XORS Auditing Report

FOR

$O token contract

https://o1.exchange/

BY

XORS Software

XORS Software
June 11, 2026

► **Prepared For:**

$O token contract
https://o1.exchange/

► **Prepared By:**

Xiangan He & Ahmed Panju

► **Contact Us:** x@xors.xyz

Contents

1 Executive Summary

XORS conducted a security assessment of $O token contract combining Slopless automated analysis with manual review and Foundry proof-of-concept exploits.

Summary of issues detected. The assessment uncovered **2 issues**, **0 of which are assessed to be of high severity or above**. The severity distribution is as follows:

- **0 Critical-severity issues**
- **0 High-severity issues**
- **0 Medium-severity issues**
- **2 Low-severity issues**

All findings have been remediated and are marked Resolved.

Disclaimer. This report is generated by automated tooling and provides no warranty of any kind, explicit or implied. The contents should not be construed as a complete guarantee that the system is secure in all dimensions. In no event shall XORS or any of its employees be liable for any claim, damages or other liability arising from, out of or in connection with the results reported here.

About Slopless. Slopless is XORS's automated security analysis engine for smart contract codebases. It runs a multi-perspective adversarial cognitive loop, surfacing candidate vulnerabilities across reentrancy, access control, arithmetic, business logic, oracle, flash-loan, and upgradeability dimensions. Each candidate is then cross-validated by a prosecutor and defense pair before it is confirmed in the report. Manual review and Foundry proof-of-concept exploits supplement the automated output where deeper protocol-specific reasoning is required.

2 Project Dashboard

Table 2.1: Application Summary.

Name	Version	Type	Platform
$O token contract	`HEAD - HEAD`	Solidity 0.8.22	EVM

Table 2.2: Engagement Summary.

Dates	Method	Consultants Engaged	Level of Effort
Jun. 11, 2026	Slopless + Manual	2	2 person-day

Table 2.3: Vulnerability Summary.

Name	Number	Resolved
Critical-Severity Issues	0	0
High-Severity Issues	0	0
Medium-Severity Issues	0	0
Low-Severity Issues	2	2
Warning-Severity Issues	0	0
Informational-Severity Issues	0	0
TOTAL	2	2

Table 2.4: Category Breakdown.

Name	Number
Security	2

3 Audit Goals and Scope

3.1 Audit Goals

The scan was scoped to provide a comprehensive automated security assessment of $O token contract. The analysis sought to identify security vulnerabilities, code quality issues, and reliability concerns using Slopless static analysis.

3.2 Classification of Vulnerabilities

Findings are classified by severity: Critical, High, Medium, and Low, based on the potential impact and likelihood of exploitation.

Table 3.1: Severity Breakdown.

	Somewhat Bad	Bad	Very Bad	Protocol Breaking
Not Likely	Info	Warning	Low	Medium
Likely	Warning	Low	Medium	High
Very Likely	Low	Medium	High	Critical

4 Vulnerability Report

In this section, we describe the vulnerabilities found during the Slopless scan. For each issue found, we log the type of the issue, its severity, location in the code base, and its current status. Table 4.1 summarizes the issues discovered:

Table 4.1: Summary of Discovered Vulnerabilities.

ID	Description	Severity	Status
XORS-L1	sweep(address) can withdraw the adapter's backing token — drain path constrained by multisig ownership	Low	Resolved
XORS-L2	bridgingEnabled gates only the outbound send — the inbound BSC→Base release is ungated (acceptable by design)	Low	Resolved

4.1 Detailed Description of Issues

4.1.1 XORS-L1: sweep(address) can withdraw the adapter's backing token — drain path constrained by multisig ownership

Severity	Low	**Type**	Security
Status	Resolved	**Location**	sweep(address)
File(s)	contracts/MyOFTAdapter.sol		
CWE	CWE-269		

Description. The Base-side contract MyOFTAdapter is a LayerZero OFT Adapter (lockbox): it holds the canonical $O ERC-20 returned by token() as locked backing, and an equal amount of $O is minted by MyOFT on BSC for every inbound message. The bridge's solvency invariant is therefore: backing held by the adapter ≥ $O circulating on BSC. The owner-only sweep(address _token) function transfers the full balance of the supplied ERC-20 to the owner and emits Swept. Because it accepts an arbitrary token address with no carve-out, it can move the backing token() itself. In the worst case a sweep of the backing balance would leave BSC-side holders with OFT that can no longer be redeemed, because the inbound release reverts when the adapter does not hold the tokens it must return.

Impact. Severity is assessed as Low. sweep() is owner-gated, and the adapter owner in the deployed configuration is a multisig — owner() resolves to the project multisig on-chain. Reaching this path therefore requires collusion or compromise of the multisig signer set rather than a single key, which materially constrains likelihood. There is no path for an unprivileged actor to trigger the withdrawal and no economic incentive for honest signers to do so.

Recommendation. Exclude the backing asset from the rescue path: sweep() should revert when the requested token is the backing asset (_token == token()), so that only stray or incidental tokens can be recovered and the backing ≥ circulating invariant holds independent of the owner. Resolution: the residual risk is accepted on the basis of multisig ownership — sweep() is reachable only through the project multisig, which bounds the likelihood to a multisig-level compromise rather than a single key. The carve-out above is recommended as defense-in-depth.

4.1.2 XORS-L2: bridgingEnabled gates only the outbound send — the inbound BSC→Base release is ungated (acceptable by design)

Severity	Low	**Type**	Security
Status	Resolved	**Location**	lzReceive / _credit
File(s)	contracts/MyOFTAdapter.sol		
CWE	CWE-284		

Description. Bridging is gated by the bridgingEnabled flag (toggled by the owner-only setBridgingEnabled). The check sits on the outbound leg — the Base→BSC send / _debit path that locks $O — so disabling bridging halts new locks and sends. The inbound leg — lzReceive / _credit, which releases locked $O when a BSC→Base message is delivered — does not consult bridgingEnabled, so redemptions on the return path continue even while bridgingEnabled is false.

Impact. Severity is assessed as Low. Gating the inbound (return) direction is not operationally required: the meaningful incident control is disabling new outbound bridging, while leaving redemptions open is generally desirable so that users holding $O on BSC cannot be trapped. The asymmetry carries no fund-loss impact — the inbound release still requires a valid LayerZero message from the configured peer — so it does not weaken the security posture of the bridge.

Recommendation. No code change is required. The team confirmed the inbound release is intentionally left open: pausing redemptions on the return path is unnecessary, and bridgingEnabled on the outbound path is the intended incident control. Resolved as accepted by design; the asymmetry is documented so operators understand that disabling bridging stops new sends but does not freeze in-flight redemptions.