AstraSec

Kodiak DEX
Security Audit Report

May 17, 2025

Contents

1 Introduction

1.1 About Kodiak DEX

1.2 Source Code

2 Overall Assessment

3 Vulnerability Summary

3.1 Overview

3.2 Security Level Reference

3.3 Vulnerability Details

4 Appendix

4.1 About AstraSec

4.2 Disclaimer

4.3 Contact

1 Introduction

1.1 About Kodiak DEX

Kodiak is Berachain's native liquidity hub, designed as a non-custodial decentralized exchange (DEX) that supports both concentrated and full-range Automated Market Makers (AMMs). It offers a vertically integrated platform combining trading, automated liquidity management, and incentivized farming. As the only DEX incubated by Berachain's Build-a-Bera accelerator, Kodiak enables seamless token launching, trading, and liquidity provision for any asset.

The audit was performed via the Hyacinth platform.

1.2 Source Code

The following source codes were reviewed during the audit:

- https://github.com/Kodiak-Finance/kodiak-periphery/pull/23
- CommitID: a67e673
- https://github.com/Kodiak-Finance/kodiak-core/pull/117
- CommitID: 2fd8d69

And these are the final versions representing all fixes implemented for the issues identified in the audit:

- https://github.com/Kodiak-Finance/kodiak-periphery/pull/23
- CommitID: dd381a1
- https://github.com/Kodiak-Finance/kodiak-core/pull/117
- CommitID: ce7a271

2 Overall Assessment

This report has been compiled to identify issues and vulnerabilities within the Kodiak DEX protocol. Throughout this audit, we identified a total of 3 issues spanning various severity levels. By employing auxiliary tool techniques to supplement our thorough manual code review, we have discovered the following findings.

Severity	Count	Acknowledged	Won't Do	Addressed
Critical	–	–	–	–
High	–	–	–	–
Medium	1	–	–	1
Low	2	2	–	–
Informational	–	–	–	–
Undetermined	–	–	–	–

3 Vulnerability Summary

3.1 Overview

Click on an issue to jump to it, or scroll down to see them all.

~~M-1~~ Unprotected Residual Tokens in KodiakExecutor::execute()

L-1 Incompatibility with Deflationary Tokens

L-2 Potential Risks Associated with Centralization

3.2 Security Level Reference

In web3 smart contract audits, vulnerabilities are typically classified into different severity levels based on the potential impact they can have on the security and functionality of the contract. Here are the definitions for critical-severity, high-severity, medium-severity, and low-severity vulnerabilities:

Severity	Acknowledged
C-X (Critical)	A severe security flaw with immediate and significant negative consequences. It poses high risks, such as unauthorized access, financial losses, or complete disruption of functionality. Requires immediate attention and remediation.
H-X (High)	Significant security issues that can lead to substantial risks. Although not as severe as critical vulnerabilities, they can still result in unauthorized access, manipulation of contract state, or financial losses. Prompt remediation is necessary.
M-X (Medium)	Moderately impactful security weaknesses that require attention and re-mediation. They may lead to limited unauthorized access, minor financial losses, or potential disruptions to functionality.
L-X (Low)	Minor security issues with limited impact. While they may not pose significant risks, it is still recommended to address them to maintain a robust and secure smart contract.
I-X (Informational)	Warnings and things to keep in mind when operating the protocol. No immediate action required.
U-X (Undetermined)	Identified security flaw requiring further investigation. Severity and impact need to be determined. Additional assessment and analysis are necessary.

3.3 Vulnerability Details

3.3.1 [M-1] Unprotected Residual Tokens in KodiakExecutor::execute()

TARGET	CATEGORY	IMPACT	LIKELIHOOD	STATUS
KodiakMetaRouter.sol	Business Logic	High	Low	Addressed

The KodiakExecutor::execute() function contains a critical oversight in its token handling logic where it fails to properly account for and return residual input tokens or intermediate tokens after swap operations. These stranded tokens become freely claimable by anyone. Malicious actors can monitor the contract for residual balances and extract them at will.

kodiak-periphery-meta-aggregator - KodiakMetaRouter.sol

```
241 function execute(
242     address inputToken,
243     uint256 amount,
244     address outputToken,
245     address router,
246     bytes calldata data
247 ) external {
248     require(msg.sender == metaRouter);
249     inputToken.safeApprove(router, amount);
250     (bool success, bytes memory returnData) = router.call(data);
251     if (!success) {
252         assembly {
253             revert(add(32, returnData), mload(returnData)) //return full revert data
254         }
255     }
256     inputToken.safeApprove(router, 0);
257     uint256 outputTokenBalance = outputToken.balanceOf(address(this));
258     if(outputTokenBalance > 0) {
259         outputToken.safeTransfer(msg.sender, outputTokenBalance);
260     }
261 }
```

Remediation Add validation for the router and ensure the function only transfers the outputToken obtained from the swap, not all outputToken held by the contract.

3.3.2 [L-1] Incompatibility with Deflationary Tokens

TARGET	CATEGORY	IMPACT	LIKELIHOOD	STATUS
KodiakMetaRouter.sol	Business Logic	Low	Low	Acknowledged

The KodiakMetaRouter::swap() function, designed to swap input.token for output.token, relies on input.token.safeTransferFrom() (line 110) to move tokens, assuming strict ERC20 compliance. However, this creates a vulnerability with deflationary tokens (transfer-on-fee tokens), which deduct fees during transfers. The function does not verify the actual amount received by the contract, leading to discrepancies between expected and received token amounts. This can cause incorrect accounting, transaction failures, or unexpected behavior during swaps, as the contract lacks mechanisms to handle such non-standard token transfers.

kodiak-periphery-meta-aggregator - KodiakMetaRouter.sol

```
89  function swap(
90      InputAmount memory input,
91      OutputAmount calldata output,
92      SwapData calldata swapData,
93      FeeData calldata feeData
94  ) external payable nonReentrant {
95      require(input.token != output.token, "KodiakMetaRouter: Input and output tokens must be different");
96      require(feeData.feeQuote >= output.minAmountOut, "KodiakMetaRouter: Invalid fee quote");
97
98      if(output.unwrap) {
99          require(output.token == address(wbera), "KodiakMetaRouter: Invalid output token");
100     }
101
102     //Do input validation and send input tokens to executor (after wrapping BERA if needed)
103     if(input.wrap) {
104         require(msg.value == input.amount, "KodiakMetaRouter: Invalid BERA amount");
105         require(input.token == address(wbera), "Invalid input token");
106         wbera.deposit{value: msg.value}();
107         input.token.safeTransfer(address(executor), input.amount);
108     } else {
109         require(msg.value == 0, "KodiakMetaRouter: Nonzero BERA amount");
110         input.token.safeTransferFrom(msg.sender, address(executor), input.amount);
111     }
112
113     SwapOutput memory swapOutput = _executeSwapAndApplyFee(input,output, swapData,feeData);
114     ...
115 }
```

Remediation Avoid using this function to swap deflationary (transfer-on-fee) tokens.

3.3.3 [L-2] Potential Risks Associated with Centralization

TARGET	CATEGORY	IMPACT	LIKELIHOOD	STATUS
Multiple Contracts	Security	Medium	Low	Acknowledged

In the Kodiak DEX protocol, the existence of a privileged owner account introduces centralization risks, as it holds significant control and authority over critical operations governing the protocol. In the following, we show the representative function potentially affected by the privileges associated with the privileged account.

kodiak-core-feature-nftWrapper - FungibleFactory.sol

```
68 //Whitelist a fungible implementation to be allowed to be deployed by the factory
69 function setAllowedImplementation(address _implementation, bool _allowed) external onlyOwner {
70     require(isImplementationAllowed[_implementation] != _allowed, "FungibleFactory: No change needed");
71     isImplementationAllowed[_implementation] = _allowed;
72     if (_allowed) {
73         allowedImplementations.push(_implementation);
74     } else {
75         uint256 length = allowedImplementations.length;
76         for (uint256 i = 0; i < length; ++i) {
77             if (allowedImplementations[i] == _implementation) {
78                 allowedImplementations[i] = allowedImplementations[length - 1];
79                 allowedImplementations.pop();
80                 break;
81             }
82         }
83     }
84     emit AllowedImplementationSet(_implementation, _allowed);
85 }
```

Remediation To mitigate the identified issue, it is recommended to introduce multi-sig mechanism to undertake the role of the privileged account. Moreover, it is advisable to implement timelocks to govern all modifications to the privileged operations.

Response By Team This issue has been confirmed by the team.

4 Appendix

4.1 About AstraSec

AstraSec is a leading blockchain security firm dedicated to delivering high-quality smart contract audits for blockchain-based protocols. Backed by a team of experienced blockchain specialists, AstraSec is committed to excellence, precision, and client satisfaction. Our auditors have conducted comprehensive reviews for numerous well-known DeFi projects, bringing deep technical expertise and industry insight to every engagement. With a rigorous approach to security and a strong understanding of blockchain ecosystems, AstraSec is a trusted partner for projects seeking to build with confidence.

4.2 Disclaimer

This audit report is provided for informational purposes only and does not constitute legal, financial, or investment advice. The observations, recommendations, and conclusions presented herein are based on the information and conditions available to us at the time of the audit and may be subject to limitations, unknown risks, or future changes. While reasonable efforts have been made to ensure the accuracy and completeness of this report, AstraSec assumes no liability for any errors, omissions, or decisions made based on its content.
Users are advised to carefully evaluate the information provided in this audit report using their own independent judgment and, where appropriate, seek professional advice before making any decisions. AstraSec shall not be held liable for any outcomes arising from the use of this report, including but not limited to any losses or damages resulting from reliance on its contents.This audit report is for reference purposes only and should not be regarded as a substitute for legal agreements, formal documentation, or contractual obligations.

4.3 Contact

Phone	+86 156 0639 2692
Email	contact@astrasec.ai
Twitter	https://x.com/AstraSecAI