OpenZeppelin | security

Morpheus MOR OFT Token Audit

April 25, 2024

Table of Contents

Summary

Type	DeFi
Timeline	From 2024-03-27 To 2024-03-27
Languages	Solidity

Total Issues	4 (4 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	2 (2 resolved)
Notes & Additional Information	2 (2 resolved)
Client Reported Issues	0 (0 resolved)

Scope

We audited the MorpheusAIs/SmartContracts repository at the `5e1222e` commit.

Only the `MOROFT.sol` was in the scope of this audit.

System Overview

The `MOROFT` contract is an ERC-20 token. It inherits from the LayerZero's `OFT` contract, which allows the token to bridge across different networks.

The `MOROFT` contract defines a `mint` function which can only be called by the `minter_address` to mint tokens.

The contract also allows users to burn their tokens or burn tokens from approved address.

Security Model and Trust Assumptions

Instead of importing the `LayerZero-v2` contracts as dependencies, the Morpheus' `SmartContracts` repository has a `@layerzerolabs` folder where the OFT and OAPP contracts are stored. At the time of this audit, the `OFT` contract present in the `SmartContracts` repository has small changes compared to the `OFT` contract present in the `LayerZero-v2` contracts repository. For instance, the internal `_debit` function in the Morpheus' version accepts `_from` as an input parameter and burns tokens from this address, however, in LayerZero's version, tokens are burned from `msg.sender`. Although, the `MOROFT` contract inherits from `OFT.sol`, it does not call the `OFT` contract's internal `_credit` and `_debit` functions for minting and burning of tokens.

The Morpheus team decided to copy the contracts to the `@layerzerolabs` folder instead of importing the dependency because the `ethers-v5` dependency in the `LayerZero-v2` conflicts with the `@typechain/ethers-v6` dependency of Morpheus' repository.

The audit of Morpheus' `@layerzerolabs` folder and of `LayerZero-v2` contracts was out of scope of this audit. It is assumed that these contracts work as intended.

Privileged Roles

The `MOROFT` contract inherits the role of `owner` via the chain of inheritance from the `OAppCore` contract. This privileged role is transferred to the `_delegate` address provided as an input to the constructor.

This `_delegate` address is also the delegate for the LayerZero's OApp endpoint.

Additionally, the contract allows only the `minter_` address to mint the tokens. This `minter_` is initialised in the constructor.

Low Severity

L-01 Lack of Input Validation

The `constructor` of the `MOROFT` contract does not verify if the values for the `_delegate` and `_minter` input addresses are non-zero. While the `_delegate` address is validated for zero address in the `OAppCore` contract, the `_minter` addresses are never validated.

To prevent initializing these addresses to zero, consider adding proper checks.

Update: *Resolved in pull request #28 at commit 8242004.*

L-02 Incorrect Docstring

To improve the readability of the codebase, consider correcting the docstring above the `IMOROFT` interface, which states that the token is capped, however, this is not reflected in the implementation of the `MOROFT` contract.

Update: *Resolved in pull request #28 at commit 95b4217.*

Notes & Additional Information

N-01 Missing Docstring

The `MOROFT` contract does not have a contract definition and none of its functions have docstrings.

Consider thoroughly documenting all functions (and their parameters) that are part of any contract's public API. Since the docstrings are present in the `IMOROFT` contract, this can be achieved by using the `@inheritdoc` tag as mentioned in the Ethereum Natural Specification Format (NatSpec).

Additionally, the `MOROFT` contract allows only the `minter_` address to mint the tokens. Consider documenting this in the docstring above the `mint` function.

Update: *Resolved in pull request #32 at commit 52c69c1.*

N-02 Lack of Security Contact

Providing a specific security contact (such as an email or ENS name) within a smart contract significantly simplifies the process for individuals to communicate if they identify a vulnerability in the code. This practice proves beneficial as it permits the code owners to dictate the communication channel for vulnerability disclosure, eliminating the risk of miscommunication or failure to report due to lack of knowledge on how to do so. Additionally, if the contract incorporates third-party libraries and a bug surfaces in these, it becomes easier for the maintainers of those libraries to make contact with the appropriate person about the problem and provide mitigation instructions.

The `MOROFT` contract does not have a security contact.

Consider adding a NatSpec comment containing a security contact on top of the contract definition. Using the `@custom:security-contact` convention is recommended as it has been adopted by the OpenZeppelin Wizard and the ethereum-lists.

Update: *Resolved in pull request #29 at commit 8d8e771.*

Conclusion

No high-severity issues were detected in the contract. However, some suggestions have been made to enhance the codebase's readability.