QuillAudits

AUDIT REPORT

April , 2025

For

ALLO

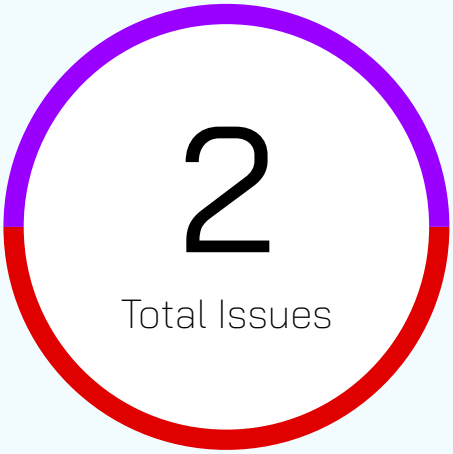
Table of Content

Executive Summary

Project name	Allo
Overview	RWAToken is a custom ERC20-like token with a fixed total supply of 10 billion tokens and a built-in blacklist functionality. Unlike standard ERC20 tokens, RWAToken does not inherit from OpenZeppelin's ERC20 contract but instead implements its own balance, transfer, and allowance management. Only the contract owner can mint tokens, ensuring controlled token distribution. The contract introduces a blacklist mechanism that allows the owner to block specific addresses from transferring or receiving tokens, providing an extra layer of compliance and security. The _transfer, _approve, and _spendAllowance functions handle token movement and approvals while enforcing blacklist restrictions.
Audit Scope	The scope of this Audit was to analyze the RWAToken Smart Contracts for quality, security, and correctness
Source Code link	https://bsc-scan.com/token/0xad5fC6e3bFAE1228aE4148A07495680E5dE4C62
Contracts in Scope	RWAToken
Branch	NA
Commit Hash	NA
Language	Solidity
Blockchain	BSC

Method	Manual Analysis, Functional Testing, Automated Testing
Review 1	1st April 2025
Updated Code Received	2nd April 2025
Review 2	2nd April
Fixed In	https://bscscan.com/address/0x965908b15F647722fB9676DB2457b3D1B64e79d8#code

Number of Issues per Severity



High	1 (50.00%)
Medium	0 (0.00%)
Low	0 (0.00%)
Informational	1 (50.00%)

Severity

Issues	High	Medium	Low	Informational
Open	0	0	0	0
Resolved	**1**	0	0	**1**
Acknowledged	0	0	0	0
Partially Resolved	0	0	0	0

Checked Vulnerabilities

- ✓ Access Management
- ✓ Arbitrary write to storage
- ✓ Centralization of control
- ✓ Ether theft
- ✓ Improper or missing events
- ✓ Logical issues and flaws
- ✓ Arithmetic Computations Correctness
- ✓ Race conditions/front running
- ✓ SWC Registry
- ✓ Re-entrancy
- ✓ Timestamp Dependence
- ✓ Gas Limit and Loops
- ✓ Exception Disorder
- ✓ Gasless Send
- ✓ Use of tx.origin
- ✓ Malicious libraries
- ✓ Compiler version not fixed
- ✓ Address hardcoded
- ✓ Divide before multiply
- ✓ Integer overflow/underflow
- ✓ ERC's conformance
- ✓ Dangerous strict equalities
- ✓ Tautology or contradiction
- ✓ Return values of low-level calls
- ✓ Missing Zero Address Validation
- ✓ Private modifier
- ✓ Revert/require functions
- ✓ Multiple Sends
- ✓ Using suicide
- ✓ Using delegatecall
- ✓ Upgradeable safety
- ✓ Using throw

- [x] Using inline assembly
- [x] Unsafe type inference
- [x] Style guide violation
- [x] Implicit visibility level.

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments, match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- Implementation of ERC standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools And Platforms Used For Audit

Remix IDE, Foundry, Solhint, Mythril, Slither, Solidity statistical analysis.

Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational Issues

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

Security vulnerabilities identified that must be resolved and are currently unresolved.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/ impact of the security issue, but are not completely resolved.

High Severity Issues

A blacklisted spender can successfully call the transferFrom function

Resolved

Path

RWAToken

Function

transferFrom

Description

Addresses that are given approval before they were blacklisted can still successfully call the transferFrom function. The issue is due to an insufficient check in the transferFrom function to confirm that the caller (msg.sender), is also not a blacklisted address.

```
function transferFrom(
    address from,
    address to,
    uint256 amount
) public notBlacklisted(from) notBlacklisted(to) returns (bool) {
    _spendAllowance(from, msg.sender, amount);
    _transfer(from, to, amount);
    return true;
}
```

Recommendation

add the notBlacklisted check for msg.sender on the transferFrom function

POC

1. Alice approves Bob 200 RWA token
2. Bob was blacklisted later by the contract owner
3. Bob notices he cannot spend his possessed tokens but has 200 RWA token allowance from Alice.
4. Bob calls successfully the transferFrom function with Alice as from and his new address that has no blacklist

Teams Comment

Allo Team removed the Blacklist function

Informational Severity Issues

Use of Floating Solidity Version

Resolved

Path

RWAToken

Function

pragma solidity ^0.8.26;

Description

The contract uses a floating solidity version to deploy the contract. This implies that the compiler version can choose to use from version 0.8.26 up to the latest which could cause deployment to use the latest version with unrecovered or known issues.

Recommendation

Use a fixed solidity pragma version that has been tested.

General Recommendation

Resolved

Description

This is a custom ERC20 token created to serve as a RWA token for the Allo protocol. It was designed so that the circulating supply is tracked while the total supply handles the absolute maximum of the RWA token. However, it is important to mention where this token differs from the standard token. Total supply for standard token tracks the existing supply of token in circulation. For standard tokens with max cap, there is a utility library that ensures that total supply does not exceed the max cap. In the case of Allo, users might consider total supply as the tokens in supply rather than see the circulating supply handling this. We recommend to retain totalSupply as the amount of tokens in supply, while utilizing the cap utils from Openzeppelin library. This way every function abides to the standard ERC20 token and the blacklist is done in the _update function, which is invoked in almost all functions.

Functional Tests

Some of the tests performed are mentioned below:

- Should check for all possible reverts for non-blacklisted and blacklisted address
- Should check that a blacklisted address cannot send or receive tokens
- Should confirm that circulating supply does not exceed total supply.

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of Allo. We performed our audit according to the procedure described above.

issues of High and informational severity issues were found.

Disclaimer

At QuillAudits, we have spent years helping projects strengthen their smart contract security. However, security is not a one-time event—threats evolve, and so do attack vectors. Our audit provides a security assessment based on the best industry practices at the time of review, identifying known vulnerabilities in the received smart contract source code.

This report does not serve as a security guarantee, investment advice, or an endorsement of any platform. It reflects our findings based on the provided code at the time of analysis and may no longer be relevant after any modifications. The presence of an audit does not imply that the contract is free of vulnerabilities or fully secure.

While we have conducted a thorough review, security is an ongoing process. We strongly recommend multiple independent audits, continuous monitoring, and a public bug bounty program to enhance resilience against emerging threats.

Stay proactive. Stay secure.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem

7+ Years of Expertise	**1M+** Lines of Code Audited
$30B+ Secured in Digital Assets	**1400+** Projects Secured

Follow Our Journey

AUDIT REPORT

April , 2025

For

ALLO

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillaudits.com