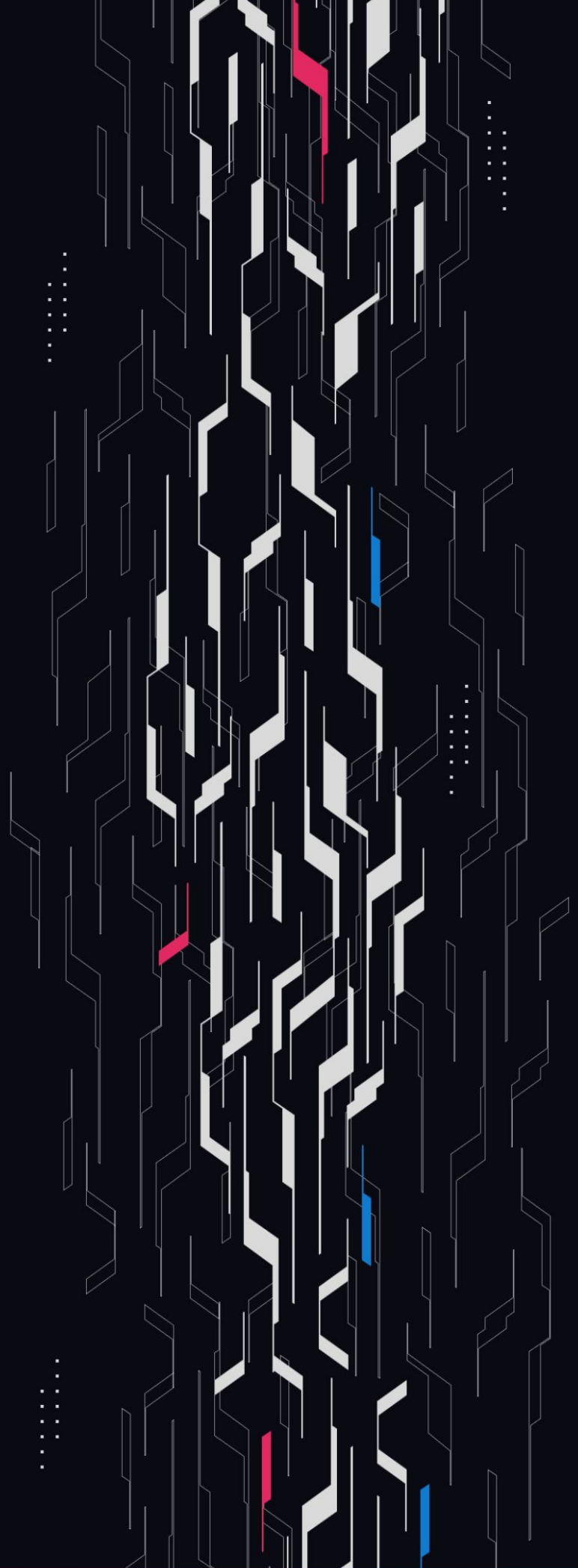


GA GUARDIAN

Orderly OFT

Security Assessment

July 1st, 2024



Summary

Audit Firm Guardian

Prepared By Daniel Gelfand, Zdravko, Wafflemakr, Osman Ozdemir, Giraffe, Michael Lett

Client Firm Orderly

Final Report Date July 1st, 2024

Audit Summary

Orderly engaged Guardian to review the security of its OFT and OFT adapter. From June 3rd to June 17th, a team of 6 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Notice that the examined smart contracts are not resistant to internal exploit. For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

✓ Verify the authenticity of this report on Guardian's GitHub: <https://github.com/guardianaudits>

📊 Code coverage & PoC test suite: <https://github.com/GuardianAudits/oft-fuzzing>

Table of Contents

Project Information

Project Overview 4

Audit Scope & Methodology 5

Smart Contract Risk Assessment

Invariants Assessed 7

Findings & Resolutions 9

Addendum

Disclaimer 16

About Guardian Audits 17

Project Overview

Project Summary

Project Name	OFT
Language	Solidity
Codebase	https://gitlab.com/orderlynetwork/orderly-v2/oft-token
Commit(s)	Initial Commit: e8b7aad0b1e812be934e5c180f2a2569b044bb37 Final Commit: 08dccdd381a5a922c39c7d3b8774b7dfd3142b0a

Audit Summary

Delivery Date	July 1, 2024
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● High	2	0	0	0	0	2
● Medium	2	0	0	0	0	2
● Low	2	0	0	1	0	1

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: <i>High</i>	Impact: <i>Medium</i>	Impact: <i>Low</i>
Likelihood: <i>High</i>	● Critical	● High	● Medium
Likelihood: <i>Medium</i>	● High	● Medium	● Low
Likelihood: <i>Low</i>	● Medium	● Low	● Low

Impact

- High**

Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.
- Medium**

A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.
- Low**

Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

- High**

The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.
- Medium**

An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.
- Low**

Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.
Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.















Invariants Assessed

During Guardian’s review of the OFT contracts, fuzz-testing with [Foundry](#) was performed on the protocol’s main functions. Given the dynamic interactions and the potential for unforeseen edge cases in the protocol, fuzz-testing was imperative to verify the integrity of several system invariants.

Throughout the engagement the following invariants were assessed for a total of 1,000,000+ runs up to a depth of 500 with a prepared Foundry fuzzing suite.

ID	Description	Tested	Passed	Remediation	Run Count
<u>OT-01</u>	Total Supply of ORDER should always be 1,000,000,000	✓	✓	✓	1,000,000+
<u>OT-02</u>	Allowance Matches Approved Amount	✓	✓	✓	1,000,000+
<u>OT-03</u>	ERC20 Balance Changes By Amount For Sender And Receiver Upon Transfer	✓	✓	✓	1,000,000+
<u>OT-04</u>	ERC20 Balance Remains The Same Upon Self-Transfer	✓	✓	✓	1,000,000+
<u>OT-05</u>	ERC20 Total Supply Remains The Same Upon Transfer	✓	✓	✓	1,000,000+
<u>OT-06</u>	Source Token Balance Should Decrease On Send	✓	✓	✓	1,000,000+
<u>OT-07</u>	Adapter Token Balance Should Increase On Send	✓	✓	✓	1,000,000+
<u>OT-08</u>	Adapter Token Total Supply Should Not Change On Send	✓	✓	✓	1,000,000+
<u>OT-09</u>	Source OFT Total Supply Should Decrease On Send	✓	✓	✓	1,000,000+
<u>OT-10</u>	Outbound Nonce Should Increase By 1 On Send	✓	✓	✓	1,000,000+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
<u>QT-11</u>	Max Received Nonce Should Increase By 1 on IzReceive			N/A	1,000,000+
<u>QT-12</u>	Destination Token Balance Should Increase on IzReceive				1,000,000+
<u>QT-13</u>	Adapter Token Balance Should Decrease on IzReceive				1,000,000+
<u>QT-14</u>	Adapter Token Total Supply Should Not Change on IzReceive				1,000,000+
<u>QT-15</u>	Destination OFT Total Supply Should Increase on IzReceive				1,000,000+

Findings & Resolutions

ID	Title	Category	Severity	Status
H-01	Layerzero Pathway will be Continuously Blocked	DoS	● High	Resolved
H-02	_acceptNonce Should Not Be Called	Logical Error	● High	Resolved
M-01	No Storage Gaps In Upgradable Contracts	Upgradability	● Medium	Resolved
M-02	Ordered Nonce Flag Should Only Be Set Once	Logical Error	● Medium	Resolved
L-01	Implementation Contracts Can Be Initialized By Anyone	Upgradability	● Low	Resolved
L-02	Owner Can Revoke Ownership	Logical Error	● Low	Acknowledged

H-01 | Layerzero Pathway will be Continuously Blocked

Category	Severity	Location	Status
DoS	● High	OFT.sol	Resolved

Description

Currently, the OFT contract allows anyone to set the address(0) as a token recipient on the destination chain. However, this is a transaction that will always revert because the OpenZeppelin's implementation does not allow minting to the 0 address. An attacker can leverage this to block the LayerZero pathway.

An example:

- The nonce on both chains is 5.
 - An attacker successfully sends their minting tx and the nonce becomes 6.
 - The transaction is received on the destination chain, but fails.
 - Since the app uses ordered delivery, all subsequent transactions will be blocked until 6 is resolved.
- As a result, an attacker can keep blocking the LayerZero pathway.

Recommendation

Don't allow sending cross-chain packages that mint to the address(0).

Resolution

Orderly Team: The issue was resolved in commit [c2e1991](#).

H-02 | `_acceptNonce` Should Not Be Called

Category	Severity	Location	Status
Logical Error	● High	Global	Resolved

Description

`OrderOFT` and `OrderAdapter` allow the owner to burn, nilify, skip and clear nonces. When doing so, a call to `_acceptNonce` is initiated.

This is problematic for:

- burning: The nonce to be burnt will always be less than the `maxReceivedNonce`. However, in ordered delivery the `_acceptNonce` function reverts if the specified nonce is not `max + 1`. This will result in burning not working when ordered delivery is turned on.
- nilifying: The `_acceptNonce` will let us nilify only the packet with `nonce = max + 1`. After nilifying the `maxReceivedNonce` will be set to the nilified nonce. Now even if the nilified package gets reverified, it will not be possible to execute it.
- clearing: If there is a need to clear a nonce which is bigger than `maxNonce + 1`, it will be impossible because of the ordered nonce.

In addition, these actions can be initiated by a delegate address and not by the `OFTApp` directly. This will cause a discrepancy between the `OrderOFT` and `LayerZero` `maxReceivedNonce` and the messaging will be blocked because `acceptNonce` will always revert from this moment on.

Recommendation

Do not call `_acceptNonce` when burning, nilifying, nor clearing. Furthermore, add a setter `setMaxReceivedNonce` to update the mapping as necessary and even consider passing a `maxReceivedNonce` argument in each of the functions which can be used to update the mapping during the execution of the owner functions.

This is because since there can be multiple in-flight packets and one of them is burned/nulled/cleared, and now the order enforcement will lead to DoS. Furthermore, a delegate can burn/nullify/clear outside the `Oapp` so the setter is absolutely necessary.

Resolution

Orderly Team: The issue was resolved in commit [e32d7c0](#).

M-01 | No Storage Gaps In Upgradable Contracts

Category	Severity	Location	Status
Upgradability	● Medium	Global	Resolved

Description

The system uses upgradable contracts. The following parent contracts don't use storage gaps, which will result in a corrupted storage if a variable is added/removed:

OFT:

- [OFTAdapterUpgradable](#)
- [OFTCoreUpgradable](#)
- [OFTUpgradable](#)
- [OAppCoreUpgradable](#)
- [OAppReceiverUpgradable](#)
- [OAppSenderUpgradable](#)
- [OAppUpgradable](#)

Recommendation

Add storage gaps to the upgradable contracts.

Resolution

Orderly Team: The issue was resolved in commit [8dae135](#).

Guardian Team: Storage gaps and used storage slots should add up to 50 which is not the case for most of the current contracts.

M-02 | Ordered Nonce Flag Should Only Be Set Once

Category	Severity	Location	Status
Logical Error	● Medium	Global	Resolved

Description

The OrderOFT and OrderAdapter contracts are intilialized with orderedNonce flag enabled, although owner can turn this flag on and off as pleased using setOrderedNonce. When orderedNonce is set to false, messages can be executed in any order, increasing the maxReceivedNonce.

If the orderedNonce is turned on back again, there will be issues with messages that were not executed with lower nonces, as the only acceptable nonce will be maxReceivedNonce + 1

Recommendation

Remove the owner function to set the orderedNonce flag.
Alternatively, only allow to turn it off and never be able to turn it back on again.

Resolution

Orderly Team: The issue was resolved in commit [e32d7c0](#).

L-01 | Implementation Contracts Can Be Initialized By Anyone

Category	Severity	Location	Status
Upgradability	● Low	Global	Resolved

Description

OrderOFT, OrderAdapter are implementation contracts designed to be called through a proxy contract. However, the initialize function can be directly called by anyone.

This would allow a malicious actor to set storage variables such as `lzEndpoint` and `owner` on the implementation contract. While no direct risks were found, it is general good practice to prevent `initialize` from being called.

Recommendation

Use `disableInitializer` in a constructor. See OpenZeppelin's [guide](#) for more details.

Resolution

Orderly Team: The issue was resolved in commit [cdb53b6](#).

L-02 | Owner Can Revoke Ownership

Category	Severity	Location	Status
Logical Error	● Low	Global	Acknowledged

Description

All Ownable contracts allow the owner to renounce their ownership. This can leave the contracts in an unexpected state and hinder the functioning of the protocol.

Recommendation

Consider utilizing Ownable2Step.

Resolution

Orderly Team: This operation will be carefully checked.

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit <https://guardianaudits.com>

To view our audit portfolio, visit <https://github.com/guardianaudits>

To book an audit, message <https://t.me/guardianaudits>