SPHINXSHIELD

Security Assessment

KlubCoin

Dec 15th, 2023

Disclaimer: SphinxShield conducts security assessments on the provided source code exclusively.
Conduct your own due diligence before deciding to use any info listed at this page.

Evaluation Outcomes

Security Score

Review	Score
Overall Score	94/100
Auditor Score	91/100

Review by Section	Score
Manual Scan Score	51/57
Advance Check Score	18/19

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.

Audit Passed

PASSED

Table of Contents

Summary

This audit report is tailored for **KlubCoin**, aiming to uncover potential issues and vulnerabilities within the **KlubCoin** 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

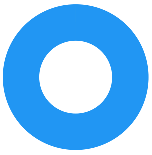
Project Summary

Project Name	KlubCoin
Blockchain	Ethereum
Language	Solidity
Codebase	https://etherscan.io/address/0x29d7139271398d0c2e22523fda06e023dcb07f8f
Commit	8bcb51a7f0ba1e31cfac3d951a52fc7151cc82af0bee4f9a1007d77b2527726b

Audit Summary

Delivery Date	Dec 15th, 2023
Audit Methodology	Static Analysis, Manual Review
Key Components	KlubCoin.sol

Vulnerability Summary



Total Findings	Resolved	Mitigated	Partially Resolved	Aknowledged	Declined
11	0	0	0	11	0

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	11	0	0	11	0
● Discussion	0	0	0	0	0

Audit Scope

ID	File	KECCAK256 or SHA256 Checksum
KBC	KlubCoin.sol	0x81c088635ab73b4b63cb28d0297eee2072f276af351799014b3d7c3775587ae7

Understandings

KlubCoin is a vibrant Ethereum-based cryptocurrency designed for clubbers, festival enthusiasts, and electronic music devotees. Deployed on the Ethereum blockchain, the KlubCoin contract incorporates various functions and mechanisms to manage its operations. Here's a detailed breakdown of its key components and functionalities:

Token Information

- Token Name: Klub Coin
- Symbol: KLUB
- Decimals: 18
- Total Supply: 1,000,000,000 KLUB

Ownership and Authorization

- The contract is owned by the address specified during deployment (admin).
- The admin address has exclusive access to privileged functions, restricted by the onlyOwner modifier.

Minting and Burning

- The initial supply of KlubCoin is minted to the deployer's address (admin).
- The burn function allows the admin to destroy a specified amount of KlubCoin tokens, reducing the total supply.

Contract Functionality

- KlubCoin supports standard ERC-20 functions such as transfer, allowance, and approval.

Additional Functionality

- The contract includes a burn function, allowing the admin to decrease the total supply.

Code Structure

- The KlubCoin contract is derived from the OpenZeppelin ERC-20 implementation.
- It includes the necessary functions and modifiers for ERC-20 compliance.
- The contract uses OpenZeppelin's Context, IERC20, IERC20Metadata, and ERC20 libraries.

Project Overview

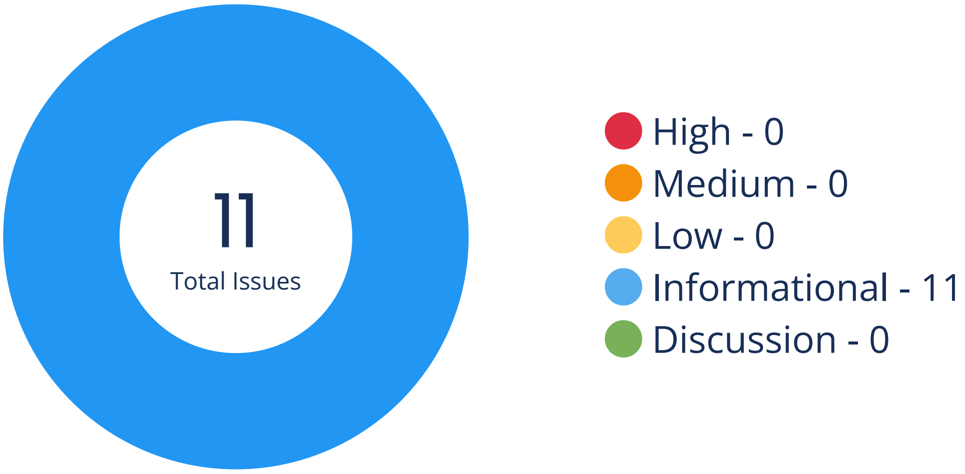
- KlubCoin aims to be the first global cryptocurrency for clubbers, festival goers, and electronic music fans.
- It introduces an innovative "Party and Rewards" model, offering cashback in tokens, exclusive events, NFT whitelisting, tickets for sold-out events, and meet-and-greets with famous DJs.

Security

- The contract has an ownership structure that allows privileged functions only for the admin, enhancing security.
- Standard ERC-20 functions are implemented following best practices and utilizing well-established OpenZeppelin libraries.

This understanding provides insights into the key features and functions of the KlubCoin contract deployed on the Ethereum blockchain. The contract serves as a vital component of the KlubCoin project, governing various aspects of its operation, including ownership, minting, burning, and user interactions.

Findings



Location	Title	Scope	Severity	Status
KlubCoin.sol:329,348	Function Visibility Can Be External	ERC20	● Informational	Aknowledged
KlubCoin.sol:6,33,118,148,512	Floating Pragma	Global	● Informational	Aknowledged
KlubCoin.sol:306,350,377,378,383,404,427,432,461,462,528	Use CustomError Instead of String	ERC20	● Informational	Aknowledged
KlubCoin.sol:6,38	Recommend to Follow Code Layout Conventions	IERC20	● Informational	Aknowledged
KlubCoin.sol:306,350,377,378,383,427,432,461,462	Long String in revert/require	ERC20	● Informational	Aknowledged
KlubCoin.sol:33,118,148,512	Specify Multiple Compiler Versions	Global	● Informational	Aknowledged
KlubCoin.sol:104,110	Unused Events	IERC20	● Informational	Aknowledged

Location	Title	Scope	Severity	Status
KlubCoin.sol:516	Variables Can Be Declared as Immutable	KlubCoin	● Informational	Aknowledged
KlubCoin.sol:377,378,404,427,461,462	Use Assembly to Check Zero Address	ERC20	● Informational	Aknowledged
KlubCoin.sol:518	Too Many Digits	KlubCoin	● Informational	Aknowledged
KlubCoin.sol:148,512	Inconsistent Solidity Version	Global	● Informational	Aknowledged

Optimization Suggestion - Function Visibility Can Be External

Title	Severity	Location	Status
Function Visibility Can Be External	● Informational	KlubCoin.sol:329,348	Aknowledged

Description

Functions that are not called should be declared as external.

Optimization Suggestion - Floating Pragma

Title	Severity	Location	Status
Floating Pragma	● Informational	KlubCoin.sol:6,33,118, 148,512	Aknowledged

Description

Contracts should be deployed with fixed compiler version which has been tested thoroughly or make sure to lock the contract compiler version in the project configuration. Locked compiler version ensures that contracts will not be compiled by untested compiler version.

Optimization Suggestion - Use CustomError Instead of String

Title	Severity	Location	Status
Use CustomError Instead of String	● Informational	KlubCoin.sol:306,350, 377,378,383,404,427,432,461,462,528	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	KlubCoin.sol:6,38	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	KlubCoin.sol:306,350,377,378,383,427,432,461,462	Aknowledged

Description

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

Optimization Suggestion - Specify Multiple Compiler Versions

Title	Severity	Location	Status
Specify Multiple Compiler Versions	● Informational	KlubCoin.sol:33,118,148,512	Aknowledged

Description

Multiple compiler versions are specified in one contract file.

Optimization Suggestion - Unused Events

Title	Severity	Location	Status
Unused Events	● Informational	KlubCoin.sol:104,110	Aknowledged

Description

Unused events increase contract size and gas usage at deployment.

Optimization Suggestion - Variables Can Be Declared as Immutable

Title	Severity	Location	Status
Variables Can Be Declared as Immutable	● Informational	KlubCoin.sol:516	Aknowledged

Description

The solidity compiler of version 0.6.5 introduces immutable to modify state variables that are only modified in the constructor. Using immutable can save gas.

Optimization Suggestion - Use Assembly to Check Zero Address

Title	Severity	Location	Status
Use Assembly to Check Zero Address	● Informational	KlubCoin.sol:377,378, 404,427,461,462	Aknowledged

Description

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

Optimization Suggestion - Too Many Digits

Title	Severity	Location	Status
Too Many Digits	● Informational	KlubCoin.sol:518	Aknowledged

Description

The number is too long, and it is easy to make mistakes when modifying and maintaining.

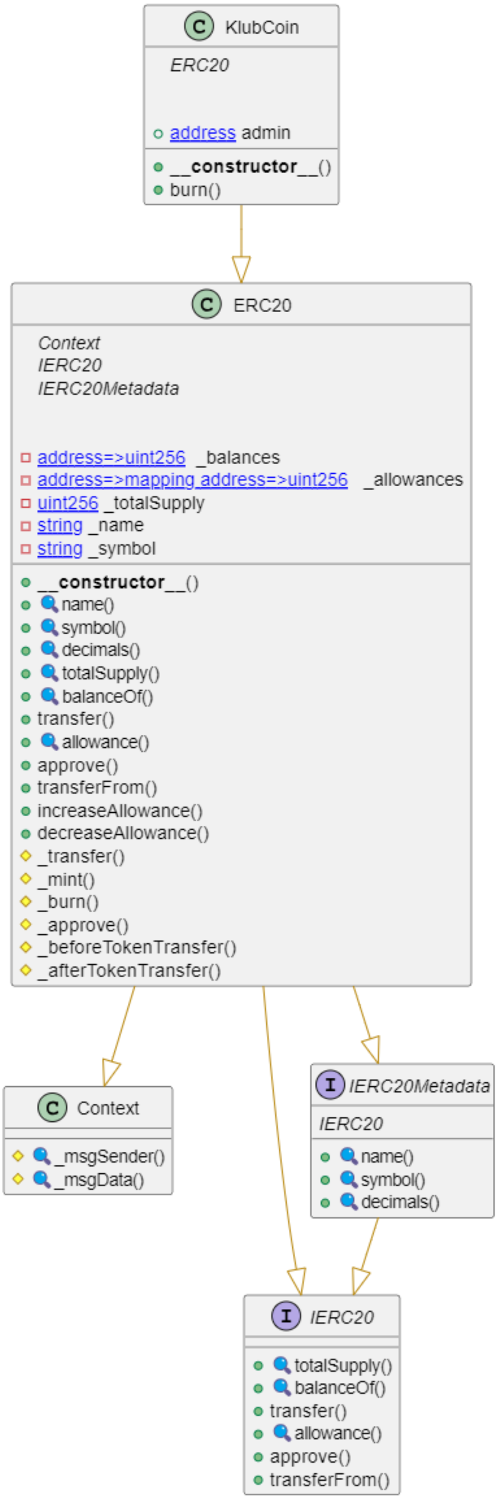
Optimization Suggestion - Inconsistent Solidity Version

Title	Severity	Location	Status
Inconsistent Solidity Version	● Informational	KlubCoin.sol:148,512	Aknowledged

Description

The source files have different solidity compiler ranges referenced. This leads to potential security flaws between deployed contracts depending on the compiler version chosen for any particular file. It also increases the cost of maintenance as different compiler versions have different semantics and behavior.

PlantUML


C KlubCoin
ERC20
address admin
__constructor__()
burn()
C 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()
C Context
_msgSender()
_msgData()
I IERC20Metadata
IERC20
name()
symbol()
decimals()
I 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. Floating Pragma: Ensure that your Solidity pragma remains consistent for added contract security.
3. Use CustomError Instead of String: Opt for custom error codes instead of string error messages for more efficient contract operation.
4. Recommend to Follow Code Layout Conventions: Strict adherence to established code layout conventions can significantly improve code readability and maintainability.
5. Long String in revert/require: Long revert or require strings can increase gas usage and should be optimized for gas efficiency.
6. Specify Multiple Compiler Versions: Clearly define and specify the Solidity compiler versions to maintain compatibility and ensure consistent behavior.
7. Unused Events: Remove any unused events from the contract code to enhance code clarity and reduce unnecessary gas consumption.
8. Variables Can Be Declared as Immutable: Variables that do not change after initialization can be declared as immutable to enhance security and readability.
9. Use Assembly to Check Zero Address: Optimized assembly checks can be employed to verify zero addresses efficiently.
10. Too Many Digits: Limit the use of excessive digits in numeric values to avoid potential precision errors and reduce gas consumption.
11. Inconsistent Solidity Version: Maintain consistency in Solidity versions across the contract to ensure compatibility and minimize unexpected behaviors.

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://klubcoin.net/

Network Security

High | 0 Attentions

Application Security

High | 3 Attentions

DNS Security

High | 5 Attentions

Network Security

✓ 9 Passed

i 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

✓ 9 Passed   ⓘ 3 Attention

Missing X-Frame-Options Header	YES ⓘ
Missing HSTS header	NO ✓
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	NO ✓
Support SSL Protocols	NO ✓
Support TLS Weak Version	NO ✓

DNS Health

✓ 7 Passed   ℹ 5 Attention

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

Social Media Checks

✓ 5 Passed   ⓘ 5 Failed

X (Twitter)	↗	PASS ✓
Facebook		FAIL ✗
Instagram	↗	PASS ✓
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	NO ✖

Project Maturity Metrics

Emerging	LOW
Token Launch Date	2023.11.01 10:45 (UTC)
Token Market Cap (estimate)	$1.22M
Token/Project Age	49 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	**YES** ✓
CoinGecko	**YES** ✓
Others	**YES** ✓

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

The coin is available on KuCoin, Bilaxy, and ExMarkets.

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.