SPHINXSHIELD

Security Assessment

Gojo Coin

Nov 15th, 2023

Evaluation Outcomes

Security Score

Review	Score
Overall Score	92/100
Auditor Score	92/100

Review by Section	Score
Manual Scan Score	42/57
Advance Check Score	17/19

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

[illegible]

Table of Contents

Summary

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

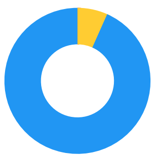
Project Summary

Project Name	Gojo Coin
Blockchain	Binance Smart Chain
Language	Solidity
Codebase	https://bscscan.com/address/0x0F9d74c91f8Cdcc2B5eFdb78cC8c41e94f13551a
Commit	d4b2c5c3e29fb031cdef30e83a70c0c062c2f222bd73b54163793f5727ca2ac0

Audit Summary

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

Vulnerability Summary



15	0	0	0	15	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	1	0	0	1	0
🔵 Informational	14	0	0	14	0
🟢 Discussion	0	0	0	0	0

Audit Scope

ID	File	KECCAK256 or SHA256 Checksum
GJC	GojoCoin.sol	0x4809f72e5f91fe89078bb409f09ab977c67c8c09db9a99f361284978112b2020

Understandings

Gojo Coin is a BEP-20 token deployed on the Binance Smart Chain (BSC), inspired by the character Gojo from the popular Jujutsu Kaisen series. Below is a breakdown of key components and functionalities within the Gojo Coin contract:

Token Information

- Token Name: Gojo Coin
- Symbol: GojoCoin
- Decimals: 9
- Total Supply: 100,000,000,000 GojoCoin

Fee Distribution

Gojo Coin transactions incur a total fee, which is distributed across various components:

- Reflective Fee (RFI): A percentage of the fee is redistributed to all holders.
- Marketing Fee: A portion of the fee is allocated to marketing efforts.

Fee Management

The contract allows the owner to manage various fees:

- Set Tax: Configure the reflective fee and marketing fee, providing flexibility in managing the fee structure.
- Set Fee Multipliers: Adjust fee percentages for buy, sell, and transfer transactions.
- Set Fee Receivers: Designate addresses to receive various fee components, such as auto-liquidity, marketing, and others.

Tax Exemption

The contract provides the owner with the capability to exempt specific addresses from fees. This feature can be utilized for whitelisting specific wallets or contracts.

Ownership and Authorization

The contract owner has the authority to authorize specific addresses, granting them access to privileged functions. These functions, protected by the onlyOwner modifier, are essential for configuring the contract and address attributes.

Transaction Limits

To prevent excessive token movement, the contract enforces transaction limits, ensuring that users do not exceed defined limits.

Swap Mechanism

Gojo Coin employs a swap mechanism to manage liquidity. When a specified threshold of tokens is reached, a portion of the contract's balance is swapped to BNB via the PancakeSwap Router. This action temporarily affects the token's price, and the remaining balance is then supplied to the Gojo Coin-BNB liquidity pool.

Trading Control

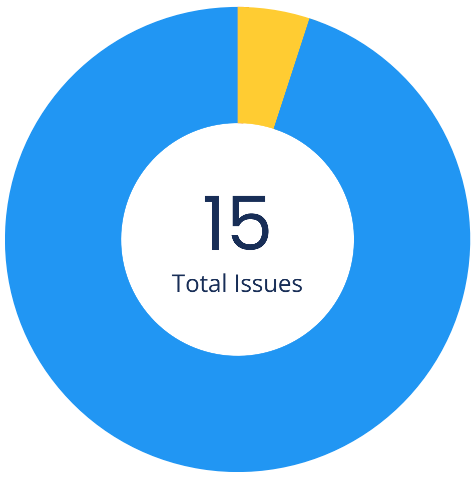
Trading can be restricted based on conditions defined by the owner. This feature ensures that trading remains closed until specific requirements are met.

Additional Functionality

The contract includes additional functions, such as clearing stuck ETH, clearing tokens, and more, enhancing its overall functionality.

This understanding provides insights into the key features and functions of the Gojo Coin contract deployed on the Binance Smart Chain. It serves as a vital component of the Gojo Coin project, governing various aspects of its operation, including fees, liquidity, and user interactions.

Findings



● High - 0
● Medium - 0
● Low - 1
● Informational - 14
● Discussion - 0

Location	Title	Scope	Severity	Status
GojoCoin.sol:155,388	Clarify Return Value	GojoCoin	● Low	Aknowledged
GojoCoin.sol:21,98,114	Recommend to Follow Code Layout Conventions	Ownable	● Informational	Aknowledged
GojoCoin.sol:182,298,310	No Check of Address Params with Zero Address	GojoCoin	● Informational	Aknowledged
GojoCoin.sol:223,291,299,300,361,362,406,407,457,459,460,461	Cache State Variables that are Read Multiple Times within A Function	GojoCoin	● Informational	Aknowledged
GojoCoin.sol:408	Use ++i/--i Instead of i++/i--	GojoCoin	● Informational	Aknowledged
GojoCoin.sol:445	Use != 0 Instead of > 0 for Unsigned Integer Comparison	GojoCoin	● Informational	Aknowledged
GojoCoin.sol:204,208,212,217,226,230,235,249,254,265,270,275,298,310,317	Function Visibility Can Be External	GojoCoin	● Informational	Aknowledged

Location	Title	Scope	Severity	Status
GojoCoin.sol:3	Floating Pragma	Global	● Informational	Aknowledged
GojoCoin.sol:349,358,367,397	Optimizable Return Statement	GojoCoin	● Informational	Aknowledged
GojoCoin.sol:35,44,193,243,259,280,291,299,311,423,424,443,444,445,446,452,537,544	Use CustomError Instead of String	Ownable	● Informational	Aknowledged
GojoCoin.sol:501	ReentrancyGuard Should Modify External Function	GojoCoin	● Informational	Aknowledged
GojoCoin.sol:44,243,259,291,423,424,443,444,445,446,537	Long String in revert/require	Ownable	● Informational	Aknowledged
GojoCoin.sol:126,127,164	Variables Can Be Declared as Immutable	GojoCoin	● Informational	Aknowledged
GojoCoin.sol:510	Get Contract Balance of ETH in Assembly	GojoCoin	● Informational	Aknowledged
GojoCoin.sol:44,423,424,434,443,444	Use Assembly to Check Zero Address	Ownable	● Informational	Aknowledged

Optimization Suggestion - Clarify Return Value

Title	Severity	Location	Status
Clarify Return Value	🟡 Low	GojoCoin.sol:155,388	Aknowledged

Description

The returned variable is specified in the function signature, but it still calls the return statement to return a local variable defined in the function body or state variable. It is necessary to clarify whether the returned value meets expectations.

Optimization Suggestion - Recommend to Follow Code Layout Conventions

Title	Severity	Location	Status
Recommend to Follow Code Layout Conventions	🔵 Informational	GojoCoin.sol:21,98,114	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 - No Check of Address Params with Zero Address

Title	Severity	Location	Status
No Check of Address Params with Zero Address	🔵 Informational	GojoCoin.sol:182,298,310	Aknowledged

Description

The input parameter of the address type in the function does not use the zero address for verification.

Optimization Suggestion - Cache State Variables that are Read Multiple Times within A Function

Title	Severity	Location	Status
Cache State Variables that are Read Multiple Times within A Function	● Informational	GojoCoin.sol:223,291, 299,300,361,362,406,407,457,459,460,461	Aknowledged

Description

When a state variable is read multiple times in a function, using a local variable to cache the state variable can avoid frequently reading data from storage, thereby saving gas.

Optimization Suggestion - Use ++i/--i Instead of i++/i--

Title	Severity	Location	Status
Use ++i/--i Instead of i++/i--	● Informational	GojoCoin.sol:408	Aknowledged

Description

Compared with i++, ++i can save about 5 gas per use. Compared with i--, --i can save about 3 gas per use in for loop.

Optimization Suggestion - Use != 0 Instead of > 0 for Unsigned Integer Comparison

Title	Severity	Location	Status
Use != 0 Instead of > 0 for Unsigned Integer Comparison	● Informational	GojoCoin.sol:445	Aknowledged

Description

For unsigned integers, use !=0 for comparison, which consumes less gas than >0. When compiler optimization is turned off, about 3 gas can be saved. When compiler optimization is turned on, no gas can be saved.

Optimization Suggestion - Function Visibility Can Be External

Title	Severity	Location	Status
Function Visibility Can Be External	● Informational	GojoCoin.sol:204,208,212,217,226,230,235,249,254,265,270,275,298,310,317	Aknowledged

Description

Functions that are not called should be declared as external.

Optimization Suggestion - Floating Pragma

Title	Severity	Location	Status
Floating Pragma	● Informational	GojoCoin.sol:3	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 - Optimizable Return Statement

Title	Severity	Location	Status
Optimizable Return Statement	● Informational	GojoCoin.sol:349,358,367,397	Aknowledged

Description

The returned variable is specified in the function signature, but the return statement is still displayed in the function body, which will increase gas consumption.

Optimization Suggestion - Use CustomError Instead of String

Title	Severity	Location	Status
Use CustomError Instead of String	● Informational	GojoCoin.sol:35,44,193,243,259,280,291,299,311,423,424,443,444,445,446,452,537,544	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 - ReentrancyGuard Should Modify External Function

Title	Severity	Location	Status
ReentrancyGuard Should Modify External Function	● Informational	GojoCoin.sol:501	Aknowledged

Description

The reentrancy guard modifier should modify the external function, because reentrancy vulnerabilities often occur in external calls.

Optimization Suggestion - Long String in revert/require

Title	Severity	Location	Status
Long String in revert/require	● Informational	GojoCoin.sol:44,243,259,291,423,424,443,444,445,446,537	Aknowledged

Description

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

Optimization Suggestion - Variables Can Be Declared as Immutable

Title	Severity	Location	Status
Variables Can Be Declared as Immutable	● Informational	GojoCoin.sol:126,127, 164	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 - Get Contract Balance of ETH in Assembly

Title	Severity	Location	Status
Get Contract Balance of ETH in Assembly	● Informational	GojoCoin.sol:510	Aknowledged

Description

Using the selfbalance and balance opcodes to get the ETH balance of the contract in assembly saves gas compared to getting the ETH balance through address(this).balance and xx.balance. When compiler optimization is turned off, about 210-250 gas can be saved, and when compiler optimization is turned on, about 50-100 gas can be saved.

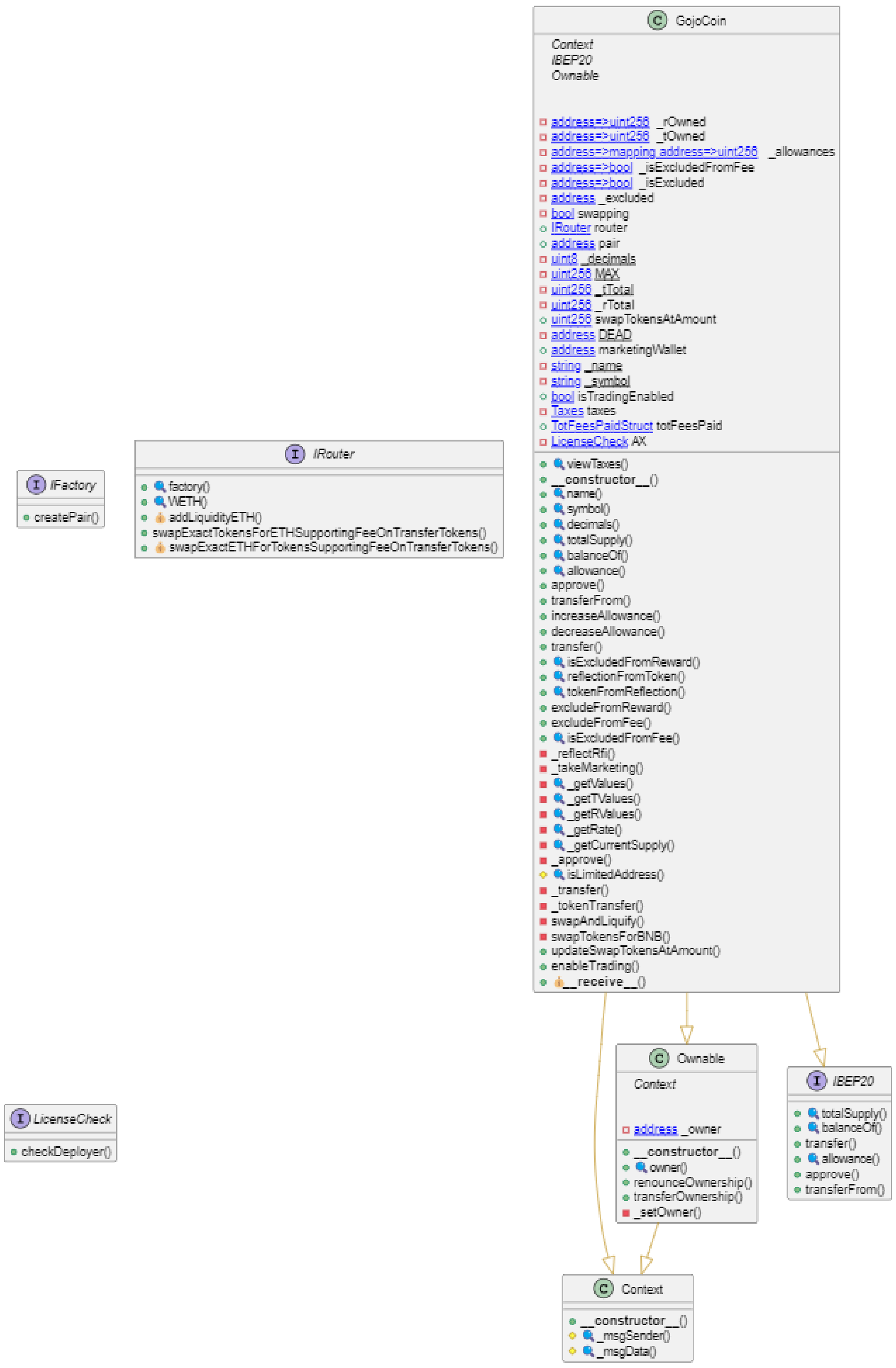
Optimization Suggestion - Use Assembly to Check Zero Address

Title	Severity	Location	Status
Use Assembly to Check Zero Address	● Informational	GojoCoin.sol:44,423,424,434,443,444	Aknowledged

Description

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

PlantUML



GojoCoin
Context
IBEP20
Ownable
address=>uint256 _rOwned
address=>uint256 _tOwned
address=>mapping address=>uint256 _allowances
address=>bool _isExcludedFromFee
address=>bool _isExcluded
address _excluded
bool swapping
IRouter router
address pair
uint8 _decimals
uint256 MAX
uint256 _tTotal
uint256 _rTotal
uint256 swapTokensAtAmount
address DEAD
address marketingWallet
string _name
string _symbol
bool isTradingEnabled
Taxes taxes
TotFeesPaidStruct totFeesPaid
LicenseCheck AX
viewTaxes()
__constructor__()
name()
symbol()
decimals()
totalSupply()
balanceOf()
allowance()
approve()
transferFrom()
increaseAllowance()
decreaseAllowance()
transfer()
isExcludedFromReward()
reflectionFromToken()
tokenFromReflection()
excludeFromReward()
excludeFromFee()
isExcludedFromFee()
_reflectRfi()
_takeMarketing()
_getValues()
_getTValues()
_getRValues()
_getRate()
_getCurrentSupply()
_approve()
isLimitedAddress()
_transfer()
_tokenTransfer()
swapAndLiquify()
swapTokensForBNB()
updateSwapTokensAtAmount()
enableTrading()
__receive__()
IFactory
createPair()
IRouter
factory()
WETH()
addLiquidityETH()
swapExactTokensForETHSupportingFeeOnTransferTokens()
swapExactETHForTokensSupportingFeeOnTransferTokens()
Ownable
Context
address _owner
__constructor__()
owner()
renounceOwnership()
transferOwnership()
_setOwner()
IBEP20
totalSupply()
balanceOf()
transfer()
allowance()
approve()
transferFrom()
LicenseCheck
checkDeployer()
Context
__constructor__()
_msgSender()
_msgData()

Appendix

<u>Finding Categories</u>

Security and Best Practices

1. Clarify Return Value: Clarify the return values of functions for better code readability and understanding. Clearly document the expected return types and values.
2. Recommend to Follow Code Layout Conventions: Adherence to established code layout conventions significantly improves code readability and maintainability. Follow consistent and widely accepted code formatting standards.
3. No Check of Address Params with Zero Address: Address parameters should undergo validation checks to ensure that they are not set to the zero address, preventing potential security vulnerabilities.
4. Cache State Variables that are Read Multiple Times within A Function: Optimize gas usage by caching state variables that are read multiple times within a function. Minimize redundant state variable reads.
5. Use ++i/--i Instead of i++/i--: Prefer using ++i and --i for incrementing and decrementing variables to enhance efficiency in certain scenarios.
6. Use != 0 Instead of > 0 for Unsigned Integer Comparison: In unsigned integer comparisons, use != 0 instead of > 0 for improved code clarity and consistency.
7. Function Visibility Can Be External: Enhance gas efficiency by setting functions to external visibility if they are accessible only from within the contract. Use external visibility where applicable.
8. Floating Pragma: Ensure that your Solidity pragma remains consistent. A well-defined Solidity pragma contributes to added contract security and stability.
9. Optimizable Return Statement: Optimize return statements to improve gas efficiency. Streamline return statements to reduce unnecessary complexity.
10. Use CustomError Instead of String: Opt for custom error codes instead of string error messages for more efficient contract operation. Custom error codes can improve gas efficiency and reduce contract size.
11. ReentrancyGuard Should Modify External Function: When implementing a ReentrancyGuard, consider modifying external functions to minimize potential reentrancy vulnerabilities.
12. Long String in revert/require: Long revert or require strings can increase gas usage. Optimize long strings for gas efficiency without compromising informative error messages.
13. Variables Can Be Declared as Immutable: Declare variables as immutable if they do not change after initialization. This enhances security and readability by clearly indicating the immutability of certain values.
14. Get Contract Balance of ETH in Assembly: Optimize gas usage by using assembly to retrieve the contract balance of ETH. Assembly may provide more efficient balance-checking mechanisms.
15. Use Assembly to Check Zero Address: Employ optimized assembly checks to verify zero addresses efficiently. Assembly can enhance gas efficiency in certain scenarios.

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://gojocoin.com/

Network Security

High | 0 Attentions

Application Security

High | 4 Attentions

DNS Security

High | 4 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

✓ 7 Passed

ⓘ 4 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	NO	✓
Support SSL Protocols	NO	✓
Support TLS Weak Version	NO	✓

DNS Health

✓ 6 Passed | ⓘ 4 Attention

Missing SPF Record	YES ⓘ
Missing DMARC Record	YES ⓘ
Missing DKIM Record	NO ✓
Ineffective SPF Record	YES ⓘ
SPF Record Contains a Softfail Without DMARC	YES ⓘ
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

✓ 2 Passed   i 8 Failed

X (Twitter)	↗	PASS ✓
Facebook		FAIL ✗
Instagram		FAIL ✗
TikTok		FAIL ✗
YouTube		FAIL ✗
Twich		FAIL ✗
Telegram	↗	PASS ✓
Discord		FAIL ✗
Medium		FAIL ✗
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	2023.11.20 15:00 (UTC)
Token Market Cap (estimate)	$56,203
Token/Project Age	1 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	NO ✗
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.