SPHINXSHIELD

Security Assessment

Oracle Layer2

Nov 8th, 2023

Evaluation Outcomes

Security Score

Review	Score
Overall Score	97/100
Auditor Score	94/100

Review by Section	Score
Manual Scan Score	51/57
Advance Check Score	19/19

Scoring System

This scoring system is provided to gauge the overall value of the audit. The maximum achievable score is 100, but reaching this score requires the project to meet all assessment requirements.

Our updated passing score is now set at 80 points. If a project fails to achieve at least 80% of the total score, it will result in an automatic failure.

Please refer to our notes and final assessment for more details.

Table of Contents

Summary

This audit report is tailored for **Oracle Layer2**, aiming to uncover potential issues and vulnerabilities within the **Oracle Layer2** project's source code, along with scrutinizing contract dependencies outside recognized libraries. Our audit comprises a comprehensive investigation involving Static Analysis and Manual Review techniques.

Our audit process places a strong emphasis on the following focal points:

1. Rigorous testing of smart contracts against both commonplace and rare attack vectors.
2. Evaluation of the codebase for alignment with contemporary best practices and industry standards.
3. Ensuring the contract logic is in harmony with the client's specifications and objectives.
4. A comparative analysis of the contract structure and implementation against analogous smart contracts created by industry frontrunners.
5. An exhaustive, line-by-line manual review of the entire codebase by domain experts.

The outcome of this security assessment yielded findings spanning from critical to informational. To uphold robust security standards and align with industry norms, we present the following security-driven recommendations:

1. Elevate general coding practices to optimize source code structure.
2. Implement an all-encompassing suite of unit tests to account for all conceivable use cases.
3. Enhance codebase transparency through increased commenting, particularly in externally verifiable contracts.
4. Improve clarity regarding privileged activities upon the protocol's transition to a live state.

Overview

Project Summary

Project Name	Oracle Layer2
Blockchain	Ethereum
Language	Solidity
Codebase	https://etherscan.io/token/0x4De914f49f04C943dbdE7e4f8017D08B51ad657C
Commit	aff02b7139658968be9d9f99d890fb0838caa25c6d02b3cd68d9d3e132c1c4ed

Audit Summary

Delivery Date	Nov 8th, 2023
Audit Methodology	Static Analysis, Manual Review
Key Components	Oracle.sol

Vulnerability Summary

6	0	0	0	6	0
Total Findings	Resolved	Mitigated	Partially Resolved	Aknowledged	Declined

Vulnerability Level	Total	Pending	Declined	Aknowledged	Resolved
High	0	0	0	0	0
Medium	0	0	0	0	0
Low	0	0	0	0	0
Informational	6	0	0	6	0
Discussion	0	0	0	0	0

Audit Scope

ID	File	KECCAK256 or SHA256 Checksum
ORC	Oracle.sol	0x4848cd71fbf0ac7fe7a0b2b86a3a87aec90e90b771af2cdabf93773adb60ead678

Understandings

Oracle Layer2 is an Ethereum-based project that introduces innovative solutions using zk rollup and layer 2 technologies to enhance decentralized applications (dApps) on blockchain platforms. The project's core mission is to optimize the efficiency and cost-effectiveness of decentralized finance (DeFi) applications while upholding the same level of security and trust that users typically associate with traditional financial institutions.

Token Information

- Token Name: Oracle Layer2
- Symbol: ORACLE
- Decimals: 18
- Total Supply: 1,000,000 ORACLE

Tax Distribution (Not applicable)

The Oracle Layer2 contract doesn't implement a tax distribution mechanism.

Fee Management (Not applicable)

Fee management features aren't present in the Oracle Layer2 contract.

Tax Exemption (Not applicable)

There are no tax exemption functionalities in the Oracle Layer2 contract.

Ownership and Authorization

The contract owner can authorize specific addresses to access privileged functions. These functions are restricted by the onlyOwner modifier and are used for configuring the contract and address attributes.

Transaction Limits (Not applicable)

Transaction limits to prevent excessive token movement are not enforced by the Oracle Layer2 contract.

Swap Mechanism (Not applicable)

There are no tax exemption functionalities in the Oracle Layer2 contract.

Open Trading (Not applicable)

There are no restrictions on trading based on specific conditions set by the contract owner in the Oracle Layer2 contract.

Additional Functionality

The Oracle Layer2 contract primarily implements standard ERC20 token functionality, including transferring, approving, and managing allowances for token holders. It doesn't include additional functionalities like clearing stuck ETH, clearing tokens, or other custom functions.

Findings

- High - 0
- Medium - 0
- Low - 0
- Informational - 6
- Discussion - 0

Location	Title	Scope	Severity	Status
Oracle.sol:53,57,61,65,69,73,78,82,87,105,110	Function Visibility Can Be External	ERC20	Informational	Aknowledged
Oracle.sol:94,112,125,126,131,143,176,177	Use CustomError Instead of String	ERC20	Informational	Aknowledged
Oracle.sol:5	Recommend to Follow Code Layout Conventions	ERC20	Informational	Aknowledged
Oracle.sol:94,112,125,126,131,176,177	Long String in revert/require	ERC20	Informational	Aknowledged
Oracle.sol:125,126,143,176,177	Use Assembly to Check Zero Address	ERC20	Informational	Aknowledged
Oracle.sol:183,189	Empty Function Body	ERC20	Informational	Aknowledged

Optimization Suggestion - Function Visibility Can Be External

Title	Severity	Location	Status
Function Visibility Can Be External	Informational	Oracle.sol:53,57,61,65,69,73,78,82,87,105,110	Aknowledged

Description

Functions that are not called should be declared as external.

Optimization Suggestion - Use CustomError Instead of String

Title	Severity	Location	Status
Use CustomError Instead of String	Informational	Oracle.sol:94,112,125,126,131,143,176,177	Aknowledged

Description

When using require or revert, CustomError is more gas efficient than string description, as the error message described using CustomError is only compiled into four bytes. Especially when string exceeds 32 bytes, more gas will be consumed. Generally, around 250-270 gas can be saved for one CustomError replacement when compiler optimization is turned off, 60-80 gas can be saved even if compiler optimization is turned on.

Optimization Suggestion - Recommend to Follow Code Layout Conventions

Title	Severity	Location	Status
Recommend to Follow Code Layout Conventions	Informational	Oracle.sol:5	Aknowledged

Description

In the solidity document(https://docs.soliditylang.org/en/v0.8.17/style-guide.html), there are the following conventions for code layout: Layout contract elements in the following order: 1. Pragma statements, 2. Import statements, 3. Interfaces, 4. Libraries, 5. Contracts. Inside each contract, library or interface, use the following order: 1. Type declarations, 2. State variables, 3. Events, 4. Modifiers, 5. Functions. Functions should be grouped according to their visibility and ordered: 1. constructor, 2. receive function (if exists), 3. fallback function (if exists), 4. external, 5. public, 6. internal, 7. private.

Optimization Suggestion - Long String in revert/require

Title	Severity	Location	Status
Long String in revert/require	Informational	Oracle.sol:94,112,125,126,131,176,177	Aknowledged

Description

If the string parameter in the revert/require function exceeds 32 bytes, more gas will be consumed.

Optimization Suggestion - Use Assembly to Check Zero Address

Title	Severity	Location	Status
Use Assembly to Check Zero Address	Informational	Oracle.sol:125,126,143,176,177	Aknowledged

Description

Using assembly to check zero address can save gas. About 18 gas can be saved in each call.

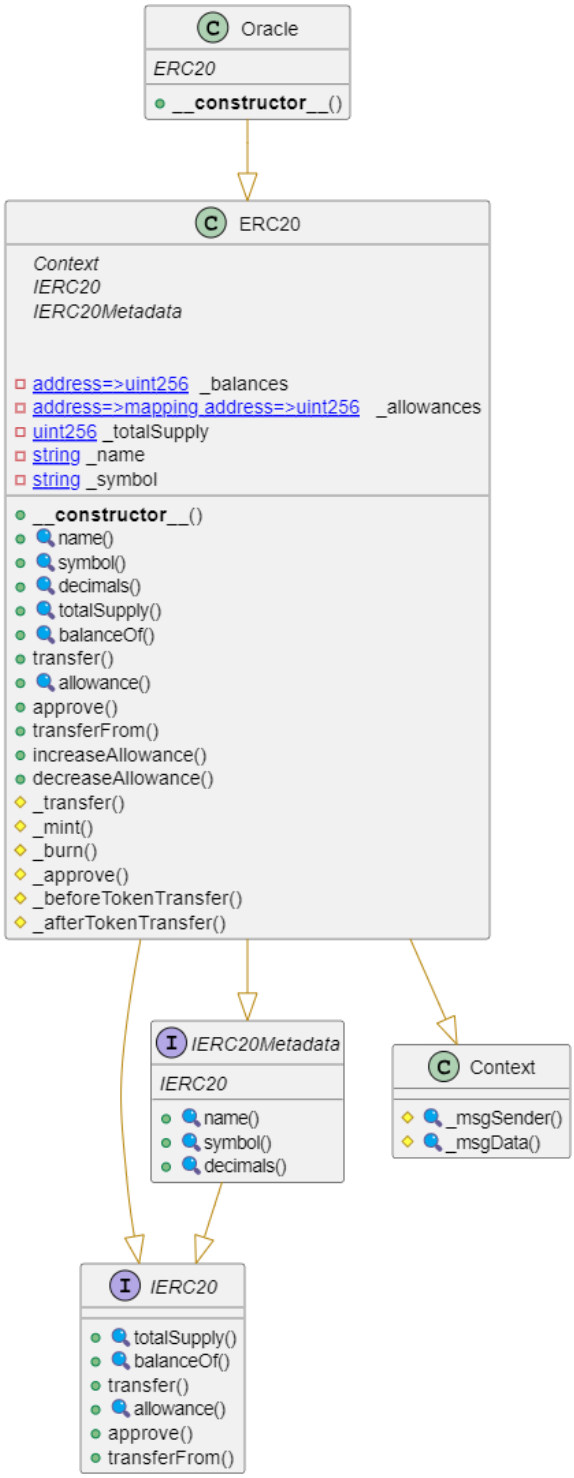
Optimization Suggestion - Empty Function Body

Title	Severity	Location	Status
Empty Function Body	Informational	Oracle.sol:183,189	Aknowledged

Description

The body of this function is empty.

PlantUML


Oracle
ERC20
__constructor__()
ERC20
Context
IERC20
IERC20Metadata
address=>uint256 _balances
address=>mapping address=>uint256 _allowances
uint256 _totalSupply
string _name
string _symbol
__constructor__()
name()
symbol()
decimals()
totalSupply()
balanceOf()
transfer()
allowance()
approve()
transferFrom()
increaseAllowance()
decreaseAllowance()
_transfer()
_mint()
_burn()
_approve()
_beforeTokenTransfer()
_afterTokenTransfer()
IERC20Metadata
IERC20
name()
symbol()
decimals()
Context
_msgSender()
_msgData()
IERC20
totalSupply()
balanceOf()
transfer()
allowance()
approve()
transferFrom()

Appendix

Finding Categories

Security and Best Practices

1. Function Visibility Can Be External: Enhance gas efficiency by setting functions to external visibility if they are accessible only from within the contract.
2. Use CustomError Instead of String: Opt for custom error codes instead of string error messages for more efficient contract operation.
3. Recommend to Follow Code Layout Conventions: Strict adherence to established code layout conventions can significantly improve code readability and maintainability.
4. Long String in revert/require: Long revert or require strings can increase gas usage and should be optimized for gas efficiency.
5. Use Assembly to Check Zero Address: Optimized assembly checks can be employed to verify zero addresses efficiently.
6. Empty Function Body: Functions should not contain empty bodies, as this can introduce vulnerabilities.

KECCAK256 or SHA256 Checksum Verification

Checksum verification is a critical component of smart contract development. It ensures the integrity of contract deployment and code execution by confirming that the bytecode being executed matches the intended source code. The following details the KECCAK256 and SHA256 checksum verification process.

KECCAK256 Checksum Verification:

- Checksum Definition: KECCAK256 is a cryptographic hashing function used in Ethereum to create a checksum of the contract bytecode. It is part of the Ethereum Name Service (ENS) standard.
- Use Cases: KECCAK256 checksums are used in ENS for verification of Ethereum addresses. They help prevent unintended transfers due to typos or errors.
- Checksum Process: The KECCAK256 checksum is created by taking the SHA3 hash of the lowercase hexadecimal Ethereum address, and then converting it to the corresponding checksum address by replacing characters with uppercase letters.

SHA256 Checksum Verification:

- Checksum Definition: SHA256 is a widely used cryptographic hash function, often employed to verify the integrity of data and contracts.
- Use Cases: SHA256 checksums are widely used in software development, including the verification of software downloads and smart contracts.
- Checksum Process: The SHA256 checksum is generated by applying the SHA256 hashing algorithm to the content of the contract. This results in a fixed-length hexadecimal value that is compared to the expected value to verify the contract's integrity.

Importance of Checksum Verification:

- Checksum verification ensures that smart contracts are executed as intended, preventing tampering and security vulnerabilities.
- It is a security best practice to verify that the deployed bytecode matches the intended source code, reducing the risk of unexpected behavior.

Best Practices:

- Always use checksum verification in situations where it is essential to verify Ethereum addresses or contract integrity.
- Implement checksum verification to ensure that contract deployment and interactions occur as intended.
- Verify the validity of contract deployments and the integrity of the code during development and deployment phases.

Website Scan

https://protocoloracle.com/

Network Security

High | 0 Attentions

Application Security

High | 6 Attentions

DNS Security

High | 1 Attentions

Network Security

9 Passed

0 Attention

FTP Service Anonymous LOGIN	NO
VNC Service Accesible	NO
RDP Service Accesible	NO
LDAP Service Accesible	NO
PPTP Service Accesible	NO
RSYNC Service Accesible	NO
SSH Weak Cipher	NO
SSH Support Weak MAC	NO
CVE on the Related Service	NO

Application Security

5 Passed | 6 Attention

Missing X-Frame-Options Header	YES
Missing HSTS header	YES
Missing X-Content-Type-Options Header	YES
Missing Content Security Policy (CSP)	YES
HTTP Access Allowed	NO
Self-Signed Certificate	NO
Wrong Host Certificate	NO
Expired Certificate	NO
SSL/TLS Supports Weak Cipher	YES
Support SSL Protocols	NO
Support TLS Weak Version	YES

DNS Health

9 Passed | 1 Attention

Missing SPF Record	NO
Missing DMARC Record	YES
Missing DKIM Record	NO
Ineffective SPF Record	NO
SPF Record Contains a Softfail Without DMARC	NO
Name Servers Versions Exposed	NO
Allow Recursive Queries	NO
CNAME in NS Records	NO
MX Records IPs are Private	NO
MX Records has Invalid Chars	NO

Social Media Checks

4 Passed

6 Failed

X (Twitter)	PASS
Facebook	FAIL
Instagram	FAIL
TikTok	FAIL
YouTube	FAIL
Twich	FAIL
Telegram	PASS
Discord	PASS
Medium	PASS
Others	FAIL

Recommendation

To enhance project credibility and outreach, we suggest having a minimum of three active social media channels and a fully functional website.

Social Media Information Notes

Unspecified Auditor Notes

Notes from the Project Owner

Fundamental Health

KYC Status

SphinxShield KYC	NO
3rd Party KYC	PinkSale

Project Maturity Metrics

Somewhat Developed	**MEDIUM**
Token Launch Date	**2023.11.01 16:45 (UTC)**
Token Market Cap (estimate)	**$438.38K**
Token/Project Age	**124 Days**

Recommendation

We strongly recommend that the project undergo the Know Your Customer (KYC) verification process with SphinxShield to enhance transparency and build trust within the crypto community. Furthermore, we encourage the project team to reach out to us promptly to rectify any inaccuracies or discrepancies in the provided information to ensure the accuracy and reliability of their project data.

Coin Tracker Analytics

Status

CoinMarketCap	NO
CoinGecko	YES
Others	NO

Recommendation

We highly recommend that the project consider integrating with multiple coin tracking platforms to expand its visibility within the cryptocurrency ecosystem. In particular, joining prominent platforms such as CoinMarketCap and CoinGecko can significantly benefit the project by increasing its reach and credibility.

CEX Holding Analytics

Status

Not available on any centralized cryptocurrency exchanges (CEX).

Recommendation

To increase your project's visibility and liquidity, we recommend pursuing listings on centralized cryptocurrency exchanges. Here's a recommendation you can use:

We strongly advise the project team to actively pursue listings on reputable centralized cryptocurrency exchanges. Being listed on these platforms can offer numerous advantages, such as increased liquidity, exposure to a broader range of traders, and enhanced credibility within the crypto community.

To facilitate this process, we recommend the following steps:

1. Research and Identify Suitable Exchanges: Conduct thorough research to identify centralized exchanges that align with your project's goals and target audience. Consider factors such as trading volume, reputation, geographical reach, and compliance with regulatory requirements.
2. Meet Compliance Requirements: Ensure that your project is compliant with all necessary legal and regulatory requirements for listing on these exchanges. This may include Know Your Customer (KYC) verification, security audits, and legal documentation.
3. Prepare a Comprehensive Listing Proposal: Create a detailed and persuasive listing proposal for each exchange you intend to approach. This proposal should highlight the unique features and benefits of your project, as well as your commitment to compliance and security.
4. Engage in Communication: Establish open lines of communication with the exchange's listing team. Be prepared to address their questions, provide requested documentation, and work closely with their team to facilitate the listing process.
5. Marketing and Community Engagement: Promote your project within the exchange's community and among your own supporters to increase visibility and trading activity upon listing.
6. Maintain Transparency: Maintain transparency and provide regular updates to your community and potential investors about the progress of listing efforts.
7. Be Patient and Persistent: Listing processes on centralized exchanges can sometimes be lengthy. Be patient and persistent in your efforts, and consider seeking the assistance of experts or advisors with experience in exchange listings if necessary.
8.

Remember that listing on centralized exchanges can significantly impact your project's growth and market accessibility. By following these steps and maintaining a professional, compliant, and communicative approach, you can increase your chances of successfully getting listed on centralized exchanges.

Disclaimer

SphinxShield, its agents, and associates provide the information and content contained within this audit report and materials (collectively referred to as "the Services") for informational and security assessment purposes only. The Services are provided "as is" without any warranty or representation of any kind, either express or implied. SphinxShield, its agents, and associates make no warranty or undertaking, and do not represent that the Services will:

- Meet customer's specific requirements.
- Achieve any intended results.
- Be compatible or work with any other software, applications, systems, or services.
- Operate without interruption.
- Meet any performance or reliability standards.
- Be error-free or that any errors or defects can or will be corrected.

In addition, SphinxShield, its agents, and associates make no representation or warranty of any kind, express or implied, as to the accuracy, reliability, or currency of any information or content provided through the Services. SphinxShield assumes no liability or responsibility for any:

- Errors, mistakes, or inaccuracies of content and materials.
- Loss or damage of any kind incurred as a result of the use of any content.
- Personal injury or property damage, of any nature whatsoever, resulting from customer's access to or use of the Services, assessment report, or other materials.

It is imperative to recognize that the Services, including any associated assessment reports or materials, should not be considered or relied upon as any form of financial, tax, legal, regulatory, or other advice.

SphinxShield provides the Services solely to the customer and for the purposes specifically identified in this agreement. The Services, assessment reports, and accompanying materials may not be relied upon by any other person or for any purpose not explicitly stated in this agreement. Copies of these materials may not be delivered to any other person without SphinxShield's prior written consent in each instance.

Furthermore, no third party or anyone acting on behalf of a third party shall be considered a third-party beneficiary of the Services, assessment reports, and any accompanying materials. No such third party shall have any rights of contribution against SphinxShield with respect to the Services, assessment reports, and any accompanying materials.

The representations and warranties of SphinxShield contained in this agreement are solely for the benefit of the customer. Accordingly, no third party or anyone acting on behalf of a third party shall be considered a third-party beneficiary of such representations and warranties. No such third party shall have any rights of contribution against SphinxShield with respect to such representations or warranties or any matter subject to or resulting in indemnification under this agreement or otherwise.

About

SphinxShield, established in 2023, is a cybersecurity and auditing firm dedicated to fortifying blockchain and cryptocurrency security. We specialize in providing comprehensive security audits and solutions, aimed at protecting digital assets and fostering a secure investment environment.

Our accomplished team of experts possesses in-depth expertise in the blockchain space, ensuring our clients receive meticulous code audits, vulnerability assessments, and expert security advice. We employ the latest industry standards and innovative auditing techniques to reveal potential vulnerabilities, guaranteeing the protection of our clients' digital assets against emerging threats.

At SphinxShield, our unwavering mission is to promote transparency, security, and compliance with industry standards, contributing to the growth of blockchain and cryptocurrency projects. As a forward-thinking company, we remain adaptable, staying current with emerging trends and technologies to consistently enhance our services.

SphinxShield is your trusted partner for securing crypto ventures, empowering you to explore the vast potential of blockchain technology with confidence.