Smart Contract Audit Report

# Project: AUDD Stablecoin (ERC-20 Token)

Date: October 7, 2024

Client: AUDD

Auditor: Rafsun Sheikh (GitHub: rafsunsheikh)

# 1. Introduction

This report presents the findings of the security audit conducted for the AUDD stablecoin smart contract, which follows the ERC-20 standard. The objective of the audit was to review the codebase for potential vulnerabilities and ensure the security of the contract, especially given its role in financial transactions. The audit was carried out using automated and manual analysis, focusing on best practices in smart contract development, including secure coding principles and industry-standard checks.

# 2. Audit Scope

The scope of the audit covered the **StableCoin.sol** smart contract within the AUDD codebase, with particular focus on security, functionality, and compliance with the ERC-20 standard. The analysis addressed key areas such as:

- Centralization risks

- Checks for address validation (e.g., address(0))

- Optimization of function visibility (public vs external)

- Best practices for protecting against reentrancy attacks

# 3. Methodology

The audit involved the following steps:

- Automated Vulnerability Scanning: Using AuditWizard and associated tools to detect common vulnerabilities in Ethereum smart contracts.

- Manual Code Review: A thorough line-by-line review of the Solidity code to assess logic flaws, best practices, and potential risks.

- Severity Classification: Each issue is assigned a severity level, ranging from low to critical, based on its potential impact on the system.

# 4. Findings and Recommendations

## 4.1 Centralization Risk for Trusted Owners

**Description**: Contracts have owners with privileged rights to perform admin tasks, which introduces a risk of malicious updates or unauthorized fund movements.

**Severity**: Low

**Code Location**: StableCoin.sol, Line 829

## 4.2 Missing Checks for address(0) When Assigning Values

**Description**: No check is performed when assigning values to address state variables, which could lead to inadvertent assignment of invalid addresses (i.e., address(0)).

**Severity**: Low

**Code Location**: StableCoin.sol, Lines 848, 1338, 1340, 1362, 1417, 1601

## 4.3 Public Functions Not Used Internally Should Be Marked as external

**Description**: Functions declared as public can be accessed both internally and externally, but those not accessed internally should be marked external for gas optimization.

**Severity**: Low

**Code Location**: StableCoin.sol, Lines 1161, 1169, 1212, 1186, 1200, 1257, 1280, 1300, 1586, 1625

## 4.4 Reentrancy Protection (nonReentrant Modifier Order)

**Description**: The nonReentrant modifier should occur before all other modifiers to mitigate the risk of reentrancy attacks effectively.

**Severity**: Low

**Code Location**: StableCoin.sol, Lines 1874, 2062

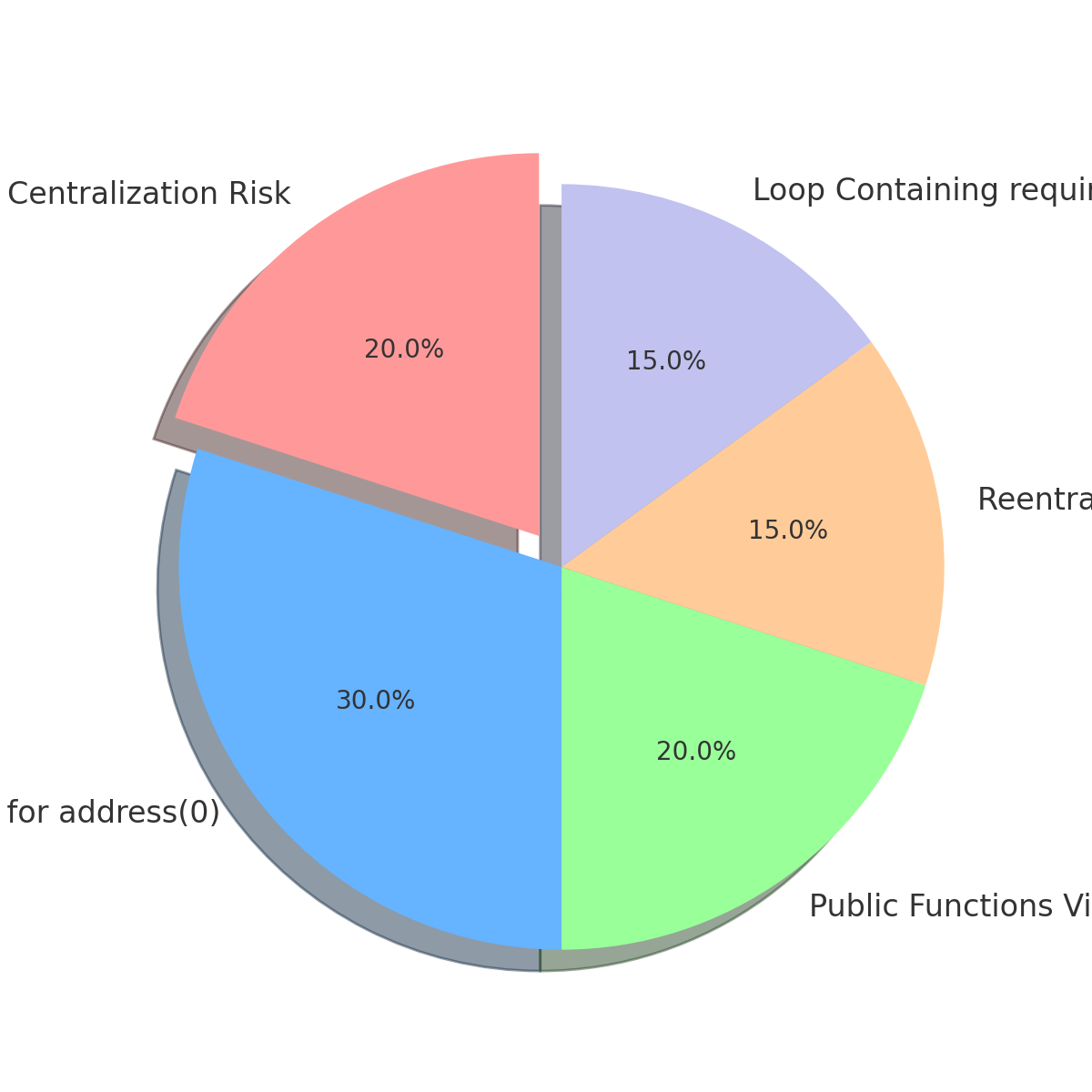
## 4.5 Loop Contains require/revert Statements

**Description**: Using require or revert within a loop can cause the entire transaction to fail if a single condition is not met. This should be avoided to allow transactions to handle failures gracefully.

**Severity**: Low

**Code Location**: StableCoin.sol, Lines 1825, 1855, 2007

## 4.5 Infographic: Distribution of Findings by Category



# 5. Conclusion

Overall, the audit of the AUDD ERC-20 token smart contract revealed no critical vulnerabilities. The majority of the findings were of low severity and primarily involved best practices for code optimization and centralization risks associated with admin privileges.  
By addressing the provided recommendations, the stability and security of the contract can be further enhanced. The AUDD token appears robust, with no high or critical issues discovered.