GUARDIAN

Orderly

Solana Staking

Security Assessment

March 21st, 2025

Summary

Audit Firm Guardian

Prepared By Daniel Gelfand, Nicholas Chew, Rodion,
Zdravko Hristov, Shrenik, Raj Kumar

Client Firm Orderly

Final Report Date March 21, 2025

Audit Summary

Orderly engaged Guardian to perform a security review of their new feature which enables users to stake assets cross-chain from Solana. From the 24th of February to the 5th of March, a team of 6 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Issues Detected Throughout the engagement 2 High/Critical issues were uncovered and promptly remediated by the Orderly team.

For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

🔗 Blockchain network: **Solana**

✅ Verify the authenticity of this report on Guardian's GitHub: https://github.com/guardianaudits

Table of Contents

Project Overview

Project Summary

Project Name	Orderly
Language	Solidity
Codebase	https://gitlab.com/orderlynetwork/orderly-v2/omnichain-ledger https://gitlab.com/orderlynetwork/orderly-v2/solana-proxy
Commit(s)	Initial commit for omnichain-ledger: cbd3cfbdb28df24289a48d1f1ec7a8d07eca6de6 Initial commit for solana-proxy: 22e18ea5851c73fcb517e114bdcef73080398a02 Final commit for omnichain-ledger: be89754dcbc5c4af2ae741d4dee3ae96ae771f70 Final commit for solana-proxy: d0328936432a021bab6eeae63808d693b38d4eb4

Audit Summary

Delivery Date	March 21, 2025
Audit Methodology	Static Analysis, Manual Review, Test Suite

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	1	0	0	0	0	1
● High	1	0	0	1	0	0
● Medium	4	0	0	2	0	2
● Low	29	0	0	15	2	12

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	● Critical	● High	● Medium
Likelihood: *Medium*	● High	● Medium	● Low
Likelihood: *Low*	● Medium	● Low	● Low

Impact

High	Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.
Medium	A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.
Low	Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High	The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.
Medium	An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.
Low	Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts. Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Findings & Resolutions

ID	Title	Category	Severity	Status
C-01	Rewards Can Be Claimed Through send_request.rs	Validation	● Critical	Resolved
H-01	Misconfigured OApp Config Results In Failed Messages	Configuration	● High	Acknowledged
M-01	ClaimReward Backward Fee Is Always Paid	Unexpected Behavior	● Medium	Acknowledged
M-02	ledgerOappSend Fails Due To Fees Not Sent	Logical Error	● Medium	Resolved
M-03	Receiver_token_account May Not Be Initialized	DoS	● Medium	Resolved
M-04	Destination Gas For Payload Types Not Set	Logical Error	● Medium	Acknowledged
L-01	Common Options When Sending Messages To EVM Chains And Solana	Unexpected Behavior	● Low	Resolved
L-02	Incorrect AdminRoleTransferred Event	Informational	● Low	Resolved
L-03	Stakes From Solana May Not Have Empty Payload	Best Practices	● Low	Resolved
L-04	Contracts Lack Withdraw Functions	Logical Error	● Low	Partially Resolved
L-05	Transfer_admin() Does Not Follow Two-Step Ownership	Best Practices	● Low	Acknowledged
L-06	Unused Constraint In send_request.rs	Warning	● Low	Resolved
L-07	Precision Loss When Sending To Solana	Informational	● Low	Acknowledged
L-08	Backward Fee May Not Match payloadType2LzOptions	Warning	● Low	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
L-09	Users Charged Backwards Fee Even If Their Request Is A No-Op	Informational	● Low	Acknowledged
L-10	orderAmountForCollect Should Round Up	Rounding	● Low	Acknowledged
L-11	Proofs Can Be Submitted In A Paused State	Validation	● Low	Resolved
L-12	Solana Chained ID Ahead By One	Warning	● Low	Resolved
L-13	Unordered Execution May Increase unlockTimestamp	Unexpected Behavior	● Low	Acknowledged
L-14	OApp Delegate Should Be Multisig	Warning	● Low	Acknowledged
L-15	Wrong Quote When Paying With Lz Token	Unexpected Behavior	● Low	Resolved
L-16	Redundant Message payloadType Check	Superfluous Code	● Low	Resolved
L-17	compose_msg Not Verified	Validation	● Low	Acknowledged
L-18	Solana Restart Related Risks	Unexpected Behavior	● Low	Acknowledged
L-19	CancelAllVestingRequests Can Still Be Submitted	Best Practices	● Low	Partially Resolved
L-20	Rate Limiter Vulnerable To Resource Exhaustion	Unexpected Behavior	● Low	Acknowledged
L-21	Transfer Function Used	Warning	● Low	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
L-22	LedgerOApp Can Receive Messages In Paused State	Validation	● Low	Resolved
L-23	Missing Storage Gap In LzTestData	Warning	● Low	Acknowledged
L-24	Insecure Proxy Initialization	Validation	● Low	Acknowledged
L-25	Redundant Mappings	Warning	● Low	Resolved
L-26	Pause On Lz_receive Lead To Stuck Tokens	DoS	● Low	Acknowledged
L-27	USDC Amount Is Not Converted To Local Decimals	Decimals	● Low	Acknowledged
L-28	Rate Limiter Can Be Updated But Is Not Implemented	Informational	● Low	Acknowledged
L-29	ledgerOappSend Fails When Options Unset	Logical Error	● Low	Resolved

C-01 | Rewards Can Be Claimed Through send_request.rs

Category	Severity	Location	Status
Validation	● Critical	send_request.rs	Resolved

Description

When claiming rewards from Solana, the send_claim program is expected to be used where the user first submits a proof, then the SendClaim function encodes the user's rewards and merkle root into the payload.

However, an attacker may bypass send_claim and use the send_request program instead to claim rewards. SendRequest does not validate that the payload type is not ClaimRewardSolana. The attacker is therefore able to set their desired claim amount and easily obtain a merkle root to match the current active root of the desired distribution.

On the Ledger chain, MerkleDistributor handles reward claims from Solana differently, without requiring a merkle proof, as it expects the check to be performed on Solana. However, by bypassing send_claim the attacker is able to claim any arbitrary amount from the distribution, draining the LedgerOCCManager of all ORDER tokens.

Recommendation

In send_request: 58 validate that the payload type is not ClaimRewardSolana .

Resolution

Orderly Team: The issue was resolved in commit cd0eaa8.

H-01 | Misconfigured OApp Config Results In Failed Messages

Category	Severity	Location	Status
Configuration	● High	layerzero.config.ts	Acknowledged

Description

Block Confirmation Mismatch

In layerzero.config.ts, messages sent from Solana to Orderly require 10 block confirmations. However, on the Orderly chain, the receive configuration requires 32 block confirmations (as seen in the ReceiveUln302 contract: Explorer Link, using the getUlnConfig function with OApp address 0x68835941c7C300bFEF44D1D68b83798791901eB8 and remoteId 30168).

This mismatch causes the sending OApp (Solana) to proceed with only 10 confirmations, while the receiving OApp (Orderly) rejects any message with fewer than 32 confirmations. As a result, messages will be blocked until either:

1. The sending OApp increases the outbound block confirmation requirement to 32, or

2. The receiving OApp decreases the inbound confirmation threshold to 10.

NOTE: Increasing the outbound block confirmation will also mitigate against any re-org risk.

DVN Mismatch

Similarly, layerzero.config.ts sets the required DVN to a LayerZero Labs address (4VDjp6XQaxoZf5RGwiPU9NR1EXSZn2TP4ATMmiSzLfhb). However, on the Orderly chain, the required DVN points to a dead DVN (0x690b1857EaA8c55850547d7C22148C0B99a71dCd). This discrepancy means that:

1. The sending OApp (Solana) pays the LayerZero Labs DVN to verify packets,

2. But the receiving OApp (Orderly) relies on a dead DVN, which performs no verification.

As a result, messages will be blocked until the receiving OApp updates its configuration to remove or replace the dead DVN.

Recommendation

1. Ensure that block confirmation settings match on both the sending and receiving OApps.

2. Update the DVN configuration to ensure that both the sending and receiving OApps reference the same, active DVN.

Resolution

Orderly Team: Acknowledged.

M-01 | ClaimReward Backward Fee Is Always Paid

Category	Severity	Location	Status
Unexpected Behavior	● Medium	send_claim.rs	Acknowledged

Description

When users claim rewards both from EVM chains and Solana, they pay backward fee to cover the message fees from Orderly network back to theirs. However, some `distributionIds` are rewarding only `esORDER`.

When these tokens are claimed, they are being staked in the system and no backwards message is sent. In result, users who claim their `esORDER` rewards pay an unfair fee.

Recommendation

You can track which `distributionId` is rewarding `esORDER` on the vault sides and don't charge fee if the user specified `distributionId` matches it.

Resolution

Orderly Team: Acknowledged.

M-02 | ledgerOappSend Fails Due To Fees Not Sent

Category	Severity	Location	Status
Logical Error	● Medium	LedgerOApp: 116, LedgerOCCManager: 203	Resolved

Description

With the `ILedgerOapp(ledgerOappAddr).ledgerOappSend(occMessage);` call from the `OCCManager`, ETH value is not sent to the `LedgerOApp`.

Because native value is not forwarded, and Orderly does not subsidize cross-chain messages, `_lzSend` will revert due to lack of messaging fee provided (error `NotEnoughNative`) and all messages back to Solana through the OApp will be permanently DoS'd.

Recommendation

Forward native value to the `LedgerOApp` so that the `ledgerOappSend` function call can succeed. This is achieved by:

1. Making the function `ledgerOappSend` `payable`

2. In `LedgerOCCManager`, calculate and forward the native fee.

Resolution

Orderly Team: The issue was resolved in commit 05fe8af.

M-03 | Receiver_token_account May Not Be Initialized

Category	Severity	Location	Status
DoS	● Medium	solana_proxy/src/lz_receive.rs	Resolved

Description PoC

lz_receive is called by the executor after a user sends a request with the ClaimUsdcRevenue payload type to the Ledger.

In the LzReceive struct, it is expected that the receiver_token_account is already initialized by the user. However, there is a possibility that this account is not initialized. If that happens, the executor's call to lz_receive will always revert with error "AccountNotInitialized", causing a DOS and loss of fees.

Recommendation

Initialize the account if it is not already initialized with init_if_needed.

Resolution

Orderly Team: The issue was resolved in commit 28077ba.

M-04 | Destination Gas For Payload Types Not Set

Category	Severity	Location	Status
Logical Error	● Medium	programs/solana-proxy/src/instructions	Acknowledged

Description

When sending a message from Solana to Orderly, the anticipated gas cost for executing lzReceive must be specified. Currently, this is determined by get_enforced_options, which provides a single gas value based on SEND or SEND_AND_CALL.

However, in LedgerOApp, lzReceive processes various payload types, each requiring different gas amounts. Using a single gas value results in users either: a) overpaying for simple operations, or b) underpaying, causing transactions to fail due to insufficient gas.

Recommendation

Consider implementing a mapping of payload type to destination gas in the Solana proxy, similar to the existing payloadType2DstGas mapping in OCCManager.sol.

Resolution

Orderly Team: Acknowledged.

L-01 | Common Options When Sending Messages To EVM Chains And Solana

Category	Severity	Location	Status
Unexpected Behavior	● Low	LedgerOCCManager.sol	Resolved

Description

According to the LZ docs, when sending messages to Solana, the specified `addExecutorLzReceiveOption` parameters are `compute_units` and lamports.

Currently, `LedgerOCCManager.buildOCCLedgerMsg()` will use the same `addExecutorLzReceiveOption(_oftGas, 0)`. This may result in unexpected behavior because the gas and `compute_units` are not the same.

In addition, you should send at least 1500000 lamports as mentioned in the docs.

Recommendation

Consider passing different values when sending to Solana.

Resolution

Orderly Team: The issue was resolved in commit 0401f8e.

L-02 | Incorrect AdminRoleTransferred Event

Category	Severity	Location	Status
Informational	● Low	solana-proxy/src/instructions/transfer_admin.rs	Resolved

Description

InTransferAdmin:apply the event emits the newly set admin as the old_admin, because it uses ctx.accounts.proxy_config.admin after is has already been updated. Consequently, the AdminRoleTransferred emits inaccurate information.

Recommendation

Cache the old admin and use that cached value in the event instead.

Resolution

Orderly Team: The issue was resolved in commit 5915655.

L-03 | Stakes From Solana May Not Have Empty Payload

Category	Severity	Location	Status
Best Practices	● Low	LedgerOCCManager.sol	Resolved

Description

When users stake their `ORDER` from Solana, they can pass arbitrary data in the compose message. Each of these fields are validated except `chainedEventId` and `payload`. The `chainedEventId` parameter is not used if the messages comes from Solana, so that's okay.

The payload however is further passed to `OmnichainLedgerV1.ledgerRecvFromVault()`. Even though it's not used there, for additional safety it's better to not use it.

Recommendation

Use empty bytes for the payload.

Resolution

Orderly Team: The issue was resolved in commit 4c9f0c8.

L-04 | Contracts Lack Withdraw Functions

Category	Severity	Location	Status
Logical Error	● Low	LedgerOApp.sol, programs/solana-proxy/src/instructions/	Partially Resolved

Description

The LedgerOApp contract includes a receive function, which allows it to accept native ETH. However, it lacks a withdrawal function, meaning any excess ETH that accumulates cannot be recovered when the contract is no longer in use. This could lead to stranded funds.

Other related contracts, such as ProxyLedger and LedgerOCCManager, already include a withdrawTo function, suggesting that LedgerOApp should have one for consistency. Similarly, the proxy_token_account is designed to hold protocol-owned USDC, which users claim.

However, there is no function to withdraw surplus USDC, potentially leaving funds locked in the contract. Since LedgerOCCManager includes a withdrawOrderTo function for ORDER token withdrawals, it would make sense for the Solana proxy to also provide a mechanism to withdraw USDC.

Recommendation

A withdrawTo function should be added to LedgerOApp to allow for the recovery of excess ETH. Additionally, a withdrawUSDCTo function should be implemented in proxy_token_account to ensure surplus USDC can be withdrawn when necessary.

Resolution

Orderly Team: The issue was resolved in commit b670398.

L-05 | Transfer_admin() Does Not Follow Two-Step Ownership

Category	Severity	Location	Status
Best Practices	● Low	transfer_admin.rs	Acknowledged

Description

In the current implementation, the admin is the single point of failure in the `apply()` function inside of the `transfer_admin.rs` instruction due to the single-step transfer.

Recommendation

Set pending admin first and then the pending admin has to accept the ownership

Resolution

Orderly Team: Acknowledged.

L-06 | Unused Constraint In send_request.rs

Category	Severity	Location	Status
Warning	● Low	send_request.rs: 36	Resolved

Description

In send_request.rs, the constraint backward_fee.order_backward_fee < msg_fee.native_fee @ProxyError:InsufficientMessagingFee is commented out and remains unused. This is inconsistent with the send_claim.rs instruction, where the constraint is actively enforced.

Currently, no overflow risk is present due to the overflow-checks = true setting in Cargo.toml. However, if this setting were to change in the future, the absence of this constraint could introduce an overflow risk.

Recommendation

Consider uncommenting and enforcing the constraint in send_request.rs to maintain consistency with send_claim.rs and mitigate potential overflow risks if overflow checks are ever disabled in future configurations.

Resolution

Orderly Team: The issue was resolved in commit c6a597a.

L-07 | Precision Loss When Sending To Solana

Category	Severity	Location	Status
Informational	● Low	Global	Acknowledged

Description

EVM `$ORDER` has 18 decimals precision, but the token on Solana has 10 decimals. This means users will suffer small losses on token bridging.

Recommendation

Document this so the users are aware.

Resolution

Orderly Team: Acknowledged.

L-08 | Backward Fee May Not Match payloadType2LzOptions

Category	Severity	Location	Status
Warning	● Low	set_backward_fee.rs, send_claim.rs, send_request.rs	Acknowledged

Description

When sending a message from Solana to Orderly, if ORDER tokens are expected in the backward message, an `order_backward_fee` is charged. However, this fee is applied uniformly across all payload types, without accounting for potential differences in gas requirements.

On the Orderly chain, the `LedgerOApp` contract maps payload types to gas options, meaning that different payload types may require varying amounts of gas.

This mismatch can lead to inconsistencies between the backward fee charged on Solana and the actual gas sent back from Orderly to Solana.

Recommendation

Consider implementing a mapping of payload types to backward fees on Solana instead of applying a single `order_backward_fee` across all payload types. Otherwise, document this behavior.

Resolution

Orderly Team: Acknowledged.

L-09 | Users Charged Backwards Fee Even If Their Request Is A No-Op

Category	Severity	Location	Status
Informational	● Low	OmnichainLedgerV1.sol	Acknowledged

Description

When users initiate withdrawal requests or they claim their vestings or rewards, the amount they are eligible to may be 0. In this case, a cross-chain message won't be executed, but they will still have paid to it in a form of a backwards fee.

Recommendation

No code changes needed, but document this so the users are careful.

Resolution

Orderly Team: Acknowledged.

L-10 | orderAmountForCollect Should Round Up

Category	Severity	Location	Status
Rounding	● Low	Staking.sol: 190	Acknowledged

Description

In the function `_unstakeOrderNow`, the penalty for early unstaking is calculated as:

```
orderAmountForCollect = (_amount * UNSTAKE_NOW_COLLECT_PERCENT) / 100; // 5% of
withdrawn amount
```

However, this calculation rounds down due to Solidity's integer division behavior, which benefits the user instead of the protocol.

Recommendation

To ensure the penalty always rounds up in favor of the protocol, modify the calculation using ceiling division:

```
orderAmountForCollect = (_amount * UNSTAKE_NOW_COLLECT_PERCENT + 99) / 100;
```

Resolution

Orderly Team: Acknowledged.

L-11 | Proofs Can Be Submitted In A Paused State

Category	Severity	Location	Status
Validation	● Low	send_claim.rs	Resolved

Description

The SubmitProof:apply() function in send_claim.rs doesn't check if the proxy is currently paused or not. This allows users to submit proofs during a paused state and can result in unexpected behaviors.

Recommendation

Consider disabling proof submissions if the proxy is paused.

Resolution

Orderly Team: The issue was resolved in commit 205027f.

L-12 | Solana Chained ID Ahead By One

Category	Severity	Location	Status
Warning	● Low	LedgerOCCManager.sol: 293	Resolved

Description

For EVM chains, the first message sent cross-chain has a `chainedEventId` of 0. In contrast, the first cross-chain message sent from SOL has a `solanaChainEventId` of 1 due to the pre-increment: `srcEid = solanaEid + solanaChainEventId`.

This difference may be unexpected for Orderly's off-chain systems.

Recommendation

Be aware of this difference and adjust the SOL `eventId` as necessary for off-chain systems to function appropriately.

Resolution

Orderly Team: The issue was resolved in commit 3f4524b.

L-13 | Unordered Execution May Increase unlockTimestamp

Category	Severity	Location	Status
Unexpected Behavior	● Low	orderly-omnichain-solana-team2/contracts/lib/Staking.sol: ln 198	Acknowledged

Description

Transactions sent via the Solana proxy to the Orderly network may not execute in the same order they were submitted. This unordered execution can lead to unexpected results.

For example, consider the following scenario:

A user intends to first execute a `withdrawOrder` transaction, since the `unlockTimestamp` has been reached, and subsequently initiate a new `CreateOrderUnstakeRequest`.

The user submits the `withdrawOrder` request via `send_request` on the Solana proxy, immediately followed by a `CreateOrderUnstakeRequest`.

However, due to the non-sequential execution of transactions on the Orderly network, the `CreateOrderUnstakeRequest` might execute before the `withdrawOrder`.

If this happens, the `unlockTimestamp` is extended by an additional `unstakeLockPeriod`, causing the subsequent `withdrawOrder` transaction to fail.

When the `withdrawOrder` transaction fails due to the extended lock period, the fees previously paid for its submission through the Solana proxy are forfeited.

Recommendation

Clearly document this behavior.

Resolution

Orderly Team: Acknowledged.

L-14 | OApp Delegate Should Be Multisig

Category	Severity	Location	Status
Warning	● Low	set_delegate.rs	Acknowledged

Description

The `proxy_config`'s delegate has critical controls over the OApp's configuration. To maximize safety, it is recommended for the delegate to be a multi-sig.

Recommendation

Consider using a multi-sig for the OApp's delegate.

Resolution

Orderly Team: Acknowledged.

L-15 | Wrong Quote When Paying With Lz Token

Category	Severity	Location	Status
Unexpected Behavior	● Low	quote_claim.rs; quote_send.rs	Resolved

Description

When sending crosschain requests through the solana-proxy, users specify the maximum fee they are ready to pay in both native tokens and `lz token`.

```
pub native_fee: u64, pub lz_token_fee: u64
```

However, `quote_claim.rs` and `quote_send.rs` hardcode `pay_in_lz_token` to `false` when it quotes the endpoint. Because of this, the quoting mechanism won't work for `lz token` payments even though the solana proxy allows such messages.

Recommendation

Allow the user to specify whether they want to pay with `lz token`.

Resolution

Orderly Team: The issue was resolved in commit 784be09.

L-16 | Redundant Message payloadType Check

Category	Severity	Location	Status
Superfluous Code	● Low	LedgerOCCManager.sol: 193	Resolved

Description

The require statement would never revert as the check is already performed in the if condition in the function `ledgerSendToVault()`.

Recommendation

Consider removing the require statement.

Resolution

Orderly Team: The issue was resolved in commit 53f8eea.

L-17 | compose_msg Not Verified

Category	Severity	Location	Status
Validation	● Low	oft/src/instructions/send.rs	Acknowledged

Description

For staking from Solana, the OFT is transferred with the composed message carrying the `OCCVaultMessage` to be processed on the Ledger Chain.

The compose message is intended to be created by Orderly's frontend, however any user can specify a compose message and custom front ends could be built.

This opens the user to potential issues such as if the composed message contents are malformed and reverts upon `lzCompose` execution, the user would have lost their funds since the OFT was burnt on src chain.

Recommendation

Clearly document these risks to users.

Resolution

Orderly Team: Acknowledged.

L-18 | Solana Restart Related Risks

Category	Severity	Location	Status
Unexpected Behavior	● Low	set_oft_config.rs: 34-40, set_pause.rs: 21	Acknowledged

Description

The internal state in the `solana-proxy` and `OFTStore` would be reverted to previous state in case if the Solana chain restarts.

Recommendation

Implement an check to verify the `LastRestartSlot` state to detect outdated configuration.

Resolution

Orderly Team: Acknowledged.

L-19 | CancelAllVestingRequests Can Still Be Submitted

Category	Severity	Location	Status
Best Practices	● Low	ProxyLedger.sol, send_request.rs	Partially Resolved

Description

The CancelAllVestingRequest payload is no longer supported, but it was left in the payload enum, so the payload types won't be shifted and the system will remain backwards compatible.

However, the code still allows sending such request from the vault sides and the users who initiate them will pay fees for a 100% reverting transaction.

Recommendation

Consider not allowing CancelAllVestingRequests to be sent.

Resolution

Orderly Team: The issue was resolved in commit 4136571.

L-20 | Rate Limiter Vulnerable To Resource Exhaustion

Category	Severity	Location	Status
Unexpected Behavior	● Low	peer_config.rs: 53-63, send.rs: 67-69	Acknowledged

Description

The rate limiter's implementation utilizes a basic token bucket model without transaction weighting, making it vulnerable to large transaction dominance.

An attacker can submit a small number of resource-intensive transactions, consuming the rate limit and preventing other users from submitting transactions, leading to potential DoS and resource starvation.

Recommendation

To prevent large transaction dominance, implement a tiered token bucket system. Create separate buckets for distinct transaction size ranges (e.g., small, medium, large).

This ensures dedicated capacity for all transaction sizes, preventing large transactions from monopolizing resources and ensuring fairness.

Resolution

Orderly Team: Acknowledged.

L-21 | Transfer Function Used

Category	Severity	Location	Status
Warning	● Low	orderly-omnichain-solana-team2/contracts/lib/LedgerOCCManager.sol: ln 359	Resolved

Description

In the `LedgerOCCManager`'s `withdrawTo` function, the owner can withdraw ether to a specified address. This is currently done using a low-level transfer function.

However, `transfer()` only forwards 2300 gas, which is insufficient for the recipient to execute any non-trivial logic in a `receive()` or fallback function, such as when recipient is a multisig wallet.

Recommendation

Consider using the low-level call function instead of the transfer function.

Resolution

Orderly Team: The issue was resolved in commit be89754.

L-22 | LedgerOApp Can Receive Messages In Paused State

Category	Severity	Location	Status
Validation	● Low	LedgerOApp.sol	Resolved

Description

The `LedgerOApp` can be paused by calling the `pause()` function. However, `_lzReceive()` doesn't use the `whenNotPaused` modifier which makes it possible for the proxy to receive messages while it's paused.

In most cases, the proxy on the Solana chain should be paused as well, but in case the protocol team wants to pause only receiving messages on Orderly Network, it won't work.

In addition, even if both proxies are paused, any messages sent before the pause, but still not received, can be executed.

Recommendation

Be aware of this behavior and consider implementing the `whenNotPaused` modifier on `lzReceive()`.

Resolution

Orderly Team: Resolved.

L-23 | Missing Storage Gap In LzTestData

Category	Severity	Location	Status
Warning	● Low	OCCAdapterDataLayout.sol	Acknowledged

Description

The storage layout for `LedgerOCCManager` is:

• `LzTestData` (no gap)

• `LedgerAccessControl` (5 gap slots)

• `OCCAdapterDatalayout` (50 gap slots)

• `LedgerOCCManager` (45 gap slots)

`LzTestData` does not have a storage gap, so caution should be taken not to introduce variables for future upgrades which would lead to collision issues.

Recommendation

Do not introduce new variables to `LzTestData` for future upgrades.

Resolution

Orderly Team: Acknowledged.

L-24 | Insecure Proxy Initialization

Category	Severity	Location	Status
Validation	● Low	init_proxy.rs	Acknowledged

Description

The current implementation of the proxy initialization functionality is susceptible to front-running attacks as there is no any checks for the signer address.

It has to be ensured that there is no possibility for malicious users to front-run the critical functionality of the protocol. As there is no sufficient validation, the function can be called by any account.

Recommendation

Consider adding the `ProgramData` account and check for the signer to be the expected `upgrade_authority` from the `program_data`.

Resolution

Orderly Team: Acknowledged.

L-25 | Redundant Mappings

Category	Severity	Location	Status
Warning	● Low	LedgerOApp.sol	Resolved

Description

The `LedgerOApp` contract on the Ledger chain has its own `chainId2Eid` and `eid2ChainId` mappings. If the mappings are updated, the update is not reflected in the mappings within of the `LedgerOCCManager` contract.

This may lead to inconsistent endpoint<>chain id's across the two contracts. Furthermore, the `chainId2Eid` is not even utilized beyond the setter function `setChainId2Eid`.

Recommendation

Consider having the `LedgerOApp` query the `LedgerOCCManager`'s eid <> chain id mappings to have one source of truth.

Resolution

Orderly Team: The issue was resolved in commit f1108aa.

L-26 | Pause On Lz_receive Lead To Stuck Tokens

Category	Severity	Location	Status
DoS	● Low	lz_receive.rs	Acknowledged

Description

Currently, Pause is used on `lz_receive` in both the Solana proxy and Solana OFT. Suppose a user sends order tokens from the Ledger to Solana. When the user initiates the call on the Ledger the send is not paused; however, when the Executor calls `lz_receive` on Solana OFT, it is paused.

This means that the user does not receive the order tokens on both chains, causing them to become stuck in the middle. It also causes every Executor call to revert.

Recommendation

Ensure a retry system is implemented and documented.

Resolution

Orderly Team: Acknowledged.

L-27 | USDC Amount Is Not Converted To Local Decimals

Category	Severity	Location	Status
Decimals	● Low	ProxyLedger.vaultRecvFromLedger()	Acknowledged

Description

When users claim their USDC revenue, they will be credited usdcRevenueAmount on the source chain. This amount is in the same decimal precision for all chains (most likely 6 decimals).

However, there are some LayerZero supported networks, like BNB, for example, where the USDC token has different decimals (on BNB it's 18). This will cause a severe loss of funds for these users.

Recommendation

Convert the amount of USDC received to its local decimals amount when it's received in ProxyLedger.vaultRecvFromLedger()

Resolution

Orderly Team: Acknowledged.

L-28 | Rate Limiter Can Be Updated But Is Not Implemented

Category	Severity	Location	Status
Informational	● Low	set_peer_config.rs: 58	Acknowledged

Description

The set_peer_config.rs file includes functionality to update the rate limiter. However, the rate limiter is not implemented in any of the Solana proxy instructions, making this update function redundant.

Recommendation

Review whether the update_rate_limiter function is necessary.

Resolution

Orderly Team: Acknowledged.

L-29 | ledgerOappSend Fails When Options Unset

Category	Severity	Location	Status
Logical Error	● Low	LedgerOApp.sol: 128	Resolved

Description

In the ledgerOappSend function, gas for the destination chain (Solana) is retrieved from the payloadType2LzOptions mapping and appended to options. However, if payloadType2LzOptions is unset, it returns a value of zero for gas.

This results in a stuck message on Solana, as no gas is paid for the executor to process the transaction. This behavior differs from the design in the LedgerOCCManager contract, which handles EVM flows differently.

In the buildOCCLedgerMsg function, if _dstGas = 0, a default value of 200000 is assigned to prevent execution failures (see LedgerOCCManager.sol#L130.

Furthermore, the likelihood of payloadType2LzOptions being unset is high because:

• The live LedgerOCCManager contract on the Orderly chain has payloadType2DstGas unset for all payload types.

• No test files or scripts indicate that the protocol intends to set payloadType2LzOptions.

Recommendation

Modify the ledgerOappSend function to assign a default gas value if typeOptions.gas = 0

Resolution

Orderly Team: The issue was resolved in commit 46dbcf5.

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits