PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For LayerZero (EndpointV2 Sui)

07 October 2025

paladinsec.co   info@paladinsec.co

Table of Contents

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's EndpointV2, Call, ULN, OApp, OFT, PTB Builders and DNV packages on the Sui network. It is a diff-audit of the changes in the contract compared with a previous audit done by Paladin. 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
URL	https://layerzero.network/
Platform	SUI
Language	Move
Preliminary Contracts	https://github.com/LayerZero-Labs/monorepo/tree/main/packages/layerzero-v2/sui/contracts Commit: 81c17a63820599fd55f025963a329aed1a92a280 Checked against: de19d28bea6a0bf38384dd5b7d2dd841d11466f5
Resolution	N/A (all issues acknowledged)

1.2 Contracts Assessed

- contracts/endpoint-v2/sources/
  - endpoint_v2.move
  - internal/messaging_composer.move
  - internal/oapp_registry.move
- contracts/dynamic-call/
  - call/sources/call.move
  - call/sources/call_cap.move
  - multi-call/sources/multi_call.move
- contracts/message-libs/uln-302/sources/
  - uln_302.move
  - internal/send_uln.move
- contracts/message-libs/uln-common/sources/
  - dvn_assign_job.move
  - dvn_get_fee.move
  - dvn_verify.move
  - executor_assign_job.move
  - executor_get_fee.move
- contracts/workers/worker-common/sources/
  - worker_common.move
  - worker-infos/worker_info_v1.move
- contracts/workers/worker-registry/sources/
  - worker_registry.move
- contracts/workers/dvns/dvn/sources/
  - dvn_move
  - dvn_witness.move
  - hashes.move
  - internal/multisig.move
  - dvn-infos/dvn_info_v1.move
- contracts/workers/dvns/dvn-layerzero/sources/
  - dvn_layerzero.move
- contracts/oapps/oapp/sources/
  - oapp.move
  - endpoint_calls.move
  - oapp-infos/oapp_info_v1.move
- contracts/oapps/oft/oft-common/sources/
  - compose_transfer.move
  - migration.move
  - oft_compose_msg_codec.move

- oft_composer_manager.move
- contracts/oapps/oft/oft/sources/
  - oft.move
  - oft_impl.move
  - oft_ptb_builder.move
  - internal/pausable.move
  - oft-infos/oft_info_v1.move
- contracts/ptb-builders/msglib-ptb-builders/uln-302-ptb-builder/sources/
  - uln_302_ptb_builder.move

1.3 Overview of Key Changes

EndpointV2

- `register_oapp(oapp, oapp_info)` now takes serialized `oapp_info` and returns the created `MessagingChannel` address
- New/renamed APIs: `set_oapp_info`, `get_oapp_info`; composer equivalents `register_composer(...)`: `address`, `set_composer_info`, `get_composer_info`
- Docstrings and entry functions expanded; "`lz_receive_info/lz_compose_info`" naming fully replaced by "`oapp_info/composer_info`"
- Composer APIs
  - `register_composer` now takes `composer_info` (previously `lz_compose_info`) and returns the compose queue address
  - `set_composer_info` renamed from `set_lz_compose_info`
  - `get_composer_info` renamed from `get_lz_compose_info`

messaging_composer

- Data model
  - `ComposerInfo` → `ComposerRegistration`
  - Field `lz_compose_info` → `composer_info`
- Events
  - `ComposerRegisteredEvent` field renamed to `composer_info`
  - `LzComposeInfoSetEvent` → `ComposerInfoSetEvent`
- API changes
  - `register_composer` now returns compose_queue address and accepts `composer_info` (no non-empty constraint)
  - `set_composer_info` renamed from `set_lz_compose_info`; drops non-empty validation
  - `get_composer_info` renamed from `get_lz_compose_info`

- Removed strict checks
- EInvalidLZComposeInfo and related assertions removed; empty composer_info is allowed

OApp

- OApp objects are forced to be shared; APIs take &OApp or its object address

- New oapp::oapp_info_v1 codec (stores oapp_object, next_nonce_info, lz_receive_info, extra_info) used as Endpoint oapp_info

- New oapp::endpoint_calls helpers wrap endpoint entry points (set delegate, (init/clear/skip/burn), set libraries, set config, etc.)

- Internal fields changed (oapp_cap, admin_cap), with new getters: oapp_cap_id, admin_cap

Registries (internal modules)

- OApp registry: OAppInfo → OAppRegistration with oapp_info field; events renamed (OAppInfoSetEvent, etc.); empty oapp_info now allowed

- Composer registry: ComposerInfo → ComposerRegistration with composer_info; events renamed (ComposerInfoSetEvent, etc.); register_composer returns queue address

ULN 302 and shared call types

- Common worker call params moved to uln-common (dvn_get_fee, dvn_assign_job, executor_get_fee, executor_assign_job, new dvn_verify::VerifyParam)

- Uln302.verify now consumes a Call<VerifyParam, Void> and asserts the caller instead of raw params

- Minor safety: endpoint-caller assertions added in send/confirm flows

Dynamic call/multi-call

- Old `call/multi_call` removed; reintroduced under `dynamic-call/*` with updated packages

OFT core

- New upgrade/migration model: `upgrade_version` constant and checks, `MigrationCap`, `MigrationTicket`, `migrate`
- `Pausable` extracted to `oft::pausable`; OFT now holds Pausable instead of a raw bool
- Fee-by-path: `oft_fee` now stores `default_fee_bps` plus per-`dst_eid` overrides; events/getters added; can unset per-EID fee
- Rate limiter events simplified (no package address); explicit drop methods for cleanup
- Send/quote/receive signatures now pass &OApp where relevant; `lz_receive_with_compose` now uses `oft_common::OFTComposerManager`
- View API expanded: default/effective fee getters, admin/migration/oapp object accessors, upgrade version, etc.

OFT initialization and PTB

- Removed OFT "factory" pattern; added `oft::oft_impl` with `OFTInitTicket` and `init_oft` / `init_oft_adapter` that share the OFT object and return `MigrationCap`.
- `oft_ptb_builder` merged under the OFT package; now passes `OApp` object to calls; helper `oft_package()` for latest package resolution.

Workers (DVN/Executor)

- New `worker_registry::worker_registry` (shared) to store versioned worker info for off-chain discovery
- New codecs: `worker_common::worker_info_v1`, `dvn::dvn_info_v1`, `executor::executor_info_v1`; DVN/Executor register themselves on creation
- Worker-common adds supported message-lib tracking (set/remove/check/assert) and events; allowlist APIs retained
- DVN: `create_dvn` shares object and returns its address; new signed admin flows (`set_supported_message_lib`, `set_worker_info`); `MultiCall` usage via `borrow_next`
- Executor: analogous owner-only APIs (`set_supported_message_lib`, `set_worker_info`); validation asserts supported message lib

Message-libs and PTB builders

- Multiple `Move.toml`/locks reduced and dependencies updated
- ULN PTB builder updated to use `uln-common`, `MultiCall`, and decoupled types.

1.4 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
● Governance	0	-	-	-
● High	0	-	-	-
● Medium	0	-	-	-
● Low	4	-	-	4
● Informational	3	-	-	3
Total	**7**	-	-	**9**

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.

1.4.1 endpoint_v2.move

ID	Severity	Summary	Status
01	INFO	Unauthenticated alert functions enable event spamming and log forgery in `lz_recieve_alert` and `lz_compose_alert`	ACKNOWLEDGED

1.4.2 oapp_registry.move

No issues found.

1.4.3 messaging_composer.move

No issues found.

1.4.4 call.move

No issues found.

1.4.5 call_cap.move

No issues found.

1.4.6 multi_call.move

No issues found.

1.4.7 uln_302.move

No issues found.

1.4.8 send_uln.move

No issues found.

1.4.9 dvn_assign_job.move

No issues found.

1.4.10 dvn_get_fee.move

No issues found.

1.4.11 dvn_verify.move

No issues found.

1.4.12 executor_assign_job.move

No issues found.

1.4.13 executor_get_fee.move

No issues found.

1.4.14 worker_common.move

No issues found.

1.4.15 worker_info_v1.move

No issues found.

1.4.16 worker_registry.move

No issues found.

1.4.17 dvn.move

ID	Severity	Summary	Status
02	LOW	`verify` does not enforce the supported message-lib allowlist	ACKNOWLEDGED
03	LOW	PTB builder one-time init is inconsistently enforced between admin and signatures paths	ACKNOWLEDGED
04	INFO	Missing upper bound on multiplier basis points enables fee mispricing	ACKNOWLEDGED

1.4.18 dvn_witness.move

No issues found.

1.4.19 hashes.move

No issues found.

1.4.20 multisig.move

ID	Severity	Summary	Status
05	LOW	Quorum can be set to 1, collapsing multi-signature into single-signature	ACKNOWLEDGED

1.4.21 dvn_info_v1.move

No issues found.

1.4.22 dvn_layerzero.move

No issues found.

1.4.23 oapp.move

No issues found.

1.4.24 endpoint_calls.move

ID	Severity	Summary	Status
06	LOW	`set_delegate` permits zero-address delegate	ACKNOWLEDGED

1.4.25 oapp_info_v1.move

No issues found.

1.4.26 compose_transfer.move

No issues found.

1.4.27 migration.move

ID	Severity	Summary	Status
07	INFO	`object::delete` should be replaced with `id.delete`	ACKNOWLEDGED

1.4.28 oft_compose_msg_codec.move

No issues found.

1.4.29 oft_composer_manager.move

No issues found.

1.4.30 oft.move

No issues found.

1.4.31 oft_impl.move

No issues found.

1.4.32 oft_ptb_builder.move

No issues found.

1.4.33 oapp_registry.move

No issues found.

1.4.34 pausable.move

No issues found.

1.4.35 oft_info_v1.move

No issues found.

1.4.36 uln_302_ptb_builder .move

No issues found.

2 Findings

2.1 endpoint_v2.move

The endpoint_v2 module orchestrates the public send, receive and compose surface for OApps and executors. It wires quoting and sending through msglibs, verifies and clears inbound messages, and constructs the dynamic call to the OApp ensuring end-to-end atomicity within a single PTB. It also exposes admin and OApp configuration via the manager, and small helpers for supported EIDs and channel state.

Key changes compared to the previous commit

- `register_oapp(oapp, oapp_info)` now takes serialized `oapp_info` and returns the created `MessagingChannel` address.
- New/renamed APIs: `set_oapp_info`, `get_oapp_info`; composer equivalents `register_composer(...)`: `address`, `set_composer_info`, `get_composer_info`.
- Docstrings and entry functions expanded; "`lz_receive_info/lz_compose_info`" naming fully replaced by "`oapp_info/composer_info`".

- Composer APIs
  - `register_composer` now takes `composer_info` (previously `lz_compose_info`) and returns the compose queue address.
  - `set_composer_info` renamed from `set_lz_compose_info`.
  - `get_composer_info` renamed from `get_lz_compose_info`.

2.1.1 Issues & Recommendations

Issue #01	Unauthenticated alert functions enable event spamming and log forgery in `lz_receive_alert` and `lz_compose_alert`
Severity	INFORMATIONAL
Description	The alert helpers are callable by any holder of a `CallCap` and do not authenticate the caller as an authorized executor, nor correlate alerts to real, verifiable messages. This enables arbitrary emission of alert events with forged metadata, inflating logs and confusing off-chain monitoring and incident pipelines.
Recommendation	Require authentication and/or on-chain correlation before emitting alerts.
Resolution	ACKNOWLEDGED The team commented: "The executor in the endpoint is intentionally designed to be permissionless. Off-chain systems will properly filter and process events based on the executor address to ensure that only valid events are handled."

2.2 oapp_registry.move

The `oapp_registry` module maintains authoritative mappings from an OApp's identity to its owned channel and configuration, and enforces "OApp-only" invariants for sensitive operations. It exposes asserts and lookups used throughout Endpoint to validate ownership and existence.

2.2.1 Issues & Recommendations

No issues found.

2.3 messaging_composer.move

The `messaging_composer` module defines the composer identity and the `ComposeQueue` used for compose callbacks. It supports enqueuing compose messages by hash and index, clearing on execution with a delivered sentinel to block replay, and simple presence checks for tooling.

2.3.1 Issues & Recommendations

No issues found.

2.4 call.move

The `call` module defines the generic `Call<Param, Result>` execution primitive and its capability `CallCap`. It provides a "hot-potato" pattern for composing sub-calls and finalizing them atomically within a single PTB. Helpers cover creating a call, batching children, destroying and finishing children, and completing the parent, plus a small `multi_call` utility for transient fan-out. The type system (no drop on Call) enforces that calls are consumed or the PTB reverts.

There are no differences between the two commits.

2.4.1 Issues & Recommendations

No issues found.

2.5 call_cap.move

The `call_cap` module defines `CallCap` capability objects used to authorize cross-contract call operations in LayerZero on Sui. It supports two capability sources: individual caps, whose identity is the cap's own UID, and package caps, created via a one-time witness so their identity is the package address—allowing all instances from the same package to share authorization. This is encoded via CapType (Individual or Package(address)), eliminating external mappings between cap IDs and package addresses. The module provides creation and identification helpers so both protocol-level (package) and per-instance (individual) operations can be authorized consistently.

This contract was audited from the ground up as it was not in the previous audit.

2.5.1 Issues & Recommendations

No issues found.

2.6 multi_call.move

The `multi_call` module provides a lightweight container for batching multiple dynamic Call invocations inside a single PTB. It lets a caller collect child calls, then finalize the batch atomically. It is intended for transient, in-PTB fan-outs and complements the call module's parent and child pattern.

2.6.1 Issues & Recommendations

No issues found.

2.7 uln_302.move

The ULN 302 (Ultra Light Node) module defines a configurable cross-chain messaging library that provides decentralized verification through executors and DVNs (Decentralized Verifier Networks). It establishes a flexible, bidirectional security model for OApps, enabling both outbound and inbound message protection with per-endpoint customization.

The module implements a staged "call-and-confirm" pattern for quoting fees, sending messages, and verifying deliveries, ensuring atomic execution across all participants. ULN 302 exposes helpers for executor assignment, DVN verification, per-OApp configuration, and system-wide defaults.

2.7.1 Issues & Recommendations

No issues found.

2.8 send_uln.move

The `send_uln` module defines the generic `Call<Param, Result>` execution primitive and its capability `CallCap`. It provides a "hot-potato" pattern for composing sub-calls and finalizing them atomically within a single PTB. Helpers cover creating a call, batching children, destroying and finishing children, and completing the parent, plus a small `multi_call` utility for transient fan-out. The type system (no `drop` on `Call`) enforces that calls are consumed or the PTB reverts.

2.8.1 Issues & Recommendations

No issues found.

2.9 dvn_assign_job.move

The `dvn_assign_job` module provides `AssignJobParam`, a thin wrapper around `dvn_get_fee::GetFeeParam` for submitting actual DVN verification jbs after quote. By reusing the exact fee quote parameters as the job assignment payload, it keeps the quote and send flows consistent and type safe. Includes a constructor from the base params and getter to access underlying fee parameters.

2.9.1 Issues & Recommendations

No issues found.

2.10 dvn_get_fee.move

The `dvn_get_fee` module defines the `GetFeeParam` payload used to quote DVN verification fees for cross-chain message verification. It captures the destination endpoint (`dst_eid`), encoded `packet_header`, `payload_hash`, required confirmations, the source sender address, and DVN-specific options, along with a constructor and getters. This struct standardizes how callers request DVN fee calculations, ensuring all routing and security-relevant fields (like confirmations and payload hash) are present and typed.

2.10.1 Issues & Recommendations

No issues found.

2.11 dvn_verify.move

The `dvn_verify` module is a ULN Common structure defining `VerifyParam` for DVNs to submit packet header, payload hash, and observed confirmations to ULN during inbound verification.

This contract was audited from the ground-up as it is a new file.

2.11.1 Issues & Recommendations

No issues found.

2.12 executor_assign_job.move

The `executor_assign_job` module provides `AssignJobParam`, wrapping `executor_get_fee::GetFeeParam` for submitting execution jobs using the same parameters as fee quoting. This mirrors the DVN pattern to maintain consistency between quote and dispatch, enabling straightforward promotion of a quoted execution into a concrete job with a constructor and a getter to access the base fee parameters.

2.12.1 Issues & Recommendations

No issues found.

2.13 executor_get_fee.move

The `executor_get_fee` module defines the `GetFeeParam` payload for quoting execution fees with the executor role. It includes the destination endpoint (`dst_eid`), source sender address, expected `calldata_size`, and execution options (e.g., LZ receive options), plus a constructor and getters. This standardizes the information needed for cost estimation of message execution on the destination chain..

2.13.1 Issues & Recommendations

No issues found.

2.14 worker_common.move

The `worker_common` module is the shared worker core used by DVN and Executor. It encapsulates the worker's deposit address, price feed, fee library, default multiplier, pause state, ACLs (allow/deny), supported option types, and the worker CallCap. It also provides admin and owner-gated configuration, ACL enforcement, pause checks, and read-only accessors, as well as emits events on configuration changes.

2.14.1 Issues & Recommendations

No issues found.

2.15 worker_info_v1.move

The `worker_info_v1` module defines a standardized way to store and handle information about a worker in the worker data format. It introduces a structure called `WorkerInfoV1`, which contains two main fields: a `worker_id` that identifies the type of worker, and a worker_info field that holds additional details about the worker as a byte array.

The module allows developers to create new instances of this structure, retrieve its data, and securely convert it to and from a byte representation. To ensure compatibility and data integrity, it includes a version constant (`WORKER_INFO_VERSION = 1`) and error checks that prevent invalid or outdated data from being used.

When encoding, the worker information is serialized into a versioned byte vector using Binary Canonical Serialization (BCS), making it suitable for storage or transmission. When decoding, the version is verified and the data is reconstructed, ensuring the information matches the expected format.

2.15.1 Issues & Recommendations

No issues found.

2.16 worker_registry.move

The `worker_registry` module provides a secure, on-chain registry for worker metadata, where:

- Workers can self-register or update their own info (protected via `CallCap`).
- Others can look up worker info based on address.
- Updates emit events for traceability.
- It includes utilities for testing and event validation.

This is useful in decentralized systems needing verifiable worker participation (e.g., cross-chain agents, oracles, validators, or off-chain compute nodes).

2.16.1 Issues & Recommendations

No issues found.

2.17 dvn.move

The `dvn` module is the core DVN worker for LayerZero v2 on Sui. It owns the worker capability and DVN identity (VID), maintains destination-specific fee config, an ECDSA-secp256k1 multisig committee and quorum, and a replay-protection hash table. It integrates with ULN-302 to quote and assign verification jobs via `MultiCall`, enforces ACL and paused status, and verifies inbound packets after asserting DVN signatures over standardized payloads. It also emits events on destination-configuration changes.

2.17.1 Issues & Recommendations

Issue #02	`verify` does not enforce the supported message-lib allowlist
Severity	● LOW SEVERITY
Description	The `verify` entrypoint accepts an arbitrary `target_message_lib` and relies solely on signer approval of that address. Other flows enforce a `supported_message_libs` policy, but `verify` bypasses that control, allowing payloads to target contracts outside the curated set. This widens the attack surface, for example if signers are compromised or a payload is constructed incorrectly, and breaks the invariant that DVN interactions only occur through approved message libraries.
Recommendation	We recommend enforcing the same allowlist policy on verify by invoking `assert_supported_message_lib(target_message_lib)` before dispatch.
Resolution	● ACKNOWLEDGED The team commented: "The multisig holds the highest level of authority and is fully responsible for its own signatures. Other interfaces require additional checks only because they are publicly accessible, unlike the multisig operations."

Issue #03	PTB builder one-time init is inconsistently enforced between admin and signatures paths
Severity	● LOW SEVERITY
Description	The admin path `init_ptb_builder_move_calls` enforces a one-time initialization invariant and rejects subsequent re-initialization attempts. The signatures path `set_ptb_builder_move_calls` does not check the initialization `ptb_builder_initialized` flag and can overwrite the configuration after it is supposedly sealed, resetting the builder state. This creates inconsistent semantics and enables governance-level reconfiguration that operators may not expect.
Recommendation	We recommend normalizing semantics by enforcing the same initialization guard in the signatures path. If controlled reconfiguration via governance is desired, replace the "only once" invariant with an explicit, versioned update flow.
Resolution	● ACKNOWLEDGED The team commented: "The behavior of `init_ptb_builder_move_calls` is intentional. It allows convenient initialization of the DVN's PTB during the first setup. The multisig retains the authority to modify it later in case of upgrades or if the initial configuration proves suboptimal."

Issue #04	Missing upper bound on multiplier basis points enables fee mispricing
Severity	● INFORMATIONAL
Description	Both DVN destination configuration and Worker default configuration accept `multiplier_bps` as `u16` without enforcing a 10,000 bps cap. An admin can set values up to 65,535 bps (>100%), producing inflated or anomalous fees and violating assumptions that bps ≤ 10,000. Even if downstream logic clamps values during use, the misconfiguration happens at the point of setting, allowing policy drift and inconsistent fee behavior.
Recommendation	Consider adding an upper bound at configuration time by rejecting any `multiplier_bps > 10,000`. If zero is disallowed by policy, also enforce `multiplier_bps > 0` for further hardening.
Resolution	

2.18 dvn_witness.move

The `dvn_witness` module is a simple, zero-data struct representing a "witness" (proof or marker) for something related to LayerZero in the DVN system. It can be used for type checking or gatekeeping in other modules, where the presence of this witness might be required to proceed with certain actions (like verifying cross-chain messages, permissions, or protocol steps).

It is a common pattern in smart contracts and formal verification systems to use such phantom/witness types to enforce protocol rules at the type level.

2.18.1 Issues & Recommendations

No issues found.

2.19 hashes.move

The hashes module is a deterministic payload and hash builder for DVN authorization. It produces 4-byte function selectors, serialized payloads, and keccak256 hashes for DVN privileged flows: `verify`, `set_dvn_signer`, `set_quorum`, `quorum_change_admin`, `set_allowlist`, `set_denylist`, and set_pause. This module ensures off-chain pre-images match on-chain verification.

2.19.1 Issues & Recommendations

No issues found.

2.20 multisig.move

The `multisig` module is a lightweight multi-signature engine for DVN. Maintains an uncompressed secp256k1 signer set and a configurable quorum, emits events on signer and quorum updates, and verifies signatures by recovering pubkeys from a keccak-signed payload. It is used by `dvn::dvn` to gate privileged actions via off-chain approvals and on-chain checks.

2.20.1 Issues & Recommendations

Issue #05	Quorum can be set to 1, collapsing multi-signature into single-signature
Severity	● LOW SEVERITY
Description	The governance path allows `set_quorum` to be set with a value of 1, which reduces the committee to a single point of failure. With `quorum=1`, any one committee key can authorize privileged operations that are supposed to require multi-party consent like admin changes, fee and price-feed updates, ACL changes or PTB builder configuration. This violates the core threat model of a multi-signature design, makes key compromise or operator error catastrophic, and enables accidental misconfiguration that silently weakens security guarantees.
Recommendation	We recommend enforcing a minimum quorum ≥ 2 at the contract boundary for both initial creation and subsequent updates.
Resolution	● ACKNOWLEDGED The team commented: "Our design follows the convention of other VMs' multisig implementations, such as Gnosis and Sui's official multisig, both of which allow a quorum of 1. This provides flexibility for testing, single-operator environments, or specific governance setups where such configuration is acceptable."

2.21 dvn_info_v1.move

The `dvn_info_v1` module provides a minimal and versioned metadata container (`DVNInfoV1`) for DVN systems, with reliable (de)serialization support. It is useful for storing or transmitting DVN metadata in a consistent, version-controlled format—critical in decentralized systems where data integrity and forward compatibility are important.

2.21.1 Issues & Recommendations

No issues found.

2.22 dvn_layerzero.move

The `dvn_layerzero` module is the bootstrap for the DVN package. It mints and transfers the package `CallCap` to the sender, and provides `init_dvn` to safely create and share a configured `dvn::dvn` instance. It also ensures the provided worker capability originates from this package before construction.

2.22.1 Issues & Recommendations

No issues found.

2.23 oapp.move

The oapp module is the core framework for building omnichain apps on Sui using LayerZero v2. It encapsulates the app's identity (via a capability-bound address), admin authority, a registry of trusted cross-chain peers, enforced execution policies, and transient state for tracking in-flight sends.

This module integrates with the LayerZero endpoint to estimate costs, construct outbound messages, and validate inbound messages by asserting the caller is the endpoint and the sender matches a configured peer.

Admin-only configuration governs which peers are trusted and which execution options are mandated per destination and message type. Read-only accessors expose the app and admin addresses and current configuration.

2.23.1 Issues & Recommendations

No issues found.

2.24 endpoint_calls.move

The `endpoint_calls` module contain admin wrappers to operate the endpoint: register OApp/channel, set delegate/OApp info, init channels, clear/skip/nilify/burn inbound state, and configure send/receive libraries with grace/timeout.

2.24.1 Issues & Recommendations

Issue #06	**`set_delegate` permits zero-address delegate**
Severity	LOW
Description	`set_delegate` accepts `@0x0`, which may inadvertently disable delegation semantics or break downstream assumptions.
Recommendation	Add `assert!(new_delegate != @0x0, EInvalidDelegate)` (or reuse an existing invalid-address error) before forwarding to the endpoint.
Resolution	● ACKNOWLEDGED The team commented: "Allowing the delegate to be set to 0x0 is intentional — it serves as a way to remove or clear the existing delegate."

2.25 oapp_info_v1.move

The `oapp_info_v1` module is a versioned `OAppInfo` bundle that encodes `oapp_object`, `next_nonce_info`, `lz_receive_info`, and `extra_info` for endpoint storage and execution planning.

2.25.1 Issues & Recommendations

No issues found.

2.26 compose_transfer.move

The `compose_transfer` module is a traceable transfer object that wraps tokens and context (`from`, `guid`) for composer execution, enabling audited routing to composer deposit addresses.

2.26.1 Issues & Recommendations

No issues found.

2.27 migration.move

The migration module contain the following primitives: `MigrationCap` and typed `MigrationTicket` that bundles OFT components (OFT cap, treasury or escrow, extras) with enforced mutual exclusivity and safe dismantling/rehydration.

2.27.1 Issues & Recommendations

Issue #07	**`object::delete` should be replaced with `id.delete`**
Severity	INFO
Description	In 2024 edition, UID exposes `delete()` directly. Using `object::delete(id)` is legacy style and flagged by the style guide.
Recommendation	Replace `object::delete(id)` with `id.delete()`.
Resolution	● ACKNOWLEDGED

2.28 oft_compose_msg_codec.move

The `oft_compose_msg_codec` module specifies the payload used when invoking compose logic after a transfer is received. It encodes the original packet sequence, source endpoint, local decimal amount, the source side initiator, and an opaque message for the composer. The codec ensures a stable minimal format for on chain parsing and preserves execution context needed for downstream workflows.

2.28.1 Issues & Recommendations

No issues found.

2.29 oft_composer_manager.move

The `oft_composer_manager` module is a registry for composer deposit addresses and compose transfer routing. It builds and transfers `ComposeTransfer` to registered deposit addresses, indexing by composite key and emitting events.

2.29.1 Issues & Recommendations

No issues found.

2.30 oft.move

The `oft` module implements the Omnichain Fungible Token on Sui, integrating with LayerZero via an embedded OApp to handle secure cross chain messaging and peer verification.

The contract supports two treasury models, mint and burn or escrow and release, so tokens can either be burned on send and minted on receive, or escrowed on send and released on receive to preserve supply invariants. Amounts are normalized across chains using shared decimals with deterministic rounding to remove dust, and events are emitted for creation, sends, and receipts for off chain tracking.

Inbound messages undergo endpoint and peer authentication and can deliver tokens directly to recipients or trigger compose flows that enqueue follow up work for registered composers. The contract provides views into configuration and state and a pause switch for emergency control.

2.30.1 Issues & Recommendations

No issues found.

2.31 oft_impl.move

The `oft_impl` module is a bootstrap that issues an `OFTInitTicket` at initialization and provides `init_oft` / `init_oft_adapter` to consume the ticket, verify OApp binding, delegate to OFT, and return admin/migration caps.

2.31.1 Issues & Recommendations

No issues found.

2.32 oft_ptb_builder.move

The `oft_ptb_builder` contain PTB utilities that generate the serialized `lz_receive` execution plan for endpoint registration and dynamically build receive PTBs (standard vs compose) with upgrade-stable package resolution.

2.32.1 Issues & Recommendations

No issues found.

2.33 oapp_registry.move

The `oapp_registry` is an endpoint-side OApp registry that binds an OApp to its `messaging_channel`, stores `oapp_info` and a delegate, and exposes reads for routing and authorization.

2.33.1 Issues & Recommendations

No issues found.

2.34 pausable.move

The `pausable` is an embeddable pausable component that exposes `set_pause`, `is_paused`, and `assert_not_paused`, and emits a single state-change event.

2.34.1 Issues & Recommendations

No issues found.

2.35 oft_info_v1.move

The `oft_info_v1` is a versioned OFT metadata carrying the latest OFT package address and OFT object address; encodes/decodes for oapp_info_v1.extra_info to keep endpoint metadata aligned across upgrades.

2.35.1 Issues & Recommendations

No issues found.

2.36 uln_302_ptb_builder.move

The `uln_302_ptb_builder` module implements the library-specific side of PTB construction. It keeps a registry of worker PTBs and returns the `quote`, `send`, and `set_config` `MoveCall` sequences for ULN-302. It exposes `build_quote_ptb`, `build_send_ptb`, `build_set_config_ptb` and a `set_worker_ptbs` path for workers to register their PTBs. The calls are completed and destroyed using the builder's internal `CallCap` using a "hot-potato" pattern.

2.36.1 Issues & Recommendations

No issues found.

PALADIN
BLOCKCHAIN SECURITY