BAILSEC.IO

OFFICE@BAILSEC.IO

X: @BAILSECURITY

TG: @HELLOATBAILSEC

FINAL REPORT:

TelosOFTV1.2 - Update Audit

(differential)

May 2024

Disclaimer:

Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

The content of this assessment is not an investment. The information provided in this report is for general informational purposes only and is not intended as investment, legal, financial, regulatory, or tax advice. The report is based on a limited review of the materials and documentation provided at the time of the audit, and the audit results may not be complete or identify all possible vulnerabilities or issues. The audit is provided on an "as-is," "where-is," and "as-available" basis, and the use of blockchain technology is subject to unknown risks and flaws.

The audit does not constitute an endorsement of any particular project or team, and we make no warranties, expressed or implied, regarding the accuracy, reliability, completeness, or availability of the report, its content, or any associated services or products. We disclaim all warranties, including the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We assume no responsibility for any product or service advertised or offered by a third party through the report, any open-source or third-party software, code, libraries, materials, or information linked to, called by, referenced by, or accessible through the report, its content, and the related services and products. We will not be liable for any loss or damages incurred as a result of the use or reliance on the audit report or the smart contract.

The contract owner is responsible for making their own decisions based on the audit report and should seek additional professional advice if needed. The audit firm or individual assumes no liability for any loss or damages incurred as a result of the use or reliance on the audit report or the smart contract. The contract owner agrees to indemnify and hold harmless the audit firm or individual from any and all claims, damages, expenses, or liabilities arising from the use or reliance on the audit report or the smart contract.

By engaging in a smart contract audit, the contract owner acknowledges and agrees to the terms of this disclaimer.

1. Project Details

Important:
Please ensure that the deployed contract matches the source-code of the last commit hash.

Project	TelosOFTV2 - Update Audit (differential)
Website	telos.net
Type	OFT
Language	Solidity
Methods	Manual Analysis
Github repository	https://github.com/telosnetwork/TelosOFT/tree/f33925b1d2d19b84913bfe36621f29a0311acf38/contracts
Resolution 1	https://github.com/telosnetwork/TelosOFT/tree/c930f7121ffacab294eaa91c3c0980ea21a5a315

2. Detection Overview

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	1	1		
Medium	3	3		
Low				
Informational	3	3		
Governance				
Total	7	7		

2.1 Detection Definitions

Severity	Description
High	The problem poses a significant threat to the confidentiality of a considerable number of users' sensitive data. It also has the potential to cause severe damage to the client's reputation or result in substantial financial losses for both the client and the affected users.
Medium	While medium level vulnerabilities may not be easy to exploit, they can still have a major impact on the execution of a smart contract. For instance, they may allow public access to critical functions, which could lead to serious consequences.
Low	Poses a very low-level risk to the project or users. Nevertheless the issue should be fixed immediately
Informational	Effects are small and do not post an immediate danger to the project or users
Governance	Governance privileges which can directly result in a loss of funds or other potential undesired behavior

3. Detection

Bailsec was tasked with a differential audit of TelosOFTV2:

Old Commit:

https://github.com/LayerZero-Labs/solidity-examples/tree/c04e7d211b1b610f84761df943e6a38b0a53d304/contracts

New Commit:

https://github.com/telosnetwork/TelosOFT/tree/f33925b1d2d19b84913bfe36621f29a0311acf38/contracts

Files in Scope:

1. contracts/token/upgrades/ProxyOFTUpgrade.so
2. contracts/token/upgrades/TelosNativeOFTUpgrade.sol
3. contracts/token/upgrades/TelosOFTUpgrade.sol
4. contracts/lzApp/LzApp.sol
5. contracts/lzApp/NonblockingLzApp.sol
6. contracts/token/BaseOFTV2.sol
7. contracts/token/OFTCoreV2.sol
8. contracts/token/OFTV2.sol
9. contracts/token/ProxyOFTV2.sol
10. contracts/token/TelosNativeOFTV2.sol
11. contracts/token/TelosOFT.sol

Primary Update Overview:
The key update in this differential audit involves the implementation of upgradeability for all important base contracts and their dependencies. Moreover, three new contracts were developed, namely:

a) ProxyOFTUpgrade
b) TelosNativeOFTUpgrade
c) TelosOFTUpgrade

Security Considerations: The main change in the architecture is the implementation of upgradeability, using the UUPSUpgradeable pattern. This will expose the following risk

a) Incorrect initialization: If the new contracts do not properly initialize all necessary variables, this will result in issues. Furthermore, all dependencies must be initialized.

b) Incorrect storage layout for dependencies: The TelosOFTV2 architecture is heavily dependent on inheritance. This means that the OFTV2 contract inherits multiple other contracts. All inherited contracts should correctly expose a gap[] variable, ensuring smooth future upgrades.

c) Upgrade issues: If a previous implementation is already existent, it is important that the storage layout is not violated. This is not the case here.

d) Correctness of upgrade privilege: Since the UUPSUpgradeable standard is chosen for upgradeability, the _authorizeUpgrade function must be correctly overridden, allowing only a privileged address to invoke the upgrade.

Disclaimer: This audit involves only the changes provided by the corresponding diffchecker files. Please be advised that for issues which are reported outside of the diffchecker scope, an additional resolution must be scheduled. A differential audit is always a constrained task because not the full codebase is re-audited. This will have inherent consequences if intrusive changes have side-effects on parts of a codebase/module, which is not part of the audit scope.

Libraries

BytesLib

The BytesLib library is a simple helper contract which allows for manipulating bytes data types.

No change detected

ExcessivelySafeCall

The ExcessivelySafeCall library is a simple helper contract that executes low-level calls in a safe manner.

No change detected.

LZApp

LzLib

The LzLib is the main LayerZero library which handles the encoding and decoding of different parameters and type conversions.

No change detected.

LzApp

The LzApp contract is responsible for different parameter settings via governance, gas checks and the send/receive logic.

It mainly handles incoming cross-chain transfers from the LzEndpoint and outgoing cross-chain transfers towards the LzEndpoint and specific validations such as trustedRemote checkup, general parameter validations for cross-chain transfers and gas checks.

The following changes were detected (https://diffchecker.com/55NwKJ2W):

a) Ownable dependency was changed to OwnableUpgradeable.
b) UUPSUpgradeable has been inherited.
c) Constructor has been changed to __LzApp_init and immutable lzEndpoint was changed to mutable lzEndpoint
d) gap[] has been introduced at the end of the storage layout
e) authorizeUpgrade was incorporated, ensuring no unprivileged upgrades can happen.

Issue_01	UUPSUpgradeable is not initialized
Severity	**Informational**
Description	The contract inherits the UUPSUpgradeable contract as the used proxy implementation. This contract is however not initialized. It is important to mention that this contract is used in 4 different deployed versions: a) OFTV2 b) ProxyOFTV2 c) TelosNativeOFT d) TelosOFT Initialization is therefore not mandatory to happen within this contract but can also happen within the other contracts. However, it only happens for scenario d) TelosOFT. In all other deployments, UUPSUpgradeable stays uninitialized.

Recommendations	Due to the fact that the initializer does not expose any logic: https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/master/contracts/proxy/utils/UUPSUpgradeable.sol#L65 there is no risk in it not being initialized. However, if it is desired to fix this issue, we recommend initializing it within the LzApp contract and remove the init call within the TelosOFT contract.
Comments / Resolution	Resolved.

NonblockingLzApp

The NonBlockingLzApp inherits the LzApp and exposes functionality to make messages “non-blocking”.

If a message reverts, it will not block the channel but will rather be stored into a failedMessages mapping where it can then be resent again.

The following changes were detected (https://www.diffchecker.com/10dYT4oW/):

a) Constructor has been changed to __nonblockingLzApp_init and LzApp is initialized
b) gap[] has been introduced at the end of the storage layout

No issues found.

Token / Parent Contracts

OFTCoreV2

The OFTCoreV2 contract inherits the NonblockingLzApp/LzApp and exposes functionality for fee estimation and the core logic for cross-chain receipts and transfers.

The following changes were detected (https://www.diffchecker.com/BGBe8ldY/):

a) Constructor has been changed to __OFTCoreV2_init and NonblockingLzApp is initialized
b) Immutable keyword has been removed from sharedDecimals variable.
c) gap has been introduced at the end of the storage layout

No issues found.

BaseOFTV2

The BaseOFTV2 contract inherits OFTCoreV2/NonblockingLzApp/LzApp and orchestrates the entry points for outgoing cross-chain transfers. Moreover it exposes view-only functions for native fee calculation.

The following changes were detected (https://www.diffchecker.com/GTHP3DWN/):

a) ERC165Upgradeable was imported instead of ERC16 and supportsInterface override was adjusted
b) Constructor has been changed to __BaseOFTV2_init and OFTCoreV2 is initialized
c) gap has been introduced at the end of the storage layout

Issue_02	ERC165Upgradeable is not initialized
Severity	**Informational**
Description	The contract imports the ERC165Upgradeable contract but does not initialize it. This will not expose any harm but is against best-practices.
Recommendations	Consider initializing the ERC165Upgradeable contract.
Comments / Resolution	Resolved.

Token / Base Contracts

The following contracts are meant to be the Base contracts for deployments, they will inherit all dependencies and parent contracts from above.

OFTV2

The OFTV2 contract is the standard OFTV2 implementation which inherits the following contracts and dependencies

a) ERC20Upgradeable
b) BaseOFTV2
c) ERC165Upgradeable
d) OFTCoreV2
e) NonblockingLzApp
f) LzApp
g) UUPSUpgradeable
h) OwnableUpgradeable

It exposes the essential debit and credit functions that are invoked upon incoming or outgoing cross-chain transfers and handles the minting/burning tokens to/from users.

The following changes were detected (https://www.diffchecker.com/pGJgyV12/):

a) ERC20Upgradeable is imported and inherited instead of ERC20
b) Constructor has been changed to __OFTV2_init, BaseOFTV2 and ERC20Upgradeable is initialized
c) gap[] has been introduced at the end of the storage layout
d) Immutable keyword has been removed from ld2sdRate variable.

Issue_03	OFTV2 contract is not deployable as stand-alone contract
Severity	**High**
Description	Generally speaking, the OFTV2 contract is similar to the TelosOFT contract in their implementation. In the standard L0 OFT architecture, this contract is also deployed as a standalone contract. This will not work due to the initialize implementation being marked as private and onlyInitializing. There is essentially no entry point to initialize this contract.
Recommendations	Consider if Telos intends to use this contract as base contract or only as abstract contract for the TelosNativeOFT. In the latter scenario, this issue can be savely marked as resolved with this comment. If however it is desired to deploy this as a standalone contract, the initialize function must be publicly callable and must expose the initializer modifier.
Comments / Resolution	Resolved: An additional public initialize function has been exposed. While this works, it is important to mention that this contract now serves two distinct purposes: a) Used as parent contract b) Used as deployed base

	We recommend to create one distinct contract for each purpose: a) OFTV2: This contract should only expose __OFTV2_init and not the initialize function. This is used as parent contract only. b) OFTV2Base: This contract should only expose initialize and not __OFTV2_init. This is used as deployed base.

ProxyOFTV2

The ProxyOFTV2 contract is the OFT implementation for the non standard OFT token. In this context, the ERC20 token is abstracted and not part of the core OFT structure, but externally invoked.

Instead of minting and burning the token, just normal transfers are executed upon cross-chain interactions. From the OFT contract to the recipient or vice-versa.

The following contracts and dependencies are inherited:

a) BaseOFTV2
b) ERC165Upgradeable
c) OFTCoreV2
d) NonblockingLzApp
e) LzApp
f) UUPSUpgradeable
g) OwnableUpgradeable

The following changes were detected (https://www.diffchecker.com/6X4TdgZ0/):

a) SafeERC20Upgradeable is imported and inherited instead of SafeERC20
b) Immutable keyword from storage variables has been removed
c) Constructor has been changed to initialize and BaseOFTV2 is initialized

Issue_04	Lack of gap[] variable
Severity	**Medium**
Description	The contract is inherited by the ProxyOFTUpgrade contract and therefore also exposes use-case as parent contract. All parent contracts in proxy implementations should expose a gap[] variable to ensure smooth upgrades without potential storage collisions in the scenario if the storage layout is altered.
Recommendations	Consider implementing a gap[] variable.
Comments / Resolution	Resolved.

TelosNativeOFT

The TelosNativeOFT contract is the OFT implementation for the native Telos token. Additional to the OFT structure, it implements a wrapper functionality, allowing users to wrap their native Telos token to the TelosNativeOFT token, which can then be used for cross-chain transfers.
Furthermore, this contract allows the direct initiation of outgoing cross-chain transfers by providing the corresponding msg.value, which removes the need to wrap the native token beforehand and does this atomically during the send/sendAndCall functions.

The following contracts and dependencies are inherited:

a) ReentrancyGuardUpgradeable
b) OFTV2
c) ERC20Upgradeable
d) BaseOFTV2
e) ERC165Upgradeable
f) OFTCoreV2

g) NonblockingLzApp
h) LzApp
i) UUPSUpgradeable
j) OwnableUpgradeable

The following changes were detected (https://www.diffchecker.com/GJphL7Ss/):

a) The ReentrancyGuard was changed to ReentrancyGuardUpgradeable
b) The constructor was changed to initialize, the ReentrancyGuardUpgradeable and OFTV2 contracts are initialized
c) Custom errors were renamed from NativeOFTV2 to TelosOFT

Issue_05	Lack of gap[] variable
Severity	**Medium**
Description	The contract is inherited by the TelosNativeOFTUpgrade contract and therefore also exposes use-case as parent contract. All parent contracts in proxy implementations should expose a gap[] variable to ensure smooth upgrades without potential storage collisions in the scenario if the storage layout is altered.
Recommendations	Consider implementing a gap[] variable.
Comments / Resolution	Resolved.

TelosOFT

The TelosOFT contract is the standard OFT implementation, similar to the OFTV2 contract but with changed variable names.

The following contracts and dependencies are inherited:

a) ERC20Upgradeable
b) BaseOFTV2
c) ERC165Upgradeable
d) OFTCoreV2
e) NonblockingLzApp
f) LzApp
g) UUPSUpgradeable
h) OwnableUpgradeable

The following changes were detected (https://www.diffchecker.com/ZSUjh5z7/):

a) ERC20Upgradeable is imported and inherited instead of ERC20
b) Constructor has been changed to initialize and BaseOFTV2, ERC20Upgradeable and UUPSUpgradeable is initialized
c) Immutable keyword has been removed from ld2sdRate variable.

Issue_06	Lack of gap variable
Severity	**Medium**
Description	The contract is inherited by the TelosOFTUpgrade contract and therefore also exposes use-case as parent contract. All parent contracts in proxy implementations should expose a gap variable to ensure smooth upgrades without potential storage collisions in the scenario if the storage layout is altered.
Recommendations	Consider implementing a gap variable.

Comments / Resolution	Resolved.

Issue_07	Redundant import of ERC20
Severity	**Informational**
Description	The ERC20 contract is not used anymore but still imported. This can confuse third-party reviewers.
Recommendations	Consider removing it.
Comments / Resolution	Resolved.

Upgrades

ProxyOFTUpgrade

The ProxyOFTUpgrade contract is a simple extension for the ProxyOFTV2 contract, exposing reinitializing logic. If reinitialized, the newProperty variable is set to true and _initialized is set to 2, indicating that an upgrade to version 2 has happened.

No issues found.

TelosNativeOFTUpgrade

The TelosNativeOFTUpgrade contract is a simple extension for the TelosNativeOFTUpgrade contract, exposing reinitializing logic. If reinitialized, the newProperty variable is set to true and _initialized is set to 2 indicating that an upgrade to version 2 has happened.

No issues found

TelosOFTUpgrade

The TelosOFTUpgrade contract is a simple extension for the TelosOFTUpgrade contract, exposing reinitializing logic. If reinitialized, the newProperty variable is set to true and _initialized is set to 2, indicating that an upgrade to version 2 has happened.

No issues found.