OKX DEX Router Solana

Security Audit Report

prepared by

OKX Web3
Audit Team

2024.05.10

Contents

1. Overview

1.1 About OKX DEX Router

OKX DEX Router a decentralized exchange (DEX) aggregator project based on Solana, 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	Solana	Language	Rust
Repository			
Base Commit			
Final Commit			

1.3 Audit Scope

```
src
├── adapters
│   ├── aldrin.rs
│   ├── lifinity.rs
│   ├── meteora.rs
│   ├── mod.rs
│   ├── raydium.rs
│   ├── spl_token_swap.rs
│   ├── stable_swap.rs
│   ├── whirlpool.rs
│   └── [… other adapters]
├── constants.rs
├── error.rs
├── instructions
│   ├── commission_from_swap.rs
│   ├── commission_proxy_swap.rs
│   ├── commission_swap.rs
│   ├── common.rs
│   ├── from_swap.rs
│   ├── mod.rs
│   ├── proxy_swap.rs
│   └── swap.rs
├── lib.rs
└── utils.rs
    ├── mod.rs
    └── token.rs
```

1.4 Revision History

Version	Date	Commit	Description
V1.0	20240207	229bc2b	Base Audit

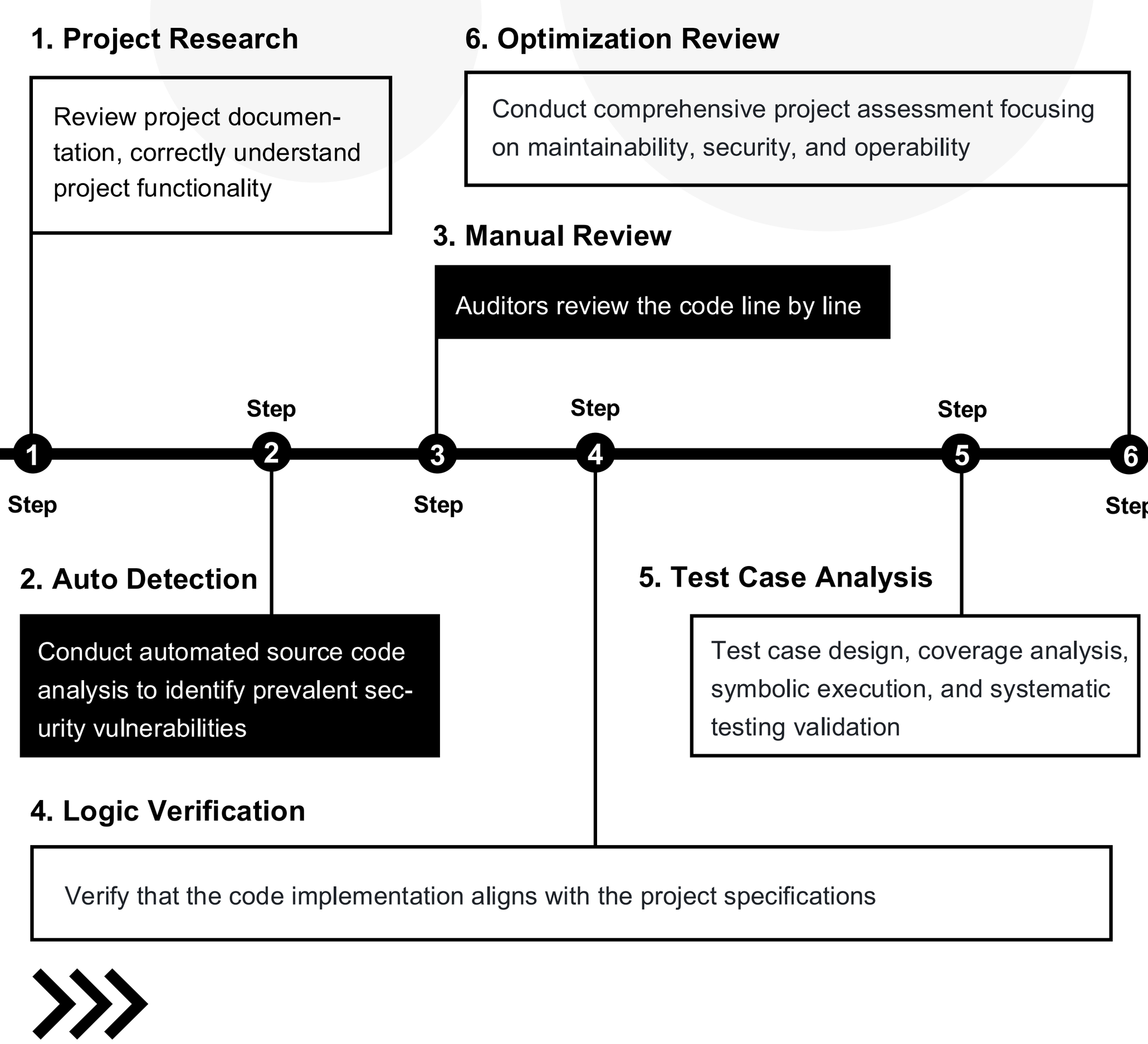
2. Audit Summary

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	3	4

- **Vulnerability list:**

No.	Severity	Vulnerability	Category	Status
1	Low	Single-Hop Destination Validation Bypass	Coding	Fixed
2	Info	Insufficient Memory Allocation for Instruction.data	Coding	Fixed
3	Info	Misleading Error Message for Equality Check	Coding	Fixed
4	Info	Optimizable Compute Unit	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 Single-Hop Destination Validation Bypass

Location	File	Status	Severity
Line 313-325	dexrouter.rs	Fixed	! Low

- **Description**

When the length of hops is 1, the value of j is 0, which satisfies both j == 0 and j == hops.len() - 1. If the if...else if... solution is used to check, the else if... branch will be skipped, and the consistency of to_account and destination_token_account cannot be guaranteed.

- **Related Code**

```
313    if j == 0 {
314
315        require!(
316            ctx.accounts.source_token_account.key() == hop_accounts.from_account,
317            ErrorCode::InvalidSourceTokenAccount
318        );
319    } else if j == hops.len() - 1 {
320
321        require!(
322            ctx.accounts.destination_token_account.key() == hop_accounts.to_account,
323            ErrorCode::InvalidDestinationTokenAccount
324        );
325    }
```

- **Recommendation**

It is recommended to modify the code logic to cover the scenario when the length of hops is 1, such as using if...if... for judgment.

- **Project Team Feedback**

Team Response	Fixed in commit a20505a
Re-audit Result	Confirmed

3.2 Insufficient Memory Allocation for Instruction.data

Location	File	Status	Severity
-	Multiple Files	Fixed	!Info

- **Description**

The value of size_of::<SplTokenSwapArgs>() is 16, so the with_capacity function allocates 16 bytes of space for data. However, in subsequent operations, 1 + 8 + 8 = 17 bytes of data are inserted into data.

- **Related Code**

```
153     let mut data = Vec::with_capacity(size_of::<SplTokenSwapArgs>());
154     data.push(1);
155     data.extend_from_slice(&amount_in.to_le_bytes());
156     data.extend_from_slice(&1u64.to_le_bytes());
```

- **Recommendation**

Therefore, it is recommended to directly allocate enough space when executing the with_capacity function to avoid reallocating memory, thereby improving code efficiency.

- **Project Team Feedback**

Team Response	Fixed in commit a20505a
Re-audit Result	Confirmed

3.3 Misleading Error Message for Equality Check

Location	File	Status	Severity
Line 100	dexrouter.rs	Fixed	!Info

- **Description**

The check is whether the two values are equal, but the error code is that the total amount must be less than or equal to the amount, which does not match the check.

- **Related Code**

```
 99   require!(
100       total_amounts == args.amount_in,
101       ErrorCode::TotalAmountsMustBeLessThanOrEqualToAmountIn
```

- **Recommendation**

It is recommended to change TotalAmountsMustBeLessThanOrEqualToAmountIn to TotalAmountsMustBeEqualToAmountIn.

- **Project Team Feedback**

Team Response	Fixed in commit a20505a
Re-audit Result	Confirmed

3.4 Optimizable Compute Unit

Location	File	Status	Severity
Line 120-121	dexrouter.rs	Fixed	!Info

- **Description**

The find_program_address() function internally uses a loop to call create_program_address() in order to locate a PDA account that does not on the Ed25519 curve. This process can result in significant CU consumption depending on the specific seeds provided.

- **Related Code**

```
80   pub fn handler<'a>(ctx: Context<'_, '_, 'a, 'a, SwapAccounts<'_>>, args: SwapArgs) -> Result<()> {
119
120      // get authority_pda & bump_seed
121      let (authority_pda: Pubkey, bump_seed: u8) = Pubkey::find_program_address(seeds: &[SEED_SA], ctx.program_id);
122
123      // Swap by Routes
124      let mut offset: usize = 0;
125      let remaining_accounts: &[AccountInfo<'a>] = ctx.remaining_accounts;
126      let signers_seeds: &[&[&[u8]]] = &[&[SEED_SA, &[bump_seed]]];
```

```
497      #[allow(clippy::same_item_push)]
498      pub fn try_find_program_address(seeds: &[&[u8]], program_id: &Pubkey) -> Option<(Pubkey, u8)> {
499          // Perform the calculation inline, calling this from within a program is
500          // not supported
501          #[cfg(not(target_os = "solana"))]
502          {
503              let mut bump_seed: [u8; 1] = [std::u8::MAX];
504              for _ in 0..std::u8::MAX {
505                  {
506                      let mut seeds_with_bump: Vec<&[u8]> = seeds.to_vec();
507                      seeds_with_bump.push(&bump_seed);
508                      match Self::create_program_address(&seeds_with_bump, program_id) {
509                          Ok(address: Pubkey) => return Some((address, bump_seed[0])),
510                          Err(PubkeyError::InvalidSeeds) => (),
511                          _ => break,
512                      }
513                  }
514                  bump_seed[0] -= 1;
515              }
516              None
517          }
```

- **Recommendation**

The PDA account can be computed off-chain, and then the bump seeds can be stored on-chain or passed in to avoid this additional cost.

- **Project Team Feedback**

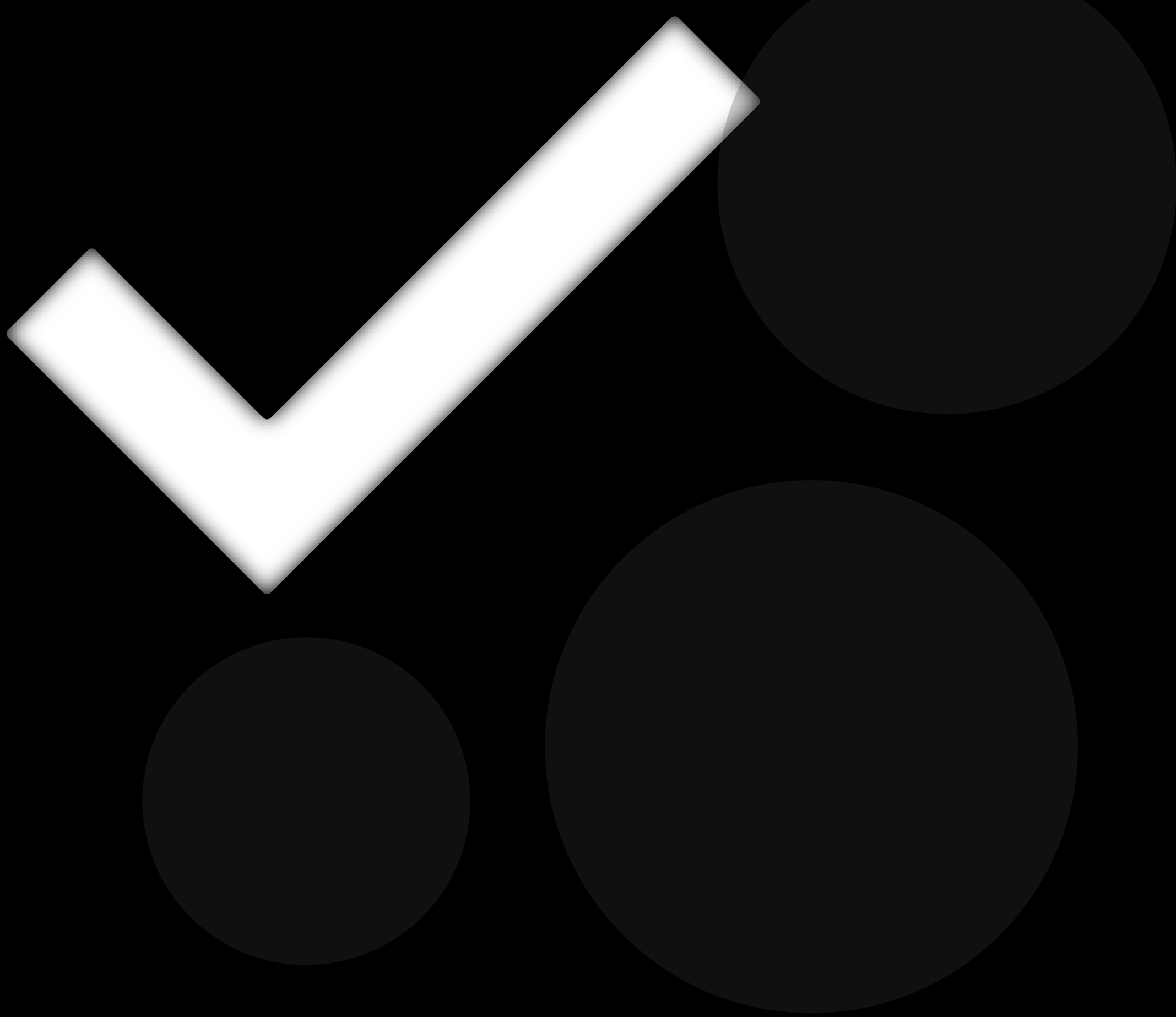
Team Response	Fixed in commit a20505a
Re-audit Result	Confirmed

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 Rust-based smart contracts running on Solana chain, examining design, architecture, and implementation to identify potential vulnerabilities

OKX Web3 Audit Team