



# Smart Contract Audit

FOR  
VERB

DATED :04 Jan, 2024



# AUDIT SUMMARY

---

**Project name** – Verb AI

**Date:** 04 Jan, 2024

**Scope of Audit-** Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

**Audit Status:** **PASSED**

## Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	0	0	1
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0

---

# USED TOOLS

---

## Tools:

### 1- Manual Review:

A line by line code review has been performed by audit ace team.

**2- BSC Test Network:** All tests were conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

### 3- Slither :

The code has undergone static analysis using Slither.

### Testnet version:

The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:

<https://testnet.bscscan.com/address/0x44dce68f17fced8a8bd782b4ca1d85327a51576c#code>

---



# Token Information

---

**Token Address:**

0xea40BA42EE2e35F36eb63770F23A96Ec47e14091

**Name:** Verb AI

**Symbol:** VERB

**Decimals:** 18

**Network:** Binance Smart Chain

**Token Type:** BEP-20

**Owner:**

0x554627b7b89FAFa8e29bba08Ca6c25e27a6a9667

**Deployer:**

0x554627b7b89FAFa8e29bba08Ca6c25e27a6a9667

**Token Supply:** 10,000,000,000,000,000

**Checksum:** aEde641126e217b2b455d49e77fc4fg2

**Testnet:**

<https://testnet.bscscan.com/address/0x44dce68f17fced8a8bd782b4ca1d85327a51576c#code>

---



# TOKEN OVERVIEW

---

**Buy Fee: 0-0%**

---

**Sell Fee: 0-0%**

---

**Transfer Fee: 0-0%**

---

**Fee Privilege: Owner**

---

**Ownership: Owned**

---

**Minting: None**

---

**Max Tx: Yes**

---

**Blacklist: No**

---



# AUDIT METHODOLOGY

---

The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
  - Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
  - Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
  - Test coverage analysis determines whether the test cases are covering the code and how much code is exercised when we run the test cases.
  - Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
  - Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.
-



# VULNERABILITY CHECKLIST

---

- |                                    |                               |
|------------------------------------|-------------------------------|
| ✓ Return values of low-level calls | ✓ <b>Gasless Send</b>         |
| ✓ Private modifier                 | ✓ Using block.timestamp       |
| ✓ Multiple Sends                   | ✓ Re-entrancy                 |
| ✓ Using Suicide                    | ✓ Tautology or contradiction  |
| ✓ Gas Limitand Loops               | ✓ Timestamp Dependence        |
| ✓ Address hardcoded                | ✓ Revert/require functions    |
| ✓ Exception Disorder               | ✓ Use of tx.origin            |
| ✓ Using inline assembly            | ✓ Integer overflow/underflow  |
| ✓ Divide before multiply           | ✓ Dangerous strict equalities |
| ✓ Missing Zero Address Validation  | ✓ Using SHA3                  |
| ✓ Compiler version not fixed       | ✓ Using throw                 |
-



# CLASSIFICATION OF RISK

## Severity

## Description

◆ Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
◆ High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
◆ Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
◆ Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
◆ Gas Optimization / Suggestion	A vulnerability that has an informational character but is not affecting any of the code.

## Findings

### Severity

### Found

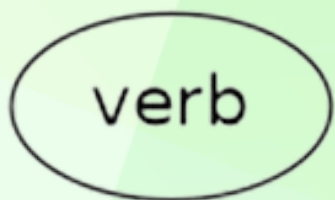
◆ Critical	0
◆ High-Risk	0
◆ Medium-Risk	0
◆ Low-Risk	0
◆ Gas Optimization / Suggestions	1





# INHERITANCE TREE

---





# STATIC ANALYSIS

---

```
INFO:Detectors:
Pragma version^0.8.17 (verb.sol#5) allows old versions
solc-0.8.22 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Contract verb (verb.sol#7-55) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
verb.constructor() (verb.sol#18-22) uses literals with too many digits:
- totalSupply = 10000000000000000 * 10 ** uint256(decimals) (verb.sol#19)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
verb.decimals (verb.sol#10) should be constant
verb.name (verb.sol#8) should be constant
verb.symbol (verb.sol#9) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
verb.totalSupply (verb.sol#11) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:verb.sol analyzed (1 contracts with 93 detectors), 8 result(s) found
```



# FUNCTIONAL TESTING

---

## 1- Approve (passed):

<https://testnet.bscscan.com/tx/0xf61d81d95af0881f5d75625616a7f3dcb29c70b2f92892cce52661c82f745aed>

## 2- Transfer (passed):

<https://testnet.bscscan.com/tx/0x8c0e074eabb5b26a86cd004ec2adaeda2289895f42a09205efb1ef6fc12ec010>

---



# MANUAL TESTING

---

**Optimization**

**Severity:** Informational

**Subject:** Floating Pragma Solidity Version

**Status:** Open

**Overview:**

It is considered best practice to pick the latest compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

```
pragma solidity ^0.8.0;
```



# DISCLAIMER

---

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed. The Auditace team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Auditace receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token. The Auditace team disclaims any liability for the resulting losses.

---



# ABOUT AUDITACE

---

We specializes in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.



**<https://auditace.tech/>**



**[https://t.me/Audit\\_Ace](https://t.me/Audit_Ace)**



**[https://twitter.com/auditace\\_](https://twitter.com/auditace_)**



**<https://github.com/Audit-Ace>**

---