PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For LayerZero
(OVault)

16 July 2025

paladinsec.co   info@paladinsec.co

Table of Contents

Disclaimer

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

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

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

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

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

1 Overview

This report has been prepared for LayerZero's OVault 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/ondoprotocol/gm-lz-bridge/tree/2214e20491619070d43da049dadf509901724b78/
Resolution #1	https://github.com/LayerZero-Labs/devtools/pull/1617/commits/41e6b52b1c766ea593bb38a2c524ab69c4db4152 https://github.com/LayerZero-Labs/devtools/commit/b8258f14c37d7da0d32fc1666f07996411d65c91

1.2 Contracts Assessed

Name	Contract	Live Code Match
`OVault`		
`OVaultComposer`		
`OVaultUpgradeable`		
`IOFTWithDecimalConversionRate`		
`IOVaultComposer`		

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
● High	0	-	-	-
● Medium	2	2	-	-
● Low	2	2	-	-
● Informational	10	6	-	4
Total	**14**	**10**	-	**4**

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 OVault

ID	Severity	Summary	Status
01	INFO	First depositors into `OVault` are at risk of receiving 0 shares	ACKNOWLEDGED

1.3.2 OVaultComposer

ID	Severity	Summary	Status
02	MEDIUM	`depositSend()` and `redeemSend()` can cause loss of funds or reverts	✓ RESOLVED
03	MEDIUM	Missing slippage check on `depositSend` and `redeemSend`	✓ RESOLVED
04	LOW	`_refundAddress` in `depositSend()` and `redeemSend()` will get ignored if destination is the HUB chain	✓ RESOLVED
05	LOW	Unnecessary dust removal in slippage check edge case	✓ RESOLVED
06	INFO	Maximum approval of `OVault` and `shareOFT` in constructor	ACKNOWLEDGED
07	INFO	Failed messages could be forced to be executed in the opposite of the legit call path	ACKNOWLEDGED
08	INFO	`OVaultComposer` can receive ETH, but there is no function to retrieve it	ACKNOWLEDGED
09	INFO	`send()` uses transfer instead of `safeTransfer`	✓ RESOLVED
10	INFO	`_send()` does not emit an event in every case	✓ RESOLVED
11	INFO	Possible erroneous git merge resulted in duplicate code	✓ RESOLVED
12	INFO	Typographical issues	✓ RESOLVED

1.3.3 OVaultUpgradeable

ID	Severity	Summary	Status
13	INFO	Missing `initialize` function	✓ RESOLVED

1.3.4 IOFTWithDecimalConversionRate

No issues found.

1.3.5 IOVaultComposer

ID	Severity	Summary	Status
14	INFO	Unused code	✓ RESOLVED

2 Findings

2.1 OVault

`OVault` is a wrapper on ERC4626 and will only be deployed on a single EVM network called the HUB network. The Vault takes in an asset token and produces share tokens, as such we have an asset OFT and a share OFT. These OFTs can be deployed on any number of networks, and will function as a HUB-Spoke model. The main focus of this audit is the composer that we are building to automate the following actions on the ERC4626:

- `deposit()`
- `redeem()`

2.1.1 Issues & Recommendations

Issue #01	First depositors into `OVault` are at risk of receiving 0 shares
Severity	● INFORMATIONAL

Description	As described in OZ's ERC4626 implementation:
	
	`* [CAUTION]`
	`* ====`
	`* In empty (or nearly empty) ERC-4626 vaults, deposits are at high risk of being stolen through frontrunning`
	`* with a "donation" to the vault that inflates the price of a share. This is variously known as a donation or inflation`
	`* attack and is essentially a problem of slippage. Vault deployers can protect against this attack by making an initial`
	`* deposit of a non-trivial amount of the asset, such that price manipulation becomes infeasible. Withdrawals may`
	`* similarly be affected by slippage. Users can protect against this attack as well as unexpected slippage in general by`
	`* verifying the amount received is as expected, using a wrapper that performs these checks such as`
	`* https://github.com/fei-protocol/ERC4626#erc4626router-and-base[ERC4626Router].`
	
	—
	
	First depositors into `OVault` are at risk of receiving 0 shares if they get frontrun by an attacker that mints 1 wei share and then transfers underlying tokens to the `OVault`.

Recommendation	Consider mitigating this known ERC4626 issue by depositing funds to the OVault upon deployment. Also consider introducing slippage checks in `redeemSend()` and `depositSend()`.
Resolution	● ACKNOWLEDGED The team will document this behaviour.

2.2 OVaultComposer

`OVaultComposer` is a cross-chain vault composer built on LayerZero's messaging protocol. It allows users to deposit assets into an ERC4626 vault and receive shares on another chain, or redeem shares on one chain and receive assets on another. The contract acts as a middleware between asset/share OFT (Omnichain Fungible Token) contracts and an ERC4626 vault.

Features:

- Cross-chain deposit and redemption flows
- Handles messaging between chains via LayerZero
- Built-in error handling with refund and retry mechanisms for failed messages
- Slippage protection for vault operations

2.2.1 Privileged Functions

- `lzCompose`

2.2.2 Issues & Recommendations

Issue #02 — **depositSend() and redeemSend() can cause loss of funds or reverts**

Severity — MEDIUM SEVERITY

Location

```
function depositSend(
        SendParam calldata _sendParam,
        address _refundAddress
   ) external payable nonReentrant {
    IERC20(ASSET_ERC20).safeTransferFrom(msg.sender,
address(this), _sendParam.amountLD);

    _executeOVaultAction(
             ASSET_OFT,
             _sendParam.amountLD
          );

    _send(
              SHARE_OFT, // _oft
             _sendParam, // _sendParam
              msg.value, // _totalMsgValue
              _refundAddress // _refundOverpayAddress
    );
}
```

Description

depositSend() deposits assets into the OVault and sends the shares either to an address on the HUB chain or to a destination chain via the SharesOFT.

The issue is that depositSend() assumes that assets and shares have a 1:1 ratio which easily could be changed by sending underlying tokens to the OVault.

This is assumed because if 100e18 assets are specified in sendParam and the deposit mints only 90e18 shares, then the code will attempt to send 100e18 shares either to an address or to the sharesOFT which causes a revert.

In redeemSend, there is the same issue — if a user specifies 100e18 amountLD in sendParam, 110e18 assets could be returned but only 100e18 assets would be sent, resulting in 10e18 assets being locked in OVaultComposer or reverting depending on minAmountLD.

	Additionally, `quoteDepositSend` and `quoteRedeemSend` do not take into account the vault deposit/redeem and just assume a 1:1 conversion.
Recommendation	Consider re-assigning `sendParam.amountLD` after the deposit / redeem with the correct value. For the quote functions, consider using `previewDeposit`/`previewRedeem`.
Resolution	✓ RESOLVED

Issue #03	**Missing slippage check on `depositSend` and `redeemSend`**
Severity	● MEDIUM SEVERITY
Description	In case a user specifies the HUB chain as the `destEID` when using `depositSend()` or `redeemSend()`, a slippage check should be included because for the Hub chain, a simple transfer is executed to the receiver. This means an attacker can sandwich the send operation, resulting in less OFT being transferred to the user.
Recommendation	Consider using `_executeOVaultActionWithSlippageCheck`.
Resolution	✓ RESOLVED

Issue #04	**`_refundAddress` in `depositSend()` and `redeemSend()` will get ignored if destination is the HUB chain**
Severity	LOW SEVERITY
Description	If a user specifies the HUB chain as the `destEID` when using `depositSend()` or `redeemSend()`, then if the transfer of native funds to the recipient fails, the funds will be sent to `REFUND_OVERPAY_ADDRESS` instead of `_refundOverpayAddress`. `REFUND_OVERPAY_ADDRESS` should be used if there is overpaying when calling `OFT.send()`, or if the recipient of the native funds cannot receive them in case the message comes from a different source than the HUB chain. If a user calls `depositSend()` or `redeemSend()`, having their native funds sent to the refund overpay address despite providing `_refundAddress` might be an unexpected outcome.
Recommendation	Possible solutions: - Consider documenting the behaviour if the logic will stay the same. - Use a different logic than `_send` in `depositSend()` and `redeemSend()` that will actually return the native funds in case of revert to the `_refundAddress`.
Resolution	✓ RESOLVED

Issue #05	Unnecessary dust removal in slippage check edge case
Severity	● LOW SEVERITY
Description	When the execution of `lzCompose()` reaches `try this.executeOVaultActionWithSlippageCheck(callerOFT, amount, composeSendParam.minAmountLD) returns (`, there is an edge case where the dust removal that happens in `_checkSlippage()` is not needed. If we send 100e18 assets from chain1 to HUB and include a compose message that will automatically deposit the assets, then the user must specify `minAmountLD` in the compose message. Let's assume local decimals are 18 and shared are 6. The decimal conversion rate is going to be 1e12. If `minAmountLD` is 90e18 + 1e12 and the deposit `OVault` function mints 90e18 + 0.9e12, then the call would revert because when dust is removed 90e18 will be left. Dust should be removed when the `dstEID` is another chain and the amount will be sent to an OFT for bridging but if the `dstEID` is the HUB chain then the minted shares should simply be transferred to the recipient so there will be no need to remove the dust when comparing `minAmountLD` as the recipient will receive the dust amount as well.
Recommendation	If the HUB chain is the `dstEID` of the compose message, consider comparing the amount with `minAmountLD` without removing the dust.
Resolution	✓ RESOLVED

Issue #06	Maximum approval of `OVault` and `shareOFT` in constructor
Severity	INFORMATIONAL
Description	The composer contract will be holding funds and there is a possibility that `OVault` might be an upgradable contract. If it is compromised, then the approvals could be used to drain the composer.
Recommendation	Consider approving when needed instead of using maximum approve.
Resolution	ACKNOWLEDGED

Issue #07	Failed messages could be forced to be executed in the opposite of the legit call path
Severity	INFORMATIONAL
Description	`retryWithSwap()` and `refund()` can be both called when a message is with status `CanRefundOrRetryWithSwap`. An attacker could front run a legit call to these functions and force the failed message to be executed in the opposite action—refunded or retried.
Recommendation	Unfortunately there is no easy solution. We included this issue so the team will be aware.
Resolution	ACKNOWLEDGED The team stated: "This is entirely possible and a known limitation of a permissionless design. We can add this to the docs where we tell developers to go with a ownable model on the retry states if they do not want this to happen."

Issue #08	**`OVaultComposer` can receive ETH, but there is no function to retrieve it**
Severity	● INFORMATIONAL
Description	`receive() external payable {}` allows ETH to be received, but currently there is no function which enables a trusted actor to retrieve it.
Recommendation	Implement a guarded function to collect the ETH from the contract if there is more than needed.
Resolution	● ACKNOWLEDGED

Issue #09	**`send()` uses `transfer` instead of `safeTransfer`**
Severity	● INFORMATIONAL
Description	`SafeERC20` is used everywhere in the contracts except in the `send()`.
Recommendation	Consider changing `token.transfer(_receiver, _amountLD);` to `IERC20(token).safeTransfer(_receiver, _amountLD);`
Resolution	✓ RESOLVED

Issue #10	**`_send()` does not emit an event in every case**
Severity	● INFORMATIONAL
Description	Within `_sendParam.dstEid != HUB_EID`, the `_send()` function does not emit an event.
Recommendation	Consider adding an event emission.
Resolution	✓ RESOLVED

Issue #11	Possible erroneous git merge resulted in duplicate code
Severity	● INFORMATIONAL
Description	`OVaultComposer` contains several blocks that are repeated verbatim—e.g., duplicate import lines for SafeERC20, two identical uses of SafeERC20 for IERC20 directives, two successive assignments to each immutable decimal-conversion variable in the constructor, and redundant state updates such as failedMessages[_guid] = … and duplicate branch checks in `retry()`. These repetitions strongly suggest an unresolved merge conflict or an accidental copy-paste during a recent commit.
Recommendation	Consider re-analyzing the code to remove all the repeated codeblocks.
Resolution	✓ RESOLVED

Issue #12	Typographical issues
Severity	INFORMATIONAL

Description	Remove the following lines since they are declared twice:
	
	Lines 7, 23, 71-72, 114, 123, 136, 235.
	
	—
	
	Consider adding natspec comments and documenting each param on each external function for better code readability.
	
	—
	
	`send() handles the HUB-chain shortcut with an early if (…) { …; return; }, whereas _send() handles the same case with an if … else … ladder. The mixed style is purely cosmetic but increases the chance of future maintenance errors when the two functions need to stay in sync. Furthermore, send() can just call`
	`IOFT(_oft).send{ value: _totalMsgValue }(`
	`                _sendParam,`
	`                MessagingFee(_totalMsgValue, 0),`
	`                _refundOverpayAddress`
	`            );`
	
	—
	
	The `delete` of `failedMessages` on refund/retry is obsolete because `sendFailedMessage` already deletes that mapping entry on L304:
	`delete failedMessages[_guid];`

Recommendation	Consider making the recommended changes.
Resolution	✓ RESOLVED

2.3 OVaultUpgradeable

`OVaultUpgradeable` is the upgradable version of `OVault`.

2.3.1 Issues & Recommendations

Issue #13	**Missing `initialize` function**
Severity	● LOW SEVERITY
Description	The contract cannot be used properly without adding an `initialize` function with the initializer modifier. Inside this function, the following functions should be called: `__ERC20_init()`, `__ERC4626_init()` and `__Context_init()`.
Recommendation	Consider implementing the following recommendation.
Resolution	✓ RESOLVED

2.4 IOFTWithDecimalConversionRate

`IOFTWithDecimalConversionRate` is the interface for the OFT that exposes `decimalConversionRate`.

2.4.1 Issues & Recommendations

No issues found.

2.5 IOVaultComposer

`IOVaultComposer` is the interface for the `OVaultComposer`.

2.5.1 Issues & Recommendations

Issue #14	Unused code
Severity	● INFORMATIONAL
Description	Several code are unused: - `OnlyAsset` error - `CanNotWithdraw` error - `OnlyShare` error
Recommendation	Consider removing the unused code.
Resolution	✓ RESOLVED

PALADIN

BLOCKCHAIN SECURITY