

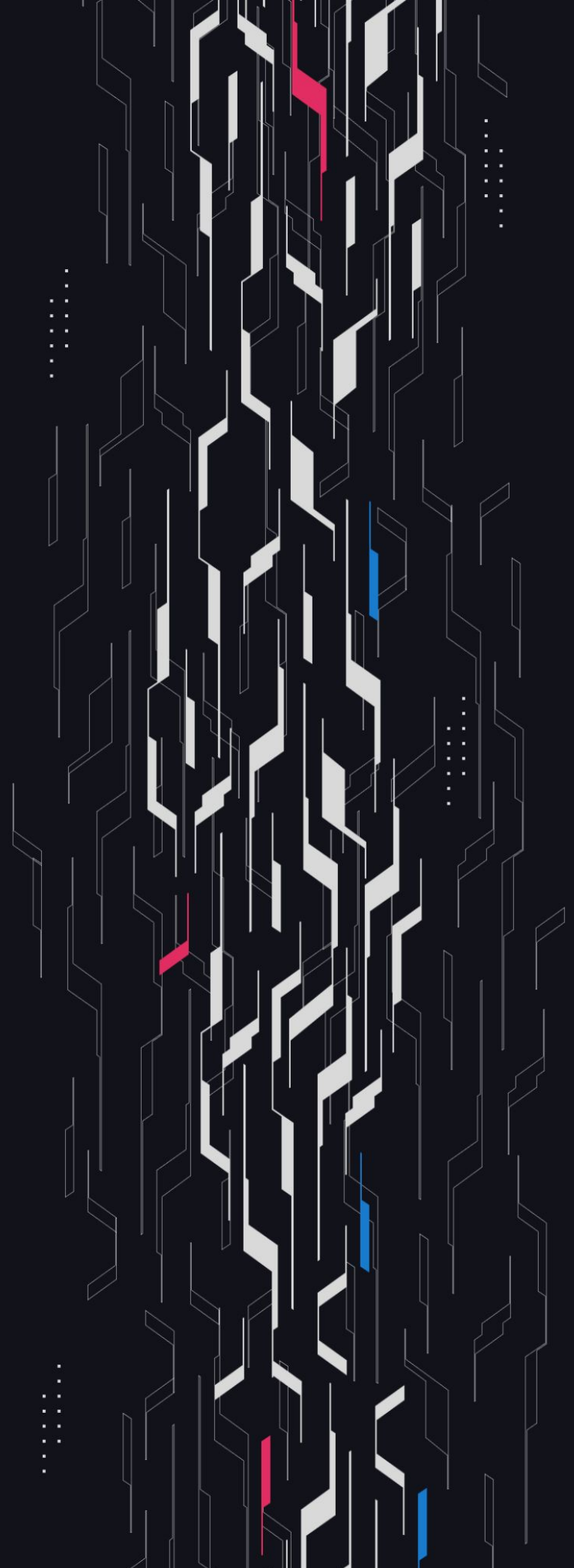
GA GUARDIAN

LayerZero

Console Review

Security Assessment

March 26th, 2026



Summary

Audit Firm Guardian

Prepared By Owen Thurm, Daniel Gelfand, Minato Namakazi, Wafflemakr

Client Firm LayerZero

Final Report Date March 26, 2026

Audit Summary

LayerZero engaged Guardian to perform three rounds of security review on the Console codebase. From January 19th to March 25th, a team of four auditors conducted a comprehensive assessment of the source code. All findings identified throughout the three rounds have been recorded in the following report.

Confidence Ranking

Given the lack of High and Critical issues detected during the main review, Guardian assigns a Confidence Ranking of 5 to the protocol. Guardian advises the protocol to consider periodic review with future changes. For detailed understanding of the Guardian Confidence Ranking, please see the rubric on the following page.

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

Guardian Confidence Ranking

Confidence Ranking	Definition and Recommendation	Risk Profile
5: Very High Confidence	Codebase is mature, clean, and secure. No High or Critical vulnerabilities were found. Follows modern best practices with high test coverage and thoughtful design. Recommendation: Code is highly secure at time of audit. Low risk of latent critical issues.	0 High/Critical findings and few Low/Medium severity findings.
4: High Confidence	Code is clean, well-structured, and adheres to best practices. Only 1 Significant issue was uncovered per week. Design patterns are sound, and test coverage is strong. Recommendation: Suitable for deployment after remediations; consider periodic review with changes.	0-1 High/Critical findings per engagement week and little to no Medium severity issues. Varied Low severity findings.
3: Moderate Confidence	Medium-severity and occasional High-severity issues found. Code is functional, but there are concerning areas (e.g., weak modularity, risky patterns). No critical design flaws, though some patterns could lead to issues in edge cases. Recommendation: Address issues thoroughly and consider a targeted follow-up audit depending on code changes.	1-2 High/Critical findings per engagement week.
2: Low Confidence	Code shows frequent emergence of Critical/High vulnerabilities. Audit revealed recurring anti-patterns, weak test coverage, or unclear logic. These characteristics suggest a high likelihood of latent issues. Recommendation: Post-audit development and a second audit cycle are strongly advised.	2-4 High/Critical findings per engagement week. Or additional High/Critical findings uncovered in remediation review which have not been resolved and confirmed by Guardian. round
1: Very Low Confidence	Code has systemic issues. Multiple High/Critical findings (≥ 5/week), poor security posture, and design flaws that introduce compounding risks. Safety cannot be assured. Recommendation: Halt deployment and seek a comprehensive re-audit after substantial refactoring.	≥ 5 High/Critical findings and overall systemic flaws.

Table of Contents

Project Information

Project Overview 5

Audit Scope & Methodology 6

Smart Contract Risk Assessment

Round One Findings & Resolutions 9

Round Two Findings & Resolutions 83

Round Three Findings & Resolutions 90

Addendum

Disclaimer 93

About Guardian 94f

Project Overview

Project Summary

Project Name	LayerZero
Codebase(s)	https://github.com/LayerZero-Labs/monorepo-external
Commit(s)	Final Commit: 9a1436b0dad246e67dc0f44070091288dabd255c

Audit Summary

Delivery Date	3/26/2026
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● High	0	0	0	0	0	0
● Medium	2	0	0	0	0	2
● Low	2	0	0	0	0	2
● Info	72	0	0	64	1	7

Audit Scope & Methodology

contract,source,total,comment
ERC20Plus.sol,51,123,55
OFTBurnMint.sol,23,45,20
OFTBurnSelfMint.sol,21,43,20
OFTLockUnlock.sol,9,19,8
OFTNative.sol,9,19,8
OFTBurnMintAlt.sol,23,33,8
OFTBurnSelfMintAlt.sol,21,31,8
OFTLockUnlockAlt.sol,9,19,8

Nexus.sol,265,537,204
NexusAlt.sol,16,43,23
NexusERC20.sol,75,174,74
NexusERC20Guard.sol,23,48,20
NexusOFT.sol,79,165,63
NexusOFTAlt.sol,38,73,26
NexusMsgCodec.sol,35,111,64

NexusFeeConfigModule.sol,83,172,60
NexusModule.sol,34,92,45
NexusPauseModule.sol,98,195,61
NexusRateLimiterModule.sol,56,125,52

OFTRegistryBaseUpgradeable.sol,89,189,71
OFTRegistryRBACUpgradeable.sol,21,55,27
TokenScalesBaseUpgradeable.sol,41,107,52

RateLimiterBaseUpgradeable.sol,272,590,254
RateLimiterRBACUpgradeable.sol,31,83,42
PauseByIDBaseUpgradeable.sol,55,127,57
PauseByIDRBACUpgradeable.sol,32,74,31
EnumerableSetPagination.sol,36,69,21
FeeConfigBaseUpgradeable.sol,54,125,56
FeeConfigRBACUpgradeable.sol,17,45,22
FeeAccountingBaseUpgradeable.sol,38,93,45
FeeAccountingRBACUpgradeable.sol,14,38,19
AllowlistBaseUpgradeable.sol,94,184,70
AllowlistRBACUpgradeable.sol,22,60,29
AccessControl2StepUpgradeable.sol,69,142,54
PauseBaseUpgradeable.sol,43,107,50
PauseRBACUpgradeable.sol,18,48,23

interfaces (multiple),~150,~600,~400

```
source count: {  
  total: 5290,  
  source: 2208,  
  comment: 2424,  
  commentToSourceRatio: 1.10  
}
```

scope constraints:
oapp (non-upgradeable) → diff vs LZ devtools @306ee69
oapp (upgradeable) → diff vs LZ devtools @306ee69
oft (non-upgradeable) → diff vs LZ devtools @306ee69
oft (upgradeable) → diff vs LZ devtools @306ee69

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: <i>High</i>	Impact: <i>Medium</i>	Impact: <i>Low</i>
Likelihood: <i>High</i>	● Critical	● High	● Medium
Likelihood: <i>Medium</i>	● High	● Medium	● Low
Likelihood: <i>Low</i>	● 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.

Round One Findings & Resolutions

ID	Title	Category	Severity	Status
M-01	Nexus composeFrom Is Not Original Sender	Logical Error	● Medium	Resolved
M-02	Missing Overflow Validation	Validation	● Medium	Resolved
L-01	quoteOFT Doesn't Account For Fees	Warning	● Low	Resolved
L-02	quoteOFT Misses outboundEnabled	Logical Error	● Low	Resolved
I-01	Some Tokens Cannot Bridge Back Full Supply	Warning	● Info	Acknowledged
I-02	IMintableBurnable Return Values Ignored	Warning	● Info	Acknowledged
I-03	Pausing Asymmetry Between Debit And Credit	Documentation	● Info	Acknowledged
I-04	Misleading Fee Balance View Function	Warning	● Info	Acknowledged
I-05	Allowlist Not Enforced On Approvals And Permit	Warning	● Info	Acknowledged
I-06	Approvals And Permit Usable During Global Pause	Warning	● Info	Acknowledged
I-07	Owner Must Pass Downscaled Value	Documentation	● Info	Acknowledged
I-08	Closed Rate Limit Allows Zero-Amount Messaging	Warning	● Info	Acknowledged
I-09	Withdraw Fees May Drain Balance On Override	Logical Error	● Info	Acknowledged

Round One Findings & Resolutions

ID	Title	Category	Severity	Status
I-10	Incorrect Registry Storage Location	Documentation	● Info	Resolved
I-11	Allowlist Mode Switching Preserves Inactive Set	Warning	● Info	Acknowledged
I-12	Fee Withdrawal Blocked By Allowlist Settings	Documentation	● Info	Acknowledged
I-13	Allowlist Can Block Fund Recovery	Warning	● Info	Acknowledged
I-14	Upgradeable Immutables Can Change Behavior	Warning	● Info	Acknowledged
I-15	Exempt Users Can Reduce Net Accounting	Warning	● Info	Acknowledged
I-16	Zero Amounts Cause Rate Limit Precision Loss	Rounding	● Info	Acknowledged
I-17	Rate Limits Consumed DoS	Warning	● Info	Resolved
I-18	Lacking Configuration List Duplicate Check	Validation	● Info	Acknowledged
I-19	Nexus And Portal Tokens Should Have Low Supply	Warning	● Info	Acknowledged
I-20	Unlimited Transfers With Net Accounting	Logical Error	● Info	Acknowledged
I-21	Non-exempt Users May Bypass Inbound Rate Limits	Warning	● Info	Acknowledged
I-22	De-registration Risk	Warning	● Info	Acknowledged

Round One Findings & Resolutions

ID	Title	Category	Severity	Status
I-23	Minting Bypasses Global Pause	Warning	● Info	Acknowledged
I-24	Recover Bypasses Validations	Warning	● Info	Acknowledged
I-25	Missing InvalidLocalDecimals Error	Validation	● Info	Resolved
I-26	Incorrect Max Amount Reported	Unexpected Behavior	● Info	Acknowledged
I-27	Enforced Options Cannot Be Applied Individually	Compatibility	● Info	Acknowledged
I-28	Some Tokens Incompatible	Compatibility	● Info	Acknowledged
I-29	Lacking ERC7802 Compliance	Compatibility	● Info	Acknowledged
I-30	Trapped ETH Through IzReceive	Warning	● Info	Acknowledged
I-31	Redundant Sources Of Truth	Best Practices	● Info	Resolved
I-32	quoteSend Ignores Pausability Or Rate Limits	Warning	● Info	Acknowledged
I-33	Breaking MsgInspector Interface Update	Compatibility	● Info	Acknowledged
I-34	Mints Can Occur While Token Is Paused	Documentation	● Info	Acknowledged
I-35	Payouts Spend Fees From Same Pool	Warning	● Info	Acknowledged

Round One Findings & Resolutions

ID	Title	Category	Severity	Status
I-36	Scale Decimals Are Not Exposed	Compatibility	● Info	Partially Resolved
I-37	quoteOFT Does Not Consult Guard	Unexpected Behavior	● Info	Acknowledged
I-38	Initialize Must Be Invoked During Deployment	Best Practices	● Info	Acknowledged
I-39	Nexus Doesn't Support Native Inner Tokens	Suggestion	● Info	Acknowledged
I-40	New Network Supports Allows Circumvention	Warning	● Info	Acknowledged
I-41	Lacking Pause Idempotency Check	Best Practices	● Info	Acknowledged
I-42	Event Log Manipulation With Reentrancy	Events	● Info	Acknowledged
I-43	NexusAlt Lacks Sanity Check	Validation	● Info	Resolved
I-44	Missing MinterBurnerCallFailed Implementation	Superfluous Code	● Info	Resolved
I-45	Mint/Burn Result Ignored By PortalOFTMintBurn	Compatibility	● Info	Acknowledged
I-46	Credit Can Consume An Unexpected Amount Of Gas	DoS	● Info	Acknowledged
I-47	Unsynchronized Limits Can Lead To Stuck Funds	Warning	● Info	Acknowledged
I-48	Rate Limit Bypass Through Exempt Address	Warning	● Info	Acknowledged

Round One Findings & Resolutions

ID	Title	Category	Severity	Status
I-49	Fee-on-Transfer/Rebasing Tokens Not Supported	Warning	● Info	Acknowledged
I-50	Tokens With Inconsistent Decimals Incompatible	Compatibility	● Info	Acknowledged
I-51	PortalOFTNative Stuck Message Risk	Warning	● Info	Acknowledged
I-52	Missing Events	Events	● Info	Resolved
I-53	Rate Limit Quotes Can Be Misleading	Unexpected Behavior	● Info	Acknowledged
I-54	Outside Tokens Do Not Follow The Guard	Compatibility	● Info	Acknowledged
I-55	Fee Rounds Down	Rounding	● Info	Acknowledged
I-56	Quote Misses Underlying Token Pause State	Unexpected Behavior	● Info	Acknowledged
I-57	Fees Inflated For Small Sends	Rounding	● Info	Acknowledged
I-58	Rate Limits Retroactively Affected	Unexpected Behavior	● Info	Acknowledged
I-59	Net Accounting Abused For One Sided Rate	Rounding	● Info	Acknowledged
I-60	Rate Limit Configurations Sandwiched	Frontrunning	● Info	Acknowledged
I-61	Incorrect To Address Used	Logical Error	● Info	Acknowledged

Round One Findings & Resolutions

ID	Title	Category	Severity	Status
I-62	Default Rate Limit Cannot Apply Across Tokens	Compatibility	● Info	Acknowledged
I-63	nexusQuoteOFT Misses Exemptions	Warning	● Info	Acknowledged
I-64	nexusQuoteOFT Does Not Consult The MsgInspector	Unexpected Behavior	● Info	Acknowledged

M-01 | Nexus composeFrom Is Not Original Sender

Category	Severity	Location	Status
Logical Error	● Medium	Nexus.sol	Resolved

Description

In Nexus, both compose identity fields resolve to the NexusOFT wrapper instead of the original user. Standard OFT behavior preserves the OFT contract as `_from` and the end user as `composeFrom`, but the Nexus call flow causes `OFTMsgCodec.encode()` to capture the wrapper as the compose initiator. As a result, receiving contracts cannot reliably identify or authenticate the original sender. Any downstream logic that depends on user-specific authorization or accounting sees every compose as coming from the same wrapper address.

Recommendation

Use Nexus-specific message encoding that writes the original user into `composeFrom` instead of relying on the current `msg.sender` context.

Resolution

LayerZero: Resolved.

M-02 | Missing Overflow Validation

Category	Severity	Location	Status
Validation	● Medium	DecimalUtils.sol	Resolved

Description [PoC](#)

In the `DecimalUtils` file the `_toSD` does not perform any validation to ensure that the result of the decimal conversion does not overflow a `uint64`.

```
...
function _toSD(uint256 _amountLD) internal view virtual returns (uint64 amountSD) {
return uint64(_amountLD / DECIMAL_CONVERSION_RATE);
}
...
```

As a result, large crosschain transfers for tokens with a high supply can result in the sender burning a huge amount of tokens and receiving a very small amount of tokens out, representing a complete loss of assets.

Recommendation

Add validation on the result of `_amountLD / DECIMAL_CONVERSION_RATE` to ensure that it does not overflow a `uint64` before casting it as such.

Resolution

LayerZero: Resolved.

L-01 | quoteOFT Doesn't Account For Fees

Category	Severity	Location	Status
Warning	● Low	Global	Resolved

Description

The `quoteOFT` function returns `maxAmountLD` directly from the rate limiter's available outbound capacity. However, the rate limiter enforces limits on `amountReceivedLD` (post-fee amount), while `maxAmountLD` is meant to represent the maximum `amountSentLD` (pre-fee amount) a user can submit. When fees are configured, users can actually send more than the reported maximum. For example, with a rate limit of 1000 and a 10% fee, users can send approximately 1111 tokens (resulting in 1000 received), but `quoteOFT` reports a maximum of 1000. This causes UIs and integrations relying on this value to understate available capacity.

Recommendation

Consider accounting for the fee in `quoteOFT` for the `maxAmountLD` calculation.

Resolution

LayerZero: Resolved.

L-02 | quoteOFT Misses outboundEnabled

Category	Severity	Location	Status
Logical Error	● Low	Global	Resolved

Description

The `nexusQuoteOFT` and `quoteOft` on the Portal uses the `maxAmountLD` as presented by the `getRateLimitUsages` function. However this function does not account for when the config bitmap of the rate limit indicates that the outbound rate limit is disabled. As a result the `nexusQuoteOFT` function will present inaccuracies in the `maxAmountLD` when the `outboundEnabled` config value for a rate limit is `false`.

Recommendation

Take into consideration whether the outbound pathway is enabled or not for the rate limit in the quote functions.

Resolution

LayerZero: Resolved.

I-01 | Some Tokens Cannot Bridge Back Full Supply

Category	Severity	Location	Status
Warning	● Info	Global	Acknowledged

Description

Due to the way the fee logic works, in some cases it is impossible for a networks supply to go entirely to zero while a fee is configured.

Consider the following example:

- A token uses a lockbox OFT on Network A, and a a MintBurn OFT on Network B
- Network B has an outgoing fee of 50 BIPs and a token supply of X, backed by X tokens on Network A in the lockbox
- Token on Network B has 18 decimals and 6 shared decimals
- Even if all X tokens are transferred back to Network A, a fee of $50X/10000$ will be taken and kept on Network B
- Since the dust removal logic favors counting dust removed as fees, a remaining amount that is below the conversion rate will just simply not be transferrable back
- Such a dust amount is guaranteed to always exist as long as the fee does not get truncated to zero

Recommendation

Be aware that a dust supply of a token will be trapped on a network unless the fee is removed.

Resolution

LayerZero: Acknowledged.

I-02 | IMintableBurnable Return Values Ignored

Category	Severity	Location	Status
Warning	● Info	Nexus.sol: 272, 388	Acknowledged

Description

The IMintableBurnable interface shows that both `burn()` and `mint()` return `bool success`. However, in Nexus.sol, these return values are ignored. If the underlying token implementation returns false instead of reverting, the operations will fail silently, leading to:

- Tokens not actually burned but message sent (unbacked mint on destination)
- Tokens not actually minted but message processed (user doesn't receive funds)

Recommendation

Check return values or document that tokens that revert on failure should be used.

Resolution

LayerZero: Acknowledged.

I-03 | Pausing Asymmetry Between Debit And Credit

Category	Severity	Location	Status
Documentation	● Info	Nexus.sol: 376	Acknowledged

Description

The `whenNotPaused(_nexusId)` modifier appears only on `_debit`, which is the send path. The `_credit` function used for receiving tokens has no pause check. This means that pausing stops outbound transfers but allows inbound which may be unexpected.

Recommendation

Document this asymmetry clearly.

Resolution

LayerZero: Acknowledged.

I-04 | Misleading Fee Balance View Function

Category	Severity	Location	Status
Warning	● Info	Nexus.sol: 404	Acknowledged

Description

The Nexus's fees accumulated is simply calculated as `IERC20(_token).balanceOf(address(this));` within function `getFeeBalance`. It should be clearly noted to users that if tokens are directly sent to the Nexus, they'll be counted as withdrawable fees.

Recommendation

Clearly document this behavior.

Resolution

LayerZero: Acknowledged.

I-05 | Allowlist Not Enforced On Approvals And Permit

Category	Severity	Location	Status
Warning	● Info	PortalERC20.sol	Acknowledged

Description

The allowlist checks are only applied to token movement functions (transfer/transferFrom/burn) but not to approval state changes. Blacklisted or non-whitelisted users can set or increase allowances through functions `approve/increaseAllowance/decreaseAllowance` and `permit` while disallowed by the allowlist. These allowances can be executed immediately once the owner/spender becomes allowlisted.

Recommendation

Clearly document that approval changes do not verify allowlist.

Resolution

LayerZero: Acknowledged.

I-06 | Approvals And Permit Usable During Global Pause

Category	Severity	Location	Status
Warning	● Info	PortalERC20.sol	Acknowledged

Description

Functions such as `approve` and `permit` are not gated by `whenNotGloballyPaused`. While transfers are paused, actors can set allowances and signatures that can be executed immediately after unpause, which should be documented.

Recommendation

Clearly document this behavior.

Resolution

LayerZero: Acknowledged.

I-07 | Owner Must Pass Downscaled Value

Category	Severity	Location	Status
Documentation	● Info	RateLimiterUpgradeable.sol	Acknowledged

Description

Rate-limit configs/states expect downscaled units when SCALE_DECIMALS > 0, but only `getRateLimitUsages` upscales for display, so an admin passing base decimal limits will silently set much larger caps and lead to a misconfigured limit.

Recommendation

Add explicit documentation on `IRateLimiterCore` setters and `RateLimiterUpgradeable` admin functions stating inputs are downscaled units.

Resolution

LayerZero: Acknowledged.

I-08 | Closed Rate Limit Allows Zero-Amount Messaging

Category	Severity	Location	Status
Warning	● Info	RateLimiterCoreUpgradeable.sol	Acknowledged

Description

Setting the rate limit to closed (limit = 0) blocks non-zero transfers, but zero-amount sends still pass the rate-limit checks. This allows messages (including optional compose payloads) to be emitted even when operators expect the rate limiter to fully halt activity. If downstream logic assumes that "closed" means no cross-chain messages, this creates an unexpected bypass for message-only actions.

Recommendation

If the intended behavior is to block all messaging when closed, add an explicit amount > 0 check in the send path. Otherwise, document that "closed" only blocks value transfer and that pause must be used to halt messaging.

Resolution

LayerZero: Acknowledged.

I-09 | Withdraw Fees May Drain Balance On Override

Category	Severity	Location	Status
Logical Error	● Info	FeeCoreUpgradeable.sol: 166	Acknowledged

Description

FeeCoreUpgradeable comments suggest overrides may track fees via `balanceOf(this)` instead of storage. That pattern is safe for mint/burn OFTs, but not for lock/unlock OFTs where the contract holds both user deposits and accrued fees.

If an override followed the documentation literally in a lock/unlock setup, `withdrawFees()` could treat the full contract balance as fees and drain user funds along with the real fee balance. Current implementations are safe, but the guidance is too broad.

Recommendation

Clarify in the `getFeeBalance()` and `_resetFeeBalance()` documentation that balance-based fee tracking is only safe when contract balances do not also include user-held liquidity.

Resolution

LayerZero: Acknowledged.

I-10 | Incorrect Registry Storage Location

Category	Severity	Location	Status
Documentation	● Info	OFTRegistryCoreUpgradeable.sol	Resolved

Description

In the OFTRegistryCoreUpgradeable contract the OFT_REGISTRY_CORE_STORAGE_LOCATION is stored as 0x5e17fe86d24c27c5744e7e62d5c0e4b546c352da7d7ccf57d4d9d4da22a06900 with a comment indicating it is computed as keccak256(abi.encode(uint256(keccak256("layerzerov2.storage.ofregistrycore")) - 1)) & ~bytes32(uint256(0xff)). However keccak256(abi.encode(uint256(keccak256("layerzerov2.storage.ofregistrycore")) - 1)) & ~bytes32(uint256(0xff)) evaluates to 0x3061291cb11960a6131e0dc3984d775ee9ea1d64c1334ba80fc584e9cc0bf400 which does not match the storage location used.

Recommendation

Correct the comment or the storage location bytes.

Resolution

LayerZero: Resolved.

I-11 | Allowlist Mode Switching Preserves Inactive Set

Category	Severity	Location	Status
Warning	● Info	AllowlistCore.sol	Acknowledged

Description

When switching between Blacklist and Whitelist modes via `setAllowlistMode()`, the contract only updates the active mode but does not clear the inactive set. Both `blacklistedSet` and `whitelistedSet` persist in storage regardless of the current mode. If the owner toggles modes and later switch back, previously configured entries immediately become active again. For example, blacklisting users, switching to Whitelist mode, then returning to Blacklist mode will re-activate all previous blacklist entries without additional transactions.

Recommendation

Clearly document this behavior.

Resolution

LayerZero: Acknowledged.

I-12 | Fee Withdrawal Blocked By Allowlist Settings

Category	Severity	Location	Status
Documentation	● Info	Global	Acknowledged

Description

In whitelist or blacklist modes, `PortalERC20.transfer` and `NexusERC20.transfer` requires the caller and recipient to be allowlisted. Nexus uses transfer when withdrawing fee balances to a recipient. If `address(Nexus)` or the recipient is not allowlisted (or is blacklisted), `withdrawFees` reverts, leaving fee balances stuck in the Nexus contract.

Recommendation

Clearly document that the Nexus must be allowlisted as well as the recipient.

Resolution

LayerZero: Acknowledged.

I-13 | Allowlist Can Block Fund Recovery

Category	Severity	Location	Status
Warning	● Info	Global	Acknowledged

Description

PortalERC20.recoverFunds and NexusERC20.recoverFunds explicitly revert when the `_from` address is allowlisted. Fee withdrawal from Nexus uses token transfers, which require `address(Nexus)` to be allowlisted in whitelist mode. As a result, if the protocol allowlists Nexus to enable fee withdrawals, fund recovery from the Nexus address is always blocked by design.

Recommendation

Clearly document this behavior

Resolution

LayerZero: Acknowledged.

I-14 | Upgradeable Immutables Can Change Behavior

Category	Severity	Location	Status
Warning	● Info	Global	Acknowledged

Description

Several upgradeable contracts rely on immutable values (e.g., SCALE_DECIMALS, decimal conversion rates). Upgrading to an implementation compiled with different immutables silently changes behavior without storage migration, potentially altering rate-limit math and token decimal handling.

Recommendation

Document this and ensure deployments are done without changing immutables.

Resolution

LayerZero: Acknowledged.

I-15 | Exempt Users Can Reduce Net Accounting

Category	Severity	Location	Status
Warning	● Info	RateLimiterCoreUpgradeable.sol	Acknowledged

Description

In the `_applyRateLimit` function users who are marked as exempt can avoid the rate limits, as expected.

However, when a rate limit has net accounting enabled, an exempt user can abuse this. Consider the following scenario:

- Chain A is the current chain
- Chain A has the inbound and outbound rate limits enabled
- Chain A has net accounting enabled
- Chain A currently has a 5e18 usage on the outbound rate limit, and the maximum is 10e18
- An exempt user bridges out 5e18 tokens and avoids the rate limit
- The exempt user then transfers these tokens to a different address under their control
- The exempt user then bridges them back to Chain A
- Since Chain A uses net accounting, the Chain’s outbound rate limit usage is decreased to 0.

Recommendation

Be aware of this ability from exempt addresses, and consider this carefully for setups that use both exempt addresses and net accounting. All exempt addresses must be trusted entirely.

Resolution

LayerZero: Acknowledged.

I-16 | Zero Amounts Cause Rate Limit Precision Loss

Category	Severity	Location	Status
Rounding	● Info	RateLimiterCoreUpgradeable.sol	Acknowledged

Description

Zero value sends are allowed, however they do update the rate limit decay which uses round down division.

If a smaller limit is used relative to the length of the window for a rate limit, some precision loss can be forced through zero amount transfers maliciously occurring at a low interval to keep the rate limits down marginally.

Recommendation

Be aware of this rounding and consider disallowing zero value OFT send.

Resolution

LayerZero: Acknowledged.

I-17 | Rate Limits Consumed DoS

Category	Severity	Location	Status
Warning	● Info	Global	Resolved

Description

It is possible for anyone to purposely consume the rate limits when net accounting is not enabled by simply transferring out and then back into the src chain.

Recommendation

Be aware of and document this loophole so that those who would like to avoid it can use net accounting.

Resolution

LayerZero: Resolved.

I-18 | Lacking Configuration List Duplicate Check

Category	Severity	Location	Status
Validation	● Info	RateLimiterCoreUpgradeable.sol	Acknowledged

Description

In the `RateLimiterCoreUpgradeable` contract `_setRateLimitConfigs` and `_setRateLimitStates` configuration functions there is no validation that the `param.id` has not already been configured in the list before.

If a parameter with the same id were to be provided it could lead to an unexpected configuration overwrite.

Recommendation

Be aware of this lacking validation and consider adding it to validate against any potential mishaps during configuration. Otherwise be sure to review configurations to ensure that only unique param ids are being configured.

Resolution

LayerZero: Acknowledged.

I-19 | Nexus And Portal Tokens Should Have Low Supply

Category	Severity	Location	Status
Warning	● Info	Global	Acknowledged

Description

The Nexus and Portal implementations use a value of 0 for the `_scaleDecimals` hardcoded in their constructor.

As a result, to avoid any unexpected conditions these systems should avoid using tokens what may have a total supply greater than ~79 Billion tokens using 18 decimals.

Recommendation

Be aware of this limitation and consider it when using tokens with these systems.

Resolution

LayerZero: Acknowledged.

I-20 | Unlimited Transfers With Net Accounting

Category	Severity	Location	Status
Logical Error	● Info	RateLimiterCoreUpgradeable.sol: 326	Acknowledged

Description

When one transfer direction is disabled but `netAccountingEnabled` remains on, users can send unlimited volume through the disabled direction and still reduce usage in the opposite direction. That lets them repeatedly restore capacity on the enforced side and bypass the intended rate limit. For example, if outbound is disabled, inbound is limited, and net accounting is enabled, a user can send tokens outbound without restriction, reduce inbound usage, then immediately consume the restored inbound capacity. Repeating the cycle turns a configured limit into effectively unlimited throughput.

Recommendation

Document this behavior clearly so operators understand that disabling one direction can still affect the opposite direction when net accounting is enabled.

Resolution

LayerZero: Acknowledged.

I-21 | Non-exempt Users May Bypass Inbound Rate Limits

Category	Severity	Location	Status
Warning	● Info	RateLimiterCoreUpgradeable.sol	Acknowledged

Description

In the `_applyRateLimit` function the `_user` address provided is used to check whether the user is an exempt address for the rate limit check. On the sending side, this user is the address that initiated the action and sent the tokens.
However on the receiving side, this `_user` address represents the receiving side, and anyone can initiate transfers to this address.

Recommendation

Consider if this is the desired way to check whether a crosschain transfer should be exempt from the rate limit, or if there should be some authentication also based on the authorization of the sender address from the sending chain. Though this introduces some complexity when receiving from a non-EVM compatible network.
Generally, be aware of this ability for anyone to transfer to the addresses and contracts that are considered exempt from the inbound rate limits, and document it accordingly.

Resolution

LayerZero: Acknowledged.

I-22 | De-registration Risk

Category	Severity	Location	Status
Warning	● Info	Global	Acknowledged

Description

If a token is de-registered while there are in-flight messages, then those messages will get stuck in the channel.

If the token is re-registered or another token is registered using the same tokenId, then those stuck messages will be executable again unexpectedly.

Recommendation

To avoid any unexpected behavior, consider blacklisting a tokenId after it is de-registered and requiring that a new one is used. Furthermore, handle token registration across all chains with extreme care, as it must be done in a timely manner and with exact accuracy every time.

Resolution

LayerZero: Acknowledged.

I-23 | Minting Bypasses Global Pause

Category	Severity	Location	Status
Warning	● Info	Global	Acknowledged

Description

The `mint` function in both `PortalERC20` and `NexusERC20` lacks the `whenNotGloballyPaused` modifier, allowing addresses with the `MINTER_ROLE` to mint tokens even when the system is globally paused. While `transfer`, `transferFrom`, and `burn` are all gated by pause checks, minting remains unrestricted. Consequently token supply can increase during emergency pause conditions when all other token movement is frozen.

Recommendation

Clearly document this behavior.

Resolution

LayerZero: Acknowledged.

I-24 | Recover Bypasses Validations

Category	Severity	Location	Status
Warning	● Info	Global	Acknowledged

Description

The `recoverFunds` function in PortalERC20 and NexusERC20 validates that the source address is not allowlisted but performs no validation on the destination address. By using `super._transfer()` instead of the standard transfer function, it bypasses both pause checks and allowlist enforcement on the recipient. This allows a `FUND_RECOVERY_ROLE` holder to transfer recovered tokens to any address, including those that are blacklisted.

Recommendation

Clearly document this behavior.

Resolution

LayerZero: Acknowledged.

I-25 | Missing InvalidLocalDecimals Error

Category	Severity	Location	Status
Validation	● Info	DecimalUtils.sol	Resolved

Description

In the `DecimalUtils` utility contract there is no logic to explicitly validate the local decimals are greater than or equal to the shared decimals.

As a result, in the event that the shared decimals is specified to be larger than the local decimals the deployment will revert with a non-descriptive panic revert.

Recommendation

Consider implementing an explicit validation that the shares decimals is not greater than the local decimals and reverting with a `InvalidLocalDecimals` error similarly to the canonical OFT implementation.

Resolution

LayerZero: Resolved.

I-26 | Incorrect Max Amount Reported

Category	Severity	Location	Status
Unexpected Behavior	● Info	Nexus.sol	Acknowledged

Description

When the rate limiter is globally disabled, `getRateLimitUsages()` reports `type(uint256).max` as the maximum transferable amount. That is misleading because Nexus amounts are still bounded by OFT shared-decimal constraints, so values anywhere near `uint256.max` are not actually transferable. The canonical OFT implementation reports `type(uint64).max`, and the most accurate bound here is `type(uint64).max * DECIMAL_CONVERSION_RATE`. Returning `uint256.max` therefore overstates what callers can really send.

Recommendation

Match the canonical OFT behavior by reporting `type(uint64).max`, or return the stricter bound of `type(uint64).max * DECIMAL_CONVERSION_RATE`.

Resolution

LayerZero: Acknowledged.

I-27 | Enforced Options Cannot Be Applied Individually

Category	Severity	Location	Status
Compatibility	● Info	Nexus.sol	Acknowledged

Description

A Nexus instance may support multiple NexusOFTs with different underlying token implementations. One underlying token implementation may have special logic or callback hooks on the destination chain and thus may require more gas than other underlying tokens that are supported by the Nexus. However the combineOptions function does not allow options to be enforced per nexus id, only per Eid. This leaves the Nexus with rigid option enforcement that cannot fully adapt to the various tokens that may be a part of the Nexus.

Recommendation

Consider implementing a version of combineOptions that allows enforcedOptions to be specified on a nexus id level instead of an Eid level.

Resolution

LayerZero: Acknowledged.

I-28 | Some Tokens Incompatible

Category	Severity	Location	Status
Compatibility	● Info	Global	Acknowledged

Description

The Nexus contract uses an IMintableBurnable interface that expects a boolean return value from the mint and burn functions, however not all tokens who expose these functions return a boolean. For example, the GMX token on Arbitrum implements a MintableBaseToken where the mint and burn functions do not return any return data. These tokens are incompatible out of the box with the system as the invocations of mint and burn will revert due to unexpected lack of returndata. These tokens would need adapters to be correctly supported.

Recommendation

Consider removing the return values from the IMintableBurnable interface as they are not currently used and instead limit the compatibility of the system.

Resolution

LayerZero: Acknowledged.

I-29 | Lacking ERC7802 Compliance

Category	Severity	Location	Status
Compatibility	● Info	Global	Acknowledged

Description

The Nexus system implements minting and burning calls on a `minterBurner` contract meant for crosschain mint/burn actions. There is an EIP7802 which is intended to support exactly this, and is used by the most popular cross-chain stablecoin on OFT rails, USDT0, however the Nexus system does not adhere to this EIP and instead defines it's own `IMintableBurnable` interface.

Recommendation

Consider adopting the EIP7802 interface for the Nexus system.

Resolution

LayerZero: Acknowledged.

I-30 | Trapped ETH Through IzReceive

Category	Severity	Location	Status
Warning	● Info	Nexus.sol	Acknowledged

Description

The IzReceive function does not implement any validation that ensures that the `msg.value` provided is zero and there are no ETH rescue functions, therefore any message that is executed with nonzero `msg.value` traps that ETH in the Nexus contract.

Recommendation

Consider validating that the `msg.value` is zero in `_IzRecieve` or adding a rescue native function.

Resolution

LayerZero: Acknowledged.

I-31 | Redundant Sources Of Truth

Category	Severity	Location	Status
Best Practices	● Info	NexusOFT.sol	Resolved

Description

The NexusOFT contract stores a `TOKEN_ID` which is immutable and stored at construction time, however the Nexus stores each OFT by a `tokenId` that is provided by the owner who registers it, with no validation that this id matches the one returned from the `NexustOFT.tokenId` function. This leaves room for incorrect configurations and creates parallel tracking for the same information in the `Nexus` and `NexusOFT` contracts.

Recommendation

Consider either having the `NexusOFT` contract fetch the `NEXUS.getTokenId(address(this))` result as the implementation for the `tokenId` function (less gas efficient, but deduplicates state), or validating that the reported `tokenId()` on the `NexusOFT` instance matches the `tokenId` it's being configured for in the `registerToken` function.

Resolution

LayerZero: Resolved.

I-32 | quoteSend Ignores Pausability Or Rate Limits

Category	Severity	Location	Status
Warning	● Info	Global	Acknowledged

Description

quoteSend returns a valid MessagingFee without checking pause state or rate limits. It calls _debitView which performs pure calculations. However, the actual send calls _debit which has whenNotPaused modifier and enforces rate limits with _outflow. Users can receive valid-looking quotes for sends that will revert.

Recommendation

Clearly document this behavior to integrators.

Resolution

LayerZero: Acknowledged.

I-33 | Breaking MsgInspector Interface Update

Category	Severity	Location	Status
Compatibility	● Info	Global	Acknowledged

Description

The old IOAppMsgInspector interface defined an inspect function with the signature inspect(bytes calldata _message, bytes calldata _options), however the new IMsgInspector interface defined for Nexus and Portal exposes only a inspect(address _sender, bytes calldata _message, bytes calldata _options) signature.

As a result old message inspector implementations will not be compatible with the Nexus and Portal.

Recommendation

Be aware of this breaking change and clearly document it for those building on Nexus and Portal.

Resolution

LayerZero: Acknowledged.

I-34 | Mints Can Occur While Token Is Paused

Category	Severity	Location	Status
Documentation	● Info	NexusERC20.sol	Acknowledged

Description

The mint function states that It does not revert if the recipient is not allowlisted, as funds cannot be debited in that state, however as a byproduct of ignoring the checkTransfer validation the mint function also ignores the pause status of the underlying token.

This could be intentional to not hold up IzReceive actions and allow them to be executed, however it is important to note that mints can still occur, through receiving crosschain messages or other arbitrary mints by the MINTER_ROLE holders.

Recommendation

Be aware of this gap in pause validation and consider if it is expected for all cases in which mint can be called. Consider if the isPaused function on the Pause contract should be used to particularly check if the pause status should be consulted before allowing certain mint actions.

Resolution

LayerZero: Acknowledged.

I-35 | Payouts Spend Fees From Same Pool

Category	Severity	Location	Status
Warning	● Info	PortalOFTNative.sol, PortalOFTLockUnlock.sol	Acknowledged

Description

PortalOFTNative and PortalOFTLockUnlock accrue fees in feeBalances, but recipient credits are paid from the same native token pool. Because fees are not reserved separately, bridge payouts can consume balance that accounting still treats as withdrawable fees.

In unsupported configurations where remote supply is not fully backed by locked assets, tokens can be bridged back and drain the pool below the tracked fee balance. withdrawFees() may then revert even though fees were previously accrued.

Recommendation

Use only configurations where remote token supply remains properly backed and cannot dilute the balances relied on to pay out accrued fees.

Resolution

LayerZero: Acknowledged.

I-36 | Scale Decimals Are Not Exposed

Category	Severity	Location	Status
Compatibility	● Info	Global	Partially Resolved

Description

In both the Nexus and Portal implementations that use the `RateLimiterUpgradeable`, the `scaleDecimals` are unused and not able to be configured in the constructor. Since `SCALE_DECIMALS` is an immutable value it's not possible to configure a nonzero scale decimals for any token using Portal or Nexus out of the box.

Rate limit amounts are applied on a local decimals basis, so any tokens which may reasonably have over `type(uint96).max` (notably tokens with a totalSupply over 79 Billion tokens and 18 decimals) cannot employ the necessary scaling out of the box.

Recommendation

Consider exposing the `scaleDecimals` through the constructor to support such tokens out of the box.

Resolution

LayerZero: Partially Resolved.

I-37 | quoteOFT Does Not Consult Guard

Category	Severity	Location	Status
Unexpected Behavior	● Info	Nexus.sol	Acknowledged

Description

The quoteOFT function in Nexus does not consult the token guard to check if the caller is allowlisted for the token in the first place.

Recommendation

Consider if quoteOFT should take into account the status of the caller in the guard.

Resolution

LayerZero: Acknowledged.

I-38 | Initialize Must Be Invoked During Deployment

Category	Severity	Location	Status
Best Practices	● Info	Global	Acknowledged

Description

All upgradeable contracts in this repo rely on initialize(...) for ownership and critical configuration. If a proxy instance is deployed without calling initialize in the same transaction, any external account can front-run and call initialize first, seizing ownership/admin roles and permanently compromising the instance.

Recommendation

Be sure to initialize every deployment in the same transaction to avoid any front-running initialization risks.

Resolution

LayerZero: Acknowledged.

I-39 | Nexus Doesn't Support Native Inner Tokens

Category	Severity	Location	Status
Suggestion	● Info	Nexus.sol	Acknowledged

Description

Portal contracts can support native tokens as the underlying token for the OFT, however Nexus contracts do not expose a variant to support any native tokens as they only contain the min/burn variant.

Recommendation

Consider if this is intended and whether Nexus should be able to support a native token (ERC20 form or canonical native) as an inner token similarly to how the `PortalOFTLockUnlock` and `PortalOFTNative` versions of the Portal OFT can.

Resolution

LayerZero: Acknowledged.

I-40 | New Network Supports Allows Circumvention

Category	Severity	Location	Status
Warning	● Info	Global	Acknowledged

Description

In Nexus the fees, rate limits, and pause state are based upon the nexusId which depends on the destination EID in addition to the tokenId.

As a result, when a new peer is configured for the Nexus for a new Eid there is an opportunity to bypass all of the existing network exit pathway fees, pause state, and rate limits by sending to this new destination EID before it's fee, pause state, and rate limits are configured for each of the tokenIds.

Recommendation

Be sure to configure the pause state, fees, and rate limits for every token as necessary on a Nexus before configuring a new EID for the Nexus.

Resolution

LayerZero: Acknowledged.

I-41 | Lacking Pause Idempotency Check

Category	Severity	Location	Status
Best Practices	● Info	PauseCore.sol, PauseCoreUpgradeable.sol	Acknowledged

Description

In the `PauseCore` and `PauseCoreUpgradeable` contracts, the `_setDefaultPaused` function includes an idempotency check to validate that the configuration being made actually changes state, however no such validation exists for the configurations that are made in the `_setPaused` function for the individual pause configurations.

Recommendation

Consider introducing an idempotency check in the `_setPaused` function to match the validation performed in the `_setDefaultPaused` function.

Resolution

LayerZero: Acknowledged.

I-42 | Event Log Manipulation With Reentrancy

Category	Severity	Location	Status
Events	● Info	PortalOFTNative.sol	Acknowledged

Description

PortalOFTNative._lzReceive() calls _credit() before emitting OFTReceived. Because _credit() can transfer native value to a recipient that re-enters, another lzReceive or send can execute before the original receive path emits its completion events.

This does not appear to break core accounting, but it can cause event consumers and indexers to observe OFTReceived and PacketDelivered in an order that does not match the order in which operations actually completed.

Recommendation

Document this ordering caveat so integrators consuming OFTReceived and PacketDelivered do not assume strict completion order from event sequence alone.

Resolution

LayerZero: Acknowledged.

I-43 | NexusAlt Lacks Sanity Check

Category	Severity	Location	Status
Validation	● Info	NexusAlt.sol	Resolved

Description

The other Alt contracts verify that this is indeed a network which uses an ERC20 representation of the native token by validating that the `nativeToken` result from the endpoint is nonzero. However the `NexusAlt` contract, which does not need this token directly but is still intended only for these networks, does not implement such a validation.

Recommendation

Consider implementing sanity validation in the `NexusAlt` constructor that the `ENDPOINT.nativeToken()` result is nonzero to verify that this is indeed an Alt network.

Resolution

LayerZero: Resolved.

I-44 | Missing MinterBurnerCallFailed Implementation

Category	Severity	Location	Status
Superfluous Code	● Info	PortalOFTMintBurn.sol	Resolved

Description

The PortalOFTMintBurn contract defines a MinterBurnerCallFailed error and stipulates that it is Thrown when a low-level call to MINTER_BURNER fails.

However this error is not thrown when a low-level call to the MINTER_BURNER fails and is not used at all.

Recommendation

Consider removing the MinterBurnerCallFailed error.

Resolution

LayerZero: Resolved.

I-45 | Mint/Burn Result Ignored By PortalOFTMintBurn

Category	Severity	Location	Status
Compatibility	● Info	PortalOFTMintBurn.sol	Acknowledged

Description

The PortalOFTMintBurn implementation intends to support arbitrary mint and burn functions so long as they accept an address followed by a uint256 as parameters. However the PortalOFTMintBurn contract is not compatible with mint/burn functions that fit this signature and return false on failure rather than reverting. This is because the _callMinterBurner function simply delegates to the OpenZeppelin functionCall function which verifies the success of the call but does not assume or assert anything about the returndata contents.

Recommendation

Either explicitly document this or support asserting a returned boolean when there is nonzero returndata present.

Resolution

LayerZero: Acknowledged.

I-46 | Credit Can Consume An Unexpected Amount Of Gas

Category	Severity	Location	Status
DoS	● Info	PortalOFTNative.sol: 121	Acknowledged

Description

Because the PortalOFTNative automatically loads in the returnData into memory during _credit, this function is susceptible to gas griefing or an OOG when a large amount of bytes are returned: (bool success, bytes memory data) = payable(_to).call{ value: _amountLD }("");

The arbitrary call is also forwarded the maximum available 63/64 amount of gas in the current transaction. Which gives the to address the ability to expend more gas than desired.

Recommendation

Consider implementing a low level assembly call and specifying 0 as the length of return data to copy into memory to avoid any returndata bombs.

Furthermore consider capping the gasLimit that is forwarded to the to address to avoid any unexpected expenditures.

Resolution

LayerZero: Acknowledged.

I-47 | Unsynchronized Limits Can Lead To Stuck Funds

Category	Severity	Location	Status
Warning	● Info	Global	Acknowledged

Description

The configured rate limits can be different between chains where the Oapp contracts are deployed. If the source chain allows sending more than the destination allows for receiving, transactions with an amount between these two limits can be sent but cannot be received on the destination chain because the corresponding transactions to `IzReceive()` will revert.

Recommendation

Clearly document this risk to users.

Resolution

LayerZero: Acknowledged.

I-48 | Rate Limit Bypass Through Exempt Address

Category	Severity	Location	Status
Warning	● Info	RateLimiterCoreUpgradeable.sol: 302	Acknowledged

Description

The address-exemption feature checks `msg.sender`, not the beneficial token owner. If an exempt router, aggregator, or similar address is allowed to call `send()`, non-exempt users can route through it and bypass rate limits for that destination.

The same issue can extend to exempt EOAs using mechanisms like EIP-7702. Once an exempt caller executes the transfer path, all users behind that caller inherit the exemption in practice.

Recommendation

Treat exempt addresses as high trust, monitor how they are used, and remove exemptions if they allow third parties to bypass intended rate-limit controls.

Resolution

LayerZero: Acknowledged.

I-49 | Fee-on-Transfer/Rebasing Tokens Not Supported

Category	Severity	Location	Status
Warning	● Info	Global	Acknowledged

Description

On credit, the contract transfers `_amountLD` to the recipient without accounting for tokens that charge a transfer fee/deflate on transfer. The recipient will receive less than `_amountLD` while the protocol still considers the message fully delivered, creating an unaccounted shortfall between the escrowed amount and the actual tokens received.

Recommendation

Disallow fee-on-transfer and rebasing tokens.

Resolution

LayerZero: Acknowledged.

I-50 | Tokens With Inconsistent Decimals Incompatible

Category	Severity	Location	Status
Compatibility	● Info	Global	Acknowledged

Description

Some tokens have different decimals depending on the network being used such as USDT which has 6 decimals on most chains but 18 decimals on BNB. These tokens are incompatible with the Nexus because there is a requirement that all tokens in the nexus have the same local decimals on all networks the Nexus is deployed on.

Recommendation

Be aware of this constraint when allowing tokens to be a part of a Nexus. Alternatively, consider supporting solutions for differences in local decimals across tokens on various networks since they may not always be aligned on every network.

Resolution

LayerZero: Acknowledged.

I-51 | PortalOFTNative Stuck Message Risk

Category	Severity	Location	Status
Warning	● Info	PortalOFTNative.sol	Acknowledged

Description

The PortalOFTNative has a risk of crosschain messages being stuck if they were to target a to address which is a contract that does not accept ether.
In this case the message would be stuck in the message channel and there would be no way to rescue that Ether.

Recommendation

Document this risk and consider performing some initial offchain validations in the UI to protect users from such a risk.

Resolution

LayerZero: Acknowledged.

I-52 | Missing Events

Category	Severity	Location	Status
Events	● Info	Global	Resolved

Description

Events are missing in the following functions that could benefit from them:

- RateLimiterCoreUpgradeable._setRateLimitGlobalConfig
- FeeCoreUpgradeable._accrueFee

Recommendation

Consider implementing the appropriate events for these cases.

Resolution

LayerZero: Resolved.

I-53 | Rate Limit Quotes Can Be Misleading

Category	Severity	Location	Status
Unexpected Behavior	● Info	Global	Acknowledged

Description

The quote functions only focus on the rate limits for the sending chain and cannot include any information about the rate limit on the receiving chain, which may be misleading for large transfers. There may be a scenario where a transfer is wholly larger than the limit on a receiving chain and can in fact never fit within the limit, in which case the crosschain message would be stuck until the rate limit is raised just for this message.

Recommendation

Document this gap in the current quoting functions.

Resolution

LayerZero: Acknowledged.

I-54 | Outside Tokens Do Not Follow The Guard

Category	Severity	Location	Status
Compatibility	● Info	Global	Acknowledged

Description

In the Nexus setup when an outside token is used which does not consult the NexusERC20Guard before actions, there can be a divergence between the allowlist/blacklist status of the normal NexusERC20 tokens and this outside token.

For example, if a Nexus were to be created where one inner token was USDT while the rest were all NexusERC20 tokens, User A could be blacklisted by the NexusERC20 tokens in the guard but USDT may not have this address blacklisted. Because of this, User A may still interact with the Nexus contract through USDT usage.

Recommendation

Consider if this setup where an outside token is valid, or if only NexusERC20 tokens can be used with a Nexus hub and document this accordingly.

Resolution

LayerZero: Acknowledged.

I-55 | Fee Rounds Down

Category	Severity	Location	Status
Rounding	● Info	FeeCore.sol	Acknowledged

Description

In the getFee function, the resulting fee is rounded down in favor of the user and against the favor of the protocol.

This is typically against best practices of rounding in protocols, though the material affect is minimal in practice especially due to the fact that the subsequent de-dusting is counted as a fee.

Recommendation

Confirm if it is acceptable to LayerZero to round the fee amount down in favor of the user in this non-material way.

Resolution

LayerZero: Acknowledged.

I-56 | Quote Misses Underlying Token Pause State

Category	Severity	Location	Status
Unexpected Behavior	● Info	Nexus	Acknowledged

Description

The quoteOFT function on the NexusOFT relies entirely on the nexusQuoteOFT function from the Nexus contract. The nexusQuoteOFT function accounts for the if a particular pathway corresponding to the nexusId or if the Nexus as a whole is paused when computing the oftLimit. However the nexusQuoteOFT function does not account for if the underlying NexusERC20 token is paused. In the NexusERC20Guard contract the checkTransfer function asserts an _assertNotPaused validation for the token, however this is not reflected in the quote result.

Recommendation

Consider if the pause status of the underlying token should be taken into account when quoting and providing the oftLimit similarly to the individual pathway pauses that may occur in the Nexus contract.

Resolution

LayerZero: Acknowledged.

I-57 | Fees Inflated For Small Sends

Category	Severity	Location	Status
Rounding	● Info	Nexus.sol	Acknowledged

Description

`_debitView()` applies `_removeDust()` after subtracting fees and does not refund value rounded away by dust removal. For very small sends, that can make the effective fee far larger than the configured fee rate.

For example, after fee subtraction a send may round down to zero received tokens, effectively charging a 100% fee, or lose half its value when the remaining amount is rounded down to the nearest shared-decimal unit. The issue is most noticeable on low-value transfers.

Recommendation

If this behavior is undesirable, distinguish true fee from dust removed for rounding and refund the de-dusted amount instead of treating it as part of the sent value.

Resolution

LayerZero: Acknowledged.

I-58 | Rate Limits Retroactively Affected

Category	Severity	Location	Status
Unexpected Behavior	● Info	RateLimiterCoreUpgradeable.sol	Acknowledged

Description

In the `_setRateLimitConfigs` function the protocol is careful to checkpoint the rate limit states before updating the details of the decay slope. However this checkpointing is only done for rate limits when their specific config is being updated. Commonly, configs will actually rely on the default config, which uses id 0, for the window and limit of their rate limit. If the default config is updated then all of the rate limit states for those rate limits relying on it will be affected in a stepwise manner retroactively.

Recommendation

Be aware of this behavior when updating the global rate limit using id 0. Consider manually processing all of the rate limits that rely on it before the global rate limit when passing the list of `SetRateLimitConfigParam`'s.

Resolution

LayerZero: Acknowledged.

I-59 | Net Accounting Abused For One Sided Rate

Category	Severity	Location	Status
Rounding	● Info	RateLimiterCoreUpgradeable.sol	Acknowledged

Description

When `forwardEnabled` is false, `backwardEnabled` is true, and `netAccountingEnabled` is true, the system allows back-and-forth transfers to net usage. Because amounts are rounded up even when reducing usage, an actor can lower current usage more than intended and regain additional capacity. By splitting return transfers into smaller chunks, the actor can repeatedly round reductions in their favor and end up with less recorded usage than before the cycle. That weakens one-sided rate limits beyond the intended netting behavior.

Recommendation

Round up when increasing `forwardUsage`, but round down when decreasing the opposite-direction usage through net accounting.

Resolution

LayerZero: Acknowledged.

I-60 | Rate Limit Configurations Sandwiched

Category	Severity	Location	Status
Frontrunning	● Info	RateLimiterCoreUpgradeable.sol	Acknowledged

Description

Rate-limit configuration changes take effect immediately. When `useGlobalStateFlag` is enabled, per-path limits are replaced in one transaction by the shared global limiter, and `_setRateLimitStates` can also set exact usage values directly.

That creates a sandwich opportunity around the admin transaction: an actor can consume allowance before the update, then back-run after the new configuration lands and consume additional capacity that operators may not have intended to expose within the same block window.

Recommendation

Make sensitive rate-limit changes through a private mempool or another protected execution path, and account for sandwich exposure when changing limiter configuration.

Resolution

LayerZero: Acknowledged.

I-61 | Incorrect To Address Used

Category	Severity	Location	Status
Logical Error	● Info	NexusOFT.sol	Acknowledged

Description

In the Nexus contract, when a credit is made to the zero address the address is re-assigned to the dead address in the `_credit` function to avoid reverts with some underlying ERC20 implementations. However the `NexusOFT.nexusReceive` function receives the original `to` address as the `_to` parameter, which has not been remapped to the dead address. In this case the `OFTReceived` event and the `sendCompose` call will use the incorrect receiver address, as it will present as `address(0)` but actually be the dead address.

Recommendation

Re-map `address(0)` to the dead address in the `nexusReceive` function of the `NexusOFT` contract.

Resolution

LayerZero: Acknowledged.

I-62 | Default Rate Limit Cannot Apply Across Tokens

Category	Severity	Location	Status
Compatibility	● Info	RateLimiterCoreUpgradeable.sol	Acknowledged

Description

The RateLimiterCoreUpgradeable contract includes a default rate limit that is intended to be shared across tokens supported by the Nexus.

The Nexus does require that these tokens all use the same decimals and shared decimals, however it does not make sense for this default rate limit to apply across assets that have different prices. For example an asset with a price of \$100,000 per 1e18 tokens cannot abide by the same rate limit as an asset with a price of \$1 per 1e18 tokens.

Recommendation

Consider optionally allowing a default rate limit to be defined by a USD value of tokens being moved via an arbitrary supported oracle pricing solution.

Otherwise, be aware of this deficiency in the ability to enforce a default rate limit in a valuable way, and clearly document this for those who use Nexus to rate limit multiple tokens.

Resolution

LayerZero: Acknowledged.

I-63 | nexusQuoteOFT Misses Exemptions

Category	Severity	Location	Status
Warning	● Info	Nexus.sol	Acknowledged

Description

The `nexusQuoteOFT` function uses the `maxAmountLD` as presented by the `getRateLimitUsages` function. However this function does not account for when the sender who would initiate an OFT send is exempt when presenting the available amount.

The `nexusQuoteOFT` function provides no extra logic to raise the available amount to the `type(uint256).max` if the ultimate caller of the `quoteOft` function on the `NexusOFT` contract is exempt, and therefore portrays an inaccurate quote result for that caller.

Recommendation

Consider taking into consideration whether the caller of the `quoteOFT` function on the `NexusOFT` contract is exempt or not to portray an accurate quote response in the `nexusQuoteOFT` function. Note that this vector also applies to Portal, hence `quoteOFT` can check for exemptions there as well.

Resolution

LayerZero: Acknowledged.

I-64 | nexusQuoteOFT Does Not Consult The MsgInspector

Category	Severity	Location	Status
Unexpected Behavior	● Info	Nexus.sol	Acknowledged

Description

In the `nexusQuoteOFT` function, the msg inspector is not consulted to verify if the send may be invoked or not based upon the sender, sendParams, and options. However, the `nexusQuoteSend` function does consult the msg inspector by way of using the `_buildMsgAndOptions` function.

Recommendation

Consider if the `nexusQuoteOFT` function ought to consult the msg inspector to more accurately mimic the result of a send operation. Or consider standardizing both the `nexusQuoteOFT` and `nexusQuoteSend` functions to both not consult the msg inspector if it should not be considered. Otherwise clearly document this difference between the two quote functions.

Resolution

LayerZero: Acknowledged.

Round Two Findings & Resolutions

ID	Title	Category	Severity	Status
I-01	Stale Checkpointing Documentation	Documentation	● Info	Acknowledged
I-02	Potential Overflow In maxAmountLD	Warning	● Info	Acknowledged
I-03	AccessControl2StepUpgradeable Lacks Timelock	Warning	● Info	Acknowledged
I-04	PauseByIDBase._setPaused Lacks Idempotent Check	Warning	● Info	Acknowledged
I-05	Missing AccessControl Init Locks Admin Functions	Logical Error	● Info	Acknowledged
I-06	PreCrime Simulation Layer Entirely Removed	Warning	● Info	Acknowledged

I-01 | Stale Checkpointing Documentation

Category	Severity	Location	Status
Documentation	● Info	IRateLimiter.sol	Acknowledged

Description

IRateLimiter.checkpointRateLimits docs still state it should be used "when not using setRateLimitConfigs", but current behavior is the opposite: setRateLimitConfigs does not checkpoint state. This creates contradictory guidance for operators updating limits/windows.

Recommendation

Update the checkpointRateLimits docstring to reflect current behavior, e.g. "call before changing limits/windows..." so it is clear that rate limiter admin must trigger checkpoints, otherwise changes can apply retroactively.

Resolution

LayerZero: Acknowledged.

I-02 | Potential Overflow In maxAmountLD

Category	Severity	Location	Status
Warning	● Info	Nexus.sol	Acknowledged

Description

In `nexusQuoteOFT`, `maxAmountLD` is rescaled by fee basis points:
`maxAmountLD = (maxAmountLD * BPS_DENOMINATOR) / (BPS_DENOMINATOR - feeBps);`
This is currently safe because available amounts are tightly bounded, the unlimited `type(uint256).max` sentinel is excluded before the multiplication, and the full-fee case is handled separately. The concern is that this safety depends on those invariants remaining true. If future changes introduce scaled availability or broader bounds, this multiplication path could become an overflow or revert surface.

Recommendation

Document the assumptions that keep this safe today, and revisit the calculation if future changes expand the range of `maxAmountLD`.

Resolution

LayerZero: Acknowledged.

I-03 | AccessControl2StepUpgradeable Lacks Timelock

Category	Severity	Location	Status
Warning	● Info	AccessControl2StepUpgradeable.sol	Acknowledged

Description

AccessControl2StepUpgradeable implements a two-step DEFAULT_ADMIN_ROLE transfer modeled after OpenZeppelin's AccessControlDefaultAdminRules. However, it omits a time delay between beginDefaultAdminTransfer() and acceptDefaultAdminTransfer(). As implemented, the Acknowledged admin can accept the transfer in the very same block as the transfer is initiated without delay.

Recommendation

Clearly document this behavior or consider adding a delay.

Resolution

LayerZero: Acknowledged.

I-04 | PauseByIDBase._setPaused Lacks Idempotent Check

Category	Severity	Location	Status
Warning	● Info	PauseByIDBase.sol	Acknowledged

Description

PauseByIDBaseUpgradeable._setPaused does not check whether the new config matches the existing config before writing. Every other pause setter in the codebase reverts on same-state writes:

- PauseBaseUpgradeable._setPaused(bool) — reverts with PauseStateldempotent
- PauseByIDBaseUpgradeable._setDefaultPaused(bool) — reverts with PauseStateldempotent

But PauseByIDBaseUpgradeable._setPaused(SetPausedParam[]) unconditionally overwrites and emits, which allows the same config to be written repeatedly, emitting duplicate PauseSet events each time.

Recommendation

Consider adding an idempotent check per entry,

Resolution

LayerZero: Acknowledged.

I-05 | Missing AccessControl Init Locks Admin Functions

Category	Severity	Location	Status
Logical Error	● Info	Global	Acknowledged

Description

OAppCoreRBACUpgradeable protects management functions such as setPeer and setDelegate with onlyRole(OAPP_ADMIN_ROLE), but its initializer does not initialize AccessControl2Step. If an integrating contract calls only __OAppCoreRBAC_init(...) and skips __AccessControl2Step_init(...), no account receives DEFAULT_ADMIN_ROLE, grantRole cannot be used, and the role-gated admin paths can be locked permanently.

Recommendation

Update the docs and initialization guidance to state explicitly that inheritors must call __AccessControl2Step_init(initialAdmin), and that omitting this step can lock admin functionality.

Resolution

LayerZero: Acknowledged.

I-06 | PreCrime Simulation Layer Entirely Removed

Category	Severity	Location	Status
Warning	● Info	OFTCoreBaseUpgradeable.sol	Acknowledged

Description

The Console implementation completely removes LayerZero's PreCrime simulation subsystem from the OFT stack. In `OFTCoreBaseUpgradeable`, the inheritance chain is `OAppUpgradeable -> OAppOptionsType3BaseUpgradeable -> OAppMsgInspectionBaseUpgradeable – OAppPreCrimeSimulatorUpgradeable` is absent.

Recommendation

Consider if this intended. If so, clearly document this behavior.

Resolution

LayerZero: Acknowledged.

Round Three Findings & Resolutions

ID	Title	Category	Severity	Status
I-01	Shared Nonce Channel Enables Token-level DoS	DoS	● Info	Acknowledged
I-02	Immutable Decimals Can Change On Upgrade	Upgradeability	● Info	Acknowledged

I-01 | Shared Nonce Channel Enables Token-level DoS

Category	Severity	Location	Status
DoS	● Info	Nexus	Acknowledged

Description [PoC](#)

`setPeer(dstEid, peer)` creates a single shared LayerZero lane between Nexus contracts. At the protocol level, LayerZero identifies channels by (sender, srcEid, dstEid, receiver) and maintains a single sequential nonce per lane, with no awareness of token IDs. Although each token has its own NexusOFT, all tokens route through the same Nexus hub, so transfers for USDC, WBTC, and all other tokens share one nonce stream between chains.

On the source side, a send is accepted as long as the OFT is registered locally and a peer exists for the destination chain, but it does not verify that the token is registered on the destination Nexus. This allows a token that exists on chain A but not on chain B to still be sent. The failure only occurs on B inside `_lzReceive()`, where `InvalidTokenId(tokenId)` is triggered. By that point, the message has already been verified and stored in the LayerZero endpoint, but not cleared.

Because LayerZero uses a lazy inbound nonce, clearing later messages requires iterating through all prior nonces in the lane. If invalid packets accumulate ahead of valid ones, this linear scan becomes increasingly expensive and can run out of gas, preventing legitimate transfers from being processed.

An attacker can exploit this by repeatedly sending a token supported on A but not B (even with `amountLD = 0`). These messages are accepted on A, revert on B, and remain uncleared, creating a backlog. Since all tokens share the same Nexus lane, this backlog impacts unrelated tokens, eventually causing legitimate transfers to fail due to gas exhaustion during `_clearPayload`.

This results in a global DoS across all tokens on the affected chain pair. The attack is cheap to execute (only messaging fees), and recovery is operational, requiring manual clearing of stuck packets while user funds remain blocked until intervention.

<https://gist.github.com/GuardianAudits/6cf23d0d7c36bf1bd299aa4b22da15b8>

Recommendation

`setPeer(B)` opens the A -> B road for the whole Nexus
`registerToken(X)` decides which tokens exist on each chain
The bug needs token X exists on A, does not exist on B, but the A -> B road is still open for X
A mitigation for this can be to pause any affected route, by blocking any exact pair, tokenId X -> destination Eid B, before `nexusSend()` burns and sends
By computing the route ID with `nexus.getNexusId(tokenId, dstEid)`, then calling `setPaused(routeId)` on the pause module

Resolution

LayerZero: Acknowledged.

I-02 | Immutable Decimals Can Change On Upgrade

Category	Severity	Location	Status
Upgradeability	● Info	ERC20Plus.sol:21-92	Acknowledged

Description

ERC20Plus stores DECIMALS as an immutable set in the implementation constructor. In a proxy setup, immutables live in the implementation bytecode (not proxy storage). If the proxy is upgraded to an implementation deployed with a different constructor _decimals, decimals() will silently change while balances/allowances/totalSupply remain in the old units.

Recommendation

Do not use an immutable for decimals in an upgradeable token. Store decimals in proxy storage (set once in initialize()), and enforce that it cannot change across upgrades.

Resolution

LayerZero: Acknowledged.

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

Founded in 2022 by DeFi experts, Guardian is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian 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>