PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For LayerZero Mintable OFT

06 Jun 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 Mintable OFT 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 Mintable OFT
URL	https://layerzero.network/
Platform	Ethereum
Language	Solidity
Preliminary	https://github.com/LayerZero-Labs/mint-burn-oft-adapter/tree/b2ff8314d0c9ea65d8b1b69d9f5257ab9a8747b5/contracts
Resolution	https://github.com/LayerZero-Labs/mint-burn-oft-adapter/tree/b2ff8314d0c9ea65d8b1b69d9f5257ab9a8747b5/contracts

1.2 Contracts Assessed

Name	Contract	Live Code Match
MintBurnOFTAdapter		NOT DEPLOYED
MintBurnERC20Mock		NOT DEPLOYED
MintBurnOFTAdapter-Mock		NOT DEPLOYED

1.3 Findings Summary

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

ID	Severity	Summary	Status
1	INFO	Sending tokens does not necessarily adhere to checks-effects-interactions, which could lead to reentrancy issues in unlikely and non-standard token implementations	ACKNOWLEDGED
2	INFO	Typographical issues	ACKNOWLEDGED

1.3.2 MintBurnERC20Mock

No issues found.

1.3.3 MintBurnOFTAdapterMock

No issues found.

2 Findings

2.1 MintBurnOFTAdapter

The `MintBurnOFTAdapter` is a variant of the traditional OFTAdapter. The purpose of an `OFTAdapter` is to allow for an existing `ERC20` token to be bridged to multiple chains. It is essentially the locking contract which stores the existing `ERC20` on the original chain, and where this `ERC20` can subsequently be withdrawn from when people bridge back out of the OFT system.

While the traditional OFTAdapter will store all the outstanding tokens in its balance, the `MintBurnOFTAdapter` differs from this design by immediately burning them as soon as people bridge to other chains (eg. enter the OFT system by depositing the `ERC20` into the `MintBurnOFTAdapter`). When people bridge back to the main chain, they will receive the original already existing `ERC20`, but instead of it being transferred from the traditional OFTAdapter's balance, the `ERC20` gets re-minted at this stage instead.

This means that with the `MintBurnOFTAdapter`, the `totalSupply` at the source chain actually decreases meaning the sum of all chains their `totalSupply` represents the actual total supply. With the traditional OFTAdapter, the `totalSupply` of the origin chain represented the actual total supply.

The `MintBurnOFTAdapter` contains simple logic extending the larger OFTCore contract, which is an `OFT` and `OApp` itself. It should be noted that a full review of the OFTCore contract is out-of-scope for this assessment. This however means that the `MintBurnOFTAdapter` adheres to the `OFT` and `OApp` specifications as well.

As this adapter directly mints and burns the underlying `ERC20`, it has the benefit that no approvals need to be given to bridge in- and out of this origin chain. This generally improves UX significantly.

The `MintBurnOFTAdapter` does not support underlying tokens with less than 6 decimals, similar to the traditional OFTAdapter.

Tjhe `MintBurnOFTAdapter` requires explicit `mint` and `burn` privileges on the underlying contract. If such privileges are not granted, or are revoked, the contract will malfunction. Furthermore, it burns using the `minterBurner.burn(_from, amountSentLD)` signature, which means that the token must expose such a `burn` function where the

`account` can be selected, and not (just) a `burn(amount)` function which burns from the caller's balance.

2.1.1 Privileged Functions

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

2.1.2 Issues & Recommendations

Issue #1	**Sending tokens does not necessarily adhere to checks-effects-interactions, which could lead to reentrancy issues in unlikely and non-standard token implementations**
Severity	INFORMATIONAL
Description	The `MintBurnOFTAdapter` overrides the `_debit` function with a `burn` call to the configured burnable underlying token (or a third-party contract which can burn the underlying token). However, this `burn` call occurs in the middle of the `send` flow, meaning that it occurs after `_debitView` but before the sending is finished. If `burn` in an unlikely scenario contains a reentrancy vector (we don't know of any tokens which have this, hence why this issue is informational), the send functions themselves no longer adhere to checks-effects-interactions. By default, even if this is the case, this does not appear exploitable. However, if for some reason, `_debitView` is overridden within a token extending `MintBurnOFTAdapter` with properties such as rate limit checks or a variable fee based on send volume properties, the behavior of this additional functionality will likely be exploitable through reentrancy. Since this scenario has so many requirements that are each unlikely to happen, the issue is provided purely informationally. It should finally be noted that this behavior exists within all `OFTAdapter` implementations, and is not limited to the `MintBurnOFTAdapter`.
Recommendation	Consider documenting this behavior or perhaps adding a reentrancy guard to the send functions. Note that read-only reentrancy would still be of some concern even with a guard. If desired, the issue could be fully resolved by fully redesigning the send flow into checks-effects-interactions. However, this may have its own downsides, as certain OFTs might not be able to adhere to checks-effects-interactions due to their `_debit` function requiring interactions to figure out the `amountReceivedLD`. Thus this resolution has its own downsides. That limitation is evident in the following comment within the codebase: Line 58 `* a pre/post balance check will need to be done to calculate the amountReceivedLD.` Such a before-after check would be exploitable if the inner code allowed reentrancy, indicating that this issue is actually relevant, but not necessarily

	fixable with checks-effects-interactions, as a before-after pattern can never be written in checks-effects-interactions. A reentrancy guard would then be the only solution. For gas purposes, it may make sense to only have that guard be introduced for tokens which actually have reentrancy, and to limit the resolution of this issue to documenting this requirement.
Resolution	● ACKNOWLEDGED

Issue #2	**Typographical issues**
Severity	● INFORMATIONAL

Description	Lines 6 and 13
	
	`import { SafeERC20 } from "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";`
	`using SafeERC20 for IERC20;`
	
	This import appears unused.
	
	Line 31 and to some extend other comments
	
	`* @dev In the case of OFTAdapter, address(this) and erc20 are NOT the same contract.`
	
	These comments refer to `OFTAdapter`, a contract not visible within the scope of this repository. It may make sense to document the location of this contract at the top of the contract, as the source this contract is derived from, or to rename it within these comments to `MintBurnOFTAdapter`.

Recommendation	Consider fixing the typographical issues.
Resolution	● ACKNOWLEDGED

2.2 MintBurnERC20Mock

The `MintBurnERC20Mock` is a testing utility. **It should not be deployed as a real contract** as `mint` and `burn` are not protected. No issues were found in its context as a unit-testing contract. In case this contract was deployed as a real production token, it would have the critical vulnerability where anyone could mint tokens out of thin air, without any authorization requirements.

2.2.1 Privileged Functions

None.

2.2.2 Issues & Recommendations

No issues found.

2.3 MintBurnOFTAdapterMock

The `MintBurnOFTAdapterMock` is a simple example implementation of the `MintBurnOFTAdapter` contract, without any modifications to the default behavior. It's purpose is to serve as a testing utility.

2.3.1 Privileged Functions

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

2.3.2 Issues & Recommendations

No issues found.