# WORLDSHARDS BEP-20 TOKEN SCOPE REVIEW REPORT

# 1 EXECUTIVE SUMMARY

# 1.1 EXECUTIVE SUMMARY

This document presents the smart contracts security audit conducted by Oxorio for `Lowkick Games` WorldShards BEP-20 Token.

`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 BEP-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 1 smart contracts, encompassing 12 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 BEP-20 Token, as of audited commit `3500c62145f43e2939370932238aceaec3979749`, 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 | 0 | 0 | 0 | 0 | 0 |
| INFO | 0 | 0 | 0 | 0 | 0 |
| TOTAL | 0 | 0 | 0 | 0 | 0 |

Issue distribution by severity

Issue distribution by status

# 2 AUDIT OVERVIEW

# CONTENTS

OX☰RIO

# 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 `$SHARDS` BEP-20 Token |
| Category | Token |
| Website | lowkick.games |
| Documentation | worldshards-erc20-token/blob/main/README.md |
| Repository | github.com/lowkickgames/worldshards-erc20-token |
| Initial Commit | `3500c62145f43e2939370932238aceaec3979749` |
| Token contract address | `0x38fd4ee2ade8b4be157dfee3d6b8979c78a56145` |
| Platform | L1 |
| Network | 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 29, 2025 | The audit team commenced work on the project. |
| July 29, 2025 | Submission of the comprehensive report. |
| July 29, 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/WorldShadsToken.sol | 15 | 2 | 1 | **12** | 0% |
| | **Total** | **15** | **2** | **1** | **12** | **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 BEP-20 token contract 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** | The token contract is based on OpenZeppelin's standard implementations. | **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** |

| Category | Assessment | Result |
|----------|------------|--------|
| **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 |
|------|-------|----------|-------|---------|------|

# 2.8 CONCLUSION

A comprehensive audit was conducted on 1 smart contracts, initially revealing 0 critical and 0 major issues without warnings and info.

The subsequent commit `3500c62145f43e2939370932238aceaec3979749` was reviewed and confirmed to introduce no modifications to the `WorldShardsToken.sol` contract code. As such, the `deployedBytecode` remains consistent with the audited commit `0c0f0e14abc4f9d2b1bf241fb7eed2ad445f12b4`, ensuring the deployed contract at `0x38fd4ee2ade8b4be157dfee3d6b8979c78a56145` aligns with the audited commit under Token Scope Review Report dated January 10, 2025.

As a result, the project has passed our audit. Our auditors have verified that the WorldShards `$SHARDS` BEP-20 Token, as of audited commit `3500c62145f43e2939370932238aceaec3979749`, 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 FINDINGS REPORT

OX RIO

# 3.1 CRITICAL

No issues found

## 3.2 MAJOR

No issues found

# 3.3 WARNING

No issues found

# 3.4 INFO

No issues found

# 4. VERIFICATION

# 4.1 NETWORK SPECIFIC BEHAVIOR

## Status: PASS

All the network features affecting the protocol's operation are being studied. The virtual machine, the message transmission process within the main network, and vice versa (all distinctive network features and how they can impact the protocol's operation) are being researched.

## Results

Contracts comply with compiler version `v0.8.28` and EVM version `Paris`.

# 4.2 SCOPE CHECKING

## Status: PASS

This stage involves auditors researching the provided scope for verification, studying project dependencies, and building the protocol's architecture. Project documentation is examined. Existing tests are also run at this stage, and the test coverage level is checked. Contract mocks are investigated for logical errors. The protocol's architecture is examined for conceptual errors.

## Results

The declared scope fully covers the contract and its dependencies, and corresponds to the described architecture.

# 4.3 AUDIT REPORT INVESTIGATION

Status: <span style="color:green">PASS</span>

At this stage, the presence of an audit report is verified, along with the alignment of the scope in the report with the deployed scope. It is checked whether all critical vulnerabilities have either been fixed or there is evidence that the vulnerability cannot be fixed without posing a threat to the protocol. Recommendations and the conclusion in the report are studied, as well as the alignment of the final commit with all the recommendations.

## Results

> The contract code matches the audited commit and incorporates all the auditors' recommendations.

# 4.4 DEPLOY SCRIPT CHECK

## Status: PASS

Auditors study the deployment script for contracts, examining initialization parameters. It is verified that interrupting the protocol deployment will not lead to incorrect initialization (for example, a front-run on initialization should result in both the script's reversion and require re-deployment).

## Results

| The deployment script is correctly structured and ensures safe contract initialization.

# 4.5 DEPLOYMENT VERIFICATION

## Status: PASS

The bytecode of the deployed contracts is checked to match the final commit in the report. An additional check is performed to verify all contracts on the explorer. Further verification is conducted to confirm that the bytecode of deployed contracts cannot be altered.

### Results

The bytecode of the compiled contract matches the bytecode of the deployed contracts (except for the IPFS link at the end of the runtime bytecode).

# 4.6 INITIALIZATION PARAMETERS CHECK

## Status: PASS

At this stage, values are gathered from the storage in verified contracts, and they are checked for compliance with the parameters from the deployment script. Auditors ensure that all contracts are initialized and cannot be reinitialized by malicious users.

## Results

The initialization parameters align with the current storage values:

```
_name: "WorldShards"
_symbol: "SHARDS"
_totalSupply: 5000000000
_deployAddress: 0xdf00F45fae7dD4A19B3709FEe483347477Af68d4
```

# 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

OXRIO