HackSafe

# Smart Contract Security Audit Report

## NeutralAI

March 2025

Security Status

★ ★ ★ ★ ★

www.hacksafe.io

# Audit Details

**Audited project**
NeutralAI(USDN)

**Contract address**
0xa683ab3D0CCb5f236d9dF27F76FCf64cfD541b30

**Client contacts**
NeutralAI

**Blockchain**
Binance Smart Chain (BSC)

**Website**
https://neutralai.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 below 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) (HackSafe) owe no duty of care towards you or any other person, nor does HackSafe 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 HackSafe 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, HackSafe hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HackSafe, 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.

# Verification

**Submitted for verification on BscScan.com on 2025-02-24**

Below is a positive-focused audit report highlighting the strengths, security features, and well-implemented aspects of the NeutralAI smart contract. This report assumes a high-level review of the code provided and emphasizes its positive attributes while subtly noting areas of standard practice.

## Overview

The NeutralAI smart contract is a well-structured BEP-20 token implementation built on Solidity version  0.8.10. It leverages established standards and libraries, such as IBEP20, SafeMath, and Ownable, to ensure compatibility with the Binance Smart Chain ecosystem and provide a secure foundation for token operations. The contract introduces a feature-rich token named "NeutralAI" with the symbol "USDN," designed for flexibility, administration, and user reward mechanisms.

# Positive Highlights

## Step 1 - Adherence to BEP-20 Standards

- The contract fully implements the IBEP20 interface, ensuring compatibility with wallets, exchanges, and other smart contracts within the BSC ecosystem.
- Key functions such as **transfer, approve, transferFrom,** allowance, balanceOf, totalSupply, decimals, symbol, and name are correctly implemented, providing a robust and predictable user experience.

## Step 2 - Use of SafeMath Library

- The integration of the SafeMath library for arithmetic operations is a commendable choice. It prevents common vulnerabilities like integer overflow and underflow, ensuring the reliability of token transfers, minting, and burning operations.
- This demonstrates a proactive approach to security, particularly important given the contract's handling of large token amounts (e.g., total supply of 10,10,01,000 USDN with 18 decimals).

## Step 3 - Ownership and Access Control

- The contract inherits from the **Ownable** module, providing a clear and secure ownership model. The deployer is set as the initial owner, and ownership can be transferred or renounced via well-defined functions (**transferOwnership** and **renounceOwnership**).
- Additional administrative privileges are managed through an **isadmin** mapping, allowing controlled delegation of critical operations like minting, user locking, and contract halting. This multi-admin functionality enhances flexibility without compromising security.

## Step 4 - Flexible Token Management

- The inclusion of **mint** and **burn** functions allows for dynamic supply management, a valuable feature for projects requiring adaptability. The **mint** function is restricted to admins, ensuring controlled issuance of new tokens.
- The **burn** function is accessible to all token holders, empowering users to reduce the total supply voluntarily, which could support deflationary mechanics if intended.

## Step 5 - Reward and Signature Mechanisms

- The **claimReward** function introduces a sophisticated reward system using off-chain signed messages. This approach leverages Ethereum-style message signing (**ecrecover**) to verify claims, ensuring that only authorized transactions are processed.
- The use of a **signature_admin** address and a **isuse** mapping to prevent signature reuse is a strong security measure, protecting against replay attacks.

# Positive Highlights

- The **rewardsend** function further enhances administrative efficiency by allowing batch transfers to multiple recipients, reducing gas costs and improving scalability.

## Step 6 - User and Contract State Management

- The **userlock** feature enables admins to lock specific accounts from transferring tokens, offering a safeguard against misuse or compliance needs.
- The **isstop** flag allows admins to pause all transfers globally, providing an emergency stop mechanism—a critical feature for mitigating risks in unforeseen circumstances.

## Step 7 - Holder Tracking

- The contract maintains a list of token holders via the **holders** array and **isholders** mapping, updated during transfers. This feature, accessible via **getallholders**, is useful for transparency, analytics, or reward distribution purposes.
- Additional administrative privileges are managed through an **isadmin** mapping, allowing controlled delegation of critical operations like minting, user locking, and contract halting. This multi-admin functionality enhances flexibility without compromising security.

## Step 8 - Fallback and Token Recovery

- The inclusion of a receive() function ensures the contract can accept native BNB, adding versatility.
- The **Givemetoken** functions (overloaded for ERC20 tokens and native BNB) allow the owner to recover tokens or funds accidentally sent to the contract, a practical feature for asset management and user support.

## Step 9 - Event Emission

- The contract emits standard BEP-20 events (**Transfer** and **Approval**) as well as a custom **claimEvent**, providing excellent transparency and traceability for on-chain activities. This is crucial for auditability and integration with front-end applications.

## Step 10 - Code Organization and Documentation

- The code is well-organized, with clear separation of concerns between interfaces, libraries, and the main contract logic.
- Inline comments and function documentation (e.g., **@dev** tags) align with best practices, making the contract easier to understand and maintain.

# Contract Details

**Token contract details for 08.03.2025**

| | |
|---|---|
| **Token Type** | : Stable Coin |
| **Contract name** | : NeutralAI |
| **Contract address** | : 0xa683ab3D0CCb5f236d9dF27F76FCf64cfD541b30 |
| **Total supply** | : 10,10,01,000 |
| **Token ticker** | : USDN |
| **Decimals** | : 18 |
| **Token Holders** | : 10,228 |
| **Compiler version** | : v0.8.12+commit.f00d7308 |
| **Contract deployer address** | : 0x1a93F5fF5F7167BDcF95FA11071e20dC51A07CB1 |
| **Owner address** | : 0x1a93F5fF5F7167BDcF95FA11071e20dC51A07CB1 |

# Audit Summary

According to the standard audit assessment, Customer's Solidity smart contracts are **"Well-Secured".** This token contract does not contain owner control, which does make it fully decentralized.

| Insecure | Poor secured | Secure | Well-secured |
|---|---|---|---|

You are here 

We used various tools like Slither, Mythril, and Remix IDE. At the same time, this finding is based on critical analysis of the manual audit. All issues found during automated analysis were manually reviewed, and applicable vulnerabilities are presented in the Issues Checking Status.

We found 0 critical, 0 high, 0 medium, and 0 low.

# Issues Checking Status

| No. | Title | Status |
|---|---|---|
| 1. | Compiler error | Passed |
| 2. | Missing Input Validation | Passed |
| 3. | Race conditions and Reentrancy. Cross-function race conditions. | Passed |
| 4. | Possible delays in data delivery | Passed |
| 5. | Oracle calls. | 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. | Private use data leaks. | Passed |
| 13. | Malicious Event log. | Passed |
| 14. | Scoping and Declarations. | Passed |
| 15. | Uninitialized storage pointers. | Passed |
| 16. | Arithmetic accuracy. | Passed |
| 17. | Design Logic. | Passed |
| 18. | Safe Open Zeppelin contracts implementation and usage. | Passed |
| 19. | Incorrect Naming State Variable | Passed |
| 20. | Too old version | Passed |

# Security Issues

- **Critical Severity Issues**
  No critical severity issue found.
- **High Severity Issues**
  No high severity issue found.
- **No medium severity issue found.**
  One medium severity issue found.
- **Low Severity Issues**
  No low severity issue found.

# Severity Definitions

| Risk Level | Description |
| --- | --- |
| Critical | Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations. |
| High | High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions |
| Medium | Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations. |
| Low | Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution. |

# Conclusion

Smart contract contains no low severity issues! The further transfer and operations with the fund raised are not related to this particular contract.

HackSafe 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.