QuillAudits

AUDIT REPORT

January 2026

For

Kingz

Table of Content

Executive Summary

Project Name	KINGZ
Protocol Type	ERC20 Token
Project URL	https://kingz.win
Overview	This contract is an ERC-20 token with a presale trading lock. Transfers are unrestricted between normal wallets during presale, but interactions with marked restricted addresses (like DEX routers or pools) are blocked unless one side is exempt. The owner manages these lists during presale, and once trading is enabled, all restrictions are permanently removed and the token functions as a standard ERC-20.
Audit Scope	The scope of this Audit was to analyze the KINGZ Smart Contracts for quality, security, and correctness.
Source Code link	https://etherscan.io/address/0x7016f537cfbc5fc3078f3fed35d24415889c5ea2#code
Contracts in Scope	KingzToken.sol
Language	Solidity
Blockchain	EVM
Method	Manual Analysis, Functional Testing, Automated Testing
Review 1	13th January 2026

Verify the Authenticity of Report on QuillAudits Leaderboard:

https://www.quillaudits.com/leaderboard

Number of Issues per Severity

🟥	Critical	0(0.0%)
🟥	High	0(0.0%)
🟧	Medium	0(0.0%)
🟨	Low	1 (100%)
🟪	Informational	0(0.0%)

Severity

Issues	🟥 Critical	🟥 High	🟧 Medium	🟨 Low	🟪 Informational
Open	0	0	0	0	0
Acknowledged	0	0	0	1	0
Partially Resolved	0	0	0	0	0
Resolved	0	0	0	0	0

Summary of Issues

Issue No.	Issue Title	Severity	Status
1	Consider using Ownable2Step	Low	Acknowledged

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
- ✓ Using inline assembly
- ✓ Style guide violation
- ✓ Unsafe type inference
- ✓ 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 Static Analysis.

Types of Severity

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

🟥 Critical: Immediate and Catastrophic Impact

Critical issues are the ones that an attacker could exploit with relative ease, potentially leading to an immediate and complete loss of user funds, a total takeover of the protocol's functionality, or other catastrophic failures. Critical vulnerabilities are non-negotiable; they absolutely must be fixed.

🟥 High (H): Significant Risk of Major Loss or Compromise

High-severity issues represent serious weaknesses that could result in significant financial losses for users, major malfunctions within the protocol, or substantial compromise of its intended operations. While exploiting these vulnerabilities might require specific conditions to be met or a moderate level of technical skill, the potential damage is considerable. These findings are critical and should be addressed and resolved thoroughly before the contract is put into the Mainnet.

🟧 Medium (M): Potential for Moderate Harm Under Specific Circumstances

Medium-severity bugs are loopholes in the protocol that could lead to moderate financial losses or partial disruptions of the protocol's intended behavior. However, exploiting these vulnerabilities typically requires more specific and less common conditions to occur, and the overall impact is generally lower compared to high or critical issues. While not as immediately threatening, it's still highly recommended to address these findings to enhance the contract's robustness and prevent potential problems down the line.

🟨 Low (L): Minor Imperfections with Limited Repercussions

Low-severity issues are essentially minor imperfections in the smart contract that have a limited impact on user funds or the core functionality of the protocol. Exploiting these would usually require very specific and unlikely scenarios and would yield minimal gain for an attacker. While these findings don't pose an immediate threat, addressing them when feasible can contribute to a more polished and well-maintained codebase.

🟪 Informational (I): Opportunities for Improvement, Not Immediate Risks

Informational findings aren't security vulnerabilities in the traditional sense. Instead, they highlight areas related to the clarity and efficiency of the code, gas optimization, the quality of documentation, or adherence to best development practices. These findings don't represent any immediate risk to the security or functionality of the contract but offer valuable insights for improving its overall quality and maintainability. Addressing these is optional but often beneficial for long-term health and clarity.

Types of Issues

Open Security vulnerabilities identified that must be resolved and are currently unresolved.	**Resolved** These are the issues identified in the initial audit and have been successfully fixed.
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.

Severity Matrix

Likelihood \ Impact	🟥 High	🟧 Medium	🟨 Low
🟥 High	Critical	High	Medium
🟧 Medium	High	Medium	Low
🟨 Low	Medium	Low	Low

Impact

- **High** - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- **Low** - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

Low Severity Issues

Consider using Ownable2Step

Acknowledged

Path

KingzToken.sol

Description

The project intends to use Ownable2Step semantics (safe 2-step ownership transfer), but currently inherits from Ownable, not Ownable2Step.
 This means the owner can instantly transfer ownership in one transaction defeating the intended Two-Step security model documented in comments and project requirements.

If the owner key is compromised or misused, privileged control can be reassigned instantly without the beneficiary's acceptance.

Impact

Low

Likelihood

Low

Recommendation

Replace Ownable with Ownable2Step to enforce a safer two-step ownership transfer process.

Functional Tests

Some of the tests performed are mentioned below:

- ✓ should mint total supply to owner on deployment
- ✓ should allow only owner to enable trading
- ✓ should revert if enableTrading is called twice
- ✓ should allow normal wallet to wallet transfers during presale
- ✓ should have trading disabled by default

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.

Threat Model

1. External Dependencies & Trust Boundaries

This section enumerates everything the protocol does not fully control.

1.1 External Dependencies Table

Dependency	Type	How It Is Used	Trust Assumption	Associated Risks
ERC20.sol	Library/Base Contract	Core balance, transfer, allowance logic	Correct implementation of ERC20 invariants	N/A
OpenZeppelin ERC20Burnable	Library	Allows holders to burn tokens	Burn logic does not bypass restrictions	Unexpected Supply reduction
Ownable	Library	Owner access control	Owner key remains secure	Centralization, key compromise

2. Entry & Exit Points Analysis (Function-Level)

This is the core of the threat model. Every externally callable function must appear here.

2.1 Function Entry / Exit Table

Each function gets one table.

Contract Name	Function	Category
KingzToken.sol	constructor()	**What this function can do** Permanently lift all transfer restrictions **What this function cannot/should not do** Mint again post-deployment **Main invariant(s)** Total supply is fixed at deployment **Access level** Implicit restriction (once at deployment)
	enableTrading()	**What this function can do** Permanently lift all transfer restrictions **What this function cannot/should not do** Disable trading once enabled **Main invariant(s)** Trading enabled exactly once **Access level** Implicit restriction (once at deployment)
	setRestricted(address[], bool[])	**What this function can do** Block specific addresses during presale **What this function cannot/should not do** Restrict owner, bypass array integrity **Main invariant(s)** Owner cannot self-restrict **Access level** Owner-only

Contract Name	Function	Category
	setExempt(address[], bool[])	**What this function can do** Allow addresses to bypass presale restrictions **What this function cannot/should not do** Remove owner exemption **Main invariant(s)** Owner always exempt **Access level** Owner-only

3. Asset Flow Mapping (Critical Contracts)

This section identifies where money actually lives and how it moves.

3.1 Asset-Holding Contracts

List only contracts that custody value.

Contract	Asset Type	Custodied Assets
Kingz.sol	ERC20	Entire KINGZ supply

3.2 Asset Entry & Exit Functions

This table maps money movement, which is where most exploits happen.

Contract	Function	Asset In	Asset Out	Caller	Risk Notes
KingzToken.sol	Constructor	-	KINGZ	Deployer	Centralized initial supply
KingzToken.sol	transfer	KINGZ	KINGZ	Public	Restriction enforcement
KingzToken.sol	transferFrom	KINGZ	KINGZ	Public	Allowance misuse
KingzToken.sol	burn	KINGZ	-		Supply reduction

Note to Users/Trust Assumptions

- **Centralization Risk**

single-address control (high): Because the entire supply was minted to one recipient address at deployment, that address controls the entire circulating supply until tokens are distributed. If that address is a team, exchange, or a private wallet, they can move or sell tokens at any time. Treat tokens with a single initial holder as high-risk for sudden dumps. Verify who controls the recipient.

This audit works under the assumption that the owner role is trusted.

Closing Summary

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

No critical issues in kingz token, just 1 issue of Low severity was found.KINGZ team acknowledged the mentioned issue.

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 leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With seven years of expertise, we've secured over 1400 projects globally, averting over $3 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.

QuillAudits

7+ Years of Expertise	**1M+** Lines of Code Audited
50+ Chains Supported	**1400+** Projects Secured

Follow Our Journey

AUDIT REPORT

January 2026

For

Kingz

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillaudits.com