DEX EVM RFQ

Security Audit Report

Prepared by

OKX Web3 Audit Team

28 11 2025

Contents

1. Overview

1.1 Project Introduction

The OKX DEX RFQ system is an on-chain protocol tailored for professional market makers, enabling them to provide efficient and competitive pricing across multiple supported chains. By combining off-chain quoting with secure on-chain settlement, the protocol is designed to deliver optimal trade execution for DeFi participants.

1.2 Audit Summary

Ecosystem	EVM
Language	Solidity
Repository	https://github.com/okxlabs/Web3-DEX-EVM-PMM.git
Base Commit	e9e88d4
Final Commit	49f9359

1.3 Audit Scope

```
src/
├─ EIP712.sol
├─ OrderRFQLib.sol
├─ PmmProtocol.sol
├─ helpers/
│  └─ AmountCalculator.sol
├─ interfaces/
│  ├─ IDaiLikePermit.sol
│  ├─ IPMMSettler.sol
│  ├─ IPermit2.sol
│  └─ IWETH.sol
└─ libraries/
   ├─ ECDSA.sol
   ├─ Errors.sol
   ├─ RevertReasonForwarder.sol
   └─ SafeERC20.sol
```

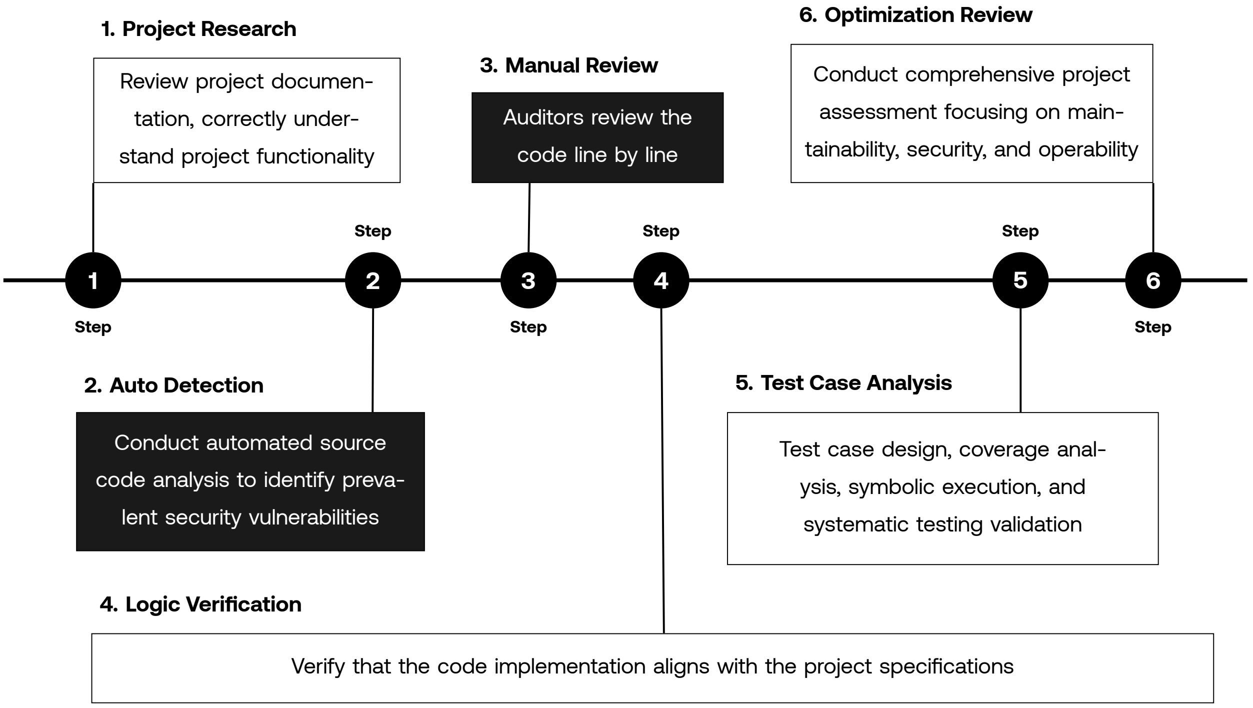
2. Audit Summary

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	2	2	4

Vulnerability list

No.	Severity	Vulnerability	Category	Status
1	Low	Possible DoS Risk in RFQ Order Handling	Denial of Service	Fixed
2	Low	Incorrect permitted.amount in Permit2 Signature Transfer	Business Logic	Fixed
3	Informational	Potential Overflow Issue in _fillOrderRFQTo with Permit2 Transfers	Business Logic	Fixed
4	Informational	Possible Reentrancy Risk in RFQ Order Handling	Time and State	Fixed

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 - Possible DoS Risk in RFQ Order Handling

Location	File	Category	Status	Severity
Line 246	PmmProtocol.sol	Denial of Service	Fixed	Low

Description

The PmmProtocol contract allows any address to act as the taker to fill an RFQ, and for each fill—even a very small partial fill, such as 1 wei—the corresponding rfqId is immediately marked as permanently invalid. This enables an attacker, after observing a legitimate match, to front-run it with a minimal amount, causing the entire order to be invalidated and all subsequent normal fills to fail. This results in a low-cost, repeatable DoS attack that severely impacts the availability and fairness of the matching process.

Related Code

```solidity
function _fillOrderRFQTo(
    OrderRFQLib.OrderRFQ memory order,
    uint256 flagsAndAmount,
    address target
) private returns (uint256 makerAmount, uint256 takerAmount) {
    if (target == address(0)) revert Errors.RFQ_ZeroTargetIsForbidden(order.rfqId);

    address maker = order.makerAddress;

    {
        // Stack too deep
        // Check time expiration
        uint256 expiration = order.expiry;
        if (expiration != 0 && block.timestamp > expiration)
            revert Errors.RFQ_OrderExpired(order.rfqId); // solhint-disable-line not-rely-on-time
        _invalidateOrder(maker, order.rfqId, 0);
    }
    ...
}

```

Recommendation

Project Team Feedback

3.2 Low - Incorrect permitted.amount in Permit2 Signature Transfer

Description

Related Code

```
function _fillOrderRFQTo(OrderRFQLib.OrderRFQ memory order, uint256 flagsAndAmount,
↪   address target)
    private
    returns (uint256 makerAmount, uint256 takerAmount)
{
    ...
    bool needUnwrap = order.makerAsset == address(_WETH) && flagsAndAmount &
    ↪   _UNWRAP_WETH_FLAG != 0;

    // Maker => Taker
    address receiver = needUnwrap ? address(this) : target;
    if (order.usePermit2) {
        if (order.permit2Signature.length > 0) {
            // permit2 signature based transfer
            IPermit2.PermitTransferFrom memory permitTransferFrom =
            ↪   IPermit2.PermitTransferFrom({
                permitted: IPermit2.TokenPermissions({token: order.makerAsset, amount:
                ↪   makerAmount}),
                nonce: order.rfqId,
                deadline: order.expiry
            });
            IPermit2.SignatureTransferDetails memory signatureTransferDetails =
                IPermit2.SignatureTransferDetails({to: receiver, requestedAmount:
                ↪   makerAmount});
            IPermit2(SafeERC20._PERMIT2).permitTransferFrom(
                permitTransferFrom, signatureTransferDetails, maker,
                ↪   order.permit2Signature
            );
        } else {
            // permit2 allowance based transfer
            IERC20(order.makerAsset).safeTransferFromPermit2(maker, receiver,
            ↪   makerAmount);
        }
    } else {
        IERC20(order.makerAsset).safeTransferFrom(maker, receiver, makerAmount);
    }
    ...
}
```

Recommendation

Project Team Feedback

3.3 Informational - Potential Overflow Issue in _fillOrderRFQTo with Permit2 Transfers

Description

Related Code

```
function _fillOrderRFQTo(
    OrderRFQLib.OrderRFQ memory order,
    uint256 flagsAndAmount,
    address target
) private returns (uint256 makerAmount, uint256 takerAmount) {
    ...
    // user: AMM->PMM
    {
        // Stack too deep
        uint256 orderMakerAmount = order.makerAmount;
        uint256 orderTakerAmount = order.takerAmount;
        uint256 amount = flagsAndAmount & _AMOUNT_MASK;
        // Compute partial fill if needed
        if (amount == 0) {
            // zero amount means whole order
            makerAmount = orderMakerAmount;
            takerAmount = orderTakerAmount;
        } else if (flagsAndAmount & _MAKER_AMOUNT_FLAG != 0) {
            if (amount > orderMakerAmount)
                revert Errors.RFQ_MakerAmountExceeded(order.rfqId);
            makerAmount = amount;
            takerAmount = AmountCalculator.getTakerAmount(
                orderMakerAmount,
                orderTakerAmount,
                makerAmount
            );
        } else {
            if (amount > orderTakerAmount)
                revert Errors.RFQ_TakerAmountExceeded(order.rfqId);
            takerAmount = amount;
            makerAmount = AmountCalculator.getMakerAmount(
                orderMakerAmount,
                orderTakerAmount,
                takerAmount
            );
        }
    }

    if (makerAmount == 0 || takerAmount == 0)
        revert Errors.RFQ_SwapWithZeroAmount(order.rfqId);

    bool needUnwrap = order.makerAsset == address(_WETH) &&
        flagsAndAmount & _UNWRAP_WETH_FLAG != 0;

    // Maker => Taker
```

Recommendation

Project Team Feedback

3.4 Informational - Possible Reentrancy Risk in RFQ Order Handling

Description

Related Code

```
function fillOrderRFQCompact(
    OrderRFQLib.OrderRFQ memory order,
    bytes32 r,
    bytes32 vs,
    uint256 flagsAndAmount
)
    external
    payable
    returns (
        uint256 filledMakerAmount,
        uint256 filledTakerAmount,
        bytes32 orderHash
    )
{
    ...
    (filledMakerAmount, filledTakerAmount) = _fillOrderRFQTo(
        order,
        flagsAndAmount,
        msg.sender
    );
    ...
}
```

Recommendation

Project Team Feedback

4. Disclaimer

5. About OKX Web3 Audit Team