Smart Contract Security Audit Report

Abbreviations

Name	Description
Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
Binance Smart Chain	A blockchain network developed by Binance, designed for fast and low-cost transactions, compatible with Ethereum's EVM, supporting smart contracts and DeFi applications.
Tron	A blockchain platform focused on decentralized entertainment and content sharing, aiming for high throughput and scalability, with its native token TRX.
Polygon	A layer-2 scaling solution for Ethereum, enhancing transaction speed and reducing costs while maintaining security, often used for DeFi and NFTs.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
ERC20	ERC20 (BEP20 in Binance Smart Chain or *x*RP20 in other chains) tokens are blockchain-based assets that have value and can be sent and received. The primary difference with the primary coin is that instead of running on their own blockchain, ERC20 tokens are issued on a network that supports smart contracts such as Ethereum or Binance Smart Chain.

Table Of Contents

1 Executive Summary

This Security Audit Report was prepared by Slowmist on Apr 31, 2025. We would like to thank the USUD team for trusting Slowmist in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the USUD Token. The scope of the audit is limited to the source code files provided to Slowmist. Slowmist completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified no vulnerable issue in the smart contracts code.

2 Management Summary

2.1 About USUD Token

USUD (Umbra Sovereign US Dollar) is a next-generation zero-fee stablecoin engineered on the UMBRA blockchain, optimized for privacy, speed, and sovereignty. Designed with built-in mixer functions and vault-backed reserves, USUD enables censorship-resistant and gasless transactions while maintaining absolute user anonymity and audit transparency.

Unlike traditional stablecoins, USUD leverages the proprietary UmbraChain protocol to ensure transaction obfuscation, stealth liquidity routing, and cross-chain mirroring across ERC20, TRC20, and native Umbra protocols.

2.2 Audit Scope

This audit focused on identifying security flaws in code and the design of the USUD Token.

Contracts are implemented on Ethereum, Binance Smart Chain, Polygon Mainnet and Tron Mainnet. The details of the deployed smart contracts are listed in the table below:

Property	Value
Token Name	USUD
Creator	- ETH, BSC & POL: 0x2674F0FEC446E320733e0ec1c7b437318f7bba49 - Tron: TRFuDXa8DVzX8V89ZigQmqcsNmywtBZT8b
Contract address	- Ethereum: 0x8e2557f09aF4a44779499bF7559720464feBdDE4 - Binance Smart Chain: 0x20b7CF1FF8BccB0382175db5086a83ee54251b13 - POL: 0x8e2557f09aF4a44779499bF7559720464feBdDE4 - Tron: TYQxXJyc7BbCrP9fAuRiJfiAxAjcNV6uzb
Solidity Compiler Version	v0.8.20+commit.a1b79de6

2.3 Audit Methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and RK87, our in-house smart contract security analysis tool.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that were considered during the audit of the smart contract:

- Integer Overflow and Underflow
- Timestamp Dependence
- Race Conditions
- Transaction-Ordering Dependence
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Reentrancy
- Logic Flaws
- Explicit visibility of functions state variables (external, internal, private and public)

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.

2.4 Disclaimer

USUD acknowledges that the security services provided by Slowmist are conducted to the best of their professional abilities but cannot guarantee 100% coverage of all security vulnerabilities. USUD understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, USUD agrees that Slowmist shall not be held responsible or liable, and shall not be charged for any hacking incidents that occur due to security vulnerabilities not identified during the audit process

2.5 Acceptance Minute

This final report served by Slowmist to the USUD will be considered an Acceptance Minute. Within 7 days, if no further responses or reports are received from the USUD, the final report will be considered fully accepted by the USUD without the signature.

3 Audit Result

3.1 Overview

The USUD Token was written in Solidity language, with the required version to be ^0.8.0. The source code was written based on OpenZeppelin's libraries: ERC20, Ownable, ERC20Permit and AccessControl. The smart contract is ERC20 implementation that have some properties (as of the report writing time):

Property	**Value**
Name	USUD
Symbol	USUD
Decimals	6
Supply	1,000,000,000

3.2 Code Correctness

- The contract correctly implements ERC20 standards, including ERC20Permit for gasless approvals.
- The decimals() function is overridden to return 6, which is consistent with the intended design.
- The constructor properly validates the supplyRecipient address to prevent minting to the zero address.

3.3 Security Analysis

- Access Control: The DEFAULT_ADMIN_ROLE is assigned to the contract deployer, and only this role can mint or burn tokens. However, there is no mechanism to renounce or transfer this role, which could pose a centralization risk.
- Reentrancy: No reentrancy vulnerabilities were identified, as the contract uses OpenZeppelin’s secure implementations.
- Integer Overflows: The contract uses Solidity 0.8.0, which includes built-in overflow protection, mitigating this risk.

3.4 Gas Efficiency

- The contract is gas-efficient, leveraging OpenZeppelin’s optimized libraries.
- The use of ERC20Permit enables gasless approvals, reducing transaction costs for users

3.4 Findings

During the audit process, the audit team had identified no vulnerable issue in the smart contracts code.

4 Version History

Version	Date	Status/Change	Created by
0.1	Apr 31, 2025	Public Report	Slowmist

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

Github
https://github.com/slowmist