OpenZeppelin | security

The Sandbox SAND-LayerZero Integration Audit

June 27, 2024

Table of Contents

Table of Contents 2

Summary 3

Scope 4

System Overview 5

Security Model and Trust Assumptions 6

Medium Severity 7

M-01 Lack of ERC-2771 Compatibility 7

Low Severity 8

L-01 Missing Docstrings 8

Notes & Additional Information 8

N-01 Function Visibility Overly Permissive 8

Conclusion 9

Summary

Type	DeFi
Timeline	From 2024-05-27 To 2024-05-31
Languages	Solidity

Total Issues	3 (3 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	1 (1 resolved)
Low Severity Issues	1 (1 resolved)
Notes & Additional Information	1 (1 resolved)

Scope

We audited the thesandboxgame/sandbox-smart-contracts repository at commit efa7a8b which introduces the integration of LayerZero OFT features with the SAND token. The SAND token contracts were previously audited at commit 45116ed. For the previously audited contracts, the scope of this audit was limited to the difference in the codebase between these commits and it was concluded that no changes had been made to the codebase.

In scope were the following files:

```
packages/oft-sand/contracts
├── OFTAdapterForSand.sol
├── OFTSand.sol
├── interfaces
│   └── IERC20Extended.sol
├── libraries
│   └── BytesUtil.sol
└── sand
    ├── ERC20BaseToken.sol
    ├── ERC20BasicApproveExtension.sol
    ├── ERC20Internal.sol
    ├── ERC2771Handler.sol
    ├── SandBaseToken.sol
    ├── WithAdmin.sol
    └── WithSuperOperators.sol
```

System Overview

The SAND token is an ERC-20-compatible token used within The Sandbox ecosystem. It serves as the basis for transactions and interactions within the platform. It is currently deployed on the Ethereum and Polygon networks, and is bridged between these networks via the Polygon POS-bridge. With the introduction of `OFTSand`, The Sandbox team wants to extend the cross-chain capabilities of the SAND token to the Base and BSC networks by integrating it with LayerZero.

The `OFTSand` contract extends from the previously audited `SandBaseToken` contract and LayerZero's `OFTCore` contract, and overrides the `_debit` and `_credit` functions for the burning and minting of SAND tokens, respectively. While the `OFTSand` contract will be deployed to each network that The Sandbox intends to support, LayerZero is integrated with the SAND token on the Ethereum mainnet via `OFTAdapter`. This is represented by the `OFTAdapterForSand` contract.

It is essential to note that The Sandbox team will continue to use the Polygon POS-bridge for bridging SAND to and from Polygon. Hence, the `OFTSand` contract will not be deployed on the Polygon network. To bridge from Polygon to another chain (such as Base), the Ethereum mainnet will be used as the intermediary.

Special consideration needs to be given to possible interactions between different bridge architecture providers, when they are being used for the same token. In this case, the SAND token is being bridged from Mainnet to Polygon using Polygon's POS-bridge, and from Mainnet to other networks (excluding Polygon) via LayerZero. However, this specific setup does not introduce additional attack surface, as both the Polygon POS-bridge, as well as the `OFTAdapterForSand` contract take custody of SAND tokens before emitting events that lead to further minting of tokens on other chains. Therefore, any form of double spending is mitigated.

During the audit, we looked at the Rate Limitation feature in the LayerZero `OApp` module, which helps prevent very high volumes of traffic. It is not configured by default and we concluded that The Sandbox team does not need to configure it.

Security Model and Trust Assumptions

In addition to those of the previous audit, the trust assumptions for this audit are:

- The `layerZeroEndpoint` is set correctly within the constructor of the `OFTAdapterForSand` and `OFTSand` contracts.
- The `owner` of the `OFTAdapterForSand` contract is the delegate of LayerZero `OApp` on the Ethereum mainnet. This address is assumed to be correct and trusted. As a delegate, this address can make any configuration changes in LayerZero's integration.
- Similarly, the `owner` of the `OFTSand` contract is the delegate of LayerZero `OApp` on the respective chain. This address is assumed to be correct and trusted.
- The peer has been set to a correct destination address and the `setPeer` function is called only after all the application configuration changes are done. Incorrectly setting up the peer can result in loss of funds during bridging.
- Thorough documentation is needed to make the users aware that in The Sandbox ecosystem, bridging to Polygon is not supported via LayerZero. It is assumed that `OFTSand` will not be deployed on the Polygon network and only POS-bridge will be used.

Medium Severity

M-01 Lack of ERC-2771 Compatibility

The SAND token is designed to be compatible with the ERC-2771 standard, integrating with the Biconomy trusted forwarder for gasless transactions. For `OFTSand` and `OFTAdapterForSand` to be compliant with ERC-2771, instances of `msg.sender` in the code need to be replaced with the `_msgSender()` function of the `ERC2771Handler` contract. Otherwise, trusted forwarders will not be able to initiate bridging via meta-transactions.

In order to send tokens across LayerZero, the `send` function of `OFTCore` needs to be called. As both contracts mentioned above inherit from `OFTCore` at version `2.3.8` which contains `msg.sender` as the hard coded `_from` parameter of the `_debit` function call at line 182, the `_debit` function will try to transfer in tokens from `msg.sender` directly. In the case of meta-transactions, `msg.sender` will be a trusted forwarder instead of the original sender, in which case the transaction will fail, as the trusted forwarder is not the custodian of the tokens. In the case of `OFTSand`, the overridden `_debit` function of `OFTSand` itself is called, whereas in the case of `OFTAdapterForSand`, the `_debit` function within `OFTAdapter` at line 74 is called.

In order to make both contracts compliant with ERC-2771, consider copying the relevant contracts into the `contracts` folder instead of importing them as a dependency, and making the necessary changes (specifically, using `_msgSender()` in the `OFTCore` contract). In addition, consider implementing thorough testing of meta-transaction compatibility before deploying to production.

Update: *Resolved in pull request #1524 at commit 5a42022. The Sandbox team stated:*

> *We added `OFTCore` to the contract folder and made the necessary changes.*

Low Severity

L-01 Missing Docstrings

Within `OFTSand.sol`, there are multiple code instances that do not have docstrings:

- The `OFTSand` contract
- The `setTrustedForwarder` function

Consider thoroughly documenting all functions (and their parameters) that are part of any contract's public API. Functions implementing sensitive functionality, even if not public, should be clearly documented as well. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

Update: *Resolved in pull request #1526 at commit a6ac40e.*

Notes & Additional Information

N-01 Function Visibility Overly Permissive

Within `OFTSand.sol`, the `token` function with `public` visibility could be limited to `external`.

To better convey the intended use of functions and to potentially realize some additional gas savings, consider changing a function's visibility to be only as permissive as required.

Update: *Resolved in pull request #1527 at commit 49f4129.*

Conclusion

This audit covered changes which introduce the integration of LayerZero OFT features with the SAND token. One medium-severity issue was found and some fixes were suggested to improve the clarity of the codebase.

While interacting with multiple blockchain networks, it is important to be aware of their own security assumptions. During this audit, we analyzed the risks of integrating SAND with both POS-bridge and LayerZero and concluded that as long as `OFTSand` is not deployed on Polygon, the individual cross-chain infrastructure would work fine independently.

The Sandbox team has been responsive and engaged in discussions with the auditors during the audit period.