PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For LayerZero Native OFTAdapter

28 Aug 2024

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 Native OFTAdapter 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 Native OFTAdapter
URL	https://www.layerzero.foundation/
Platform	Ethereum
Language	Solidity
Preliminary	https://github.com/LayerZero-Labs/devtools/blob/6813a304474a5e707c09b0eafb680da31078dc2c/packages/oft-evm/contracts/NativeOFTAdapter.sol https://github.com/LayerZero-Labs/devtools/blob/6813a304474a5e707c09b0eafb680da31078dc2c/packages/oft-evm/contracts/OFTCore.sol
Resolution	https://github.com/LayerZero-Labs/devtools/blob/6813a304474a5e707c09b0eafb680da31078dc2c/packages/oft-evm/contracts/NativeOFTAdapter.sol https://github.com/LayerZero-Labs/devtools/blob/6813a304474a5e707c09b0eafb680da31078dc2c/packages/oft-evm/contracts/OFTCore.sol

1.2 Contracts Assessed

Name	Contract	Live Code Match
NativeOFTAdapter		PENDING
OFTCore		PENDING

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
Governance	-	-	-	-
High	-	-	-	-
Medium	-	-	-	-
Low	2	-	-	2
Informational	1	1	-	-
Total	**3**	**1**	**-**	**2**

Classification of Issues

Severity	Description
Governance	Issues under this category are where the governance or owners of the protocol have certain privileges that users need to be aware of, some of which can result in the loss of user funds if the governance's private keys are lost or if they turn malicious, for example.
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 NativeOFTAdapter

ID	Severity	Summary	Status
1	LOW	Withdrawing to a contract which does not accept native gas tokens does not revert early	ACKNOWLEDGED
2	INFO	Receipt does not adhere to checks-effects-interactions	✓ RESOLVED

1.3.2 OFTCore

ID	Severity	Summary	Status
3	LOW	_toSD can overflow for specific high supply tokens, causing the contract to severely malfunction	ACKNOWLEDGED

2 Findings

2.1 NativeOFTAdapter

The `NativeOFTAdapter` can be deployed on a mainnet chain to permit users to deposit real native gas tokens into the adapter to mint OFT tokens on the connected chains, effectively bridging the real mainnet gas token into bridged OFT versions.

The adapter then acts as the "locker" for these gas tokens, keeping them safe until the users eventually bridge the tokens back into mainnet. At that point, the tokens are transferred to the wallet which bridges out.

2.1.1 Privileged Functions

- `setMsgInspector`
- `setPreCrime`
- `setEnforcedOptions`
- `setPeer`
- `setDelegate`
- `transferOwnership`
- `renounceOwnership`

2.1.2 Issues & Recommendations

Issue #1	**Withdrawing to a contract which does not accept native gas tokens does not revert early**
Severity	LOW SEVERITY
Description	Bridges back into this adapter allow for the sender to set a recipient address. This recipient can for example be a smart contract address. When set to a smart contract address, the adapter will release the native gas tokens tokens to that contract. However, as is common with solidity smart contracts, they revert on gas token receipt by default, causing these bridges to be stuck forever.
Recommendation	It may be hard to address this, as alternatives such as SELFDESTRUCT force transferring the gas token may not work on all chains and is an unpopular approach. We therefore recommend to at least carefully document this limitation, and to carefully validate this automatically on the frontend (eg. a warning could be given when it is detected that the address has bytecode on the destination chain).
Resolution	ACKNOWLEDGED The client has decided not to fix this issue because the root issue here is that the funds are being sent to the wrong address. They indicate the sender is responsible for validating the destination address and ensuring that the destination address can accept the funds.

Issue #2	Receipt does not adhere to checks-effects-interactions
Severity	INFORMATIONAL
Description	Presently the compose queueing is done after the ETH has been credited. Though potentially desired, this allows for the recipient to execute code before the compose message is actually queued. Since the queueing does not create an interaction it may be interesting to reorder these lines of code.
Recommendation	Consider whether it makes sense to invert the order of actions within the receiving. Specifically, calling `sendCompose` first within `_lzReceive` in `OFTCore`. It should be noted that this comes with the distinct downside that if `lzReceive` or `_credit` is overridden with a premature interaction, this interaction could potentially maliciously execute the compose before the funds have actually arrived to the address. It may therefore be simpler to carefully investigate and document the limitations of this present ordering, without making any modifications.
Resolution	RESOLVED The client has decided not to re-order these operations because they indicate the current ordering has been tested and the commonly used OFTCore contract has been previously audited.

2.2 OFTCore

The `OFTCore` dependency contains the core logic of all OFT contracts, including the `NativeOFTAdapter` contract this audit focuses on.

Significant control is given to the `owner` of this contract to freely configure the `OFT` and its trusted messaging parameters.

2.2.1 Privileged Functions

- `setMsgInspector`
- `setPreCrime`
- `setEnforcedOptions`
- `setPeer`
- `setDelegate`
- `transferOwnership`
- `renounceOwnership`

2.2.2 Issues & Recommendations

Issue #3	**_toSD can overflow for specific high supply tokens, causing the contract to severely malfunction**
Severity	LOW SEVERITY

Description	Lines 336-338
	
	`function _toSD(uint256 _amountLD) internal view virtual returns (uint64 amountSD) {`
	`  return uint64(_amountLD / decimalConversionRate);`
	`}`
	
	The `_toSD` function uses an unsafe `uint64` cast. In case the resulting shared decimal number is larger than `uint64`, which could be particularly the case for certain memecoins with high nominal circulating supplies, the contract will malfunction. Specifically, this function will return a smaller amount than expected.

Recommendation	Consider using `SafeCast` instead. Given that this function is used during sending, this will cause sends to revert early and permit senders to adjust their input amount down.
Resolution	ACKNOWLEDGED The client has decided not to fix this as they indicate such supplies are extremely rare and that they would lose audit coverage over this very old and battle tested contract. They will keep this limitation in mind and we hope that developers reading this issue are also careful with any high supply tokens.