {`
	
	This function appears unused.
	
	Line 52
	
	`self.expiry > 0 && self.expiry > get_current_block_heig-ht()`
	
	It is unclear why the `self.expiry > 0` check is added here.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.5 endpoint_v2::messaging_receipt

The `messaging_receipt` module is a simple utility module which defines the `MessagingReceipt` resource and is emitted whenever a message is sent by the endpoint.

Unlike other utility modules, the `MessagingReceipt` resources can not be freely created by any module. Instead, creation is exclusively done by the endpoint's `channel` module. However, `MessagingReceipt` instances can be freely copied by any module which receives one, though the data cannot be changed.

2.5.1 Privileged Functions

None.

2.5.2 Issues & Recommendations

No issues found.

2.6 endpoint_v2::msglib_manager

The `msglib_manager` module covers all the functionality for setting the default send and receive msg libraries for given `dst_eid` and `src_eid` , as well as setting the send and receive libraries for specified OApps.

The functions to set the default msg libraries (`set_default_receive_library_timeout`/ `set_default_send_library`) are only callable by the admin. There are additional functions for validating that passed msg libraries have been registered, and whether they support given eids.

2.6.1 Privileged Functions

None.

2.6.2 Issues & Recommendations

Issue #14	A positive grace_period can be provided in set_default_receive_library without it actually being used
Severity	● LOW SEVERITY
Description	The `set_default_receive_library` function can in certain cases allow caller to provide a `grace_period` larger than zero without it actually being used—specifically, when no default library is configured yet.
Recommendation	Consider asserting that the grace period is zero in such cases.
Resolution	● ACKNOWLEDGED

Issue #15	get_registered_libraries() response can be too large and run out of gas
Severity	● LOW SEVERITY
Description	Line 65 `public(friend) fun get_registered_libraries(): vector<address> {` The `get_registered_libraries` function returns a vector of unbound length. The length is dependent on how many libraries are added by the LayerZero governance. If third-party modules start using this function, a compromised governance can DoS those modules by registering a large amount of libraries.
Recommendation	Consider either capping the number of libraries that can be added, or documenting this limitation on the public view functions. It may make sense to add a length and index-based function.
Resolution	✓ RESOLVED The query now has pagination parameters.

Issue #16	The timeout setting functions emit a zero address old library
Severity	● INFORMATIONAL

Description	Lines 190, 257, 282 and 312
	
	`emit(DefaultReceiveLibraryTimeoutSet { eid: src_eid, old_lib: @0, expiry: 0 });`
	`emit(ReceiveLibraryTimeoutSet { receiver, eid: src_eid, old_lib: @0, timeout: 0 });`
	
	Both of these events appear to emit that the old library is zero. However, it seems that this is definitely not always the case.
	
	This issue will also be resolved if this old library parameter has the semantical meaning of the timeout's currently configured library. Setting it to `@0` is then accurate as the timeout is removed.

Recommendation	Consider emitting the old library for this parameter.
Resolution	✓ RESOLVED

Issue #17	Unnecessary validation that timeout is active prior to removing
Severity	● INFORMATIONAL
Description	Line 310 `assert!(timeout::is_active(&timeout), ENO_TIMEOUT_TO_DELETE);` The check for whether the timeout is active prior to removing it is not strictly required.
Recommendation	Consider removing the this check.
Resolution	● ACKNOWLEDGED

Issue #18	Typographical issues
Severity	● INFORMATIONAL

Description	Line 41
	`inline fun get_receive_library_timeout(src_eid: u32): address {`
	The name of this function is hardly representative of what it actually returns.
	Line 102
	`public(friend) fun get_default_send_library(dst_eid: u32): address {`
	This function is slightly inconsistent with the receive equivalent, which also checks that the `eid` is supported.
	Lines 249 and 286
	`// This can still be acheived by explicitly setting the OApp to the default library with a grace period`
	"acheived" should be "achieved".
	Lines 451-454
	`const ELIBRARY_DOES_NOT_SUPPORT_RECEIVE: u64 = 5;`
	`const ELIBRARY_DOES_NOT_SUPPORT_SEND: u64 = 6;`
	`const ENOT_AUTHORIZED: u64 = 7;`
	`const ENO_OAPP_RECEIVE_LIBRARY_TO_DELETE: u64 = 8;`
	These error codes seem to be unused.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.7 endpoint_v2::endpoint

The `endpoint` module is the primary module within the LayerZero Aptos Endpoint V2 deployment. Almost all external interactions go directly through the `endpoint` module, as all other modules are mostly isolated from external systems by design.

There are functionalities for OApps to register themselves, as well as set core parameters such as send and receive msg libraries to use, and configuring pathways. It also exposes the functions required for managing cross chain messaging, including validating packets, setting the state for packets, and clearing received messages.

The endpoint also exposes the functionality in the `channels` module, including sending messages (`send`) and getting quoted costs (`quote`). Additonally, there are functions for triggering alerts in the case of failures.

2.7.1 Privileged Functions

None.

2.7.2 Issues & Recommendations

Issue #19	**Lack of enforcement that messages get cleared during execution makes things difficult for executors, potentially causing them to accidentally double spend**
Severity	● MEDIUM SEVERITY
Description	An OApp is free to call its `clear` function on the endpoint. The result of this would be that when an executor executes a receipt on that OApp, the message is never actually marked as executed and can be executed again. This is particularly problematic as it makes things more complicated for said executors: It becomes more difficult to prove that they actually tried executing the message and if they do not have the right off-chain safeguards in place, they make accidentally double execute. Any such executors vulnerable to double execution could potentially get drained through `lz_receive` options with a `value` attached to them, as that `value` can be received multiple times while only having been paid once. The crux of this issue boils down to the fact that executors strictly only want to execute if they know the `clear` will be called. Luckily, Move actually provides a pattern for this, the "hot potato" pattern.
Recommendation	Consider using the "hot potato" pattern to enforce `clear` to be called, whenever an executor executes an `lz_receive`. Specifically, the `lz_receive` function should now take a `ClearRequirement` resource argument that cannot be dropped by the OApp in any way, meaning it does not have the `store` and `drop` traits. The only way to get rid of the `ClearRequirement` is to pass it into `clear`, which is the only function that takes a `ClearRequirement`, and subsequently destructs it. By implementing this pattern carefully and designing it to not allow the OApp to get rid of the requirements by clearing another message, the executor now knows that if `clear` was not called, the message will revert. It should be noted that transaction scripts must now be used to succesfully execute such a multi-step process. It may make sense for OApps and the endpoint to provide an alternative function to `lz_receive` which does not take a `ClearRequirement` to ease manual execution. That being said, generating the requirement should not be that difficult, and could be done with a `script` based manual execution as well.
Resolution	✓ RESOLVED Executors can now pass a hot potato into the oapp, which requires it to call clear for the message to not revert.

Issue #20	Lack of validation when creating alerts can result in spam
Severity	● LOW SEVERITY
Description	There is currently no validation for the signers passed to `lz_compose_alert` and `lz_receive_alert`, meaning anyone can call these functions, with any sort of data, which can result in spam.
Recommendation	Consider validating whether the signer objects passed to `lz_compose_alert` and `lz_receive_alert` are proper executors. This may have the downside that non-worker entities who still call `lz_receive` can't create this message so should be done carefully if done. Consider however validating that the message is actually pending execution, instead of blindly allowing the entity to emit any data.
Resolution	● ACKNOWLEDGED

Issue #21	Typographical issues
Severity	● INFORMATIONAL

Description

Line 85

```
/// The `expiry` is the exactl block number at which the
OApp will no longer confirm message using the provided
```

"exactl" should be "exact".

Line 89

```
/// is unset (uses the default). This will also revert if
the expirty is not set to a future block number.
```

"expirty" should be "expiry".

Lines 181 and 291

```
/// This may be used to create a record that the offchain
entity has made an unsuccessful attempted to deliver
/// This may be used to create a record that the offchain
composer has made an unsuccessful attempted to deliver a
```

"attempted" should be "attempt".

Line 198 and 306

```
address_of(executor),
```

	To be consistent with the rest of the codebase style, it may be cleaner to move the `executor` reference explicitly into this function.
	
	Line 250
	
	`/// This should be called by the lz_receive function on the OApp when it receives a message that inidicates that`
	
	"inidicates" should be "indicates".
	
	Line 330
	
	`/// This verifies of a message by storing the payload hash on the receive channel`
	
	The "of" appears unnecessary here.
	
	———
	
	It is unclear to us why composers need to register themselves in the first place.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.8 endpoint_v2::admin

The admin module is the entry point module for all privileged configuration functions, such as managing the default libraries or enabling ZRO fees.

The module defines a single authorized role, @layerzero_admin, which is set to a specific Aptos address at the time of compilation. All functions within this module can only be called with the admin's signer. The client has indicated that this address will be transitioned into an Aptos multi-signature account after successful deployment. We recommend that users validate this transition as it may only be done after this audit.

Even though the admin module lacks the recommended capability methodology of doing authentication, it mitigates some of the signer's risks by always immediately dropping the passed in admin signer, preventing an unsafe implementation from abusing the signer resource further.

2.8.1 Privileged Functions

- set_eid [removed]
- set_zro [replaced with set_zro_enabled, then both removed]
- register_library
- set_default_send_library
- set_default_receive_library
- set_default_receive_library_timeout

2.8.2 Issues & Recommendations

Issue #22	admin not explicitly dropped for set_eid and set_zro
Severity	● INFORMATIONAL
Description	Lines 15 and 24 `assert_admin(admin);` Throughout the codebase, the `admin` is immediately dropped by explicitly moving it. However, this is forgotten here, resulting in these functions not gaining the security security benefits of the signer drop method which we believe is something the client wants to do as much as possible.
Recommendation	Consider prepending `move` to the `admin` parameter.
Resolution	✓ RESOLVED These functions are no longer present.

Issue #23	Typographical issues
Severity	● INFORMATIONAL

Description	Line 25
	
	`if (zro != @0) {`
	
	To improve readability, consider replacing `@0` with `@0x0`.
	
	Line 36
	
	`/// The library first must be connected to the router node before it can be registered. Until a library is`
	
	"registered" should be added at the end.
	
	Line 49
	
	`/// Set the default receive message library for the given srcource EID`
	
	"srcource" should be "source".
	
	Line 62
	
	`/// The provided expiry is in a specific block number. The fallback library will be disabled after one the block`

	“one” seems to be redundant here. It should also be noted that some functions accept an “expiry” while others accept a “grace period”, which seems slightly inconsistent, unless there’s some business logic reason for this. Finally, `set_eid` and `set_zro` lack events.
Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.9 endpoint_v2::registration

The `registration` module is a small submodule of the endpoint which processes intial registrations of oapps and composers. Through the endpoint, OApps and composers can respectively call `register_oapp` and `register_composer`, which can only be called once for each OApp and composer module address.

When called, the OApps and composers can provide the actual module name which contains respectively the `lz_receive` and `lz_compose` entrypoint functions. The module names are stored as well.

The `registration` module can only be interacted with from the `endpoint` module, and exposes no functions directly to unprivileged modules, making it a simple subcomponent of the endpoint.

2.9.1 Privileged Functions

None.

2.9.2 Issues & Recommendations

Issue #24	Module string names have no cap on length, which can impact gas consumption
Severity	● LOW SEVERITY
Description	The `registration` module allows for OApps and composers to register themselves by indicating their module name. However, there is no enforcement on the maximum length of this name. As seen in other issues, large resources affect gas performance. An unbound name size could therefore cause the oapps' resource to be needlessly expensive. In the extreme case, performance issues may incur.
Recommendation	Consider limiting the length of these module names to a reasonable value.
Resolution	✓ RESOLVED The length is now limited to 100.

Issue #25	Typographical issues
Severity	● INFORMATIONAL

Description	Lines 35 and 39
	
	`public(friend) inline fun is_registered_oapp(oapp: address): bool {`
	`public(friend) inline fun is_registered_composer(compos-er: address): bool {`
	
	The `is_registered_oapp` and `is_registered_composer` functions both have the `inline` modifier, causing their callers to simply inline the code bodies of these functions. Since the functions actually call the `store` module instead, this requires for the `store` to explicitly befriend the modules calling these `registration` view functions, which is highly non-intuitive. Consider removing the `inline` from these two functions.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED The `inline` modifier has been dropped from these functions.

2.10 endpoint_v2::messaging_composer

The `messaging_composer` module is responsible for processing composed messages. The way composing works is very similar to the EVM endpoint. The OApp needs to call `send_compose` on message receipts, which queues a compose execution.

In a separate call, `clear_compose` will be called which actually executes the composition request (using the reverse ordering of responsibility, similar to how `lz_receive` executes). The composition is executed by calling `lz_compose` on the requested composer, which then calls `clear_compose` to validate correctness. It is the responsibility of the OApp to ensure that they configure a composer that correctly calls `clear_compose` on every `lz_compose` call.

2.10.1 Privileged Functions

None.

2.10.2 Issues & Recommendations

Issue #26	Lack of public view function to inspect whether a compose message hash exists
Severity	LOW SEVERITY
Description	Although there is a function which allows third-party modules to get specific message hashes, there is no such function for these modules to inspect whether there is such a hash in the first place. Instead, these modules need to discover this through trial-and-error, which is highly suboptimal.
Recommendation	Consider adding a `has_compose_message_hash` function.
Resolution	RESOLVED `get_compose_message_hash` now returns the zero hash. However, there is a big difference between the fact that a message has been cleared, or that it never existed in the first place. This difference is not conveyed by returning zero in both cases. This is why the client now also sets the hash to all ones once the message has been cleared.

Issue #27	Typographical issues
Severity	INFORMATIONAL

Description	Line 33
	`// Set the message hash to empty to prevent sending the messge again`
	"messge" should be "message".
	Line 127
	`const EUNAUTHORIZED: u64 = 4;`
	This error appears unused.

Recommendation	Consider fixing the typographical issues.
Resolution	RESOLVED

2.11 endpoint_v2_common::assert_no_duplicates

`assert_no_duplicates` module is a simple utility module which exposes the `assert_no_duplicates` pure function which validates that a vector has no "equal" (defined as satisfying `==`) elements. It is implemented using a simple nested for-loop.

2.11.1 Privileged Functions

None.

2.11.2 Issues & Recommendations

No issues found.

2.12 endpoint_v2_common::bytes32

`bytes32` is a simple utility module which defines the `Bytes32` type, a type representing an array of 32 bytes. It defines several utility functions such as converting from and to an Aptos Move `address`, and hashing arbitrary byte arrays into a `bytes32` using the `keccak256` algorithm.

`Bytes32` is a resource that has the `copy` and `drop` trait, meaning it should not be used as a "value bearing" resource.

It should be noted that the `Bytes32` module is not restricted to the endpoint v2 modules and anyone can interact with it.

2.12.1 Privileged Functions

None.

2.12.2 Issues & Recommendations

Issue #28	Typographical issues
Severity	● INFORMATIONAL
Description	Line 16 `public fun is_zero(bytes32: Bytes32): bool {` It is best practice to compare the reference of the resource instead, as the current `is_zero` function consumes the `bytes32`, or alternatively requires it to be copied which wastes significant amounts of computation.
Recommendation	Consider fixing the typographical issues.
Resolution	● ACKNOWLEDGED Although the client has tried to resolve this, it appears that a copy of the underlying bytes is still made.

2.13 endpoint_v2_common::config_eid_-tagged

`config_eid_tagged` is a simple utility module which facilitates the creation of `EidTagged` resources which associate a generically typed configuration with an `eid`. The configuration can be any other resource with the `store`, `drop` and `copy` abilities.

It should be noted that the `config_eid_tagged` module is not restricted to the endpoint v2 modules and anyone can interact with it.

2.13.1 Privileged Functions

None.

2.13.2 Issues & Recommendations

No issues found.

2.14 endpoint_v2_common::contract_identity

The `contract_identity` module is an authorization module that allows Aptos Move signers to create capabilities and send them to modules to denote that the signer authorized a certain action.

This has several distinct advantages. Most notably, capabilities are much more restricted in what permissions they grant to the receiver. On the other hand, passing the `signer` itself allows the receiver to potentially do anything that the `signer` can do. The other lesser advantage is that no explicit signer is needed to create the capabilities. Modules themselves can generate them ad-hoc without having to use something like a resource account.

The `contract_identity` system is therefore a more generalized implementation of the Move Capability Pattern.

The `ContractCalRef` resources, representing the capabilities, cannot be copied arbitrarily. Instead, they must be generated by the owner of the `ContractSigner` resource, which must be created once or multiple times by a `signer`, but can be re-used over time.

It should finally be noted that the `contract_identity` module is not restricted to the endpoint v2 modules. Instead, anyone can interact with it.

2.14.1 Privileged Functions

None.

2.14.2 Issues & Recommendations

Issue #29	Multiple identity generation can cause contract identities to be compromised if the underlying module address signer is compromised
Severity	● LOW SEVERITY
Description	As described above, the `contract_identity` module allows for a module address's signer to generate `ContractSigner` resources. Throughout the endpoint v2, such a signer is generated exactly once, and is then perpetually re-used to generate `ContractCalRef`. It is therefore undesired that at any point a new contract signer would be created, as this is likely with malicious intent. However, since the function to create new contract signers can be called multiple times, a hacker who is able to compromise the private keys for the `endpoint_v2` account can generate a new signer and hijack the system.
Recommendation	Consider only allowing a single signer to be created. This signer should have the `copy` trait however, as this would permit the receiving module to then spread copies to any of the other modules that need to make refs. It should be noted that this pattern works even better if the `Contract-Signer` is not indexed on the module's address, but instead on some module owned resource, and is requested directly by the module. Finally, on a separate note, having the recipient filtering work with a phantom type (instead of a shared module address) prevents a bunch of edge case issues as well, as it prevents multiple independent modules from living on the same address and accidentally permitting eachother's capabilities.
Resolution	✓ RESOLVED A `ContractSigner` can now only be created once from the underlying `signer`. After this, subsequent `ContractSigner` instances need to be copied from the existing ones. Furthermore, the recipient filtering now support phantom types which allows for significantly more granular access control, though it should be stated that the same type is sometimes used for multiple functions throughout the codebase.

Issue #30	**Typographical issues**
Severity	● INFORMATIONAL

Description	Lines 12-14
	
	`struct ContractSigner has store, drop { contract_address: address }`
	`struct ContractCallRef has drop { contract_address: address, target_contract: address }`
	
	These structs lack `view` functions to inspect the struct contents, preventing other modules from doing so.

Recommendation	Consider fixing the typographical issues.
Resolution	● ACKNOWLEDGED

2.15 endpoint_v2_common::guid

`guid` is a simple utility module to compute the `guid` of a message's data. it does so by hashing the sequence of the message's `nonce`, `src_eid`, `sender`, `dst_eid` and the `receiver` the `keccak256` algorithm.

2.15.1 Privileged Functions

None.

2.15.2 Issues & Recommendations

No issues found.

2.16 endpoint_v2_common::packet_raw

`packet_raw` is a simple utility module which provides a `RawPacket` resource which has `drop`, `copy` and `store` abilities. `RawPacket` simply wraps a raw and unstructured bytes array, allowing for it to be more explicitly typed and to have several utility functions.

It should be noted that the `contract_identity` module is not restricted to the endpoint v2 modules and anyone can interact with it and create raw packets.

2.16.1 Privileged Functions

None.

2.16.2 Issues & Recommendations

Issue #31	Typographical issues
Severity	INFORMATIONAL

Description	Line 3
	
	`prefers leave the packet bytes in a raw format to avoid unnecessary serialization/deserialization.`
	
	The line before this contains a comma too many at the end after "since" and this line should say "prefers to leave".
	
	Line 15
	
	`RawPacket { packet: move packet_bytes }`
	
	The use of the `move` keyword is unnecessary, as the compiler will already optimize this call so that this is done under the hood.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.17 endpoint_v2_common::packet_v1_codec

packet_v1_codec is a simple utility module which allows for the encoding of LayerZero V2 packets into a RawPacket and decoding the various fields from a raw packet. It also provides the assert_receive_header function which validates that the provided bytes have the correct header length, version field and dst_eid field.

A packet is defined by the following fields:

- nonce [8 bytes]
- src_eid [4 bytes]
- sender [32 bytes]
- dst_eid [4 bytes]
- receiver [32 bytes]
- guid [32 bytes]
- message [variable length]

2.17.1 Privileged Functions

None.

2.17.2 Issues & Recommendations

Issue #32	Typographical issues
Severity	● INFORMATIONAL

Description	Line 32-37
	
	The logic at the start of `newpacketv1` copies the code in `new_packet_v1_header_only_bytes`. Consider replacing this duplicate logic with a call to the latter function.
	
	Line 77
	
	`let packet_header_bytes = vector::slice(borrow_packet_bytes(&mut raw_packet), 0, GUID_OFFSET);`
	
	It is unclear why the `raw_packet` reference is marked as `mut` here.
	
	Line 85-87
	
	`let packet_bytes = packet_raw::borrow_packet_bytes(raw_packet);`
	`let pos = VERSION_OFFSET;`
	`let version = serde::extract_u8(packet_bytes, &mut pos);`
	
	This code is a repetition of the `get_version` function and violates the popular "don't repeat yourself" (DRY) programming pattern. Consider re-using the existing function.
	
	Line 163
	
	`assert!(vector::length(packet_bytes) == 81, EINVALID_PACKET_HEADER);`
	
	Unlike the rest of the module, this line of code uses a magic value, "81". Consider using a constant instead for consistency.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.18 endpoint_v2_common::send_packet

send_packet is a simple utility module which exposes the internally used SendPacket resource type, encompassing all packet fields as defined in the packet_v1_codec module.

It should be noted that the send_packet module is not restricted to the endpoint v2 modules and anyone can interact with it and create SendPacket resources.

2.18.1 Privileged Functions

None.

2.18.2 Issues & Recommendations

Issue #33	Typographical issues
Severity	● INFORMATIONAL
Description	Line 44 `(packet.nonce, packet.src_eid, packet.sender, packet.dst_eid, packet.receiver, packet.guid, packet.message)` This does not explicitly unpack the input struct, potentially leading to unnecessary copies being made. When disassembled, each field is individually copied with `ImmBorrowLoc`, `ImmBorrowField` and `ReadRef`, 22 opcodes in total. Meanwhile, when unpacking, the whole implementation of the function is just 3 opcodes: `MoveLoc`, `Unpack` and `Ret`.
Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED This is now unpacked instead.

2.19 endpoint_v2_common::serde

`serde` is a simple utility module responsible for encoding and decoding Move types such as unsigned integers, addresses and `Bytes32` resources into byte arrays (`vector<u8>`s).

The encoding scheme used by the `serde` module is not self-contained: it requires the user to define the field types/locations during decoding and the order during encoding.

2.19.1 Privileged Functions

None.

2.19.2 Issues & Recommendations

Issue #34	extract_uint is insecure for bytes equal to zero.
Severity	● INFORMATIONAL
Description	The `extract_uint` function does not properly check that `position` is in bounds when the `bytes` parameter is set to zero. This is because the `position` check is implicit while iterating over each individual byte. When the number of bytes is set to zero, no such iteration occurs. Though the function explicitly states it misbehaves for `bytes > 8`, it does not state anything for `bytes == 0`. This issue has been rated as informational as to our knowledge this is never done within the codebase, nor is it very sensible. A similar issue is present with `extract_fixed_len_bytes` and `extract-_bytes_until_end`, which do not revert when your provide `position == length` and `length == 0` due to the slice function within `vector.move` having a rather unique validation on the inputs.
Recommendation	Consider documenting that `0` bytes is not supported as well.
Resolution	✓ RESOLVED The client did not make any changes to this but indicates that the current behavior is fine.

Issue #35	Typographical issues
Severity	● INFORMATIONAL
Description	Line 11

```
/// Pposition will be update to the end of read
```

“Pposition” should be “Position”.

Lines 44, 57, 118 and 127

```
/// This function is does not use extract_uint because it
is more efficient to handle u128 as a special case
/// This function is does not use extract_uint because it
is more efficient to handle u256 as a special case
/// This function is does not use append_uint because it
is more efficient to handle u128 as a special case
/// This function is does not use append_uint because it
is more efficient to handle u256 as a special case
```

“is does not use” seems to be a typographical error. It also appears that the mathematical bracket usage with `append_u128` and `append_u256` uses one more bracket compared to `append_uint`. It may make sense to add that bracket to `append_uint` as well for more explicitness about the order of operations.

Lines 45 and 58

```
public inline fun extract_u128(input: &vector<u8>,
position: &mut u64): u128 {
public inline fun extract_u256(input: &vector<u8>,
position: &mut u64): u256 {
```

Inlining these functions seems to be inconsistent with other functions which are not inlined.

Line 83-84

```
let result = vector::slice(input, *position, vector::le-
ngth(input));
*position = vector::length(input);
```

It may make sense to store the input length in a temporary variable to potentially save on gas, though it will only be a minor optimization.

Line 90

```
assert!(vector::length(input) >= *position + 32, EINVAL-
ID_LENGTH);
```

This assertion is missing in other functions, such as `extract_fixed_len-_bytes`. It is unclear why the assertion is present within `extract_address` but not elsewhere.

Lines 121 and 130

```
let byte: u8 = (((value >> (8 * (15 - i))) & 0xFF) as
u8);
```

Inconsistent use of implicit subtraction in `append_u128` and `append_u256` reduces readability, as compared to e.g. line 100.

Line 146

```
assert!(target_size >= vector::length(&bytes), EINVAL-
ID_LENGTH);
```

The length variable is stored later on and can be moved up to be re-used here.

Line 157

```
/// Append a byte vecdotr to the of a buffer
```

This should say “vector”.

Line 199

```
/// This function create a bytes vector by applying the
function `f` to a &mut of and empty vector<u8>
```

This line contains a small typographical error as it says “of and” instead of “of a”.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.20 router_node_0::router_node

The message library `router_node` is used by the endpoint to be able to achieve some form of dynamic dispatch without upgradeability to various message libraries even though Aptos does not support it. Essentially, its goal is to be able to add new message library modules over time and have the endpoint send to them without having to upgrade the endpoint (as it will be made immutable).

This is implemented by iterating over a list of known existing message libraries and dispatching to these, and if the requested message library is not found in these, then the request is forwarded to the next router, which is upgrade-able while not being configured.

It should be noted that while it is upgradeable, any dispatches to message libraries not included in the fixed non upgradeable list should not be trusted as the upgradeable router can process the request however it wants if it has not been made immutable yet.

The following actions can be dynamically dispatched to the message libraries:

- `quote`
- `send`
- `commit_verification`
- `dvn_verify`
- `set_config`
- `get_config`
- `version`
- `is_supported_send_eid`
- `is_supported_receive_eid`

2.20.1 Privileged Functions

None.

2.20.2 Issues & Recommendations

Issue #36	Asymmetric library configurations can cause privilege escalation by LayerZero governance
Severity	● HIGH SEVERITY
Description	Both the `send` and `commit_verification` functions take an identical authorization reference, indicating that the endpoint created these calls to the `router_node` and underlying message libraries. Since this reference is identical, a single reference could in theory be replayed in both functions with arbitrary data. As the reference contains the message library, it is meant to execute to, and since all message libraries always validate that the reference contains their own address, it is impossible to re-use a reference from an upgradeable library for another library. This prevents the LayerZero team from adding a malicious upgradeable library that captures references and replays them with malicious fake data on the actual message libraries, to exploit them. However, this prevention is insufficient in the case where a message library has a different address for its send and receive component. Or especially if a message library only has e.g. a send component (for example when a new library is added to send to a special chain). This is because when the library has only been immutably added to one of the two functions, calls with that library address to the other function will not find a match. Let's say a special send library is added to the routers and those routers and that library are made immutable. Now, a normal OApp can send messages via that library. However, when someone configures that library as the receiving library, no match is found and the call gets forwarded to the upgradeable placeholder router. This is where the privilege escalation occurs. The LayerZero team can capture the reference in this upgradeable router since they can still modify the business logic of that router. Once captured, they can forward the reference directly to the actual existing sending library, with arbitrary parameters into its `send` function. This means that this privilege escalation allows for the LayerZero team to send arbitrary messages for an oapp that has an assymetric library through this privilege escalation path, which can be disastrous if their upgrade keys are ever compromised. This issue has been rated as high as it compromises the security expectations of these OApps, and messages could be forged by LayerZero governance under these circumstances.
Recommendation	

	Consider fine-tuning the authorization scheme with more idiomatic authorization specifications, such as phantom types or even better the actual data as its payload. The latter is near ideal, but quite cumbersome and verbose, hence why a phantom type per destination may be more ideal.
Resolution	✓ RESOLVED Authorization is now namespaced on the function which is called.

Issue #37	**Typographical issues and gas optimizations**
Severity	● INFORMATIONAL

Description	Line 6
	`/// Any unsed slot points to a upgradable placeholder contract, which makes appending new msglib implementations`
	"unsed" should be "unused".
	Line 30
	`if (msglib == @simple_msglib) {`
	It makes more sense to start the `switch` statements with `uln_302`, as this library is expected to be the most used. Correctly ordering the statements in order of usage should save execution cost.

Recommendation	Consider fixing the typographical issues and gas optimizations.
Resolution	✓ RESOLVED

2.21 router_node_1::router_node

The `router_node` for the secondary router represents the "to upgrade" router, which is the needed component to fulfil the safe dynamic dispatch feature of the router pattern. As discussed in the previous section, if the requested library address is not found within the immutable router, the request is forwarded to the next router, which is upgradeable, which is this router.

If a new library needs to be added, this router will be upgraded to forward to it, and if the requested library is not that address, forward to a new next upgradeable router. Subsequently, after this configuration upgrade, this router will be rendered immutable. Until this upgrade however, this router aborts on all transactions.

It is particularly important that users who use a new library version validate that the router was upgraded correctly, and subsequently was made immutable. This is because we as an audit firm can say absolutely nothing about the aspects of those future upgrades and libraries. Our audit is limited to the existing libraries described within this audit, and any request that specifically points to those.

2.21.1 Privileged Functions

None.

2.21.2 Issues & Recommendations

No issues found.

2.22 blocked_msglib::router_calls

The blocked message library is the first message library and is used as an emergency message library. If something does not work as expected, applications can move their configured message library to this library to cause all message library interactions to revert.

The blocked message library will indicate that it supports all `eid`s for sending and receiving, even though it will abort for all of them.

2.22.1 Privileged Functions

None.

2.22.2 Issues & Recommendations

No issues found.

2.23 simple_msglib::router_calls

The simple message library is a simple example message library which is not supposed to be used within production. This is because the simple message library does not contain any validation logic. Anyone can mark any message as verified for any OApp.

Specifically, `commit_verification` can be immediately called without there being any `dvn_verify` before it. In fact, `dvn_verify` is simply not implemented and reverts.

We want to reiterate that even though this library may be available in production, it **should not** be used by any application. Enabling this library is equivalent to letting your protocol be exploited.

`set_config` and `get_config` are also not implemented and revert for the simple message library.

Any `eid` is evaluated as "supported" by the simple message library.

2.23.1 Privileged Functions

None.

2.23.2 Issues & Recommendations

Issue #38	The token inputs for the send function are not actually reduced
Severity	● INFORMATIONAL
Description	The `send` function can return a real `native_fee` and `zro_fee`. However, this logic is incorrectly implemented as the `zro_fee` is returned regardless of whether a `ZRO` token input is provided. Next, both of the token inputs are never actually reduced, which begs the question on why the fees are returned in the first place, instead of having this module simply return `(0,0)` as its fees.
Recommendation	Consider either levying the fees (accordingly on whether the `zro` is some or none) or returning `(0,0)` for the fees. No changes are required if this setup of returning a fee but not levying is done to improve test coverage.
Resolution	✓ RESOLVED The client has confirmed this is desired within this test library and that it should not be used in production. No fix was made in this regards. However, the function now only returns either a ZRO or native fee.

2.24 simple_msglib::msglib

`simple_msglib::msglib` is the underlying module for the simple message library. It implements some of the underlying logic such as the admin initialization of the `eid` and the admin configurability of the messaging fee.

2.24.1 Privileged Functions

- `initialize`
- `set_messaging_fee`

2.24.2 Issues & Recommendations

Issue #39	Typographical issues
Severity	INFORMATIONAL

Description	Lines 11, 16, 17, 18
	`const PACKET_V1_VERSION: u8 = 1;`
	`const ENOT_IMPLEMENTED: u64 = 1;`
	`const EINVALID_PACKET_HEADER: u64 = 100;`
	`const EINVALID_PACKET_VERSION: u64 = 101;`
	These constants are unused.
	Lines 60 and 98
	`assert_admin(account);`
	`assert_admin(account);`
	Throughout the codebase, `move account` is used to explicitly burn the signer. However, it is omitted here.
	Line 71
	`public fun commit_verification(`
	This function should be marked as `friend` as it is not supposed to be directly called from external modules.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.25 msglib_3::router_calls

The third message library is a placeholder. It is extremely similar to the blocked message library in that it reverts for all calls. Additionally, it will indicate that it does not yet support any `eid`.

Unlike the first (blocked) and second (ULN 302) libraries, the placeholder libraries are upgradeable. This is because they are meant to be initialized via an upgrade, after which the upgrade key will be revoked. This allows for the message lib router to already point to these future message libraries, as is required due to the limitations of not having dynamic dispatch within Aptos Move.

2.25.1 Privileged Functions

None.

2.25.2 Issues & Recommendations

No issues found.

2.26 msglib_4::router_calls

The fourth message library is a placeholder, pretty much identical to the third. It is extremely similar to the blocked message library in that it reverts for all calls. Additionally, it will indicate that it does not yet support any `eid`.

Unlike the first (blocked) and second (ULN 302) libraries, the placeholder libraries are upgradeable. This is because they are meant to be initialized via an upgrade, after which the upgrade key will be revoked. This allows for the message lib router to already point to these future message libraries, as is required due to the limitations of not having dynamic dispatch within Aptos Move.

2.26.1 Privileged Functions

None.

2.26.2 Issues & Recommendations

Issue #40	Typographical issues
Severity	● INFORMATIONAL
Description	First, it may make sense to fetch all `packet` data at once within `send`. This data could even be moved into the simple message library as the `packet` is no longer used afterwards. This saves some gas given the many individual calls and copies. Next, there are small typographical differences between the two placeholder modules. Certain variables are named differently and the formatting is different, as well as `#[view]` being added to `get_config` in this module, but not in `msglib3::router_calls`. This needlessly adds a differential between the two contracts which adds overhead to code reviewers.
Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED This module has been removed.

2.27 uln_302::msglib

The `msglib` module exposes `view` functions which allow getting the current default send and receive uln configurations as well as the default executor configuration from the `uln_302_store` module. It also exposes a `view` function for gathering the treasury address, as well as checking whether a message (specified by packet header and payload hash) are verifiable, meaning a sufficient number of DVNs have verified the message.

2.27.1 Privileged Functions

None.

2.27.2 Issues & Recommendations

Issue #41	verifiable may return true even if the actual commit_verification flow reverts
Severity	● LOW SEVERITY
Description	The `verifiable` view function lacks certain checks which appear present on the actual `commit_verification` function. Notably, all of the `assert_receive_header` functions appaer to be missing.
Recommendation	Consider carefully comparing the two functions to ensure their behavior is consistent. Consider adding the `assert_receive_header` function if needed.
Resolution	✓ RESOLVED The client has added what to their needs is sufficient extra validation, notably the `is_receive_header_valid` function got added. It should be noted that the message library was updated with some extra business logic out of the scope for our audit: The ability to opt into using the worker config to configure the worker fee library.

2.28 uln_302::router_calls

The `router_calls` module is a fully implemented msg library which can be used to quote the cost for a message (`quote`), send a message (`send`), exposes a function for each DVN to call in order to verify a message (`dvn_verify`), and a function to fully verify a message (`commit_verification`). Both the `send` and `commit_verification` function calls are intended to only be called from the endpoint module and perform validation on a passed `ContractCallRef` struct. The `dvn_verify` and `quote` functions are callable by anyone.

Payments are done by sending mutable references `FungibleAsset` and `Option<FungibleAsset>` structs for native and `zro` tokens respectively in the `send` call. Any remaining tokens are required to be handled by the initial caller (e.g. depositing them back into their original account).

There are also functions for OApps to register their desired send / receive uln as well as executor configurations (`set_config`), as well as gather these OApp-speciific configurations (`get_config`). Additionally there are the `is_supported_send_eid` and `is_supported_receive_eid` functions which define whether this msg library supports a given `eid`, which are both defined as being whether there are default send and receive uln configs set specified respectively. Although not checked for either of these calls, if there are no OApp-specific or default executor configuration set, technically messages are unable to be sent either.

2.28.1 Privileged Functions

None.

2.28.2 Issues & Recommendations

Issue #42	Typographical issues
Severity	● INFORMATIONAL

Description	Line 1
	
	`/// Entrypoint for all calls that come from the Message Library Router`
	
	It may make sense to document that `quote`, `dvn_verify` and the view functions can all be called directly as well.
	
	Line 50
	
	`/// if the packet has not been verified by all required parties. This is be called in conjunction with the`
	
	This should say “to be called”.
	
	Line 81
	
	`/// Gets the ULN or Executor configuration for and eid on an OApp ..`
	
	This should say “an eid”.
	
	Lines 105-106
	
	`const ENOT_AUTHORIZED: u64 = 1;`
	`const ENOT_IMPLEMENTED: u64 = 2;`
	
	Both of these errors are unused.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED Most of these errors have been fixed.

2.29 uln_302::admin

Defines functionality for setting the default send and receive uln configs, as well as the default executor configuration. It also had the `initialize` function (update: now removed in the resolution round) for the admin to be able to initialize the `uln_302` msg library, which means setting the `eid` value for this deployment. All functions require admin authentication through the use of the signer, and all safely drop the signer after use.

2.29.1 Privileged Functions

- `set_default_uln_send_config`
- `set_default_uln_receive_config`
- `set_default_executor_config`

2.29.2 Issues & Recommendations

No issues found.

2.30 uln_302::verification

The `verification` module contains functionality for supporting the `dvn_verify` and `commit_verification` function calls in the `router_calls` module. The `dvn_verify` call ultimately just calls the `verify` function on this module, which sets the verification for the passed message in the store module. There is no validation on this call, meaning a DVN (or any caller for that matter) is able to validate any message at any time, including overwriting a previous validation for a message.

The `commit_verification` call ultimately calls the `commit_verification` function on this module and is more complex. This function is intended to be called once the required DVNs have committed their verification for a given message. At a high level, it will gather the receive uln configuration, which includes the required information on the required DVNs, and the optional DVNs with the optional DVN threshold. It will then loop over these DVNs and check whether they have verified this message with an appropriate number of confirmations. Verification entails checking that the DVN had called `dvn_verify` on this message, meaning a hash is stored in the `confirmations` table in the store module.

If the appropriate DVNs have verified this message, this call will then remove all the confirmations from this table. This is important, for example, in case this message needs to be re-verified.

2.30.1 Privileged Functions

None.

2.30.2 Issues & Recommendations

Issue #43	**check_verifiable has exponential execution complexity on the number of optional DVNs, potentially bricking receipts**
Severity	MEDIUM SEVERITY
Description	Line 67 `let dvn = vector::borrow(&configs_uln::get_optional_dvns(config), i);` To check that enough optional DVNs have signed, the whole `optional_dvn` vector is copied for every single iteration over the optional dvn vector. This means that if there are for example 100 optional DVNs, the worst case could be up to 10,000 iterations, potentially bricking all messages with these optional DVNs configured from being committed.
Recommendation	Consider caching the optional DVNs vector.
Resolution	✓ RESOLVED Vectors are now cached.

Issue #44	**Typographical issues**
Severity	● INFORMATIONAL

Description	Line 35
	`header: packet_raw::get_packet_bytes(packet_header),`
	This function is already explicitly imported. There is no need to include the `packet_raw::` namespace.
	Lines 181-182
	`const EINVALID_EID: u64 = 1;`
	`const ENOT_AUTHORIZED: u64 = 2;`
	These errors appear unused.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.31 uln_302::sending

The `sending` module implements much of the functionality to support the `quote` and `send` function calls in the `router_calls` module and is only callable from this module. Its main external functions mirror those in `router_calls`.

The first function, `quote`, first confirms that the message is valid. It then gathers the executor fees using the `executor_fee_lib_router` module, the DVN fees for all optional and required DVNs using the `dvn_fee_lib_router` module, and the treasury fee from the `treasury` module. The treasury fee can optionally be paid in ZRO rather than the native token.

The second primary external function is `send`, where the initial calculation of executor, DVN, and treasury fees follows a similar logic to `quote`. However, in `send`, tokens are actually paid to each worker respectively and the treasury. Tokens (with native and ZRO being optional) are passed in using mutable references, and the required amounts are extracted to pay each worker. The caller is expected to handle any remaining tokens in the `FungibleAsset` structs.

2.31.1 Privileged Functions

None.

2.31.2 Issues & Recommendations

Issue #45	**Worker payment events used by the worker may contain insufficient information for the worker to easily verify payment**
Severity	● LOW SEVERITY
Description	The `ExecutorFeePaid` and `DvnFeePaid` events currently contain limited information, primarily just the worker address and the fee paid to them. It may be beneficial to enhance these events so that the event itself provide enough information for the worker to fully validate that they should act on the message. This enhancement would mainly involve including the message `GUID`. Additionally, it could be useful to incorporate the worker fee library so workers have a historical record of the fee library that was used for the payment.
Recommendation	Consider at least including the message `GUID` in the events to simplify the workers' listener procedure. Good workers should still cross-reference this event with the emission of a `PacketSent` event on the actual `endpoint_v2::channels` module, as there are far fewer intermediary steps between the app and the `PacketSent` event. This makes it an ideal event to double-check as an extra security measure to ensure that false (forged) information is never verified.
Resolution	● ACKNOWLEDGED

Issue #46	The use of indexing rather than DVN lookup for assigning DVN options can break when a default or OApp-specific ULN configuration is updated prior to the message being sent
Severity	● LOW SEVERITY
Description	Line 162 `let index_option_pairs = msglib_types::worker_options::group_dvn_options_by_index(&validation_options);` There is a potential race condition due to the structure of the DVN portion in the options passed to the `send` function. Each DVN option in the concatenated array is mapped to the current set of default or OApp-specified required and optional DVNs (depending on the configuration), based on the index position of each DVN. However, if this ordering changes for the OApp—either due to modifications in the OApp-specific configuration or because it relies on the default configuration which then changes—this mapping becomes invalid, causing the options passed to be incorrect. Specifically, `index_option_pairs` indexes the options based on the current index of the DVN. However, these indices can shift as DVNs are added or removed in the `send` configuration. The app admin cannot control whether any transactions were already in flight and pending confirmation on the source chain when they modify the configuration. Thus, a race condition could potentially occur.
Recommendation	Consider encoding the DVN options bytes array such that options are mapped directly to DVNs rather than to the current index of where that DVN is in the concatenation of the required and option DVN arrays.
Resolution	● ACKNOWLEDGED As this is also present on the EVM, the client has indicated that they will not fix it.

Issue #47	Typographical issues
Severity	● INFORMATIONAL

Description

Line 49

```
let packet_header = packet_v1_codec::extract_header(raw-
_packet);
```

There must be a more efficient way than copying the whole packet just to extract its header. Perhaps consider adjusting the extract header function to take a reference.

Lines 108 and 242

```
// Assert message size supported by executor
```

This maximum message size is NOT configured by the executor, it is configured by the message sending app itself. If additional message size validation needs to be incorporated, this would have to happen at the worker fee library level.

Lines 230 and 274

```
public(friend) fun pay_executor_and_assert_size(
public(friend) fun pay_verifier(
```

These functions are marked as `friend` but should probably just be `private` instead given that they do not appear to be used outside of this module.

Lines 259 and 302

```
ensure_primary_store_exists(deposit_address, fungible_a-
sset::asset_metadata(native_token));
```

There are two instances in which `primary_fungible_store::deposit` is called, and in both cases the `ensure_primary_store_exists` check is performed. However, this is duplicated work, as this check is already done in the std `deposit` function.

Line 302

```
ensure_primary_store_exists(deposit_address, fungible_a-
sset::asset_metadata(native_token));
```

This metadata can be cached outside of the for-loop to save gas.

Line 365

```
const EWORKER_NOT_REGISTERED: u64 = 2;
```

This error appears unused.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.32 uln_302::uln_302_store

The `uln_302_store` module defines the `Store` struct, which holds all ULN send/receive configurations, as well as executor configuration data for both default (`default_configs`) and OApp-specific (`oapp_configs`) configurations. Additionally, it stores the `eid` value and a `confirmations` table that tracks whether a DVN has verified a given message.

This module also provides helper functions to retrieve or set a given ULN or executor configuration, as well as to set, retrieve, or remove verification confirmations.

2.32.1 Privileged Functions

None.

2.32.2 Issues & Recommendations

Issue #48	Store eid is only set to non-zero potentially periods after deployment
Severity	● LOW SEVERITY
Description	When the module is initialized, the store is configured with a zero `eid`. This presents the significant downside that some functions, such as `get_eid()`, may still return a value even if the `eid` is not set. Additionally, the zero `eid` can never be used, although this may be less problematic since we do not expect it to be used. It would be better not to initialize the store at all upon deployment. This way, all interactions with the store would be expected to abort, as they would almost all attempt to borrow the `Store` resource.
Recommendation	Consider not using `init_module` at all, as this would ensure that all interactions with the store will generally abort. Instead, allow the `eid` initializer to deploy the `Store` to `@uln_302`. As always, we recommend implementing a better compile-time initialization scheme to avoid this intermediary state altogether. However, as discussed, designing this may be tedious due to Aptos' limitations in this regard.
Resolution	✓ RESOLVED Though the `eid` might still be set in a separate period, its configuration has been fully removed from this module.

Issue #49	Typographical issues
Severity	● INFORMATIONAL

Description	Line 31
	
	`struct DefaultConfig has store {`
	
	The `OAppConfig` struct also has the copy and drop abilities. Though these do not appear to be necessary, it may make sense to add them here as well for consistency.
	
	Lines 83 and 88
	
	`/// Internal function that checks the whether the default config table is initialized for a given EID.`
	`/// Internal function that checks the whether the oapp config table is initialized for a given EID and OApp.`
	
	These comments appear to contain an extra "the".
	
	Line 129
	
	`/// Sets the default send configuration for a given EID. This is a raw setter and should not be used directly`
	
	This incorrectly says "default", likely from an accidental copy paste of the comment.
	
	Lines 183 and 189
	
	`public(friend) fun get_executor_config(receiver: address, eid: u32): Option<ExecutorConfig> acquires Store {`
	`public(friend) fun set_executor_config(receiver: address, eid: u32, config: ExecutorConfig) acquires Store`
	
	The `receiver` in question should likely be called `sender` instead, given that this function appears to be primarily called with the `sender` address of the OApp.
	
	Line 253
	
	`inline fun create_oapp_configs(eid: u32, oapp: address) {`
	
	This inline function appears unused.
	
	Line 263

```
public(friend) fun assert_eid_initialized(eid: u32)
acquires Store {
```

It is unclear where this is used.

The majority of the error codes appears unused. Only `EEID_NOT_INITIALIZED` appears used.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.33 uln_302::configuration

The configuration module defines functions for directly interacting with the `uln_302_store` module to set the default ULN send/receive configurations, as well as the default executor configuration. It also includes functions for setting the OApp-specific ULN send/receive configurations and executor configuration. Additionally, it has functions for retrieving the current configurations for a given OApp, which will return the default configuration if no OApp-specific configuration is set; otherwise, it will merge the OApp-specific configuration with the default configuration.

When merging the ULN send and receive configurations, the module will determine whether the OApp-specific configuration requests certain values from the default configuration.

For the ULN configurations, it will check: (1) the number of confirmations, (2) required DVNs, (3) optional DVNs, and threshold.

For the executor configuration, it will check: (1) maximum message size and (2) executor address.

Since the default configurations can be changed at any time, the generation of the current up-to-date ULN and executor configurations must always be done on the fly.

2.33.1 Privileged Functions

None.

2.33.2 Issues & Recommendations

Issue #50	Unnecessary validation in get_executor_configcan lead to wasted gas
Severity	● INFORMATIONAL

Description	Within the `get_executor_config` function, the `else` block checks for a statement which is guaranteed to abort (line 165-169):
	
	`} else {`
	`    // provide default if oapp does not have its own configuration`
	`    assert!(option::is_some(&default_config), EEID_NOT_CONFIGURED);`
	`    option::extract(&mut default_config)`
	`}`
	
	Considering this is the case, it is cleaner to simply abort.

Recommendation	Consider replacing the current else block with a simple abort: `abort EEID_NOT_CONFIGURED`
Resolution	✓ RESOLVED

Issue #51	Typographical issues
Severity	INFORMATIONAL

Description	Lines 77-88 and lines 116-117
	
	`public(friend) fun supports_receive_eid(eid: u32): bool {`
	`  option::is_some(&uln_302_store::get_default_receive_uln_config(eid))`
	`public(friend) fun supports_send_eid(eid: u32): bool {`
	`  option::is_some(&uln_302_store::get_default_send_uln_config(eid))`
	
	There must be a cheaper way to write these than literally copying the whole configuration. Consider moving these functions to the store and operating on the references there.
	
	Lines 210-213
	
	`use msglib_types::configs_uln::{get_confirmations, get_optional_dvn_threshold,`
	`    get_optional_dvns, get_required_dvns, get_use_default_for_optional_dvns, get_use_default_for_required_dvns,`
	`    new_uln_config`
	`};`
	
	Consider moving this up.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.34 uln_302::for_each_dvn

`for_each_dvn` is a simple utility module used by the `uln_302` modules to iterate over the combination of the configured required and optional DVNs. It is a simple programmatic utility library.

2.34.1 Privileged Functions

None.

2.34.2 Issues & Recommendations

Issue #52	Typographical issues
Severity	● INFORMATIONAL
Description	Line 13 `for (i in 0..(count_required + count_optional)) {` This function would likely be simpler if the for-loop was broken up into two for-loops instead. ——— It should be stated that duplicates may be included in this iteration, and the function can execute twice for an address. It may make sense to document this given that the function is `public`.
Recommendation	Consider fixing the typographical issues.
Resolution	● ACKNOWLEDGED The function is however no longer `public`.

2.35 uln_302::assert_valid_default_uln_-config

valid_default_uln_config is a utility module used within the uln_302 modules, containing the validation logic for a correct default UlnConfig configured by the LayerZero governance. It validates the following:

- That there are no duplicates within the required and optional DVNs.
- That there are at least as many optional DVNs as the configured optional_dvn_t-hreshold.
- That there is at least a single DVN required to sign by ensuring optional_dvn_th-reshold + required_dvn_count to be greater than 0.
- That at most 127 optional DVNs are configured.
- That at most 127 required DVNs are configured.
- That none of the "use default configuration" flags are enabled, as this is the default configuration.

2.35.1 Privileged Functions

None.

2.35.2 Issues & Recommendations

Issue #53	**Typographical issues**
Severity	● INFORMATIONAL

Description	Line 22
	`assert!(required_dvn_count != 0 || optional_dvn_threshold != 0, ENO_EFFECTIVE_DVN_THRESHOLD);`
	The way this statement is written is different from the style in the `assert_valid_uln_config` module. Consider using the same style for consistency.
	Line 39
	`const EZERO_CONFIRMATIONS_PROVIDED_FOR_DEFAULT_CONFIG: u64 = 8;`
	This constant appears unused.
	———
	It should be noted that it makes sense to do a non-duplicate check on the concatenation of all DVNs here. This is because we do not see the direct benefit of a DVN being in both the optional and the required list.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED Everything but the non-duplicated check has been implemented. The latter was not done to match the evm implementation.

2.36 uln_302::assert_valid_uln_config

`assert_valid_uln_config` is a utility module used within the `uln_302` modules, containing the validation logic for a correct `UlnConfig`. It validates the following:

- That there are no duplicates within the required and optional DVNs.
- That there are at least as many optional DVNs as the configured `optional_dvn_threshold`.
- That there is at least a single DVN required to sign by ensuring `optional_dvn_threshold + required_dvn_count` to be greater than 0.
- That at most 127 optional DVNs are configured.
- That at most 127 required DVNs are configured.

2.36.1 Privileged Functions

None.

2.36.2 Issues & Recommendations

Issue #54	**The non-duplicate check fails to detect if a DVN is both part of the required and optional DVNs, thus permitting a DVN to potentially contribute in both, doubling its confirmation power**
Severity	● LOW SEVERITY

Description	Line 33-34
	
	`assert_no_duplicates::assert_no_duplicates(&configs_uln::get_required_dvns(oapp_config));`
	`assert_no_duplicates::assert_no_duplicates(&configs_uln::get_optional_dvns(oapp_config));`
	
	The uniqueness of the required and optional DVN vectors is checked independently, meaning that a single DVN cannot be listed twice in either list. However, a single DVN can be registered as both an optional *and* a required DVN at the same time because there is currently no check for duplicates across the two lists.
	
	The downside of this is that when it occurs, that DVN can contribute to the validation process twice. This is undesirable because the inclusion of the DVN in the optional vector provides no additional informational benefit if it is already required, as the message will fail regardless if the DVN does not confirm.

Recommendation	Resolving this issue appears to be impossible given on-chain limitations, as the default configuration can evolve over time, and changes to the default configuration cannot check all existing configurational combinations for conflicts, nor should they. This could be checked explicitly **if** the defaults are both unset. But regardless, this issue cannot be fully resolved from our perspective given the limitations of the data structures.
Resolution	● ACKNOWLEDGED The client finds this acceptable and will keep it to stay compliant with the EVM design.

Issue #55	Typographical issues
Severity	● INFORMATIONAL
Description	Line 36 `// Optional threshold should be not be greater than count of optional dvns` "be not be" should be replaced with "not be".
Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.37 msglib_types::configs_executor

The `configs_executor` module defines the `ExecutorConfig` struct, which includes the `max_message_size` and `executor_address` fields. It provides helper functions for converting a byte array into an `ExecutorConfig` and vice versa, as well as functions for retrieving the values from the struct.

2.37.1 Privileged Functions

None.

2.37.2 Issues & Recommendations

No issues found.

2.38 msglib_types::configs_uln

The `configs_uln` module defines the `UlnConfig` struct, which specifies the necessary parameters for sending and receiving messages. This includes the number of confirmations, an array of `required_dvns`, an array of `optional_dvns`, the `optional_dvn_threshold`, and boolean variables for specifying whether to use these values from the default ULN config if the `UlnConfig` is OApp-specific.

The module also provides helper functions for creating `UlnConfig` structs and for converting them to and from a bytes array.

2.38.1 Privileged Functions

None.

2.38.2 Issues & Recommendations

Issue #56	Lack of validation during ULN configuration construction
Severity	● LOW SEVERITY
Description	The functions that return a new `UlnConfig` do not perform significant validation on the provided data. It may make sense to add more validation: 1. `new_uln_config` should not allow `required_dvns` or `optional_dvns` to be greater than 255 in number of entries, since encoding would break. 2. `optional_dvn_threshold` should be smaller than or equal to the number of `optional_dvns` . 3. A required or optional DVN should probably be present. It should be noted that (2) and (3) are likely validated at higher levels and therefore probably do not need to be validated here. However, (1) directly causes this module to misbehave and should likely be enforced here, as the lack of this enforcement breaks invariants within the module.
Recommendation	Consider at least adding a length validation of the vectors within `new_uln_config`.
Resolution	● ACKNOWLEDGED

Issue #57	Typographical issues
Severity	● INFORMATIONAL

Description	Line 93
	
	`tag_with_eid(move eid, move config)`
	
	It is unclear why these parameters are explicitly moved here, as we assume this is done implicitly anyway.
	
	———
	
	Line 127
	
	`const EINVALID_CONFIG_TYPE: u64 = 0;`
	
	This error code appears unused.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.39 endpoint_v2_common::dvn_verify_params

The dvn_verify_params module defines the DvnVerifyParams struct, which is used by DVNs to specify which packet_header and payload_hash they are verifying, as well as the specific number of confirmations they have verified.

This module includes helper functions for converting the DvnVerifyParams struct into an Any struct, effectively converting its representation into a byte array.

2.39.1 Privileged Functions

None.

2.39.2 Issues & Recommendations

Issue #58	Typographical issues
Severity	INFORMATIONAL

Description	Line 4
	
	`module endpoint_v2_common::dvn_verify_params {`
	
	It is strange that this module is defined at the `endpoint_v2_common` address instead of the `msglib_types` address.
	
	———
	
	Line 11
	
	`struct DvnVerifyParams has drop, store {`
	
	Similar types throughout the codebase also have the `copy` ability.
	
	———
	
	Line 28
	
	`(packet_header, params.payload_hash, params.confirmations)`
	
	Explicitly unpacking this `DvnVerifyParams` instance into its constituents would likely be cheaper on gas as it may avoid copies.

Recommendation	Consider fixing the typographical issues.
Resolution	PARTIALLY RESOLVED The `DvnVerifyParams` are now unpacked.

2.40 msglib_types::worker_options

The `worker_options` module defines the `IndexOptionsPair` struct, which contains the `options` and `dvn_idx` fields. These fields specify an index into an array of DVNs and the corresponding DVN option data (encoded as a byte array). This module includes various helper functions, including `insert_dvn_option`, which iteratively appends new DVN options to an array of `IndexOptionsPair` structs, and functions for converting a single concatenated byte array into an array of individual options per DVN, represented as `IndexOptionsPair` structs. Note: it is possible for a DVN to have more than one option.

More generally, this module provides functionality to extract both executor and DVN options from a single concatenated byte array, converting them into separate bytes arrays for executor and DVN option byte arrays. There are legacy options (where the option type equals 1 or 2), which include only the executor options, and `type_3` options (where the option type equals 3), which can include both executor and DVN options. The main public function for extracting these options is `extract_and_split_options`.

2.40.1 Privileged Functions

None.

2.40.2 Issues & Recommendations

Issue #59	Executional complexity of option grouping is exponential which could lead to gas usage concerns
Severity	● LOW SEVERITY
Description	The `group_dvn_options_by_index` function has a potentially exponential execution complexity as it needs to do a lookup in an array every time a DVN option is inserted. This could be a concern if the number of unique options grows large.
Recommendation	Consider whether it makes sense to redesign the option format. This may however not be possible as this format is already in use on the EVM. Optimizations could have been made by enforcing ordered options and using an ordered merge algorithm whenever options need to be merged. Most likely, this issue will need to be acknowledged.
Resolution	✓ RESOLVED The client has optimized the algorithm to benefit from when ordered options are provided, by always attempting to append to the last inserted grouping first. Users should still make sure to provide these options in an ordered manner however, as the algorithm does not sort or enforce sorting itself.

Issue #60	extract_type_3_options and extract_legacy_options both function even if an empty array is provided
Severity	● INFORMATIONAL
Description	The two public version-specific extract functions still "extract" empty option arrays even if the input does not contain the options type (3 or legacy being everything else). This may not be desired, as arguably an empty input is a malformed input.
Recommendation	Consider whether it makes sense to double check that the first two bytes are correct, given that these functions can be called directly.
Resolution	● ACKNOWLEDGED

Issue #61	Typographical issues
Severity	● INFORMATIONAL

Description	Line 138
	
	`/// serializaed options`
	
	This should be "serialized".
	
	———
	
	Lines 68 and 145
	
	`while (position < vector::length(options)) {`
	`while (position < vector::length(dvn_options_bytes)) {`
	
	The lengths of `options` and `dvn_options_bytes` can be cached to save some gas by not calling this procedure on every iteration.
	
	———
	
	The `simple_map.move` data structure from the Aptos standard library is extremely similar to the data structure of the `IndexOptionsPair` vectors. It would make sense to avoid writing excessive code and rely on such audited data structure implementations. Though it should be noted that `simple_map` suffers from the same gas-intensive complexity for its operations.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED Everything but the `simple_map` recommendation has been implemented.

2.41 treasury::treasury

The `treasury` module allows its `admin` to configure a native fee percentage of the worker fee that is sent to the LayerZero Treasury. Additionally, it allows a fixed ZRO fee to be configured as an alternative when payment is made in ZRO. The treasury address that receives the fee tokens can be configured using the `deposit_address` configuration value.

When other modules call `treasury::pay_fee`, the fee is calculated and then deducted from the provided payment asset, which can be either ZRO or native coin.

Only one treasury instance should be deployed.

2.41.1 Privileged Functions

- `update_deposit_address`
- `set_zro_metadata_address`
- `set_native_bp`
- `set_zro_fee`

2.41.2 Issues & Recommendations

Issue #62	get_fee rounds against the favor of the protocol
Severity	● INFORMATIONAL
Description	Line 98 `total_worker_fee * config().native_fee_bps / 10000` The `get_fee` function calculates the treasury fee as a percentage of the `total_worker_fee`. However, this calculation rounds down the fee, which becomes especially noticeable when `total_worker_fee` is a small number. For example, if `total_worker_fee` is less than 10,000, no fee is charged. Generally, rounding down against the protocol's favor is a violation of best practice, as it can lead to potential exploits. In this case, however, it does not appear to pose a risk, so the issue is marked as informational. Nonetheless, if the client wants to adhere strictly to best practices, addressing this could be beneficial. Additionally, this calculation slightly limits the maximum value of `total_worker_fee`, though the impact is minor given the expected size of `native_fee_bps`.
Recommendation	Consider rounding in favor of the protocol.
Resolution	✓ RESOLVED The client has indicated this is fine and prefers to keep the protocol simple. No changes were made.

Issue #63	Unnecessary additional validation of ensure_primary_store_exists wastes gas
Severity	● INFORMATIONAL

Description	Line 145-146
	
	`let metadata = fungible_asset::asset_metadata(&deposit_asset);`
	`primary_fungible_store::ensure_primary_store_exists(deposit_address, metadata);`
	
	The `ensure_primary_store_exists` check for the `deposit_address` is duplicated, as it is already checked on Line 147 in the `primary_fungible_store::deposit` call.

Recommendation	Consider removing the unnecessary validation code.
Resolution	✓ RESOLVED The additional validation has been removed.

Issue #64	Typographical issues
Severity	● INFORMATIONAL

Description	Line 37, 67, 109, 134
	
	The use of `@0` rather than `@0x0` when setting/checking addresses reduces readability. Consider consistently using `@0x0`.
	
	———
	
	Line 146
	
	`primary_fungible_store::ensure_primary_store_exists(deposit_address, metadata);`
	
	This assertion appears to be redundant with the implementation of `deposit` below it.
	
	———
	
	Line 204
	
	`const ENOT_IMPLEMENTED: u64 = 5;`
	
	This constant does not appear to be used.
	
	———
	
	We want to remind the client that there is no easy way to denote that a fee type is not supported. It may make sense to change the type of `zro_fee` and `native_fee_bps` to `Option<u64>`. This way, the `ZRO` fee can be marked as unsupported even if the `zro_address` was recently configured.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.42 dvn_fee_lib_router_0::dvn_fee_lib_router

The first DVN fee library router is an immutable router that follows the router pattern described throughout this audit. The router pattern is used to simulate dynamic dispatch in Move, which does not natively support it.

Specifically, the DVN fee library router stack routes `get_dvn_fee` requests to the requested `dvn_fee_lib` implementation, which then calculates and returns the fee for verifying the provided message.

This immutable router can route to the first and second DVN fee libraries. If the input address does not match either of these libraries, the request will be forwarded to an upgradable placeholder. Callers should exercise caution when using this placeholder until it, too, is made immutable. Such caution is generally advised whenever interacting with a router in the router pattern.

DISCLAIMER: As with all routers, it's essential to remember that all provided parameters areUNTRUSTED. Therefore, the underlying fee libraries should never assume that the provided worker address actually exists or is related to the `get_dvn_fee` request.

2.42.1 Privileged Functions

None.

2.42.2 Issues & Recommendations

No issues found.

2.43 dvn_fee_lib_router_1::dvn_fee_lib_router

The secondary DVN fee library router is an upgradable placeholder, similar to other routers in this audit, serving as a temporary solution to mimic dynamic dispatch in Move, which lacks native support for it.

When new fee libraries are added, this router is upgraded to incorporate them, and then it is made immutable. During the upgrade, a third router is introduced and linked to the end of the second router and remains upgradable until additional fee libraries are added there.

The current implementation aborts all calls.

DISCLAIMER: As with all routers, it's essential to remember that all provided parameters are UNTRUSTED. This means that the underlying fee libraries should never assume the provided worker address actually exists or is related to the `get_dvn_fee` request.

2.43.1 Privileged Functions

None.

2.43.2 Issues & Recommendations

No issues found.

2.44 executor_fee_lib_router_0::executor_fee_lib_router

The first executor fee library router is an immutable router that follows the router pattern described throughout this audit. This pattern is used to simulate dynamic dispatch in Move, which does not natively support it.

Specifically, the executor fee library router stack routes `get_executor_fee` requests to the requested `executor_fee_lib` implementation, which then calculates and returns the fee for executing the provided message.

This immutable router can route requests to the first and second executor fee libraries. If the input address does not match either of these libraries, the request is forwarded to an upgradable placeholder. Callers should exercise caution with this placeholder until it becomes immutable. Such caution is generally advised whenever interacting with a router in the router pattern.

DISCLAIMER: As with all routers, it's crucial to remember that all provided parameters are UNTRUSTED. Therefore, the underlying fee libraries should never assume that the provided worker address actually exists or is related to the `get_executor_fee` request.

2.44.1 Privileged Functions

None.

2.44.2 Issues & Recommendations

No issues found.

2.45 executor_fee_lib_router_1::executor_fee_lib_router

The secondary executor fee library router is an upgradable placeholder, similar to other routers in this audit, serving as a temporary solution to mimic dynamic dispatch in Move, which does not support it natively.

When new fee libraries are added, this router is upgraded to incorporate them and then made immutable. During this upgrade, a third router is introduced and appended to the end of the second router, remaining upgradable until further fee libraries are added there.

The current implementation aborts all calls.

DISCLAIMER: As with all routers, it's essential to remember that all provided parameters are UNTRUSTED. Consequently, the underlying fee libraries should never assume that the provided worker address actually exists or is related to the `get_executor_fee` request.

2.45.1 Privileged Functions

None.

2.45.2 Issues & Recommendations

No issues found.

2.46 executor_fee_lib_0::executor_fee_lib

The first executor fee library serves as the primary and only executor fee library. Its main purpose is to calculate the fee that OApps must pay for the executor to execute messages on the destination chain.

The calculation involves summing the gas value with any native tokens requested on the destination chain. This combined value is then translated into a local chain native token value using the oracle modules, and a markup (the "multiplier") is applied. The resulting local value is returned as the fee.

The `get_executor_fee` function is callable by anyone and is not supposed to change states.

2.46.1 Privileged Functions

None.

2.46.2 Issues & Recommendations

Issue #65	**Fee with margin calculation can be in incorrect magnitude due to inconsistent denominations**
Severity	● HIGH SEVERITY
Description	Line 106 `let fee_with_margin = (margin_usd * native_decimals_rate) / native_price_usd + fee;` In the above calculation, `native_price_usd` is configured and managed by the price oracle and is always denominated according to its specific decimal multiplier. However, `margin_usd`, which is used alongside it, is controlled by the worker and might have a different denominator. This discrepancy could be particularly relevant if the price feed adjusts its denomination to accommodate tokens with significantly smaller or larger supplies.
Recommendation	Consider documenting an expected decimal rate for `margin_usd` (eg. BP) and actually dividing by that constant. Consider adjusting `native_price_usd` with the actual in-use `denominator` of that worker, or documenting that this value is in some fixed denomination instead (right now it is documented to be in the variable `denominator`.
Resolution	● ACKNOWLEDGED The client has indicated that they will be careful with configuring these parameters. They will not directly adjust the logic as this is identical to the other deployments such as Solana and EVM. No changes were made.

Issue #66	get_executor_fee does not use the effective price feed
Severity	● MEDIUM SEVERITY
Description	Line 27 `let (price_feed, feed_address) = worker_config::get_worker_price_feed_config(worker);` The `worker_common` modules allow workers to set up a delegate worker, which can handle price feed requests on their behalf. This setup means that, instead of configuring the price feed addresses directly, a worker can rely on the price feed configuration of its delegate. However, in the `get_executor_fee` function, the delegate worker configuration is not utilized.
Recommendation	Consider using the effective price feed of the worker instead.
Resolution	✓ RESOLVED Updated to `get_effective_price_feed`.

Issue #67	**The native_cap worker safeguard can be circumvented through many smaller transactions**
Severity	● LOW SEVERITY
Description	A safeguard is added where the worker will only bridge up to a configurable `native_cap` worth of destination native tokens. However, this cap is enforced per transaction which means that if someone wants to actually bridge a significant amount of native tokens, they could just spam transactions with the worker. A safeguard has been added to limit the worker to bridging only up to a configurable `native_cap` worth of destination native tokens. However, since this cap is applied on a per-transaction basis, it leaves the system open to potential abuse. An actor could bypass the intended limit by spamming multiple transactions, each bridging up to the maximum allowed `native_cap`, to transfer a significant amount of native tokens.
Recommendation	Consider at least documenting this so that the workers are aware of this.
Resolution	● ACKNOWLEDGED

Issue #68	get_executor_fee lacks validation of the executor options
Severity	● LOW SEVERITY
Description	The `get_executor_fee` function does not validate that any of the provided options are actually supported by the worker even though there are functions to validate this. We assume that these functions were added with the goal of actually being used on the hot path.
Recommendation	Consider validating that the options are supported.
Resolution	● ACKNOWLEDGED

Issue #69	Typographical issues
Severity	● INFORMATIONAL

Description	Line 15-16
	
	`/// This checks that the messagelib is supported by the worker, the sender is allowed, and the worker is unpaused,`
	`/// and that the worker is an executor`
	
	These comments contain the word "and" twice.
	
	———
	
	Line 122
	
	`fee = (((value * ratio) / denominator) * (multiplier_bps as u128)) / 10000;`
	
	A division before multiplication occurs here. Even though this is beneficial to prevent the value from overflowing, the downside is that the `fee` is less accurate. It may make sense to up the bitspace of the math operations to `u256` and perform multiplication before division.
	
	———
	
	Line 191
	
	`total_gas = total_gas * 102 / 100;`
	
	This and other arithmetic round in favor of the user, which is typically undesirable from a best practice perspective as it violates a DeFi security pattern. However, given that there is a floor margin typically, it may be simpler to leave things as-is.

Recommendation	Consider fixing the typographical issues.

Resolution	✓ RESOLVED The first typographical error has been resolved.

2.47 executor_fee_lib_0::executor_option

The executor_option submodule is a utility module utilized by the primary executor fee library. It implements the business logic necessary to encode and decode the raw option bytes into their corresponding option data. The encoding and decoding scheme adheres to the specifications of the EVM executor options, which are as follows:

- LZ_RECEIVE(type "1"): [gas: u128, value: u128] (with value being optional)
- NATIVE_DROP (type "2"): [amount: u128, receiver: Bytes32]
- LZ_COMPOSE (type "3"): [index: u16, gas: u128, value: u128] (with value being optional)
- ORDERED_EXECUTION (type "4"): []

Since these options can appear multiple times within a single option enumeration, the following summation rules apply:

- The lz_receive values are summed together.
- The native_drop values are summed per receiver.
- The lz_compose values are summed per index.
- The ordered_execution values are upserted, meaning that execution is ordered as soon as the option is present at least once.

The module exposes the ExecutorOptions resource, which contains a decoded version of the options. This resource can be created by anyone and should be considered a data resource rather than a value resource.

2.47.1 Privileged Functions

None.

2.47.2 Issues & Recommendations

Issue #70	Typographical issues
Severity	● INFORMATIONAL

Description	Line 38
	
	`/// The gas and value and index of a specific LZ Compose operation`
	
	This line contains "and" twice.
	
	———
	
	Line 113
	
	`while (*position < vector::length(buf)) {`
	
	The length should be cached to avoid the underlying length function being called on every iteration. This optimization is likely present in other sections of the codebase as well.
	
	———
	
	Lines 118, 122, 126 and 130
	
	`if (option_type == 1) {`
	`} else if (option_type == 2) {`
	`} else if (option_type == 3) {`
	`} else if (option_type == 4) {`
	
	All of these lines contain magic values which have an actual constant associated with them at the top of this module.
	
	———
	
	The `serialize_executor_options` function can be written more cleanly by splitting up the single for-loop into four for-loops.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.48 executor_fee_lib_1::executor_fee_lib

The second executor fee library is an unimplemented upgradeable placeholder that aborts all calls. Its deployment serves the purpose of allowing it to be incorporated within the first immutable executor fee library router.

When a secondary fee library needs to be added, this module can be upgraded with that logic and made upgradeable. This approach ensures that no changes to the router are required for the addition of the first new fee library.

This pattern aligns with the design employed in all dispatch routers throughout this audit. As always, callers should exercise particular caution when calling this module until it is implemented. Furthermore, Paladin cannot provide any insights into future implementations, as we did not receive any information regarding those developments at the time of this audit. We recommend that any future implementations undergo thorough audits, peer reviews, and comprehensive test coverage.

2.48.1 Privileged Functions

None.

2.48.2 Issues & Recommendations

No issues found.

2.49 dvn_fee_lib_0::dvn_fee_lib

The first DVN fee library is the only real fee library for DVNs. It is responsible for calculating the fee, incremented by the worker premium, for DVN destination transactions. This fee is determined based on the transaction size and the execution cost on the destination chain. The library utilizes price feeds to convert these destination costs into the value of the source native token. This value is then upmarked by a worker-configurable multiplier.

DISCLAIMER: It is important to note that both the DVN and executor fee libraries utilize approximate models of the actual execution fees. We have confirmed with the client that this approach is intentional for the sake of simplicity. They have indicated that when expanding to non-EVM chains, they will assess whether the current models are adequate and, if necessary, refine them in future libraries.

2.49.1 Privileged Functions

None.

2.49.2 Issues & Recommendations

Issue #71	Fee with margin calculation can be in incorrect magnitude due to inconsistent denominations
Severity	● HIGH SEVERITY
Description	Line 101 `let fee_with_floor_margin = chain_fee + (floor_margin_usd * native_decimals_rate) / native_price_usd;` In the above calculation, `native_price_usd` is configured and managed by the price oracle and is always denominated according to its `denominator` decimal multiplier. However, `margin_usd`, which is used alongside it, is controlled by the worker and might have a different `denominator`. This discrepancy could be particularly relevant if the price feed adjusts its denomination to accommodate tokens with significantly smaller or larger supplies.
Recommendation	Consider documenting an expected decimal rate for `margin_usd` (eg. BP) and actually dividing by that constant. Consider adjusting the `native_price_usd` with the actual in-use `denominator` of that worker, or documenting that this value is in some fixed denomination instead (right now it's documented to be in the variable `denominator`.
Resolution	● ACKNOWLEDGED The client has indicated that they will be careful with configuring these parameters. They will not directly adjust the logic as this is identical to the other deployments such as Solana and EVM. No changes were made.

Issue #72	get_dvn_fee does not use the effective price feed
Severity	● MEDIUM SEVERITY
Description	Line 27 `let (price_feed, feed_address) = worker_config::get_worker_price_feed_config(worker);` The `worker_common` modules allow workers to set up a delegate worker, which can handle price feed requests on their behalf. This setup means that, instead of configuring the price feed addresses directly, a worker can rely on the price feed configuration of its delegate. However, in the `get_executor_fee` function, the delegate is not utilized.
Recommendation	Consider using the effective price feed of the worker instead.
Resolution	✓ RESOLVED Switched to `get_effective_price_feed`.

Issue #73	Calldata size is based on local multi-signature configuration instead of the remote one
Severity	● LOW SEVERITY
Description	Line 112 `let quorum = multisig::get_quorum(worker_address);` The calldata size is proportional to the multi-signature quorum on the destination chain of a message. However, this size is based on the quorum size at the source chain, which could be different. It is reasonable to expect that the quorum is equal on all chains but not guaranteed.
Recommendation	Consider either documenting this or using a per-chain configuration.
Resolution	✓ RESOLVED The client will document that the same quorum must be used on all chains.

Issue #74	Typographical issues
Severity	● INFORMATIONAL

Description	Line 84
	
	`/// Apply the premium to the fee. I takes the higher of using the multiplier or the floor margin`
	
	"I" should be "it".
	
	Line 119
	
	`EXECUTE_FIXED_BYTES + VERIFY_BYTES + total_signature_bytes + 64`
	
	A large portion of this computation can be calculated off-chain and stored in a constant to save gas.
	
	———
	
	Several constants are set as magic values instead of explicit constants. This is inconsistent as some of the other constants have been extracted into explicit constant values.
	
	———
	
	All of the arithmetic rounds in favor of the user, which is not really a big deal here but violates a security pattern. It may be worth violating it for simplicity.

Recommendation	Consider fixing the typographical issues.
Resolution	● PARTIALLY RESOLVED Some of these errors have been resolved.

2.50 dvn_fee_lib_1::dvn_fee_lib

The second DVN fee library is an upgradeable placeholder used as a preset within the router pattern. Having a pre-registered placeholder in the router allows the LayerZero team to configure and add a new fee library without needing to install it on the extra upgradeable router.

Until configured, this library is not in use and does not contain a proper implementation — it aborts all calls. Once it is upgraded with a configuration, it is essential for the team to make the module immutable. We recommend that users planning to use this implementation validate it. Furthermore, we cannot attest to the specifics of this implementation as we have not reviewed it. Therefore, we advise the client to undertake all necessary security measures, including internal testing, internal peer review, and an external security review ("audit").

2.50.1 Privileged Functions

None.

2.50.2 Issues & Recommendations

No issues found.

2.51 price_feed_module_0::feeds

The 0th price feed module serves as the primary and only active price feed module within the LayerZero Aptos endpoint V2. Its main responsibility is to price the execution costs of transactions on destination chains, including Ethereum, Arbitrum, Optimism, and other EVM chains.

This module includes business logic for calculating execution costs across standard EVM chains, which account for gas costs similar to Ethereum. It also incorporates special logic for Arbitrum-based chains that consider the additional L1 fee required for publishing rolled-up transactions, as well as similar calculations for Optimism-based chains.

As the fee will be levied in APT or ZRO but the cost is in the destination chain token (e.g. ETH) and is based on the current destination chain activity (gas cost in gwei), the feed updater periodically updates the token values within the price feed.

Feeds can be created by anyone. The feed updater is a privileged role that can be configured by the owner to be any feed address. Updaters can then call functions such as `set_price`, `set_arbitrum_traits` and `set_native_token_price_usd` to configure the current price properties of the destination chains. Price properties are specified and stored per destination chain. If the updaters become inactive, the latest published price will remain in use. Even though it is typically recommended to have a staleness check where transactions stop working if the price is too old, here it seems more desirable to use an old price instead of breaking the workers from delivering transactions if the updaters stop functioning. Therefore, we will not recommend a staleness check.

Feeds will be enabled as soon as a price is published for them. We recommend that they are fully configured before publishing any prices for them.

DISCLAIMER: The first price feed imposes certain expectations regarding the magnitudes of native tokens on destination chains. Specifically, it anticipates that the number of decimals will be within a standard range and not excessively high. For instance, 18 decimals is generally acceptable, provided that the token does not have an infinitesimally small value. However, if a chain utilizes a native token with, for example, 36 decimals or exhibits a high numerical fee in general, it is likely that this oracle will struggle to process its gas fee values due to their size being excessively large. In such cases, the client should consider reducing the denominator and increasing the bit space of the calculations from `u128` to `u256` to accommodate such chains.

2.51.1 Privileged Functions

- `enable_feed_updater [ feed owner ]`

- disable_feed_updater [feed owner]
- set_denominator [feed owner]
- set_arbitrum_compression_percent [feed owner]
- set_eid_models [feed owner]
- set_price [feed updater]
- set_arbitrum_traits [feed updater]
- set_native_token_price_usd [feed updater]

2.51.2 Issues & Recommendations

Issue #75	Gas models appear to be approximative and do not reflect current L2 models
Severity	● LOW SEVERITY
Description	The Arbitrum and Optimism fee calculation models appear to be approximative models of the actual fee calculations on these respective chains. This is not really a surprise as those models get updated and improved over time, most notably, through the recent blob storage mainnet fork which significantly reduces L1 costs for these L2 transactions. From our perspective, using an approximate model is not that big of a deal especially since most of the underspecification within the models simply causes the gas cost to be larger than it actually is, which is not a problem for the security of the system. However, it does make sense to allow for as much flexibility as possible within these models, avoiding shared and constant parameters. It is also very unclear how this model will be able to map gas costs to other non-EVM chains such as Move-based chains, Solana and Cosmos-based chains. In particular, the Optimism model seems to be rough to use with the current more efficient Optimism L1 costs.
Recommendation	Consider reflecting on the models compared to the current gas models on Arbitrum and Optimism. Consider whether they need to be re-specified to adapt recent changes. Consider at least avoiding any hardcoded parameters, most notably the fixed 3188 constant within optimism. Consider at least avoiding any shared parameters, most notably the Arbitrum parameters which are shared amongst all Arbitrum chains.
Resolution	● ACKNOWLEDGED The client has confirmed that these approximations are fine for their purposes, given that the main funds at risk would be the workers' their buffers. Workers should be careful with adjusting their parameters if someone manages to incur more costs than the fee they pay. The client has indicated that they may fine tune this over time.

Issue #76	Gas model selection contains testnet evaluations even on mainnet
Severity	● LOW SEVERITY
Description	Lines 223 and 235 `let fee = if (dst_eid_mod == 110 || dst_eid_mod == 10143 || dst_eid_mod == 20143 || type == &ARBITRUM_MODEL_TYPE( } else if (dst_eid_mod == 111 || dst_eid_mod == 10132 || dst_eid_mod == 20132 || type == &OPTIMISM_MODEL_TYPE(` Depending on the chain, a different gas cost calculation model may be chosen from the three models: Default, Arbitrum or Optimism. However, the chain endpoint id is always checked to be the respective chain or their testnet endpoint version. The latter should not have to be done on mainnet. It is bad practice to have special logic for things which are not existent on the main deployment as it needlessly increases the attack surface of the module. We also recommend generating the Optimism lookup chain id at compile time.
Recommendation	Consider implementing a "compile-time integer constant" library that decodes constant addresses as integers. This would allow the library to provide functions like `const_uint32(@arbitrum_eid)`, which would evaluate differently based on the deployment environment. The underlying mechanism would involve parsing the address into a `uint32`. However, this approach has a notable drawback: the implementation of `const_uint32` would need to be executed on the hotpath or be explicitly cached. Neither option seems particularly appealing in terms of the elegance of the module. A more refined solution might be to forgo the testnet addresses entirely and instead depend on explicit configuration for those environments.
Resolution	● ACKNOWLEDGED The client is okay with keeping these in as they do not want to clutter the codebase with complicated compile time constants, which are not supported in an elegant manner.

Issue #77	Feed ownership cannot be directly modified
Severity	● LOW SEVERITY
Description	Unlike the common pattern of using capabilities to dictate ownership of a resource, feeds are persistently owned by their creator. The drawback is that during creation, the creator needs to be clever and use a resource account or object to future proof the ownership. Otherwise, it will be difficult to ever move the owner to a different signer.
Recommendation	Consider carefully documenting that these feeds should be created within resource accounts or objects. Alternatively, consider using the capability pattern for authorization, where a `store, copy` (and optionally `drop`) capability is emitted with creation.
Resolution	● ACKNOWLEDGED

Issue #78	EID prices cannot be explicitly discontinued by a feed provider
Severity	● LOW SEVERITY
Description	Once a feed provider enables a specific EID, it will seem as if the feed always provides a price for that chain. However, if the chain is discontinued, there is no way for the provider to explicitly remove the `eid` from its prices table. This lack of removal may be desired, as it reduces the complexity for integrators, but may be seen as a limitation if a feed wants to remove such an `eid`.
Recommendation	Consider whether it makes sense to have de-publishing functions, which explicitly remove the price from the pricing table. This issue will also be resolved on the note that continuity and simplicity are preferred over this optionality, which is also valid. In that case, no changes need to be made.
Resolution	● ACKNOWLEDGED The client recommends that this is communicated through magic values. Though this might be dangerous from our perspective.

Issue #79	Arbitrum traits cannot be configured per chain
Severity	● LOW SEVERITY
Description	The `set_arbitrum_traits` and `set_arbitrum_compression_perc-ent` functions only allow for the configuration of traits for the feed globally. This means that any Arbitrum-based chain is required to follow the same `a-rbitrum_compression_percent`, `arbitrum_gas_per_l2_tx`, and `a-rbitrum_gas_per_l1_calldata_byte` values. This seems like an unrealistic assumption, as all Arbitrum-style chains may upgrade these values on a different schedule, for example, when incorporat-ing L1 blobs. A similar issue arises with the Optimism `3188` constant, which appears to already be outdated, as it is smaller in reality on the Optimism mainnet but may differ on Optimism-based chains that have not incorporated blobs.
Recommendation	Consider moving these values into a `eid` partitioned table. Consider making the Optimism constant configurable if desired.
Resolution	● ACKNOWLEDGED The client has indicated that they might finetune this over time and that for now this suffices for their purposes.

Issue #80	estimate_fee_on_send lacks details on the timestamp of the data
Severity	● LOW SEVERITY
Description	As discussed in the introductory section, feed data is always outdated and to some extend stale, as the updaters will only periodically push the gas and token prices on-chain. This means that all prices fetched are old. This is generally fine, as a markup is used by the workers to ensure that they do not undercharge. However, more advanced workers might want to base their degree of markup on the degree of staleness of the price, or revert altogether (though the latter seems undesirable to us). However. there a timestamp is not attached to the feed data.
Recommendation	Consider adding a timestamp to the `Price` resource. It is up to the client on whether this timestamp is provided off-chain, or is simply fetched on-chain at the time of the `set_price` execution. The latter is simpler and more secure, but less accurate as the prices are always going to be older than when `set_price` actually executes.
Resolution	● ACKNOWLEDGED The client has indicated that they might finetune this over time and that for now not disclosing this timestamp suffices for their purposes.

Issue #81	Fees round in favor of the user
Severity	● INFORMATIONAL
Description	The fees within all of the arithmetic round down whenever division occurs, this is unfavorable to the fee receivers which may not be desired, and could furthermore cause unexpected results on small numbers, which is a best practice to avoid. The Arbitrum logic also appears to suffer from "mul-before-div" which reduces precision.
Recommendation	Consider whether it is desired to round in favor of the user. Since the fee receiver is arguably a user of the protocol as well typically (a dvn or executor), rounding in either direction is not a best practice violation by definition, so this issue will be resolved regardless of the chosen direction, as long as it is carefully considered. Consider at least however addressing the "mul-before-div" within the Arbitrum calculations, by moving the division by 100 to the end.
Resolution	✓ RESOLVED The client confirms they are fine with this, they furthermore confirm that this matches with the evm implementation. It should be noted that the "mul-before-div" subissue was not addressed.

Issue #82	Typographical issues
Severity	● INFORMATIONAL

Description

Lines 18-19

```
// The compression percent for arbitrum (base 100)
arbitrum_compression_percent: u64,
```

Setting the precision of this value to 100 seems quite low. Perhaps setting it in basis points (which is somewhat of a standard for less precise numbers in DeFi) is slightly more precise and flexible. Additionally, this value name is semantically incorrect, as it represents 100 - compression percent, though this is much less important. It may also make sense to limit the maximum value of this as a sanitation check during the setter, though this is also less important.

Line 68

```
feed_data_mut(feed_address).denominator = denominator;
```

It may be smart to further sanitize this input by validating that it cannot be zero. This way, a division by zero abort on the hot path is guaranteed to be avoided. Having a `denominator` be specific per destination chain might also be better, as the `denominator` could be strategically set inversely to the magnitude of the fees on each specific chain to avoid overflow aborts.

Line 80

```
public entry fun set_eid_models(account: &signer, params:
vector<u8>) acquires Feed
```

We do not directly see the benefit of all these complex multi-resource configuration functions that require significant encoding and decoding. All of this aggregation can be done outside of the core contracts with scripts, a primitive which allows for doing multiple calls in a single transaction. This way, encoding could have been fully avoided in this module.

Lines 91 and 121

```
table::upsert(&mut feed_data_mut(feed_address).model_ty-
pe, dst_eid, model_type);
table::upsert(&mut feed_data_mut(feed).prices, eid,
price);
```

From a gas perspective, it may make sense to cache the mutable reference to the feed data in a variable at the top of the function.

Line 160

```
/// Checks if a feed updater can has the permission to
write to a feed
```

This line should remove "can" as it should simply say the updater "has the permission".

Line 190

```
/// @return (price ratio, gas price in the native token,
gas price per byte
```

Throughout the codebase, the unit for the `gas_per_byte` value seems to be incorrectly described. It's in the "gas" unit, compared to a "native token" cost.

Line 287

```
* (arbitrum_gas_per_l1_call_data_byte as u128);
```

This value is already of the `u128` type. The cast appears unnecessary.

Lines 307 and 313

```
let gas_per_unit_eth = price::get_gas_price_in_unit_u12-
8(ethereum_pricing);
let gas_per_unit_opt = price::get_gas_price_in_unit_u12-
8(optimism_pricing);
```

These variable names are a misnomer, they should be "unit per gas" instead.

It can be noted that the Arbitrum compression percentage can be combined with the Arbitrum gas per L1 call data byte in a single number, to save some gas, as they are always used in combination.

It may make sense to explicitly modulo all `eid`s in the configurational setters and getters as they are modulo'd by 30,000 on the hot path.

A denominator of 1e20 alongside with the math being done in u128 instead of u256 might be unnecessarily limiting to the price value limits. It may make sense to up the space for the calculations to u256 depending on expected values. However, as the eventual expected fee is way smaller than u64, current values may be fine.

	`initialize`, `enable_feed_updater`, `disable_feed_updater`, `set_denominator`, `set_arbitrum_compression_percent`, `set_eid_models`, `set_price`, `set_arbitrum_traits` and `set_native_token_price_usd` lack events.
Recommendation	Consider fixing the typographical issues.
Resolution	● PARTIALLY RESOLVED Some of these issues have been resolved.

2.52 price_feed_module_0::price

The price submodule of `price_feed_module_0` is a simple submodule that defines the `Price` utility resource. A `Price` resource comprises three details regarding an oracle price:

- `price_ratio`: The relative value of the local native currency compared to the remote currency, multiplied by the "denominator"
- `gas_price_in_unit`: The cost of a single unit of gas in the remote native currency
- `gas_per_byte`: The execution cost associated with calldata size on the remote chain

Multiplying `gas_per_byte` by `price_ratio`, `gas_price_in_unit`, and the number of bytes of calldata yields the cost associated with the calldata in the local currency on the sending chain.

The `Price` resource can be created by anyone; therefore, it should not be viewed as a controlled or valuable asset.

2.52.1 Privileged Functions

None.

2.52.2 Issues & Recommendations

Issue #83	Typographical issues
Severity	INFORMATIONAL

Description	Lines 34 and 37
	`// Gets the gas price per byte in the native token`
	`/// Gets the gas price per byte in the native token as a u128`
	The `gas_per_byte` value appears to be denominated in "gas amount" and not in any native token. Also note how these comments are inconsistent with the number of dashes at the start. This is a small inconsistency throughout the codebase.
	Line 47
	`/// Append a Eid-tagged Prices to the end of a byte buffer`
	This should say "an Eid-tagged Price" instead.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.53 price_feed_module_0::eid_model_pair

The eid_model_pair submodule of the price_feed_module_0 defines the EidModelPair utility resource. An EidModelPair resource comprises the combination of a remote endpoint EID and its gas cost model type. There are three options for gas models:

- DEFAULT_MODEL (0)
- ARBITRUM_MODEL (1)
- OPTIMISM_MODEL (2)

Although new model types can be added over time, the is_valid_model_type function will return false for them, as it is expected that this module will be immutable. It should be noted that the provided model type and destination EID are not validated during the creation of EidModelPair resources, meaning that the resource instances must not be valid.

The EidModelPair resource can be created by anyone; therefore, it should not be viewed as a controlled or valuable asset.

2.53.1 Privileged Functions

None.

2.53.2 Issues & Recommendations

Issue #84	The EidModelPair should have the storeability to be future proof
Severity	INFORMATIONAL
Description	Line 13 `struct EidModelPair has copy, drop {` The `EidModelPair` struct presently does not define the `store` ability. This prevents dependent modules from using it and storing it in global storage within another resource. This limitation is unnecessary and also inconsistent with the structs of many of the other utility modules within this repository. It limits the flexibility of future modules using this utility module.
Recommendation	Consider adding the `store` ability.
Resolution	✓ RESOLVED

2.54 price_feed_module_1::feeds

`price_feed_module_1` is an upgradeable placeholder for a future price feed module. It can be upgraded with a proper implementation and is already configured on the primary price feed router for use. This allows the secondary price feed router to remain uninstalled for the first new price feed implementation to be made available. For now, it simply aborts on all calls.

2.54.1 Privileged Functions

None.

2.54.2 Issues & Recommendations

No issues found.

2.55 price_feed_module_2::feeds

`price_feed_module_2` is an upgradeable placeholder identical to `price_feed_module_1`. It can be upgraded with a proper implementation and is already configured on the primary price feed router for use. This allows the secondary price feed router to remain uninstalled for the first new price feed implementation to be made available. For now, it simply aborts on all calls.

2.55.1 Privileged Functions

None.

2.55.2 Issues & Recommendations

No issues found.

2.56 price_feed_router_0::router

The `price_feed_router_0` module represents a router pattern similar to the message lib router. Like the message lib routers, the price feed routers are used to mimic "dynamic dispatch" as Move does not support it.

The price feed router and its underlying modules are used to calculate the fee for executing and verifying specific messages. This fee is based on several factors, such as the current gas cost at the destination, the relative token value, the message size, the DVN complexity, and more.

The router contains a single function: `estimate_fee_on_send`. This function can be called by anyone, either indirectly through the router or directly to the underlying price feed modules. It is crucial that the current and future price feed modules do not assume that these functions can only be called by privileged callers such as the worker fee libs. The router will route the request to the requested `price_feed` module.

The function is supposed to return the following values: fee, price ratio, denominator, and native token price in USD.

2.56.1 Privileged Functions

None.

2.56.2 Issues & Recommendations

No issues found.

2.57 price_feed_router_1::router

Similar to the message lib routers, the price feed routers are used to mimic "dynamic dispatch" as Move does not support it.

This module, in particular, represents the placeholder router that can still be upgraded to add new price feeds. Once it is upgraded with a list of price feeds, it will be made immutable, and its final call (if none of the feeds are matched) will be to a new upgradeable placeholder router. This router is called as the final call within `price_feed_router_0` when no matching price feed is found in the immutable list of that module.

The only function within this module, `estimate_fee_on_send`, aborts. This is because this module is not intended to be actively used until it is configured through an upgrade.

2.57.1 Privileged Functions

None.

2.57.2 Issues & Recommendations

No issues found.

2.58 worker_common::worker_config

`worker_config` is the primary module for workers, such as executors and DVNs, to register and configure themselves.

To register, workers call `initialize_for_worker`. Afterward, they can configure which msg libs they officially support, which address they want to receive their fees on, which OApp senders to allow or deny, and so forth. The full list of configuration functions can be found in the privileged functions section below.

The configuration of a worker is managed by the signer representing that worker's address. This signer can then configure worker admins and role admins. These admins are used within the actual worker implementations, which are outside the scope of this audit.

Each worker can select exactly one fee library, which uses the worker's configuration for some of its fee details.

2.58.1 Privileged Functions

- `initialize_config_module [ @worker_common signer, callable once ]`
- `set_worker_admin [ worker ]`
- `set_worker_role_admin [ worker ]`
- `set_supported_msglibs [ worker ]`
- `set_worker_pause [ worker ]`
- `set_deposit_address [ worker ]`
- `set_price_feed [ worker ]`
- `set_price_feed_delegate [ worker ]`
- `set_worker_fee_lib [ worker ]`
- `set_default_multiplier_bps [ worker ]`
- `set_supported_options_types [ worker ]`
- `set_executor_dst_config [ worker ]`
- `set_dvn_dst_config [ worker ]`
- `set_allowlist [ worker ]`
- `set_denylist [ worker ]`

2.58.2 Issues & Recommendations

Issue #85	Unsupported option types do not appear to be rejected
Severity	● MEDIUM SEVERITY
Description	The `worker_config` module allows worker signers to manage a set of supported option types. However, the options actually used do not appear to be checked anywhere in the sending stack, which permits senders to provide different options that would likely revert for fee libraries iterating over all options. Furthermore, if a specific worker wants to disable all options and sets their supported option types to an empty array, sending would not be blocked for options that are supported by the fee library itself.
Recommendation	Consider evaluating explicitly that the provided worker options are supported for the specific worker during sends.
Resolution	● ACKNOWLEDGED The client has indicated that they only use supported option types for view and that this matches https://github.com/LayerZero-Labs/monorepo/pull/68.

Issue #86	Fee library configuration lacks explicit validation that the fee library is designed for the designated worker type
Severity	● LOW SEVERITY
Description	The `set_worker_fee_lib` function lacks any validation that the fee library is respectively an executor or DVN fee library, depending on the desired type.
Recommendation	Consider whether it makes sense to use the fee library router to figure out the `worker_id` of the fee library and use the response as a sanity check for this setter's worker.
Resolution	● ACKNOWLEDGED The client has indicated they are fine with this and that they trust the worker to set the correct fee library, as the router would supposedly otherwise stop it.

Issue #87	Typographical issues
Severity	● INFORMATIONAL

Description	Line 1
	`/// Registry is for workers to register their workers to authenticate with worker peripherals used by ULNs`
	This line still refers to a registry, even though this module is no longer called that.
	Line 7
	`use endpoint_v2_common::contract_identity;`
	This import appears unused. The one place its used is where `get_caller` is used. However, `get_caller` is already imported directly and does not need to be prefixed there.
	Line 62
	`worker_id: u8,`
	This is not asserted to be an expected value, though this might be desired for future proofing against future worker ids.
	Line 113
	`/// Get the worker id for the given signer account`
	This comment is incorrectly copied here.

Line 211

```
assert!(!is_worker_paused(worker), EWORKER_ACCESS_DENIE-
D);
```

This is the incorrect error code.

Line 245

```
EWORKER_AUTH_UNSUPPORTED_MSGLIB
```

This error still refers to the old "worker auth" module name.

Line 620

```
/// Event emitted when the worker DVN destination config
is set
```

This comment is incorrect here.

Line 738

```
const EADMIN_ALREADY_EXISTS: u64 = 1;
```

This comment is unused.

Several functions start with an `assert_worker_initialized` call, but the issue is that this is done inconsistently. Many of the functions within the module lack this sanity check. This is raised as typographical as existence checks still often protect against these functions from being called.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED Most errors have been resolved.

2.59 worker_common::worker_config_store

The `worker_config_store` is the underlying storage module for the `worker_config` module. It does not expose any functions to the exterior environment. Instead, all interactions must go through the main `worker_config` module.

2.59.1 Privileged Functions

None.

2.59.2 Issues & Recommendations

Issue #88	**Executor and DVN configurations cannot be reconfigured after they are set once**
Severity	● MEDIUM SEVERITY

Description	Lines 210 and 245
	
	`table::add(executor_dst_config, executor_id, ExecutorDstConfig {`
	`table::add(dvn_dst_config, dvn_id, DvnDstConfig {`
	
	The executor and DVN specific configuration details cannot be reconfigured for some reason, even though the configuration details will almost certainly change over time. This is because the code erroneously uses `add` instead of `upsert` when changing the configuration details.
	
	Within the `Table` implementation, `add` will always abort when a record already exists at the key, preventing these configuration values from being updated.

Recommendation	Consider using `upsert` instead in case configurational adjustments are expected, which seems to be the case given that the non-specific configurations can be adjusted over time as well.
Resolution	✓ RESOLVED `upsert` is now used.

Issue #89	A worker can set itself as the price feed delegate
Severity	● LOW SEVERITY
Description	It is fully possible for a worker to configure itself as the price feed delegate. This is not a desired setup and should be prevented from our perspective. As price feed delegation is not transient anyway, the impact of this issue is low. The system also does not support "multiple hops" of delegation, so any issues with regards to infinite loops of circular dependencies do not appear to be present. It should subsequently be noted that it can set the delegate to its existing delegate as well, though there does not appear to be any side-effect from this.
Recommendation	Consider explicitly validating that circular dependencies are not set.
Resolution	● ACKNOWLEDGED The client has indicated they don't mind this possibility. No changes have been mind.

Issue #90	Configuration setters lack validation that the worker id is the correct type
Severity	● LOW SEVERITY
Description	Even though a worker can only configure itself to be a single worker type through its `worker_id` (1 is executor, 2 is DVN), the functions which allow for specifying the executor and DVN configuration are fully accessible to any worker regardless of their self-declared `worker_id`. This means that a worker at a single address can configure both its executor details and its DVN details regardless of which `worker_id` it has. This could cause confusion and issues to dependencies that do not explicitly check the `worker_id`. Validation could subsequently be added to the `get_executor_dst_config_values` and `get_dvn_dst_config_values` function as well. Though if these cannot be incorrectly set, borrow would revert due to there being nothing to borrow. So if checks are added during addition, they would no longer be required here from our perspective.
Recommendation	Consider explicitly asserting the `worker_id` within the type-specific configuration setters.
Resolution	● ACKNOWLEDGED The client has indicated they prefer to keep the module lightweight.

Issue #91	Workers get initialized in the unpaused state, even though they cannot be fully configured yet
Severity	● LOW SEVERITY
Description	Line 107 `paused: false,` When a worker registers itself via `initialize_store_for_worker`, it must subsequently call various other functions to fully configure itself. Notably, it needs to configure its own executor/DVN settings and set its price feed selection. Until this is completed, the worker is essentially unusable. It may be beneficial to explicitly communicate this by initializing the worker in a paused state, allowing the worker to asynchronously configure itself without the risk of someone using an unfinished intermediary configuration.
Recommendation	Consider whether it makes sense to start in the paused state. This issue can be resolved as well by documenting a recommended worker initialization method that avoids such conflicts (e.g. by deploying them via a script transaction). Starting in the paused state has some UX downsides to the worker after all.
Resolution	✓ RESOLVED The client has indicated that this is intended behavior and matches the EVM. No changes have been made. We recommend workers to be careful with this.

Issue #92	An intermediary configuration state exists where native_decimals_rate is unset
Severity	● INFORMATIONAL
Description	Since the `native_decimals_rate` value is chain-specific, it cannot be made a constant. Instead, the module deployer needs to configure it after deployment. In the interim state while it is not configured, this value is absent, which could lead to side effects. Intermediary states like this pre-initialization state represent an unnecessary increase in the attack surface. This issue is marked as informational, as the absence of the value (`GeneralStore` resource) will likely prevent any important transactions from executing until the initialization is completed. This contrasts with other sections in the codebase where a default value is overwritten, allowing important executions to proceed with that default value.
Recommendation	Consider either using the pattern where compiler time constants are loaded via the address toml (using a library that converts the address into a number) or deploying and configuring everything in a single transaction to avoid these intermediary states. At worst, consider ensuring during deployment that no transactions get made during the intermediary state to avoid this unnecessary additional attack surface.
Resolution	✓ RESOLVED This whole state has been removed.

Issue #93	Significant usage of vector types which have an unbound gas specification
Severity	● LOW SEVERITY
Description	The store encodes the `admins`, `role_admins`, `supported_msglibs`, and `supported_option_types` as vectors. Vector operations and loading resources with such vectors incur a gas cost that correlates with the length of the vector. This means that if these vectors grow excessively large, operations on the resource may become prohibitively expensive. This issue is rated as low because the vectors in question are expected to remain reasonably sized under normal operations. It should also be noted that throughout the codebase, `Table` is used instead of `SmartTable`, which has its own set of trade-offs regarding the addition of new entries. We recommend that the client double-check whether they should indeed be using `Table` throughout the codebase over `SmartTable`. We also noticed that within the Aptos SDK, `Table` is frequently used, and in our opinion, `Table` is often safer for the client's needs than `SmartTable`, given that `SmartTable` retains multiple entries at a single resource. Thus, we believe that `Table` is indeed the safer option.
Recommendation	Consider either documenting this, enforcing size limits on these vectors or moving to (smart) tables which do not have such concerns. Moving to Tables may not be worth the gas cost trade-off if these vectors remain small.
Resolution	● ACKNOWLEDGED The client has indicated that since these are worker specific, it is fine that workers need to carefully manage the size of these. Workers should keep this concern in mind.

Issue #94	Typographical issues
Severity	● INFORMATIONAL

Description	Line 15
	
	`// The worker ID (either Executor: 1 or DVN: 3)`
	
	A DVN has worker ID 2.
	
	Line 30
	
	`allowlist_count: u64,`
	
	Using `TableWithLength` for the `allowlist` would be a better solution as it does not repeat trusted and audited library code.
	
	Line 111
	
	`deposit_address: worker_address,`
	
	It is never checked that this address can receive the tokens in question. This may be a good sanity check to implement, though the downside is that it would block initialization even if the `deposit_address` is immediately moved afterwards. Instead, it may be more sensible to then also provide this address as a parameter into the function.
	
	Line 117
	
	`assert!(vector::length(&admins) > 0, ENO_ADMINS_PROVIDED)`
	
	It is sensible to move this check upwards, to fail early and avoid wasting unnecessary execution gas.
	
	———
	
	It should be noted that there is a large inconsistency throughout this module between using `borrow_global` and the inline `worker_store` functions. Both direct (mutable) borrows are used alongside with the inline functions, seemingly completely inconsistent.
	
	Lines 202, 221 and 239
	
	`executor_id: u32,`
	`executor_id: u32,`
	`dvn_id: u32,`
	
	This should be `executor_dst_id` or something to indicate that its a chain ID (`dst_id`).

Line 286

```
public(friend) fun set_price_fee_delegate(worker:
address, delegate: address) acquires WorkerStore {
```

This function, alongside with its getter, should probably contain "feed" and not "fee" in its name.

We wonder if it is desired that all role admins can be removed.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED Most of the aforementioned errors have been resolved.

2.60 worker_common::multisig

The `multisig` module provides a public interface for interacting with the `SigningStore` struct, which is stored per worker in the `multisig_store` module. This module includes the `quorum` value, `signers` array, and `used_hashes` table. Workers can initialize a struct stored at their address with an initial quorum and signers, as well as iteratively update both values, as long as the number of signers is greater than or equal to the quorum.

This module also includes view functions for gathering these three variables for a given worker. Updating the quorum or signers, as well as validating a hash and adding it to the `used_hashes` table, requires that the caller has a `ContractCallRef` created by the worker, which validates calls to the `@worker_common` address.

DISCLAIMER: The `multisig` module relies on vectors to store its signers. These vectors consume unbound gas proportional to the number of signers (with a complexity of at least quorum x signers). If these vectors and quorums grow in size, the execution cost may therefore grow potentially exponentially.

2.60.1 Privileged Functions

- `set_quorum [ worker ]`
- `set_signers [ worker ]`
- `set_signer [ worker ]`

2.60.2 Issues & Recommendations

Issue #95	The signature assertion functions could be vulnerable to hash mining
Severity	● MEDIUM SEVERITY
Description	A common recommendation within ECDSA implementations is to never directly use input data as the "digest" that is signed. This is because it is possible to come up with a curve of digests that map to a known public key. It is only due to the fact that the hash function cannot be pre-image attacked that the whole signature scheme is correct. However, within the `multisig` module, there is no such hashing logic present. Instead, the hash is directly provided into the `assert_all_and_add_to_history` and `assert_signatures_verified` functions. It is therefore possible for an attacker to come up with inputs into eg. the public `assert_signatures_verified` function that pass the test, even when the signers never signed those inputs.
Recommendation	Consider documenting that the callers of these functions must strictly always do the hashing step within their own module, instead of relying on the users to provide these hashes. This could also be done within this module if desired, by having it accept a variable length `message` that is then hashed using eg. `keccak256`.
Resolution	✓ RESOLVED This is now documented.

Issue #96	Typographical issues
Severity	INFORMATIONAL

Description	Line 3
	
	`// keeps track of used hashes to prevent the same command being replayed (all hashes include an expiration time, which`
	
	The expiration actually seems to be provided separately from the hash.
	
	Line 4
	
	`// allows two of the otherwise same command be called at different times).`
	
	This should state "to be called" instead.
	
	———
	
	`hash` values are of variable size, which is not typical for hashes. This should be a 32 bytes digest.

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED

2.61 worker_common::multisig_store

The `multisig_store` module defines the `SigningStore` struct, which is stored at a worker's address and includes the following variables: `quorum` (minimum signers required to validate a hash), `signers` (public keys of valid signers for this worker), and `used_hashes` (stores whether a hash has been validated previously). It contains the base functions for creating and moving a struct to a worker on initialization, as well as updating the quorum and signers while ensuring that the number of signers is greater than or equal to the quorum.

Most importantly, it includes functions for validating that the signatures passed to validate a hash are valid and correctly map to the public keys associated with a worker. This validation occurs in the `assert_all_and_add_to_history` call, which first checks that a hash has not already been used, determines if there are enough valid signatures to meet quorum, and then places the hash into the `used_hashes` table. The signature validation scheme retrieves the public key from the signatures and compares them against the `signers` array.

Note: It is possible at the time of the preliminary round to validate hashes without requiring any signers, due to the lack of a check that `quorum` is greater than zero.

2.61.1 Privileged Functions

None.

2.61.2 Issues & Recommendations

Issue #97	Worker owner (signer) has full control over multisig and can adjust its quorum and all verification properties without going through the multisig
Severity	● GOVERNANCE
Description	Even though a multisig is used, the `signer` of the `worker` can always fully reconfigure the multisig without going through it. This means that the `signer` of the multisig has full governance over its structure and can for example set the `quorum` to `1`, itself.
Recommendation	Consider whether this is desired. If so, consider documenting it. If not, consider having the set functions behind multisig validation, or recommending that the signer is kept in the worker itself and that it should be governed by the multisig.
Resolution	● ACKNOWLEDGED The client has indicated this is the desired governance setup, this is thus a responsibility for the DVN to set their signer authority to be correctly safeguarded.

Issue #98	Lacking check that the number of signers >= quorum after callingset_multisig_signers
Severity	● INFORMATIONAL
Description	The `set_multisig_signers` function lacks a check that the number of signers being set is >= the current `quorum`. While this is checked in the `multisig::initialize_for_worker` function call, it is not checked in the `multisig::set_signers` function call. This can make it impossible to validate a hash.
Recommendation	Consider adding a check to the `set_multisig_signers` function to validate that the number of new signers is >= the current `quorum`, or at a minimum add a check to all functions which interact which this function, such as `multisig::set_signers`.
Resolution	✓ RESOLVED The recommended check has been added to `set_multisig_signers`.

Issue #99	Typographical issues
Severity	● INFORMATIONAL

Description	Lines 131 and 139
	
	`assert!(vector::length(signer) == 64, EINVALID_SIGNER_LENGTH);`
	`assert!(vector::length(&multisig_signer) == 64, EINVALID_SIGNER_LENGTH);`
	
	Previous magic values within this module were encoded as constants at the top of the module. It may make sense to be consistent and encode this as a constant as well.
	
	———
	
	It could be considered within the `assert_signatures_verified_internal` function to stop evaluating the for loop as soon as the quorum is reached, or enforce that just the quorum is provided. This could save some gas but comes at the downside of being slightly less sane.
	
	———
	
	`signing_store_mut` is sometimes not used and inconsistently replaced with `borrow_global_mut`.
	
	———
	
	Throughout the module, the `SigningStore` instances are misnamed as "worker" variables. They should be named as something more semantically accurate such as "signing_store".

Recommendation	Consider fixing the typographical issues.
Resolution	✓ RESOLVED