

OKX DEX Router EVM TEE Support

Security Audit Report

prepared by

**OKX Web3
Audit Team**

2025.08.15



Contents

1. Overview

- 1.1 Project Introduction
- 1.2 Audit Summary
- 1.3 Audit Scope
- 1.4 Revision History

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 - No deadline Validation in swapWrapToWithBaseRequest
- 3.2 Info - Duplicate Constant Definitions Across Contracts

4. Disclaimer

5. About OKX Web3 Audit Team

1. Overview

1.1 About OKX DEX Router

OKX DEX Router a decentralized exchange (DEX) aggregator project based on Ethereum, with the primary function of integrating multiple DEX protocols through smart contracts to provide users with the optimal token swap paths and prices.

1.2 Audit Summary

Ecosystem	EVM	Language	Solidity
Repository	https://github.com/okx/Web3-DEX-EVM/tree/feature/whd/tee-add-methods		
Base Commit	820743d0647ac1e2095efb36716dd1f1b98682fc		
Final Commit	88a6606b76b2acc3069bde8b692e41b1f5bfc334		

1.3 Audit Scope

contracts/8/
├── DexRouter.sol
├── UnxswapV3Router.sol
├── adapter/
├── interfaces/
│ ├── AbstractCommissionLib.sol
├── libraries/
│ ├── CommissionLib.sol
│ ├── UniswapTokenInfoHelper.sol
│ └── WrapETHSwap.sol

1.4 Revision History

Version	Date	Commit	Description
V1.0	20250815	88a6606	TEE interface support

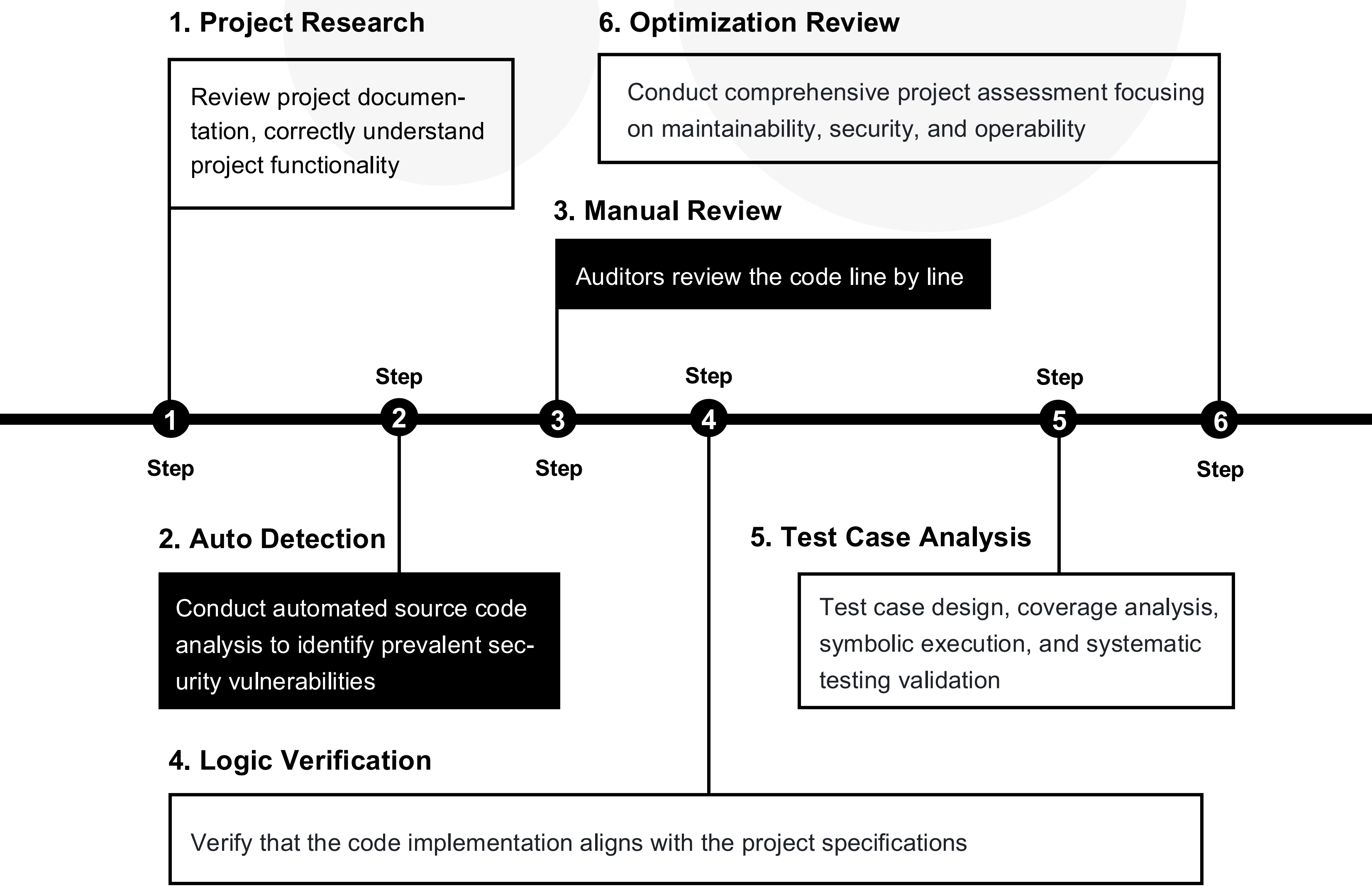
2. Audit Summary

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	No deadline Validation in swapWrapToWithBaseRequest()	Input Validation	Fixed
2	Info	Duplicate Constant Definitions Across Contracts	Coding Practice	Fixed

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 No deadline Validation in swapWrapToWithBaseRequest

Location	File	Status	Severity
Line 890	DexRouter.sol	Fixed	! Low

- Description**

The swapWrapToWithBaseRequest function accepts a BaseRequest parameter that includes a deadLine field to specify the expiration time of the transaction request. However, the current implementation does not perform any validation on the deadLine field, allowing the execution of potentially expired transaction requests. This could lead to the processing of stale or outdated requests, which may result in unintended financial consequences or unexpected behavior.

- Related Code**

```
886     function swapWrapToWithBaseRequest(  
887         uint256 orderId,  
888         address receiver,  
889         BaseRequest calldata baseRequest  
890     ) external payable {  
891         bool reversed;  
892         address fromTokenAddr = address(uint160(baseRequest.fromToken));  
893         if (fromTokenAddr == _ETH && baseRequest.toToken == _WETH) {  
894             reversed = false;  
895         } else if (fromTokenAddr == _WETH && baseRequest.toToken == _ETH) {  
896             reversed = true;  
897         } else {  
898             revert("SwapWrap: invalid token pair");  
899         }
```

- Recommendation**

Add a validation check to ensure the BaseRequest.deadLine has not expired before processing the transaction. This can be achieved by using an isExpired(baseRequest.deadLine) function.

• **Project Team Feedback**

Team Response	Fixed
Re-audit Result	Confirmed

3.2 Duplicate Constant Definitions Across Contracts

Location	File	Status	Severity
Lines 17-18	UnxswapV3Router.sol	Fixed	! Info

- Description**

The UnxswapV3Router and UniswapTokenInfoHelper contracts redundantly define two constants with identical values and purposes: one for identifying one-for-zero swaps and another for determining if WETH should be unwrapped to ETH. These constants, despite having slightly different names, serve the same function in both contracts, violating the DRY (Don't Repeat Yourself) principle. This duplication increases the risk of inconsistent updates and complicates code maintenance.

- Related Code**

```
14  contract UnxswapV3Router is IUniswapV3SwapCallback, CommonUtils {
15      using Address for address payable;
16
17      uint256 internal constant _ONE_FOR_ZERO_MASK = 1 << 255; // Mask for identifying if the swap is one-for-zero
18      uint256 private constant _WETH_UNWRAP_MASK = 1 << 253; // Mask for identifying if WETH should be unwrapped
19      bytes32 private constant _POOL_INIT_CODE_HASH =
20          0xe34f199b19b2b4f47f68442619d555527d244f78a3297ea89325f843f87b8b54; // Pool init code hash
21      bytes32 private constant _FF_FACTORY = 0xff1F98431c8aD98523631AE4a59f267346ea31F9840000000000000000;
22      // concatenation of token0(), token1() fee(), transfer() and claimTokens() selectors
```

```
13  abstract contract UniswapTokenInfoHelper is CommonUtils {
14      uint256 private constant _UNX_WETH_MASK =
15          0x4000000000000000000000000000000000000000000000000000000000000000;
16      uint256 internal constant _UNIV3_ONE_FOR_ZERO_MASK = 1 << 255; // Mask for identifying if the swap is one-for-zero
17      uint256 private constant _UNIV3_WETH_UNWRAP_MASK = 1 << 253; // Mask for identifying if WETH should be unwrapped
18
19      function _getUnxswapTokenInfo(bool sendValue, bytes32[] calldata pools)
20          internal
```

- Recommendation**

Refactor the codebase to eliminate duplicate constant definitions by consolidating them into a single location, such as a shared base contract or a dedicated constants library. This ensures consistency and simplifies maintenance.

- 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.

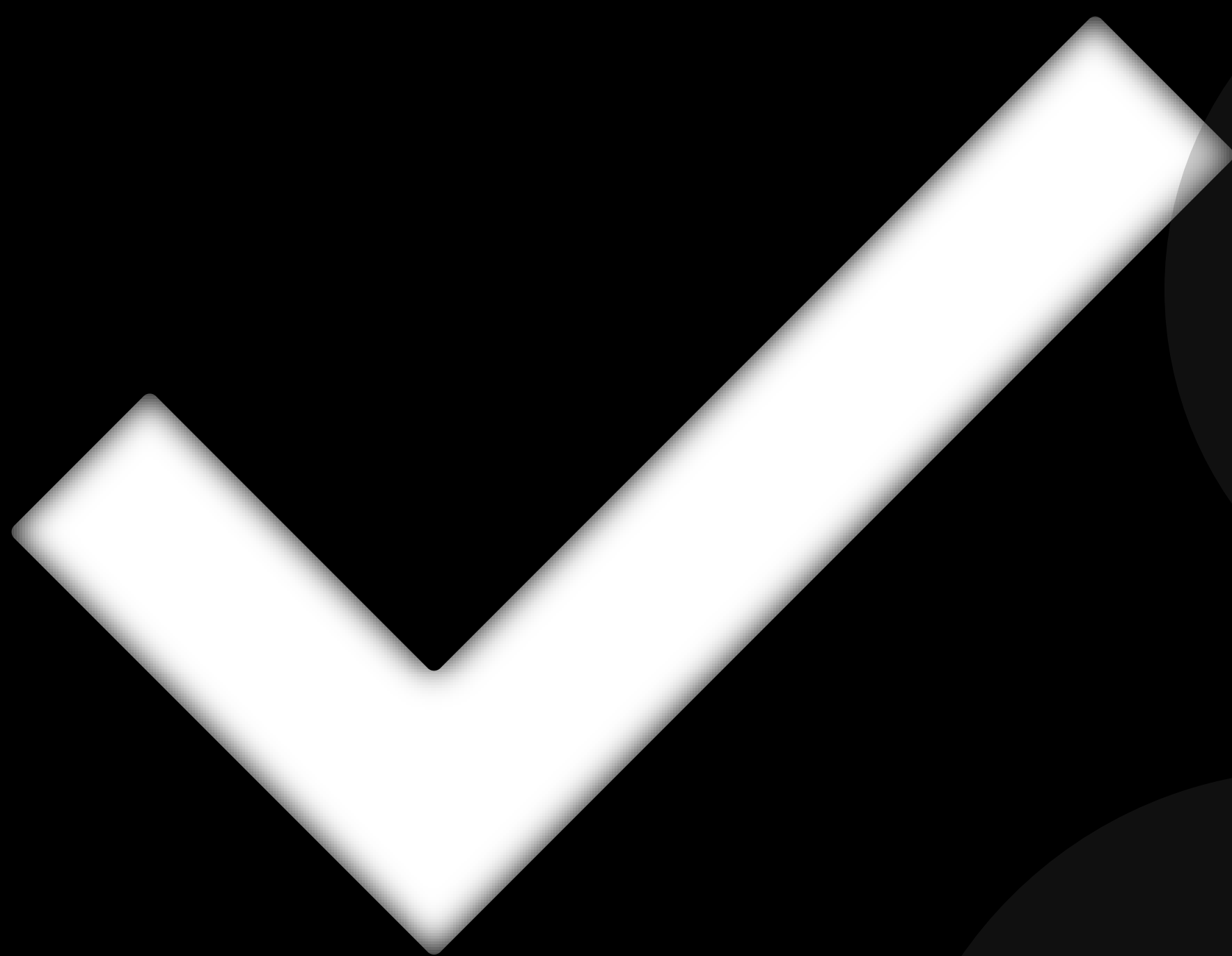
The audit report is for reference only. Its content, acquisition methods, usage, and related services cannot serve as basis for investment, taxation, legal, regulatory, or construction decisions. Without our prior written consent, the project team may not reference, cite, display, or distribute report 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 OKX DEX Router project's Solidity-based smart contracts running on EVM chains, examining design, architecture, and implementation to identify potential vulnerabilities

OKX Web3 Audit Team