WORLDSHARDS ERC20 LAYERZERO BRIDGE AUDIT REPORT

OXORIO

AUGUST 1, 2025

1 EXECUTIVE SUMMARY

OXORIO

1.1 EXECUTIVE SUMMARY

This document presents the smart contracts security audit conducted by Oxorio for `Lowkick Games` WorldShards ERC20 LayerZero Bridge.

`Lowkick Games` is an independent game development studio known for creating simple, yet engaging and innovative games. The studio focuses on offering unique gameplay experiences, often with an emphasis on intuitive mechanics and creative design. Their projects often appeal to a broad audience, making their games enjoyable for both casual and dedicated gamers alike. While the specifics of their games and achievements may vary over time, the studio is typically recognized for its dedication to quality and originality in game development.

This is a project for the ERC-20 token contract of the WorldShards Game.

The audit process involved a comprehensive approach, including manual code review, automated analysis, and extensive testing and simulations of the smart contracts to assess the project's security and functionality. The audit covered a total of 2 smart contracts, encompassing 22 lines of code. The codebase was thoroughly examined, with the audit team collaborating closely with `Lowkick Games` and referencing the provided documentation to address any questions regarding the expected behavior. For an in-depth explanation of used the smart contract security audit methodology, please refer to the Security Assessment Methodology section of this document.

Throughout the audit, a collaborative approach was maintained with `Lowkick Games` to address all concerns identified within the audit's scope. Each issue has been either resolved or formally acknowledged by `Lowkick Games`, contributing to the robustness of the project.

As a result, following a comprehensive review, our auditors have verified that the WorldShards ERC20 LayerZero Bridge, as of audited commit `95d13526fe3a3ab1b7c66920b89ead8e48dbe97a`, has met the security and functionality requirements established for this audit, based on the code and documentation provided, and operates as intended within the defined scope.

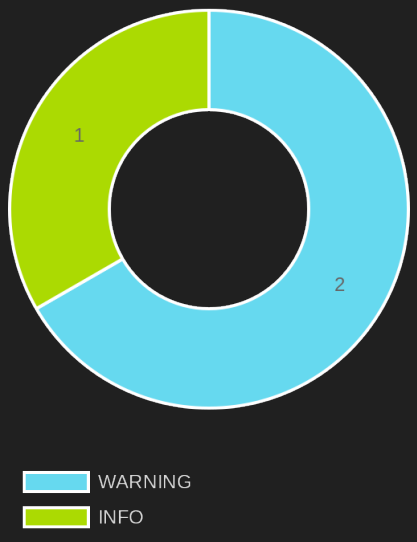
1.2 SUMMARY OF FINDINGS

The table below provides a comprehensive summary of the audit findings, categorizing each by status and severity level. For a detailed description of the severity levels and statuses of findings, see the Findings Classification Reference section.

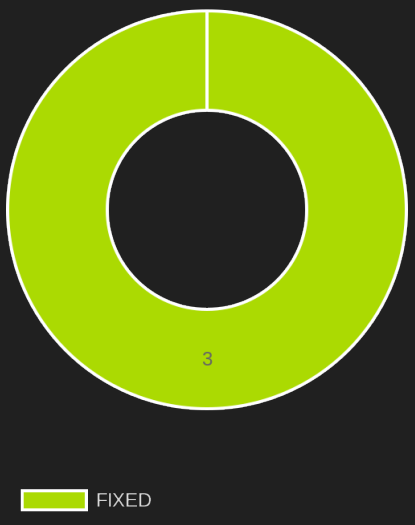
Detailed technical information on the audit findings, along with our recommendations for addressing them, is provided in the Findings Report section for further reference.

All identified issues have been addressed, with `Lowkick Games` fixing them or formally acknowledging their status.

Severity	TOTAL	NEW	FIXED	ACKNOWLEDGED	NO ISSUE
CRITICAL	**0**	0	0	0	0
MAJOR	**0**	0	0	0	0
WARNING	**2**	0	2	0	0
INFO	**1**	0	1	0	0
TOTAL	**3**	**0**	**3**	**0**	**0**



Issue distribution by severity



Issue distribution by status

2 AUDIT OVERVIEW

CONTENTS

2.1 DISCLAIMER

At the request of the client, Oxorio consents to the public release of this audit report. The information contained herein is provided "as is" without any representations or warranties of any kind. Oxorio disclaims all liability for any damages arising from or related to the use of this audit report. Oxorio retains copyright over the contents of this report.

This report is based on the scope of materials and documentation provided to Oxorio for the security audit as detailed in the Executive Summary and Audited Files sections. The findings presented in this report may not encompass all potential vulnerabilities. Oxorio delivers this report and its findings on an as-is basis, and any reliance on this report is undertaken at the user's sole risk. It is important to recognize that blockchain technology remains in a developmental stage and is subject to inherent risks and flaws.

This audit does not extend beyond the programming language of smart contracts to include areas such as the compiler layer or other components that may introduce security risks. Consequently, this report should not be interpreted as an endorsement of any project or team, nor does it guarantee the security of the project under review.

THE CONTENT OF THIS REPORT, INCLUDING ITS ACCESS AND/OR USE, AS WELL AS ANY ASSOCIATED SERVICES OR MATERIALS, MUST NOT BE CONSIDERED OR RELIED UPON AS FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER PROFESSIONAL ADVICE. Third parties should not rely on this report for making any decisions, including the purchase or sale of any product, service, or asset. Oxorio expressly disclaims any liability related to the report, its contents, and any associated services, including, but not limited to, implied warranties of merchantability, fitness for a particular purpose, and non-infringement. Oxorio does not warrant, endorse, or take responsibility for any product or service referenced or linked within this report.

For any decisions related to financial, legal, regulatory, or other professional advice, users are strongly encouraged to consult with qualified professionals.

2.2 PROJECT BRIEF

Title	Description
Client	`Lowkick Games`
Project name	WorldShards ERC20 LayerZero Bridge
Category	Bridge
Website	lowkick.games
Documentation	README.md
Repository	github.com/lowkickgames/worldshards-erc20-lz-bridge/
Initial Commit	`c231086fe83c4e60809cb1abcd3014a6c756889c`
Final Commit	`95d13526fe3a3ab1b7c66920b89ead8e48dbe97a`
Platform	L1
Network	Ethreum, BNB
Languages	Solidity
Lead Auditor	Alexander Mazaletskiy
Project Manager	Elena Kozmiryuk - elena@oxor.io

2.3 PROJECT TIMELINE

The key events and milestones of the project are outlined below.

Date	Event
July 24, 2025	Client approached Oxorio requesting an audit.
July 31, 2025	The audit team commenced work on the project.
July 31, 2025	Submission of the comprehensive report.
August 1, 2025	Submission of the final report incorporating client's verified fixes.

2.4 AUDITED FILES

The following table contains a list of the audited files. The scc tool was used to count the number of lines and assess complexity of the files.

	File	Lines	Blanks	Comments	Code	Complexity
1	contracts//WorldShardsOFTAdapter.sol	19	5	3	**11**	0%
2	contracts/WorldShardsOFT.sol	18	4	3	**11**	0%
	Total	**37**	**9**	**6**	**22**	**0%**

Lines: The total number of lines in each file. This provides a quick overview of the file size and its contents.

Blanks: The count of blank lines in the file.

Comments: This column shows the number of lines that are comments.

Code: The count of lines that actually contain executable code. This metric is essential for understanding how much of the file is dedicated to operational elements rather than comments or whitespace.

Complexity: This column shows the file complexity per line of code. It is calculated by dividing the file's total complexity (an approximation of cyclomatic complexity that estimates logical depth and decision points like loops and conditional branches) by the number of executable lines of code. A higher value suggests greater complexity per line, indicating areas with concentrated logic.

2.5 PROJECT OVERVIEW

This is a project for the WorldShards ERC20 LayerZero Bridge of the WorldShards Game.

The `$SHARDS` Token is the main resource of the web3 economy in WorldShards. As a fair launch token, `$SHARDS` has no allocation to the team or investors, ensuring it to be a fair community driven token.

The maximum circulating supply of `$SHARDS` Tokens is 5,000,000,000. Players can primarily earn `$SHARDS` through in-game drops.

2.6 CODEBASE QUALITY ASSESSMENT

The Codebase Quality Assessment table offers a comprehensive assessment of various code metrics, as evaluated by our team during the audit, to gauge the overall quality and maturity of the project's codebase. By evaluating factors such as complexity, documentation and testing coverage to best practices, this table highlights areas where the project excels and identifies potential improvement opportunities. Each metric receives an individual rating, offering a clear snapshot of the project's current state, guiding prioritization for refactoring efforts, and providing insights into its maintainability, security, and scalability. For a detailed description of the categories and ratings, see the Codebase Quality Assessment Reference section.

Category	Assessment	Result
Access Control	All tokens are allocated at deployment, with no administrative control functions in the contract.	**Not Applicable**
Arithmetic	The token contract is based on OpenZeppelin's standard implementations.	**Excellent**
Complexity	The token contract is based on OpenZeppelin's standard implementations.	**Excellent**
Data Validation	Data validation is secure and correctly implemented.	**Excellent**
Decentralization	The token implements a decentralized architecture with initial distribution allocated to a multisignature wallet address during contract deployment.	**Excellent**
Documentation	Token specifications and deployment configuration parameters are documented in the `readme.md` file	**Excellent**
External Dependencies	The contract has no external dependencies.	**Not Applicable**
Error Handling	The token contract is based on OpenZeppelin's standard implementations.	**Excellent**
Logging and Monitoring	The token contract is based on OpenZeppelin's standard implementations.	**Excellent**
Low-Level Calls	The contract has no low-level calls	**Not Applicable**
Testing and Verification	The project inherits tests from the OpenZeppelin project and includes tests for correct deployment and distribution logic.	**Excellent**

2.7 FINDINGS BREAKDOWN BY FILE

This table provides an overview of the findings across the audited files, categorized by severity level. It serves as a useful tool for identifying areas that may require attention, helping to prioritize remediation efforts, and provides a clear summary of the audit results.

File	TOTAL	CRITICAL	MAJOR	WARNING	INFO
contracts/WorldShardsOFTAdapter.sol	**3**	0	0	2	1
contracts/WorldShardsOFT.sol	**1**	0	0	1	0

2.8 CONCLUSION

A comprehensive audit was conducted on 2 smart contracts, initially revealing 0 critical and 0 major issues, along with numerous warnings and informational notes.

Key Findings Summary: The audit highlighted various attack vectors and potential vulnerabilities, with significant findings related to the absence of input validation for critical parameters, such as the `_lzEndpoint` and `_token` addresses in the `WorldShardsOFT` and `WorldShardsOFTAdapter` contracts. Specific concerns include the possibility of setting a zero address for `_lzEndpoint`, the lack of verification for `ERC20` interface compliance in `_token`, and potential complications with ownership and delegate roles due to insufficient checks in the initialize function. These issues could lead to operational failures or unintended behavior if not addressed.

Following our initial audit, `Lowkick Games` worked closely with our team to address the identified issues. The proposed changes are aimed at reinforcing role management integrity, ensuring accurate administrative permission enforcement, and enhancing code efficiency and documentation clarity to strengthen the overall security and reliability of the smart contracts. Specifically, we recommend adding checks to prevent zero-address assignments for `_lzEndpoint`, verifying `ERC20` interface compliance for `_token` using `ERC165` or low-level calls to check key functions (e.g., `balanceOf`, `totalSupply`), and ensuring distinct roles for deployer and delegate in the initialize function to avoid ownership transfer issues.

As a result, the project has passed our audit. Our auditors have verified that the WorldShards ERC20 LayerZero Bridge, as of audited commit `95d13526fe3a3ab1b7c66920b89ead8e48dbe97a`, operates as intended within the defined scope, based on the information and code provided at the time of evaluation. The robustness of the codebase has been significantly improved, meeting the necessary security and functionality requirements established for this audit.

3 AUDIT OBJECTIVES

The audit focused on identifying vulnerabilities, ensuring compliance with standards, and verifying the correctness of the implementation. The key areas of focus included:

3.1 INPUT VALIDATION

- Verification of input parameters in constructors and critical functions, such as `_lzEndpoint` and `_token`, to prevent invalid or zero-address assignments.
- Ensuring the `_token` parameter in `WorldShardsOFTAdapter` adheres to the ERC20 interface, including checks for key functions (`balanceOf`, `totalSupply`, `transfer`, `transferFrom`, `approve`, `allowance`) via ERC165 or low-level calls.

3.2 ROLE AND PERMISSION MANAGEMENT

- Analysis of ownership and delegate roles in `WorldShardsOFTAdapter`, particularly in the `initialize` function, to ensure proper separation of roles and prevent complications with ownership transfers.
- Validation of access control mechanisms to ensure only authorized entities can perform sensitive operations.

3.3 LAYERZERO INTEGRATION

- Correctness of integration with the LayerZero protocol, including the initialization of the `_lzEndpoint` address and its immutability in the `OAppCore` base contract.
- Verification of cross-chain messaging and token bridging logic to ensure secure and reliable token transfers across chains.

3.4 CODE EFFICIENCY AND GAS OPTIMIZATION

- Evaluation of gas usage in critical functions, such as token transfers and cross-chain operations, to identify potential inefficiencies.
- Review of code structure for clarity, maintainability, and adherence to best practices.

3.5 SECURITY AND VULNERABILITY ASSESSMENT

Identification of common vulnerabilities, including but not limited to:

- Reentrancy attacks.
- Front-running risks in cross-chain operations.
- Improper handling of immutable variables (e.g., `endpoint` and `innerToken`).
- Potential denial-of-service (DoS) scenarios due to invalid inputs or edge cases.
- Analysis of external dependencies, such as the LayerZero endpoint and ERC20 token contracts, for compatibility and security risks.

3.6 COMPLIANCE WITH STANDARDS

- Verification of compliance with ERC20 and ERC165 standards for token interactions.
- Ensuring adherence to LayerZero's OFT and OAppCore standards for cross-chain token bridging.

3.7 OUT OF SCOPE

The following areas were explicitly excluded from the audit:

- External dependencies not directly integrated into the contracts (e.g., LayerZero endpoint implementation details).
- Off-chain components, such as relayers or oracles, unless directly interacting with the audited contracts.

4. FINDINGS REPORT

OXORIO

4.1 CRITICAL

No issues found

4.2 MAJOR

No issues found

4.3 WARNING

W-01	Possibility to set `_lzEndpoint` to `address(0)` in `WorldShardsOFT`, `WorldShardsOFTAdapter`
Severity	**WARNING**
Status	• FIXED

Location

File	Location	Line
WorldShardsOFT.sol	contract `WorldShardsOFT` > `constructor`	8
WorldShardsOFTAdapter.sol	contract `WorldShardsOFTAdapter` > `constructor`	8

Description

In the mentioned locations, there is no check to ensure that `_lzEndpoint` is not equal to address `address(0)`.

As a result, the constructor of the base contract `OAppCore` may receive a zero `endpoint`. Moreover, since `endpoint` is an `immutable` variable, it cannot be changed later:

```
// The LayerZero endpoint associated with the given OApp
ILayerZeroEndpointV2 public immutable endpoint;

/**

 * @dev Constructor to initialize the OAppCore with the provided endpoint and delegate.
 * @param _endpoint The address of the LOCAL Layer Zero endpoint.
 */
constructor(address _endpoint) {
    endpoint = ILayerZeroEndpointV2(_endpoint);
}
```

Recommendation

We recommend adding a check to ensure that the `_lzEndpoint` parameter is not equal to `address(0)`.

Update

Fixed at `dad670ceb54785d4b8ae9a73942053eb0f7c245b`

W-02	Possible to pass a `_token` that does not implement `IERC20` in `WorldShardsOFTAdapter`
Severity	**WARNING**
Status	• FIXED

Location

File	Location	Line
WorldShardsOFTAdapter.sol	contract `WorldShardsOFTAdapter` > `constructor`	8

Description

In the `constructor` of the `WorldShardsOFTAdapter` contract, the `_token` parameter is passed as an address. However, there is no check to ensure that this address implements the ERC20 token interface.

Thus, it is possible to pass to the adapter an address that is not a fully compliant ERC20 token—one that, aside from `decimals` function, does not implement other standard functions.

Additionally, in the base contract `OFTAdapterUpgradeable`, the token is assigned to the `innerToken` variable, which is `immutable` and therefore cannot be changed later:

```
IERC20 internal immutable innerToken;

/**

 * @dev Constructor for the OFTAdapter contract.
 * @param _token The address of the ERC-20 token to be adapted.
 * @param _lzEndpoint The LayerZero endpoint address.
 * @dev _token must implement the IERC20 interface, and include a decimals() function.
 */
constructor(
    address _token,
    address _lzEndpoint
) OFTCoreUpgradeable(IERC20Metadata(_token).decimals(), _lzEndpoint) {
    innerToken = IERC20(_token);
}
```

Recommendation

We recommend adding a check to verify that the _token parameter implements the `ERC20` interface, for example, by supporting the `ERC165` standard or low-level calls (e.g., call) can be used to check the existence of key `ERC20` functions such as `balanceOf`, `totalSupply`, `transfer`, `transferFrom`, `approve`, and `allowance`. This ensures that the `_token` contract complies with the `ERC20` standard and supports the required functions, preventing potential errors when interacting with incompatible contracts.

Update

Fixed at `78d64c3c876b7d71f9590efd6651797e5fb13018`

4.4 INFO

I-01	Deployer can be both delegate and owner of the contract in `WorldShardsOFTAdapter`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
WorldShardsOFTAdapter.sol	contract `WorldShardsOFTAdapter` > function `initialize`	12

Description

In the `initialize` function of the `WorldShardsOFTAdapter` contract, the delegate is set to the `_delegate` address during the OFT initialization. The same delegate address is also set as the owner of the contract.

Therefore, if the delegate address is the same as the deployer's address, calling `transferOwnership` alone will not be sufficient to transfer ownership, as the delegate address in the OFT will also need to be updated separately.

Recommendation

We recommend adding a check in the `initialize` function to ensure that `msg.sender != _delegate`, to avoid complications with transferring ownership from the deployer in the future.

Update

Fixed at `95d13526fe3a3ab1b7c66920b89ead8e48dbe97a`

5 APPENDIX

OXORIO

5.1 SECURITY ASSESSMENT METHODOLOGY

Oxorio's smart contract security audit methodology is designed to ensure the security, reliability, and compliance of smart contracts throughout their development lifecycle. Our process integrates the Smart Contract Security Verification Standard (SCSVS) with our advanced techniques to address complex security challenges. For a detailed look at our approach, please refer to the full version of our methodology. Here is a concise overview of our auditing process:

1. Project Architecture Review

All necessary information about the smart contract is gathered, including its intended functionality and dependencies. This stage sets the foundation by reviewing documentation, business logic, and initial code analysis.

2. Vulnerability Assessment

This phase involves a deep dive into the smart contract's code to identify security vulnerabilities. Rigorous testing and review processes are applied to ensure robustness against potential attacks.

This stage is focused on identifying specific vulnerabilities within the smart contract code. It involves scanning and testing the code for known security weaknesses and patterns that could potentially be exploited by malicious actors.

3. Security Model Evaluation

The smart contract’s architecture is assessed to ensure it aligns with security best practices and does not introduce potential vulnerabilities. This includes reviewing how the contract integrates with external systems, its compliance with security best practices, and whether the overall design supports a secure operational environment.

This phase involves a analysis of the project's documentation, the consistency of business logic as documented versus implemented in the code, and any assumptions made during the design and development phases. It assesses if the contract's architectural design adequately addresses potential threats and integrates necessary security controls.

4. Cross-Verification by Multiple Auditors

Typically, the project is assessed by multiple auditors to ensure a diverse range of insights and thorough coverage. Findings from individual auditors are cross-checked to verify accuracy and completeness.

5. Report Consolidation

Findings from all auditors are consolidated into a single, comprehensive audit report. This report outlines potential vulnerabilities, areas for improvement, and an overall assessment of the smart contract's security posture.

6. Reaudit of Revised Submissions

Post-review modifications made by the client are reassessed to ensure that all previously identified issues have been adequately addressed. This stage helps validate the effectiveness of the fixes applied.

7. Final Audit Report Publication

The final version of the audit report is delivered to the client and published on Oxorio's official website. This report includes detailed findings, recommendations for improvement, and an executive summary of the smart contract's security status.

5.2 CODEBASE QUALITY ASSESSMENT REFERENCE

The tables below describe the codebase quality assessment categories and rating criteria used in this report.

Category	Description
Access Control	Evaluates the effectiveness of mechanisms controlling access to ensure only authorized entities can execute specific actions, critical for maintaining system integrity and preventing unauthorized use.
Arithmetic	Focuses on the correct implementation of arithmetic operations to prevent vulnerabilities like overflows and underflows, ensuring that mathematical operations are both logically and semantically accurate.
Complexity	Assesses code organization and function clarity to confirm that functions and modules are organized for ease of understanding and maintenance, thereby reducing unnecessary complexity and enhancing readability.
Data Validation	Assesses the robustness of input validation to prevent common vulnerabilities like overflow, invalid addresses, and other malicious input exploits.
Decentralization	Reviews the implementation of decentralized governance structures to mitigate insider threats and ensure effective risk management during contract upgrades.
Documentation	Reviews the comprehensiveness and clarity of code documentation to ensure that it provides adequate guidance for understanding, maintaining, and securely operating the codebase.
External Dependencies	Evaluates the extent to which the codebase depends on external protocols, oracles, or services. It identifies risks posed by these dependencies, such as compromised data integrity, cascading failures, or reliance on centralized entities. The assessment checks if these external integrations have appropriate fallback mechanisms or redundancy to mitigate risks and protect the protocol's functionality.
Error Handling	Reviews the methods used to handle exceptions and errors, ensuring that failures are managed gracefully and securely.
Logging and Monitoring	Evaluates the use of event auditing and logging to ensure effective tracking of critical system interactions and detect potential anomalies.
Low-Level Calls	Reviews the use of low-level constructs like inline assembly, raw `call` or `delegatecall`, ensuring they are justified, carefully implemented, and do not compromise contract security.

Category	Description
Testing and Verification	Reviews the implementation of unit tests and integration tests to verify that codebase has comprehensive test coverage and reliable mechanisms to catch potential issues.

5.2.1 Rating Criteria

Rating	Description
Excellent	The system is flawless and surpasses standard industry best practices.
Good	Only minor issues were detected; overall, the system adheres to established best practices.
Fair	Issues were identified that could potentially compromise system integrity.
Poor	Numerous issues were identified that compromise system integrity.
Absent	A critical component is absent, severely compromising system safety.
Not Applicable	This category does not apply to the current evaluation.

5.3 FINDINGS CLASSIFICATION REFERENCE

5.3.1 Severity Level Reference

The following severity levels were assigned to the issues described in the report:

Title	Description
CRITICAL	Issues that pose immediate and significant risks, potentially leading to asset theft, inaccessible funds, unauthorized transactions, or other substantial financial losses. These vulnerabilities represent serious flaws that could be exploited to compromise or control the entire contract. They require immediate attention and remediation to secure the system and prevent further exploitation.
MAJOR	Issues that could cause a significant failure in the contract's functionality, potentially necessitating manual intervention to modify or replace the contract. These vulnerabilities may result in data corruption, malfunctioning logic, or prolonged downtime, requiring substantial operational changes to restore normal performance. While these issues do not immediately lead to financial losses, they compromise the reliability and security of the contract, demanding prioritized attention and remediation.
WARNING	Issues that might disrupt the contract's intended logic, affecting its correct functioning or making it vulnerable to Denial of Service (DDoS) attacks. These problems may result in the unintended triggering of conditions, edge cases, or interactions that could degrade the user experience or impede specific operations. While they do not pose immediate critical risks, they could impact contract reliability and require attention to prevent future vulnerabilities or disruptions.
INFO	Issues that do not impact the security of the project but are reported to the client's team for improvement. They include recommendations related to code quality, gas optimization, and other minor adjustments that could enhance the project's overall performance and maintainability.

5.3.2 Status Level Reference

Based on the feedback received from the client's team regarding the list of findings discovered by the contractor, the following statuses were assigned to the findings:

Title	Description
NEW	Waiting for the project team's feedback.

Title	Description
FIXED	Recommended fixes have been applied to the project code and the identified issue no longer affects the project's security.
ACKNOWLEDGED	The project team is aware of this finding and acknowledges the associated risks. This finding may affect the overall security of the project; however, based on the risk assessment, the team will decide whether to address it or leave it unchanged.
NO ISSUE	Finding does not affect the overall security of the project and does not violate the logic of its work.

5.4 ABOUT OXORIO

OXORIO is a blockchain security firm that specializes in smart contracts, zk-SNARK solutions, and security consulting. With a decade of blockchain development and five years in smart contract auditing, our expert team delivers premier security services for projects at any stage of maturity and development.

Since 2021, we've conducted key security audits for notable DeFi projects like Lido, 1Inch, Rarible, and deBridge, prioritizing excellence and long-term client relationships. Our co-founders, recognized by the Ethereum and Web3 Foundations, lead our continuous research to address new threats in the blockchain industry. Committed to the industry's trust and advancement, we contribute significantly to security standards and practices through our research and education work.

Our contacts:

- oxor.io
- ping@oxor.io
- Github
- Linkedin
- Twitter

THANK YOU FOR CHOOSING