PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For LayerZero (EndpointV2Alt Support)

03 February 2026

paladinsec.co

info@paladinsec.co

Table of Contents

Disclaimer

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

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

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

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Paladin retains the right to re-use any and all knowledge and expertise gained during the audit process, including, but not limited to, vulnerabilities, bugs, or new attack vectors. Paladin is therefore allowed and expected to use this knowledge in subsequent audits and to inform any third party, who may or may not be our past or current clients, whose projects have similar vulnerabilities. Paladin is furthermore allowed to claim bug bounties from third-parties while doing so.

1 Overview

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

1.1 Summary

Project Name	LayerZero
URL	https://layerzero.network/
Platform	Ethereum
Language	Solidity
Preliminary Contracts	https://github.com/stargate-protocol/stargate-v2/commit/18629d1749d0a9b0cf9e103d6fba56f0710a7c6a
Resolution #1	https://github.com/stargate-protocol/stargate-v2/tree/5ee7c7c1574053c11fad6c1ffbe933df36651412

1.2 Contracts Assessed

- `StargateOFTTip20`
- `StargateOFTAlt`
- `StargateBase`
- `CreditMessagingAlt`
- `TokenMessagingAlt`
- `Transfer`
- `ITip20Minter`

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	0	-	-	-
Medium	0	-	-	-
Low	4	2	-	2
Informational	4	-	-	4
Total	**8**	**2**	-	**6**

Classification of Issues

Severity	Description
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 StargateOFTTip20

ID	Severity	Summary	Status
01	LOW	TIP20 pause does not block `mint()`, so bridge executes inbound transfers	ACKNOWLEDGED
02	LOW	TIP20 tokens do not have `transferOwnership()`	✓ RESOLVED
03	INFO	Tokens may remain stuck for a very long time due to TIP20 operational state	ACKNOWLEDGED

1.3.2 StargateOFTAlt

No issues found.

1.3.3 StargateBase

ID	Severity	Summary	Status
04	LOW	`recoverToken()` does not enforce its documented invariant and allows recovering the Stargate asset	✓ RESOLVED

1.3.4 CreditMessagingAlt

ID	Severity	Summary	Status
05	INFO	Missing `msg.value` check in `_lzReceive`	ACKNOWLEDGED

1.3.5 TokenMessagingAlt

ID	Severity	Summary	Status
06	LOW	`lzReceive` can still receive bus calls	ACKNOWLEDGED
07	INFO	Missing `msg.value` check in `_lzReceive`	ACKNOWLEDGED

1.3.6 Transfer

ID	Severity	Summary	Status
08	INFO	No event emitted when transferGasLimit is changed	ACKNOWLEDGED

1.3.6 ITip20Minter

No issues found.

2 Findings

2.1 StargateOFTTip20

`StargateOFTTip20` is a Stargate OFT variant for TIP-20 compliant bridged stablecoins on `EndpointV2Alt` chains that burns tokens on inflow and mints on outflow (instead of locking/unlocking).

2.1.1 Privileged Functions

- `transferTokenOwnership[onlyOwner]`
- `setAddressConfig[onlyOwner]`
- `setOFTPath[onlyOwner]`
- `withdrawTreasuryFee[onlyCaller]`
- `addTreasuryFee[onlyCaller]`
- `recoverToken[onlyCaller]`
- `setPause[onlyCaller]`
- `withdrawPlannerFee[onlyCaller]`
- `receiveTokenBus[onlyCaller]`
- `receiveTokenTaxi[onlyCaller]`
- `sendCredits[onlyCaller]`
- `setTransferGasLimit[onlyOwner]`

2.1.1 Issues & Recommendations

Issue #01	**TIP20 pause does not block `mint()`, so bridge executes inbound transfers**
Severity	LOW SEVERITY
Description	Pausing the token blocks transfers (`transfer`, `transferFrom`) via `notPaused`, but does not block supply-changing functions: - `mint()` does not have `notPaused` - `burn()` does not have `notPaused` So if the TIP20 admin pauses the token, bridging can become one-way depending on the direction: - Send out of Tempo can fail because Stargate needs `transferFrom` (`paused -> revert`). - Receiving into Tempo can still succeed because Stargate mints on destination and `mint()` still works while paused.
Recommendation	Check the pause state in `StargateOFTTIP20._outflow()` before minting and return `false` if paused. This way, the transaction will be stored for later retry and users can retry after unpause.
Resolution	ACKNOWLEDGED The team stated: "We'll maintain the current pause; debits (via transferFrom) will revert, while credits will succeed. This avoids undelivered cross-chain messages and retry loops. Users cannot do anything with credited tokens while paused, other than burn. Besides this, we do not expect pausing the underlying token without pausing the oft, so this will not affect the expected flow."

Issue #02	TIP20 tokens do not have `transferOwnership()`
Severity	LOW SEVERITY
Description	The `StargateOFT.transferTokenOwnership()` `onlyOwner` function attempts to transfer the ownership of the token to another contract or EOA. TIP20 operates with roles instead and the `StargateOFTTip20` contract will have to be given the `ISSUER_ROLE` to be able to mint and burn.
Recommendation	If `StargateOFTTip20` is going to be always given the admin for the issuer role, then something similar to the `StargateOFT.transferTokenOwnership()` could be implemented that transfers the admin and the `ISSUER_ROLE` to another address.
Resolution	RESOLVED

Issue #03	**Tokens may remain stuck for a very long time due to TIP20 operational state**
Severity	INFORMATIONAL
Description	In `StargateOFTTIP20`, token delivery on the destination chain depends entirely on a successful `ITIP20Minter(token).mint(_to, _amountLD) inside _outflow()`. If `_outflow()` (minting) fails on TIP20 (e.g., receiver not TIP403 authorized, supply cap exceeded), tokens are correctly cached in `unreceivedTokens` for later retry via `retryReceiveToken()`. However, the `retry` flow also depends on the same TIP20 conditions being favorable. If they are not resolved, `retry` will keep failing: - Supply cap still exceeded -> mint reverts - `ISSUER_ROLE` revoked -> mint reverts - Receiver still not TIP403 authorized -> mint reverts The transaction reverts entirely, so the cache entry is preserved and users can retry later. But if the admin never fixes these conditions, tokens remain stuck forever. This is particularly concerning during high-volume bridging where supply cap could fill up, blocking all subsequent receivers until admin intervention and this report is mainly directed to describe the supply cap exceeded scenario. Example scenario: - Supply cap nearly full. - High bridging volume. - Some mints succeed, later ones revert with `SupplyCapExceeded()`. - Tokens are cached in `unreceivedTokens`. - If `supplyCap` is never raised, tokens are permanently unclaimable.
Recommendation	Explicitly document that cross-chain delivery depends on TIP20 mint availability (authorization, supply cap, issuer role).

Resolution

ACKNOWLEDGED

The team stated that this is consistent across chains: blacklisted addresses cannot receive funds, as expected. The uint128 cap (with 6 decimals, stablecoin context) is sufficient.

2.2 StargateOFTAlt

`StargateOFTAlt` forwards ERC20 `transferFrom` calls exclusively from the `LZMultiCall` contract, acting as the token allowance holder that users approve to enable delegated token transfers.

2.2.1 Privileged Functions

- `transferTokenOwnership[onlyOwner]`
- `setAddressConfig[onlyOwner]`
- `setOFTPath[onlyOwner]`
- `withdrawTreasuryFee[onlyCaller]`
- `addTreasuryFee[onlyCaller]`
- `recoverToken[onlyCaller]`
- `setPause[onlyCaller]`
- `withdrawPlannerFee[onlyCaller]`
- `receiveTokenBus[onlyCaller]`
- `receiveTokenTaxi[onlyCaller]`
- `sendCredits[onlyCaller]`
- `setTransferGasLimit[onlyOwner]`

2.2.2 Issues & Recommendations

No issues found.

2.3 StargateBase

`StargateBase` is the core contract for all Stargate OFT variants, handling cross-chain token transfers via LayerZero messaging. It manages token inflow/outflow, path credits, fee charging, and a retry mechanism for failed deliveries. Treasury fee collection and token recovery functions are also implemented here.

2.3.1 Privileged Functions

- `setAddressConfig[onlyOwner]`
- `setOFTPath[onlyOwner]`
- `setTransferGasLimit[onlyOwner]`
- `setPause[onlyOwner]`
- `recoverToken[onlyOwner]`
- `withdrawTreasuryFee[onlyCaller(treasurer)]`
- `addTreasuryFee[onlyCaller(treasurer)]`
- `receiveTokenTaxi[onlyCaller(tokenMessaging)]`
- `receiveTokenBus[onlyCaller(tokenMessaging)]`
- `receiveCredits[onlyCaller(creditMessaging)]`
- `sendCredits[onlyCaller(creditMessaging)]`

2.3.2 Issues & Recommendations

Issue #04	**`recoverToken()` does not enforce its documented invariant and allows recovering the Stargate asset**
Severity	LOW SEVERITY

Description	The comment in `recoverToken()` states: "Reverts with `Stargate_RecoverTokenUnsupported` if the treasurer attempts to withdraw `StargateBase.token()`."
	
	However, the implementation only blocks native coin:
	`if (_token == address(0)) revert Stargate_RecoverTokenUnsupported();`
	
	There is no check preventing `_token == token`.
	
	As a result, the treasurer can call:
	`recoverToken(token, to, amount);`
	
	and transfer out the underlying Stargate asset itself.
	
	While this function is restricted to the treasurer, it violates the documented safety property and allows recovery of the core Stargate token if it resides in the contract.

Recommendation	Enforce the documented invariant by blocking recovery of the Stargate asset: `if (_token == address(0) || _token == token) revert Stargate_RecoverTokenUnsupported();`
Resolution	RESOLVED It was a documented issue.

2.4 CreditMessagingAlt

`CreditMessagingAlt` is the ALT-endpoint variant of CreditMessaging for sending credit messages with ERC-20 fee tokens. It re-quotes the messaging fee on-chain, transfers the required fee tokens from the planner to the endpoint, and explicitly rejects any msg.value.

2.4.1 Privileged Functions

- `setAssetId[onlyOwner]`
- `setMaxAssetId[onlyOwner]`
- `setPlanner[onlyOwner]`
- `setGasLimit[onlyOwner]`
- `sendCredits[onlyCaller(planner)]`
- `setPeer[onlyOwner]`
- `setDelegate[onlyOwner]`
- `renounceOwnership[onlyOwner]`
- `transferOwnership[onlyOwner]`

2.4.2 Issues & Recommendations

Issue #05	Missing `msg.value` check in `_lzReceive`
Severity	INFORMATIONAL
Description	External `lzReceive` function is payable and native drops can be received.
Recommendation	We recommend overriding `_lzReceive` and adding `_assertNoNativeValue();` in the beginning.
Resolution	ACKNOWLEDGED The team stated: "`lzReceive` originates in the Executor, which forwards `msg.value` through the Endpoint to the OApp. For correct messages this should be 0."

2.5 TokenMessagingAlt

`TokenMessagingAlt` is the ALT-endpoint variant of `TokenMessaging` that overrides `_payNative()` to return zero. This prevents any native ETH from being forwarded to the endpoint, relying on the Stargate layer to handle ERC-20 fee token transfers instead.

2.5.1 Privileged Functions

- `setAssetId[onlyOwner]`
- `setMaxAssetId[onlyOwner]`
- `setPlanner[onlyOwner]`
- `setGasLimit[onlyOwner]`
- `setMaxNumPassengers[onlyOwner]`
- `setFares[onlyOwner]`
- `setNativeDropAmount[onlyOwner]`
- `initializeBusQueueStorage[onlyOwner]`
- `setPeer[onlyOwner]`
- `setDelegate[onlyOwner]`
- `renounceOwnership[onlyOwner]`
- `transferOwnership[onlyOwner]`

2.5.2 Issues & Recommendations

Issue #06	`lzReceive` can still receive bus calls
Severity	LOW SEVERITY
Description	There are no checks that stop TokenMessaging from receiving inbound bus calls via `lzReceive() -> _lzReceive() -> _lzReceiveBus()`.
Recommendation	We recommend overriding `_lzReceive` and reverting if `BusCodec.isBus(_message)` is true.
Resolution	ACKNOWLEDGED The team stated: "This is expected. Only messages going from Tempo will have bus mode disabled. We checked the path to ensure it works as expected."

Issue #07	Missing msg.value check in `_lzReceive`
Severity	INFORMATIONAL
Description	External `lzReceive` function is payable and native drops can be received.
Recommendation	We recommend overriding `_lzReceive` and adding `_assertNoNativeValue();` in the beginning.
Resolution	ACKNOWLEDGED The team stated: "`lzReceive` originates in the Executor, which forwards `msg.value` through the Endpoint to the OApp. For correct messages this should be 0."

2.6 Transfer

`Transfer` is a utility library providing safe transfer wrappers for ERC-20 tokens and native currency. It handles non-standard token return values and provides both reverting and non-reverting transfer variants for flexible error handling across the protocol.

2.6.1 Privileged Functions

- `setTransferGasLimit[onlyOwner]`
- `renounceOwnership[onlyOwner]`
- `transferOwnership[onlyOwner]`

2.6.2 Issues & Recommendations

Issue #08	No event emitted when `transferGasLimit` is changed
Severity	INFORMATIONAL
Description	`transferGasLimit` is a mutable configuration that directly affects `transferNative(..., _gasLimited = true)` behavior. However, `setTransferGasLimit()` does not emit any event. This makes it impossible for off-chain monitoring systems to detect configuration changes that may alter native transfer behavior. If a future module relies on `_gasLimited = true`, changes to this value could affect compatibility (e.g. smart contract wallets) without any observable signal.
Recommendation	Emit an event in the `setTransferGasLimit()` function when the value changes.
Resolution	ACKNOWLEDGED

2.7 ITip20Minter

`ITip20Minter` is an interface for TIP-20 tokens that defines mint and burn functions for bridged stablecoin supply management.

2.6.2 Issues & Recommendations

No issues found.


PALADIN
BLOCKCHAIN SECURITY