



SECURITY ANALYSIS

Share  



0x5357a5c79d89c250e0be0547be66cba514f7b8fe

BINANCE (Bsc Mainnet)

[View on Bscscan](#) 

Security Score

 86.57/100

Scan duration

 5 secs

Lines of code

 86



Your Security Score is GREAT

The SolidityScan score is calculated based on lines of code and weights assigned to each issue depending on the severity and confidence. To improve your score, view the detailed result and leverage the remediation solutions provided.



This audit report has not been verified by the SolidityScan team. To learn more about our published reports, [click here](#)



Critical 0	High 0
Medium 0	Low 4
Informational 41	Gas 22

[View Audit Report PDF](#) →

THREAT ANALYSIS

Threat Score 



 High Risk
00

 Low Risk
01

 Beneficial
06

 Moderate Risk
01

 No Impact
22

 Unavailable
08

Your smart contract has been assessed and assigned a **Moderate Risk** threat score. The score indicates the likelihood of risk associated with the contract code.

Honeypot

Token is **NOT Honeypot**

The contract is not a honeypot. Code implements standard ERC-20 transfer/approve/spendAllowance: balance checks, direct allowance set and decrement. No owner redirects, fees, minting, hidden mappings, conditional zeroing, or privilege logic that would enable confiscation or allowance bypass.

Buy Tax
--

Sell Tax
--

Transfer Tax
--

Threat Summary

Market Summary



IS SOURCE CODE VERIFIED

 Beneficial



The contract's source code is verified.
Source code verification provides transparency for users interacting with smart contracts. Block explorers validate the compiled code with the one on the blockchain. This also gives users a chance to audit the contracts.



PRESENCE OF MINTING FUNCTION

 No Impact



The contract cannot mint new tokens. The `_mint` functions was not detected in the contracts.
Mint functions are used to create new tokens and transfer them to the user's/owner's wallet to whom the tokens are minted. This increases the overall circulation of the tokens.

**PRESENCE OF BURN FUNCTION**

Beneficial



The tokens can not be burned in this contract.

Burn functions are used to increase the total value of the tokens by decreasing the total supply.

**SOLIDITY PRAGMA VERSION**

Low Risk



The contract can be compiled with an older Solidity version.

Pragma versions decide the compiler version with which the contract can be compiled. Having older pragma versions means that the code may be compiled with outdated and vulnerable compiler versions, potentially introducing vulnerabilities and CVEs.

**PROXY-BASED UPGRADABLE CONTRACT**

Beneficial



This is not an upgradeable contract.

Having upgradeable contracts or proxy patterns allows owners to make changes to the contract's functions, token circulation, and distribution.

**OWNERS CANNOT BLACKLIST TOKENS OR USERS**

No Impact



Owners cannot blacklist tokens or users.

If the owner of a contract has permission to blacklist users or tokens, all the transactions related to those entities will be halted immediately.

**IS ERC-20 TOKEN**

No Impact



The contract does not use ERC-20 token standard.

ERC-20 is the technical standard for fungible tokens that defines a set of properties that makes all the tokens similar in type and value.

**PAUSABLE CONTRACTS**

No Impact



This is not a Pausable contract.

If a contract is pausable, it allows privileged users or owners to halt the execution of certain critical functions of the contract in case malicious transactions are found.

**CRITICAL ADMINISTRATIVE FUNCTIONS**

No Impact



Critical functions that add, update, or delete owner/admin addresses are not detected.

These functions control the ownership of the contract and allow privileged users to add, update, or delete owner or administrative addresses.

Owners are usually allowed to control all the critical aspects of the contract.

**CONTRACT/TOKEN SELF DESTRUCT**

No Impact



The contract cannot be self-destructed by owners.

`selfdestruct()` is a special function in Solidity that destroys the contract and transfers all the remaining funds to the address specified during the call. This is usually access-control protected.

**ERC20 RACE CONDITION**

No Impact



The contract is not vulnerable to ERC-20 approve Race condition vulnerability.

ERC-20 approve function is vulnerable to a frontrunning attack which can be exploited by the token receiver to withdraw more tokens than the allowance. Proper mitigation steps should be implemented to prevent such vulnerabilities.

**RENOUNCED OWNERSHIP**

Beneficial

The administrator has renounced their ownership.
Renounced ownership shows that the contract is truly decentralized and once deployed, it can't be manipulated by administrators.

**USERS WITH TOKEN BALANCE MORE THAN 20%**

Moderate Risk

Some addresses contains more than 20% of circulating token supply.
0xa4633c2bf2c086852b69884c4d9e5baabab286c1 100.0
. Token distribution plays an important role when controlling the price of an asset.

**OVERPOWERED OWNERS**

No Impact

The contracts have not defined any owner-controlled functions..
Giving too many privileges to the owners via critical functions might put the user's funds at risk if the owners are compromised or if a rug-pulling attack happens.

**NO COOLDOWN CODE TO HALT TRADING OR WORKFLOWS FOUND**

Beneficial

The contract does not have a cooldown feature.
Cooldown functions are used to halt trading or other contract workflows for a certain amount of time so as to prevent users from repeatedly executing transactions or buying and selling tokens.

**OWNERS WHITELISTING TOKENS/USERS**

No Impact

Owners can not whitelist tokens or users.
If the owner of a contract has permission to whitelist users or tokens, it'll be unfair toward other users or the transaction flow may not be executed impartially.

**OWNERS CAN SET/UPDATE FEES**

No Impact

Owners can not set or update Fees in the contract.

**HARDCODED ADDRESSES**

No Impact

The contract was not hardcoding addresses in the code.

**OWNERS UPDATING TOKEN BALANCE**

No Impact

The contract does not have any owner-controlled functions modifying token balances for users or the contract.

**OWNER WALLET TOKEN SUPPLY**

No Impact

The Owner's wallet contains 0 tokens which is less than 5% of the circulating token supply.

**FUNCTION RETRIEVING OWNERSHIP**

No Impact

No such functions were found
If this function exists, it is possible for the project owner to regain ownership even after relinquishing it.



**IS SPAM CONTRACT**

Unavailable

IS SPAM CONTRACT is not supported for this contract.

**MALICIOUS TYPECASTING OF ADDRESS**

No Impact

Absence of Malicious Typecasting.

The contract is free from any malicious typecasting of addresses from uint160, ensuring that it maintains the integrity of address handling. This absence of risky typecasting methods enhances the contract's security, protecting it from potential exploitation and preserving user trust.

**LIQUIDITY BURN STATUS**

Unavailable

LIQUIDITY BURN STATUS is not supported for this contract.

**LIQUIDITY LOCK STATUS**

Unavailable

LIQUIDITY LOCK STATUS is not supported for this contract.

**TOKEN SUPPLY NOT FIXED**

No Impact

The token supply in this contract is fixed, meaning no additional tokens can be minted after deployment. This ensures that the total supply remains constant, providing greater transparency and predictability for investors and users. A fixed supply reduces the risk of inflation and ensures that the token's market value is not affected by unanticipated changes in supply.

**GAS ABUSE VIA MALICIOUS MINTING**

No Impact

No such functions were found

The approve() function in the detected contract includes a .mint() function call, which is likely designed to manipulate gas usage. This pattern suggests that the contract is engaging in malicious gas abuse, causing users to unknowingly mint gas tokens and bear the financial burden.

**CODE INJECTION VIA TOKEN NAME**

No Impact

No signs of code injection via token names or symbols were found in this contract. The token name and symbol are free from potentially harmful content, such as HTML tags or JavaScript code. This reduces the risk of Cross-Site Scripting (XSS) and ensures that user interfaces displaying this information are less likely to encounter script-based security threats.

**NO HIDDEN OWNER**

Beneficial

The contract does not contain any hidden owner roles, indicating a clear ownership structure.

**OTHER ADDRESSES WITH SPECIAL ACCESS**

No Impact

No other addresses with special permissions were detected in this contract. All privileged functions are restricted to the contract owner, minimizing the risk of unauthorized access or actions. This ensures tighter control over critical functions and helps maintain the security and integrity of the contract.

**COUNTERFEIT TOKEN**

No Impact

Counterfeit tokens mimic the symbols of official tokens, misleading users into believing they are interacting with legitimate, established cryptocurrencies. These deceptive tokens pose significant risks, as users may incur financial losses or unknowingly damage the reputation of the authentic token.

 EXTERNAL CALL RISK IN CRITICAL FUNCTIONS

 No Impact



Absence external call risk in critical functions.
The critical functions in this contract do not include any external calls, minimizing the risk of unexpected state changes, and external dependencies, thereby enhancing the security and reliability of these functions.

 AI HONEYPOT

 No Impact

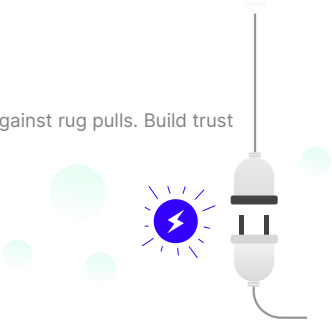


The contract is not a honeypot. Code implements standard ERC-20 transfer/approve/spendAllowance: balance checks, direct allowance set and decrement. No owner redirects, fees, minting, hidden mappings, conditional zeroing, or privilege logic that would enable confiscation or allowance bypass.

Get QuickScan API

Secure your platform with QuickScan API! It offers real-time token verification, automated contract scans, and guards against rug pulls. Build trust and stay informed.

Request for API



Contract Creator

Creators Information not available

Functions and Events

Standard Functions

allowance  approve  balanceOf  decimals  name  symbol  totalSupply  transfer  transferFrom 

Extensions

ERC20Burnable  ERC20WithAutoMinerReward 

Extra Functions

Events

Approval  Transfer 

The `allowance(owner, spender)` function confirms to the ERC20 standard.

The `approve(spender, amount)` function confirms to the ERC20 standard.

The `balanceOf(account)` function confirms to the ERC20 standard.

The `decimals()` function confirms to the ERC20 standard.

The `name()` function confirms to the ERC20 standard.

The `symbol()` function confirms to the ERC20 standard.

The `totalSupply()` function confirms to the ERC20 standard.

The `transfer(to, amount)` function confirms to the ERC20 standard.

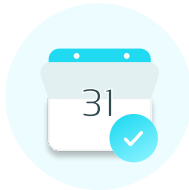
The `transferFrom(from, to, amount)` function confirms to the ERC20 standard.

The `Approval(owner, spender, value)` event is present in the contract.

The `Transfer(from, to, value)` event is present in the contract.

Why SolidityScan ?

Smart-contract scanning tool built to discover vulnerabilities & mitigate risks in your code.



Initiate Scans



Publish Reports



700+ Vulnerability Checks



Easy Integrations

Start securing your **contracts** today

Have more questions? [Talk to our team and get a demo now.](#)

[Signup For Free Trial](#)



[Pricing](#)

[Detectors](#)

[Quickscan](#)

[API Doc](#)

[SolidityScan Doc](#)

[Terms of Service](#)

[Privacy Policy](#)



 SCS Advocates

