TechRate
May, 2026

SMART CONTRACTS SECURITY AUDIT REPORT

Techrate_audits Techrate Techrate1

Audit Details

Audited project

RATGPT (RAT)

Deployer address

0x80D10f1D51a0539DaF3D6Bd2d82Af4415Ff053eE

Client contacts:

RATGPT team

Blockchain

Binance Smart Chain

Project website:

https://ratgpt.io

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and TechRate and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (TechRate) owe no duty of care towards you or any other person, nor does TechRate make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and TechRate hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, TechRate hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against TechRate, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Background

TechRate was commissioned by the RATGPT team to perform an audit of the BEP-20 smart contract deployed on Binance Smart Chain:

https://bscscan.com/token/0xc4df00D4A2e2c11B533Bf4749A759441320e13E3

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Issues Checking Status

	Issue description	Checking status
1.	**Compiler errors.**	Passed
2.	**Race conditions and Reentrancy. Cross-function race conditions.**	Passed
3.	**Possible delays in data delivery.**	Passed
4.	**Oracle calls.**	Passed
5.	**Front running.**	Passed
6.	**Timestamp dependence.**	Passed
7.	**Integer Overflow and Underflow.**	Passed
8.	**DoS with Revert.**	Passed
9.	**DoS with block gas limit.**	Passed
10.	**Methods execution permissions.**	Passed
11.	**Economy model of the contract.**	Passed
12.	**The impact of the exchange rate on the logic.**	Passed
13.	**Private user data leaks.**	Passed
14.	**Malicious Event log.**	Passed
15.	**Scoping and Declarations.**	Passed
16.	**Uninitialized storage pointers.**	Passed
17.	**Arithmetic accuracy.**	Passed
18.	**Design Logic.**	Passed
19.	**Cross-function race conditions.**	Passed
20.	**Safe Open Zeppelin contracts implementation and usage.**	Passed
21.	**Fallback function security.**	Passed

Security Issues

- High Severity Issues
- Medium Severity Issues
- Low Severity Issues

Notes:

Missing zero-address check in transfer
transfer(to, amount) does not validate to != address(0). Tokens sent to 0x0 are lost AND totalSupply is not decremented, so off-chain accounting drifts from on-chain reality.
Recommendation (short): Add require(to != address(0)); route burns through burn().
Missing zero-address check in transferFrom
Same issue in transferFrom(from, to, amount): to == address(0) is allowed.
Recommendation (short): Add require(to != address(0)).
Missing zero-address check in approve
approve(spender, amount) allows spender == address(0), polluting the allowance mapping with dead entries.
Recommendation (short): Add require(spender != address(0)).
No IERC20 inheritance / no OZ base
Plain contract RATGPT { ... } without is IERC20 or OpenZeppelin ERC20 — bypasses every battle-tested invariant; some token lists, ERC-165 introspection, and static analyzers may misclassify.
Recommendation (short): Inherit from @openzeppelin/contracts/token/ERC20/ERC20.sol.
Classic ERC-20 approve race-condition
Only approve(spender, amount) is exposed; no increaseAllowance/decreaseAllowance,
no zero-then-set enforcement. Allows the well-known approval-front-running double-spend.
Recommendation (short): Add increaseAllowance / decreaseAllowance helpers, or enforce zero-then-set.
Floating pragma ^0.8.20
Source pragma is ^0.8.20; would compile with later 0.8.x and produce different bytecode. On-chain artefact was built with 0.8.20+commit.a1b79de6.
Recommendation (short): Pin exact version: pragma solidity 0.8.20;.
Compiler optimization disabled
BscScan metadata shows Optimization: No (200 runs). Higher gas on every transfer/approve/burn forever.
Recommendation (short): Redeploy with --optimize --optimize-runs 200 (or higher).
burn(0) allowed
burn(0) succeeds, emits a spurious Transfer(msg.sender, address(0), 0) and wastes
gas. Not security-relevant.
Recommendation (short): Add require(amount > 0).

No dedicated Burn event
burn() only emits the canonical Transfer(..., address(0), ...). Off-chain analytics cannot distinguish deliberate burns from accidental sends-to-zero.
Recommendation (short): Add event Burn(address indexed from, uint256 amount) and emit it from burn().

No infinite-allowance shortcut in transferFrom
transferFrom always decrements allowance, even when it equals type(uint256).max.

Costs an extra SSTORE per transferFrom for the standard "approve max once" DEX UX.
Recommendation (short): Skip decrement when currentAllowance == type(uint256).max.

- Single-holder state. As at audit time the contract has exactly 1 holder — the deployer 0x80D10f1D51a0539DaF3D6Bd2d82Af4415Ff053eE — who holds the entire 1 000 000 000 RAT supply. Zero transfers have occurred since the constructor mint at block 95 915 749 (May-02-2026 09:43:27 UTC, tx 0x4081cfa2ec8cead4918d011cdf83b2975ffa85da61f7ea65c40343fb7b0b41a3).
- No liquidity yet. No Pancake V2 / V3 pair exists for RAT at audit time. Any DEX listing is a future event the team must orchestrate. Liquidity-locking details to be provided once a pool is created. Concentration risk (off-chain). The on-chain code grants no privileges, but the deployer holds 100 % of supply. The token's economic security currently depends entirely on what the deployer does with that balance (vesting, locking, distribution). None of this is enforced by the smart contract itself.
- Source verified on BscScan; the verified code matches the audited source byte-for-byte. Compiler v0.8.20+commit.a1b79de6.
  No proxy / no upgrade path. The deployment is a plain contract, not an EIP-1967 / UUPS / Transparent proxy. The byte-code is therefore immutable and no future upgrade can change the logic.
- Constructor mints to msg.sender (line 19) and emits Transfer(address(0), msg.sender, totalSupply) (line 20) — the initial-mint signal is correctly indexed by every standard explorer.
  No receive() / fallback() — the contract cannot receive native BNB. Any direct BNB send to the contract reverts. This eliminates the trapped-BNB footgun present in many other tokens.
- No reentrancy surface. No external calls, no call{value:...}, no token-callback hooks, no ERC-777 / ERC-1363 / ERC-223 hooks. Reentrancy is structurally impossible.
- Solidity 0.8.32 built-in arithmetic checks. Overflow / underflow protection is built into the compiler; no SafeMath is needed and none is used.
- balanceOf and allowance are public mappings; the auto-generated getters match the EIP-20 selectors 0x70a08231 and 0xdd62ed3e respectively.
- No anti-bot / anti-MEV mechanisms. No cooldown, no max-tx amount, no transfer fees, no honeypot logic. This is a clean, unmodified ERC-20.
- Honeypot signals. Static review confirms buyTax = 0 %, sellTax = 0 %, no hidden transfer hooks; the contract is not a honeypot at the byte-code level. An automated honeypot.is run will be possible once liquidity is added.

Conclusion

Smart contracts contain high severity issues! The further transfers and operations with the funds raise are not related to this particular contract.

Security score: 93.

TechRate note:
Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.