Boost TokenDistributor EVM

Security Audit Report

Prepared by

OKX Web3 Audit Team

25 11 2025

Contents

1. Overview

1.1 Project Introduction

Boost TokenDistributor EVM is a sophisticated token distribution platform built with security, gas efficiency, and scalability in mind. The platform supports both ERC20 tokens and native tokens (ETH) for flexible distribution campaigns.

1.2 Audit Summary

Ecosystem	EVM
Language	Solidity
Repository	https://github.com/okxlabs/Boost-TokenDistributor-EVM/tree/bddf501d7584ecd7a6d8bcd61989ee5171e4103a
Base Commit	bddf501d7584ecd7a6d8bcd61989ee5171e4103a
Final Commit	bddf501d7584ecd7a6d8bcd61989ee5171e4103a

1.3 Audit Scope

```
contracts/
├── DistributorFactory.sol
└── TokenDistributor.sol
```

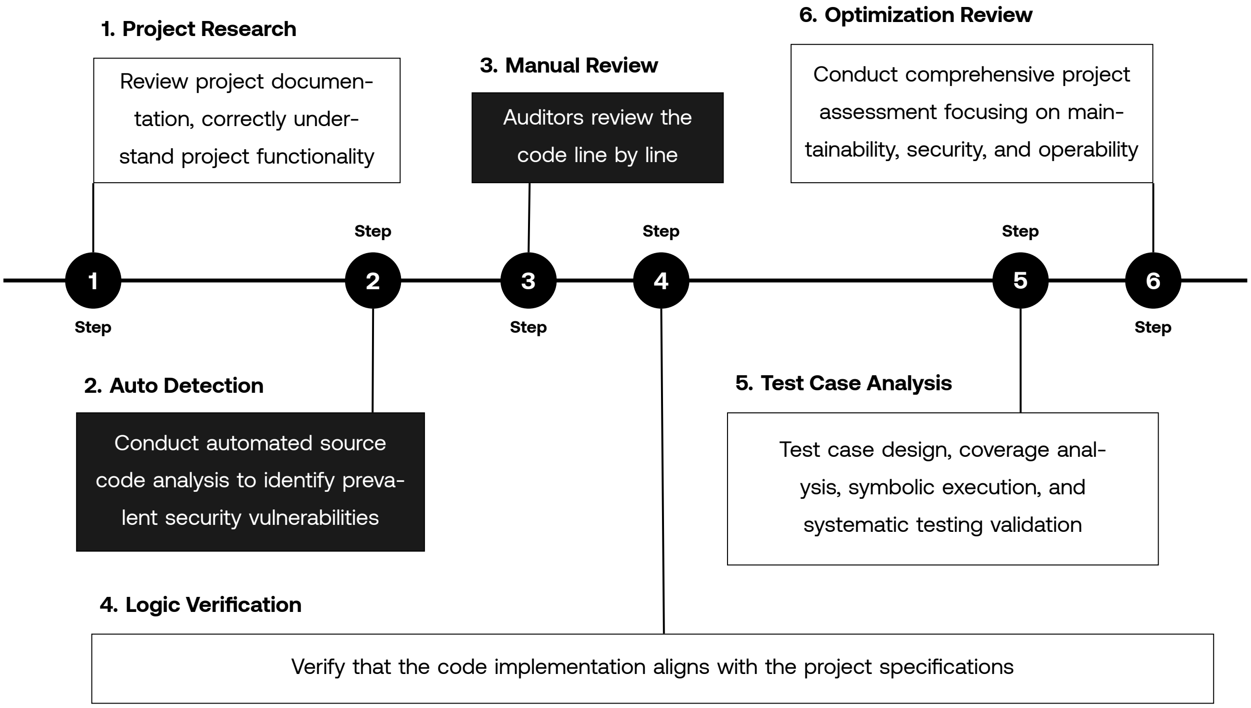
2. Audit Summary

2.1 Audit Methodology

The audit team conducted comprehensive analysis of the contract code through deep understanding of the project's design purpose, operating principles, and implementation methods. By mapping function call relationships, potential security vulnerabilities were systematically identified, with detailed problem descriptions and corresponding remediation recommendations provided.

2.2 Audit Process

The smart contract security audit follows a 6-phase process: Project Research, Automated Detection, Manual Review, Logic Verification, Test Case Analysis, and Optimization Review. During manual auditing, auditors perform comprehensive code review to identify vulnerabilities and provide detailed solutions. After completing all phases, the lead auditor communicates findings with the project team. Following the team's responses, we deliver final audit reports to the project team.


1. Project Research

Review project documentation, correctly understand project functionality

2. Auto Detection

Conduct automated source code analysis to identify prevalent security vulnerabilities

3. Manual Review

Auditors review the code line by line

4. Logic Verification

Verify that the code implementation aligns with the project specifications

5. Test Case Analysis

Test case design, coverage analysis, symbolic execution, and systematic testing validation

6. Optimization Review

Conduct comprehensive project assessment focusing on maintainability, security, and operability

2.3 Risk Classification and Description

Risk items are classified into 5 levels: Critical, High, Medium, Low, and Informational. Critical risks require immediate resolution and re-audit before final report delivery; unresolved critical risks result in audit failure.

Risk Level	Risk Description
Critical	Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
High	High risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
Medium	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
Low	Low risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
Informational	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

2.4 Results

The audit results are divided into two parts: one part is the vulnerability summary of the project audit, and the other part is the detailed vulnerability list.

Vulnerability Summary

Critical	High	Medium	Low	Informational	Total
0	0	0	1	0	1

Vulnerability list

No.	Severity	Vulnerability	Category	Status
1	Low	Centralization Risk	Centralization	Confirmed

Status Definitions

- **Open**: The audit team has notified the project team of the vulnerability, but no reasonable remediation has been implemented.
- **Fixed**: The project team has addressed the vulnerability and the fix has been verified by the audit team.

- **Confirmed**: The project team has confirmed awareness of the vulnerability risk but considers it controllable.

3. Vulnerabilities

This section outlines the risk items identified through manual review and auditing tools. Each item includes the specific file path and code location, along with the assigned risk level.

3.1 Low - Centralization Risk

Location	File	Category	Status	Severity
Multiple code instances	TokenDistributor.sol	Centralization	Confirmed	Low

Description

The current contract implementation relies on two administrator privileges: Owner and Operator. The Owner can withdraw remaining tokens from the contract after the distribution ends and before the set startTime. The Operator can set the airdrop start time and set and modify the Merkle root for claiming the airdrop. The normal operation of the contract depends on the secure management of these administrators. If these administrators are not properly protected, it will affect the normal operation of the contract and cause asset loss.

Related Code

```solidity
// Code snippet showing the vulnerability
function setTime(uint256 _startTime, uint256 _duration) external onlyOperator {
    if (_duration == 0 || _duration > MAX_DURATION) revert InvalidDuration();
    if (_startTime <= block.timestamp) revert InvalidTime();
    if (_startTime > block.timestamp + MAX_START_TIME) revert InvalidTime();
    if (block.timestamp >= startTime && block.timestamp <= endTime) revert AlreadyStarted();

    startTime = uint64(_startTime);
    endTime = uint64(_startTime + _duration);

    emit TimeSet(startTime, endTime);
}

function setMerkleRoot(bytes32 _merkleRoot) external onlyOperator {
    if (_merkleRoot == bytes32(0)) revert InvalidRoot();
    merkleRoot = _merkleRoot;

    emit MerkleRootSet(_merkleRoot);
}

function withdraw() external onlyOwner {
    // Check if distribution has ended or not set the startTime
    if (block.timestamp <= endTime) revert InvalidTime();

    uint256 balance = getBalance();
    if (balance == 0) revert NoTokens();

    transfer(msg.sender, balance);

    emit Withdrawn(msg.sender, balance);
}
```

Recommendation

The relevant administrator privileges should be properly delegated to the asset management team to manage and to record centralized risks for users.

Project Team Feedback

Team Response	The relevant permissions will be managed through the asset management platform.
Re-audit Result	Accept

4. Disclaimer

This audit report only covers the specific audit types stated herein. We assume no responsibility for unknown security vulnerabilities outside this scope.

We rely on audit reports issued before existing attacks or vulnerabilities are published. For future or new vulnerabilities, we cannot guarantee project security impact and assume no responsibility.

Our security audit analysis should be based on documents provided by the project before report release (including contract code). These materials should not contain false information, tampering, deletion, or concealment. If provided materials are false, inaccurate, missing, tampered, deleted, concealed, or modified after report release, we assume no responsibility for resulting losses and adverse effects.

The project team should understand our audit report is based on provided materials and current technical capabilities. Due to institutional technical limitations, our report may not detect all risks. We encourage continued testing and auditing by the development team and stakeholders.

The project team must ensure compliance with applicable laws and regulations.

The audit report is for reference only. Its content, acquisition methods, usage, and related services cannot serve as basis for investment, taxation, legal, regulatory, or construction decisions. Without our prior written consent, the project team may not reference, cite, display, or distribute report content to third parties. Any resulting losses shall be borne by the project team.

This report does not cover contract compiler bugs or scope beyond programming languages. Smart contract risks from underlying vulnerabilities should be borne by the project team.

Force majeure includes unforeseeable, unavoidable events like wars, natural disasters, strikes, epidemics, and legal/regulatory changes preventing contract performance. When occurring, neither party breaches contract obligations. For unaffected economic responsibilities, the project team should pay for completed work.

5. About OKX Web3 Audit Team

OKX Web3 Audit Team specializes in blockchain security with expertise in smart contract auditing, token security assessment, and Web3 security tool development. We provide comprehensive security solutions for OKX's internal Web3 projects, conduct pre-listing token audits, and develop security tools to protect OKX Web3 wallet users. Our team combines automated analysis with manual review to deliver thorough security assessments and maintain the highest security standards in the Web3 ecosystem.