

Boost Solana

Security Audit Report

prepared by

**OKX Web3
Audit Team**

2025.08.07



Contents

1. Overview

- 1.1 Project Introduction
- 1.2 Audit Summary
- 1.3 Audit Scope

2. Audit Summary

- 2.1 Audit Methodology
- 2.2 Audit Process
- 2.3 Risk Classification and Description
- 2.4 Vulnerability Checklist

3. Vulnerabilities

- 3.1 Low - Centralization Risk
- 3.2 Info - Redundant Code

4. Disclaimer

5. About OKX Web3 Audit Team

1. Overview

1.1 About Boost - TokenDistributor (Solana)

TokenDistributor (Solana) is a decentralized token distribution system built on Solana. It implements a Merkle Tree-based airdrop mechanism, enabling project teams to create rule-compliant token distribution campaigns with efficient and secure delivery.

1.2 Audit Summary

Ecosystem	Solana	Language	Rust
Repository	https://github.com/okxlabs/Boost-TokenDistributor-Solana		
Base Commit	0b004641121ceda581b7169f4f18dd8594d77f85		
Final Commit	0b004641121ceda581b7169f4f18dd8594d77f85		

1.3 Audit Scope

```
src
├── lib.rs
├── error.rs
├── constants.rs
├── event.rs
├── instructions/
│   ├── mod.rs
│   ├── create_distributor.rs
│   ├── set_time.rs
│   ├── set_merkle_root.rs
│   ├── claim.rs
│   ├── withdraw.rs
│   └── close_claim_status.rs
├── state/
│   ├── mod.rs
│   ├── distributor_state.rs
│   ├── claim_state.rs
│   └── nonce_state.rs
└── utils/
    ├── mod.rs
    ├── token.rs
    └── verify.rs
```

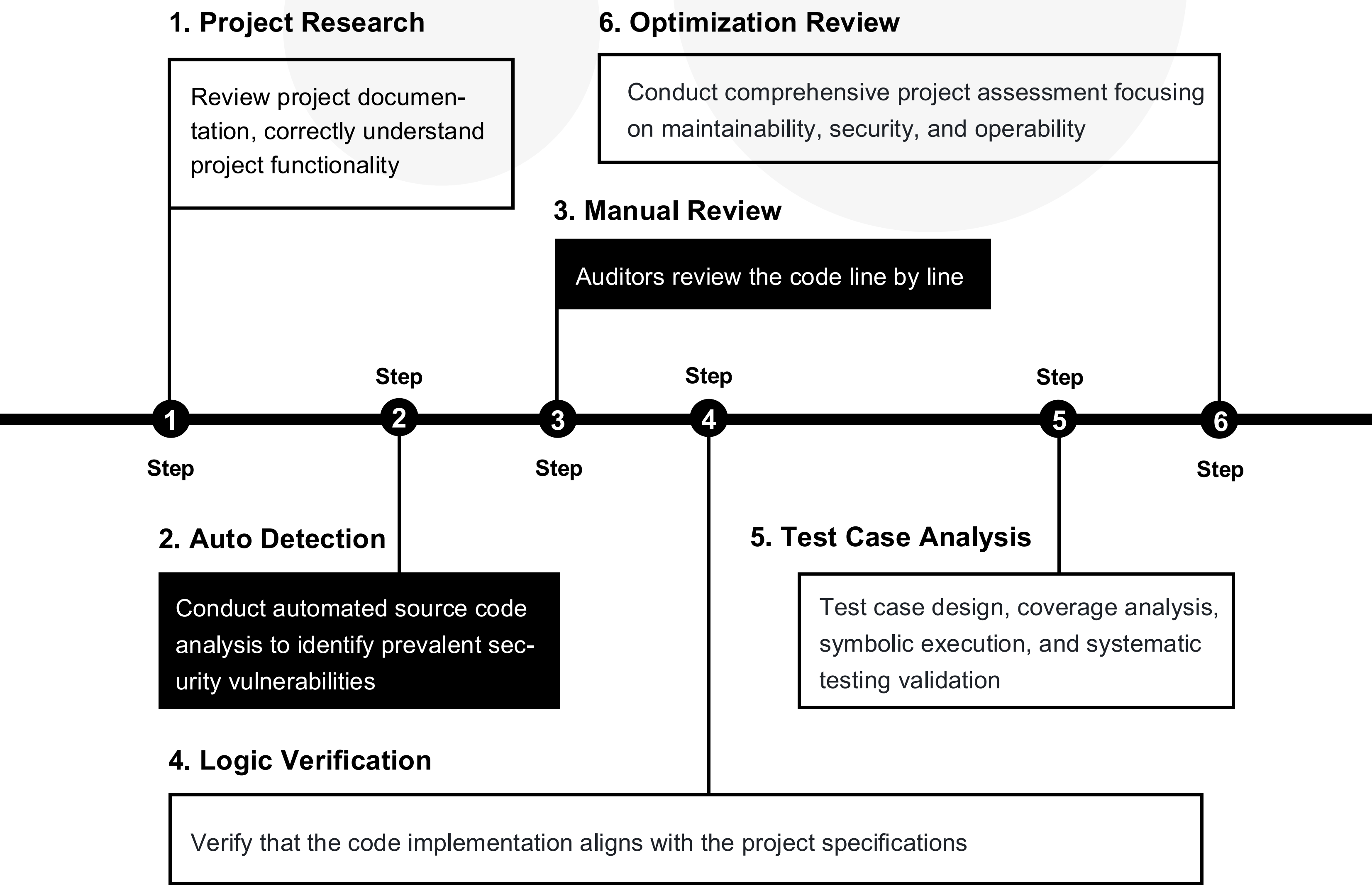
2. Audit Summary

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.



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. High risks must be addressed but are less urgent; failure to resolve also results in audit failure. Medium risks indicate potential exposure and require clear documentation of project team notification and response status without affecting report delivery. Low risks and informational items involve compliance or code detail issues that may be deferred without impacting report delivery.

Risk Level	Icon	Risk Description
Critical		Fatal risks requiring immediate resolution
High		High-risk vulnerabilities that will cause similar issues, must be resolved
Medium		Medium-risk vulnerabilities with potential impact, should be resolved
Low		Low-risk issues with improper handling or warning triggers, can be deferred
Informational		Optimization opportunities, deferrable but recommended for resolution

2.4 Vulnerability Checklist

The vulnerability checklist is 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	1	2

- **Vulnerability list:**

No.	Severity	Vulnerability	Category	Status
1	Low	Centralization Risk	Centralization Risk	Confirmed
2	Info	Redundant Code	Coding	Fixed

- **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. Detailed descriptions of the risks, recommended remediation measures, and relevant code snippets are provided to help clearly illustrate each issue.

3.1 Centralization Risk

Location	File	Status	Severity
Line 60	set_merkle_root.rs	Confirmed	! Low

- Description**

The current contract implementation relies on two administrator permissions, Owner and Operator. The Owner can withdraw the remaining tokens from the contract after the distribution is completed/before setting the trigTime. The Operator can set the airdrop start time, set and modify the Merkle root for receiving airdrops. The normal operation of the contract depends on the safe operation of these administrators. If these administrators are not properly protected, it will affect the normal operation of the contract and cause asset losses.

- Related Code**

```
60 pub fn handle_set_merkle_root(  
61     ctx: Context<SetMerkleRoot>,  
62     merkle_root: [u8; 32],  
63 ) -> Result<()> {  
64     let distributor: &mut Account<'_, TokenDistributor> = &mut ctx.accounts.distributor;  
65  
66     // Validate that the merkle root is not empty  
67     // An empty merkle root would allow no valid claims  
68     require!(merkle_root != [0; 32], TokenDistributorError::InvalidMerkleRoot);  
69  
70     // Set the merkle root for claim verification  
71     distributor.merkle_root = merkle_root;  
72  
73     // Emit event for off-chain indexing and monitoring  
74     emit_cpi!(MerkleRootSet {  
75         distributor: distributor.key(),  
76         operator: ctx.accounts.operator.key(),  
77         merkle_root,  
78     });  
79  
80     Ok(())  
81 }
```


- **Recommendation**

Properly delegate relevant administrator permissions to the asset management group for management.

- **Project Team Feedback**

Team Response	Confirmed
Re-audit Result	Accept

3.2 Redundant Code

Location	File	Status	Severity
Line 133	create_distributor.rs	Fixed	! Info

- **Description**

The owner_nonce bump is not used in the code and does not need to be stored.

- **Related Code**

```
131 // Initialize nonce state on first use
132 if owner_nonce.nonce == 0 && owner_nonce.bump == 0 {
133     owner_nonce.bump = ctx.bumps.owner_nonce;
134 }
```

- **Recommendation**

It is recommended to remove the code that stores the owner_nonce bump.

- **Project Team Feedback**

Team Response	Fixed
Re-audit Result	Confirmed

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.

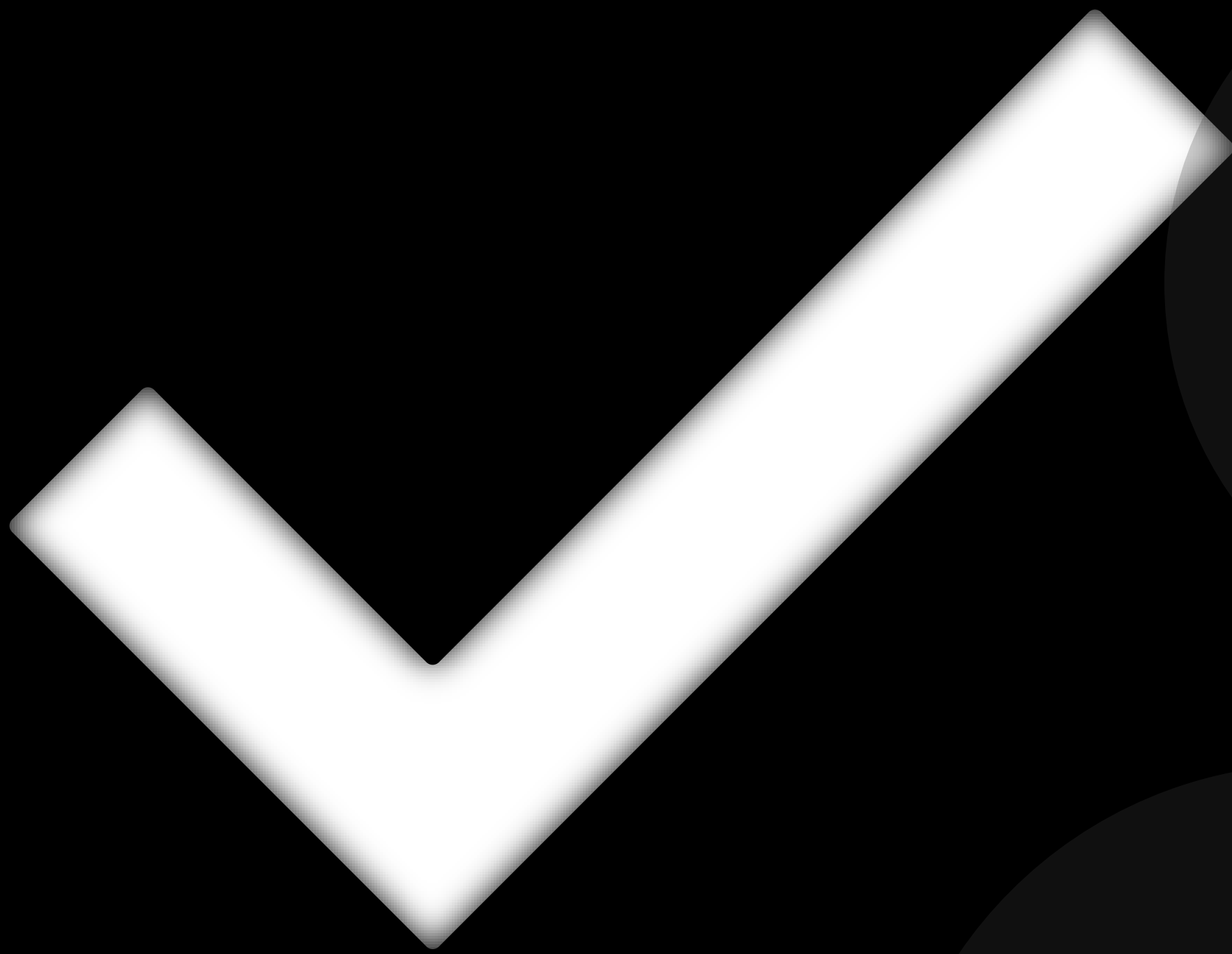
The audit report is for reference only. Its content, acquisition methods, usage, and related services cannot serve as basis for investment, taxation, legal, regulatory, or construction decisions. Without our prior written consent, the project team may not reference, cite, display, or distribute report 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 Us

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.



PASSED

This audit has been conducted to review the Boost Solana project's Rust-based smart contracts running on Solana chain, examining design, architecture, and implementation to identify potential vulnerabilities

OKX Web3 Audit Team