F Y E O

Security Code Review
Aureus Ox Swap Router

Aureus Ox

May 2025
Version 1.0

Presented by:
FYEO Inc.
PO Box 147044
Lakewood CO 80214
United States

Security Level
Public

Table of Contents

Executive Summary

Overview

Aureus Ox engaged FYEO Inc. to perform a Security Code Review Aureus Ox Swap Router.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on May 19 - May 19, 2025, and focused on the following objectives:

- To provide the customer with an assessment of their overall security posture and any risks that were discovered within the environment during the engagement.
- To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place.
- To identify potential issues and include improvement recommendations based on the results of our tests.

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the FYEO Security Team took to identify and validate each issue, as well as any applicable recommendations for remediation.

Key Findings

The following issues have been identified during the testing period. These should be prioritized for remediation to reduce the risk they pose:

- FYEO-AUREUS-OX-01 – Use of non safe* functions for transfers
- FYEO-AUREUS-OX-02 – Refund may send more than it should
- FYEO-AUREUS-OX-03 – Token with fees are not supported
- FYEO-AUREUS-OX-04 – Updating allowance may revert
- FYEO-AUREUS-OX-05 – Check inputs against the zero address
- FYEO-AUREUS-OX-06 – Check calculated amounts are greater than 0
- FYEO-AUREUS-OX-07 – Consider adding pausing functionality
- FYEO-AUREUS-OX-08 – Hardcoded number
- FYEO-AUREUS-OX-09 – No events for swaps
- FYEO-AUREUS-OX-10 – Value returned by reference and value

Based on our review process, we conclude that the reviewed code implements the documented functionality.

Scope and Rules of Engagement

The FYEO Review Team performed a Security Code Review Aureus Ox Swap Router. The following table documents the targets in scope for the engagement. No additional systems or resources were in scope for this assessment.

The source code was supplied through a private repository at https://github.com/Aureus-Ox/aureus-ox-venmo-foundry with the commit hash 995011fb93ca851ef74f8531ccbf05a3192b065b.

Remediations were submitted with commit hash 4cd5dd198aea43c09df60f17b6f863d382d55e24.

Files included in the code review
aureus-ox-venmo-foundry/ ├── src/ │ └── flare/ │ └── implementation/ │ └── OxenFlowSwapRouter.sol └── test/ └── costontwo/ └── OxenFlowSwapRouter.t.sol

Table 1: Scope

Technical Analyses and Findings

During the Security Code Review Aureus Ox Swap Router, we discovered:

- 1 finding with HIGH severity rating.
- 3 findings with MEDIUM severity rating.
- 1 finding with LOW severity rating.
- 5 findings with INFORMATIONAL severity rating.

The following chart displays the findings by severity.

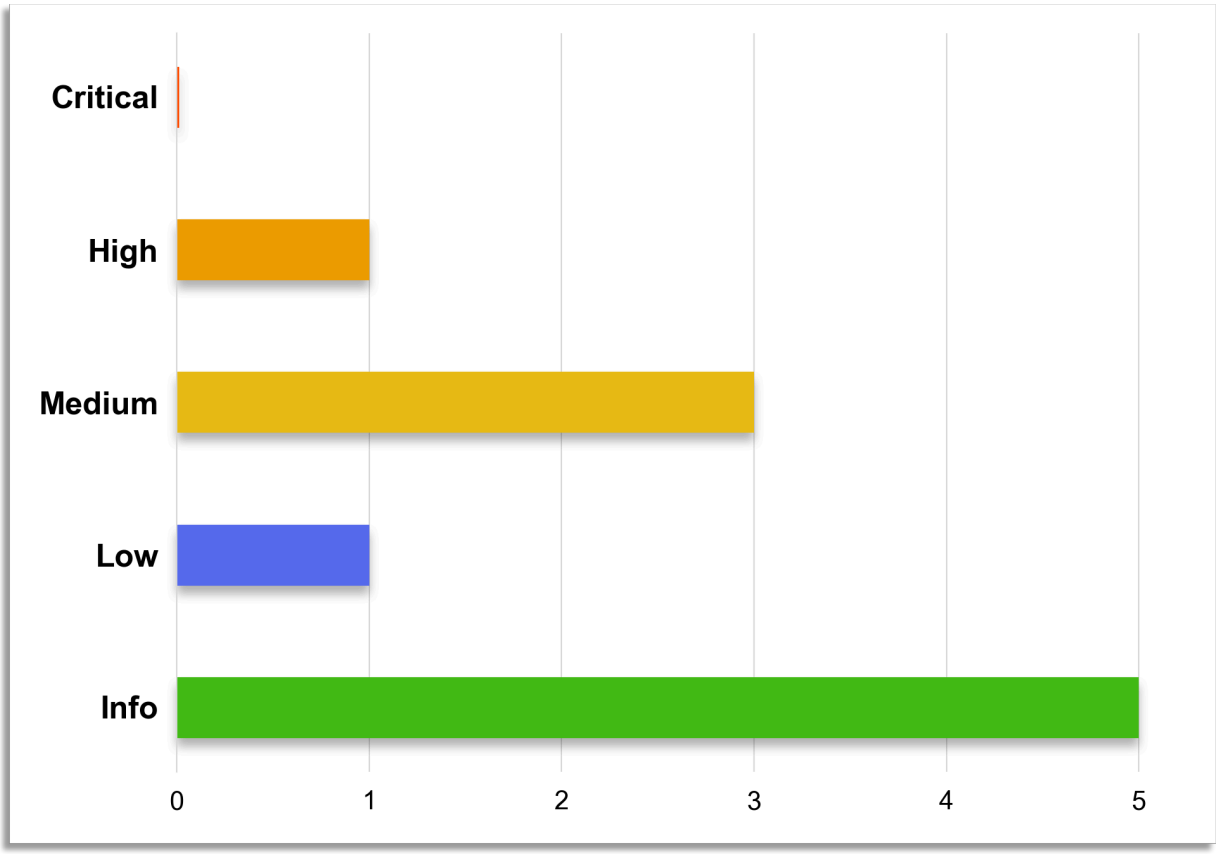

Figure 1: Findings by Severity

Findings

The *Findings* section provides detailed information on each of the findings, including methods of discovery, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

Finding #	Severity	Description	Status
FYEO-AUR EUS-OX-01	High	Use of non safe* functions for transfers	Remediated
FYEO-AUR EUS-OX-02	Medium	Refund may send more than it should	Remediated
FYEO-AUR EUS-OX-03	Medium	Token with fees are not supported	Remediated
FYEO-AUR EUS-OX-04	Medium	Updating allowance may revert	Remediated
FYEO-AUR EUS-OX-05	Low	Check inputs against the zero address	Remediated
FYEO-AUR EUS-OX-06	Informational	Check calculated amounts are greater than 0	Remediated
FYEO-AUR EUS-OX-07	Informational	Consider adding pausing functionality	Remediated
FYEO-AUR EUS-OX-08	Informational	Hardcoded number	Remediated
FYEO-AUR EUS-OX-09	Informational	No events for swaps	Remediated
FYEO-AUR EUS-OX-10	Informational	Value returned by reference and value	Remediated

Table 2: Findings Overview

The Classification of vulnerabilities

Security vulnerabilities and areas for improvement are weighted into one of several categories using, but is not limited to, the criteria listed below:

Critical – vulnerability will lead to a loss of protected assets

- This is a vulnerability that would lead to immediate loss of protected assets
- The complexity to exploit is low
- The probability of exploit is high

High - vulnerability has potential to lead to a loss of protected assets

- All discrepancies found where there is a security claim made in the documentation that cannot be found in the code
- All mismatches from the stated and actual functionality
- Unprotected key material
- Weak encryption of keys
- Badly generated key materials
- Txn signatures not verified
- Spending of funds through logic errors
- Calculation errors overflows and underflows

Medium - vulnerability hampers the uptime of the system or can lead to other problems

- Insecure calls to third party libraries
- Use of untested or nonstandard or non-peer-reviewed crypto functions
- Program crashes, leaves core dumps or writes sensitive data to log files

Low – vulnerability has a security impact but does not directly affect the protected assets

- Overly complex functions
- Unchecked return values from 3rd party libraries that could alter the execution flow

Informational

- General recommendations

Technical Analysis

The source code has been manually validated to the extent that the state of the repository allowed. The validation includes confirming that the code correctly implements the intended functionality.

Conclusion

Based on our review process, we conclude that the code implements the documented functionality to the extent of the reviewed code.

Technical Findings

General Observations

The OxenFlowSwapRouter contract sits at the heart of the wallet's swapping feature, acting as the bridge between users and Uniswap. When a user wants to trade one token for another, the wallet will call one of the swap functions - either a single-hop swap (`exactInputSingleSwapWithFee`), a multi hop swap (`exactInputSwapWithFee`), or an exact output swap (`exactOutputSwapWithFee`). Under the hood it handles everything: wrapping native FLR into WNat, approving Uniswap's router to execute the trade, performing the swap, then unwrapping FLR back to its native form if necessary. There is a small fee configured which is forwarded to a treasury wallet.

A handful of rough edges have been identified during the audit. Some critical addresses aren't validated against zero on construction or update. Token transfers and approvals rely on raw ERC-20 calls, which can silently fail or choke on nonstandard tokens (especially fee-on-transfer tokens). The refund logic may sweep up stray tokens unintentionally. In short, it nails the core swap flow but could use tuning around safety, observability, and support for edge-case tokens.

Use of non safe* functions for transfers

Finding ID: FYEO-AUREUS-OX-01
Severity: **High**
Status: **Remediated**

Description

The code uses bare `approve`, `transferFrom`, and `transfer` calls on ERC-20s. Non-standard tokens may return `false` instead of reverting, causing silent failures.

Proof of Issue

File name: src/flare/implementation/OxenFlowSwapRouter.sol
Line number: many

```
IERC20().approve(); x3
IERC20().transferFrom(); x3
IERC20().transfer(); x7
```

Severity and Impact Summary

Silent token failures can lock funds or create imbalance between contract state and on-chain balances.

Recommendation

Use OpenZeppelin's `SafeERC20`:

```
token.safeApprove(router, amount);
token.safeTransferFrom(user, address(this), amount);
token.safeTransfer(recipient, amount);
```

Refund may send more than it should

Finding ID: FYEO-AUREUS-OX-02
Severity: Medium
Status: Remediated

Description

The 'Refund unused input tokens' logic blindly refunds the entire `balanceOf(this)`, which may include tokens airdropped or sent for any other reason, not just leftover swap input.

Proof of Issue

File name: src/flare/implementation/OxenFlowSwapRouter.sol
Line number: 263

```
uint256 refund = IERC20(tokenIn).balanceOf(address(this));
if (refund > 0) {
    IERC20(tokenIn).transfer(msg.sender, refund);
}
```

Severity and Impact Summary

Could unintentionally refund third-party or donated tokens to the first caller.

Recommendation

Track the router's `tokenIn` balance before the user's swap and refund only the genuine leftover:

```
uint256 preSwap = token.balanceOf(address(this));
...
uint256 postSwap = token.balanceOf(address(this));
uint256 refund = postSwap > preSwap ? postSwap - preSwap : 0;
```

Or send out tokens to the fee wallet before transferring in user tokens (if any exist).

Token with fees are not supported

Finding ID: FYEO-AUREUS-OX-03
Severity: Medium
Status: Remediated

Description

`IERC20(tokenIn).transferFrom(msg.sender, address(this), amountIn)` assumes full receipt of `amountIn`. Fee-on-transfer tokens deduct a percentage, so the router receives less and may revert or inadvertently subsidize the swap.

Proof of Issue

File name: src/flare/implementation/OxenFlowSwapRouter.sol
Line number: 112

```
IERC20(tokenIn).transferFrom(msg.sender, address(this), amountIn);
```

Severity and Impact Summary

Tokens with fees will break the swap or lead to inequitable subsidization by the contract.

Recommendation

After `transferFrom`, compute:

```
uint256 before = token.balanceOf(address(this));
token.safeTransferFrom(...);
uint256 after = token.balanceOf(address(this));
uint256 netReceived = after - before;
```

Use `netReceived` for approvals and swapping. Or explicitly disallow fee-on-transfer tokens. Uniswap may also not support fee-on-transfer tokens.

Updating allowance may revert

Finding ID: FYEO-AUREUS-OX-04
Severity: Medium
Status: Remediated

Description

Approving the router without zeroing the prior allowance can revert to tokens that disallow non-zero to non-zero updates.

Proof of Issue

File name: src/flare/implementation/OxenFlowSwapRouter.sol
Line number: 176

```
IERC20(tokenIn).approve(address(swapRouter), params.amountIn);
```

Severity and Impact Summary

Tokens like USDT revert on `approve` if existing allowance ≠ 0.

Recommendation

Either first set allowance to zero then to the target, or use OpenZeppelin's `safeIncreaseAllowance` / `safeDecreaseAllowance`:

```
token.safeApprove(router, 0);
token.safeApprove(router, amount);
```

Check inputs against the zero address

Finding ID: FYEO-AUREUS-OX-05
Severity: **Low**
Status: **Remediated**

Description

Constructor and `setOxenFlowFeeWallet` do not validate that critical addresses (fee wallet, WNat, swapRouter) are non-zero. This can allow misconfiguration or deployment with the zero address.

Proof of Issue

File name: src/flare/implementation/OxenFlowSwapRouter.sol
Line number: 24, 35

```
constructor(ISwapRouter _swapRouter, address _oxenflowFeeWallet, uint256 _feeBps,
address _wnat) {
    swapRouter = _swapRouter;
    oxenflowFeeWallet = _oxenflowFeeWallet;
    feeBps = _feeBps;
    WNat = _wnat;
    // no require(_oxenflowFeeWallet != address(0)) etc.
}
function setOxenFlowFeeWallet(address _newWallet) external onlyRole(CONTRACT_OWNER) {
    oxenflowFeeWallet = _newWallet;
}
```

Severity and Impact Summary

Accidental zero-address configuration can break swaps or cause ETH/tokens to be sent irretrievably.

Recommendation

Add `require(_oxenflowFeeWallet != address(0), "zero fee wallet");`, `require(address(_swapRouter) != address(0))`, and `require(_wnat != address(0))` in the constructor. In `setOxenFlowFeeWallet`, require non-zero input.

Check calculated amounts are greater than 0

Finding ID: FYEO-AUREUS-OX-06
Severity: **Informational**
Status: **Remediated**

Description

Fees and user amounts may compute to zero for very small swaps.

Proof of Issue

File name: src/flare/implementation/OxenFlowSwapRouter.sol
Line number: 140, 194, 249

```
uint256 feeAmount = amountOut.mul(feeBps).div(10000);
uint256 userAmount = amountOut.sub(feeAmount);
```

Severity and Impact Summary

Small swaps incur zero fee and may return zero to the user, which is confusing.

Recommendation

Add checks:

```
require(feeAmount > 0, "fee is zero");
require(userAmount > 0, "user amount is zero");
```

Or define a minimum `amountOut`.

Consider adding pausing functionality

Finding ID: FYEO-AUREUS-OX-07
Severity: **Informational**
Status: **Remediated**

Description

The contract lacks a mechanism to pause operations in emergency or upgrade scenarios. Without `Pausable`, an exploitable bug in Uniswap pools or WNat could require suspension of this router.

Proof of Issue

File name: src/flare/implementation/OxenFlowSwapRouter.sol
Line number: 11

```
contract OxenFlowSwapRouter is AccessControl, ReentrancyGuard {
    // no Pausable or emergency-stop implemented
}
```

Severity and Impact Summary

An inability to halt trading under exceptional circumstances may cause problems.

Recommendation

Inherit OpenZeppelin's `Pausable` and add `whenNotPaused` modifiers to all swap functions. Grant a `PAUSER_ROLE` so an admin can quickly pause or unpause the router.

Hardcoded number

Finding ID: FYEO-AUREUS-OX-08
Severity: **Informational**
Status: **Remediated**

Description

Hard-coded divisor `10000` in fee calculation is a “magic number.”

Proof of Issue

File name: src/flare/implementation/OxenFlowSwapRouter.sol
Line number: 140, 194, 249

```
uint256 feeAmount = amountOut.mul(feeBps).div(10000);
```

Severity and Impact Summary

Readability / maintainability issue.

Recommendation

Define a `uint256 private constant BPS_DENOMINATOR = 10_000;` and use it in these calculations.

No events for swaps

Finding ID: FYEO-AUREUS-OX-09
Severity: **Informational**
Status: **Remediated**

Description

The contract emits `FeeCollected` but does not log the core swap details, making it harder to trace trades on-chain.

Proof of Issue

File name: src/flare/implementation/OxenFlowSwapRouter.sol
Line number: 153, 206, 260

```
emit FeeCollected(tokenOut, msg.sender, feeAmount);
// no SwapExecuted or similar event
```

Severity and Impact Summary

Reduced observability; auditors and analytics tools miss full trade data.

Recommendation

Emit an event such as:

```
event SwapExecuted(address indexed user, address tokenIn, address tokenOut, uint256
amountIn, uint256 amountOut);
```

at the end of each swap function.

Value returned by reference and value

Finding ID: FYEO-AUREUS-OX-10
Severity: **Informational**
Status: **Remediated**

Description

The call `path = _replaceTokenOutZeroAddressWithWNat(path);` takes a `bytes memory path` parameter and modifies it in place, yet also returns it. This dual approach is confusing: readers may think the assignment is necessary, when in fact the original `path` is mutated regardless of the returned value.

Proof of Issue

File name: src/flare/implementation/OxenFlowSwapRouter.sol
Line number: 57, 76

```
bytes memory newPath = path; // `path` and `newPath` alias the same memory
...
if (lastToken == address(0)) {
    for (uint i = 0; i < 20; i++) {
        newPath[len - 20 + i] = bytes20(WNat)[i];
    }
}
return newPath;
```

Severity and Impact Summary

Purely a readability / maintainability concern, but can lead to misunderstandings and future bugs if someone refactors only the return or only the mutation.

Recommendation

Have the function not return a value to remove the ambiguity.

Our Process

Methodology

FYEO Inc. uses the following high-level methodology when approaching engagements. They are broken up into the following phases.

Figure 2: Methodology Flow

Kickoff

The project is kicked off as the sales process has concluded. We typically set up a kickoff meeting where project stakeholders are gathered to discuss the project as well as the responsibilities of participants. During this meeting we verify the scope of the engagement and discuss the project activities. It's an opportunity for both sides to ask questions and get to know each other. By the end of the kickoff there is an understanding of the following:

- Designated points of contact
- Communication methods and frequency
- Shared documentation
- Code and/or any other artifacts necessary for project success
- Follow-up meeting schedule, such as a technical walkthrough
- Understanding of timeline and duration

Ramp-up

Ramp-up consists of the activities necessary to gain proficiency on the project. This can include the steps needed for familiarity with the codebase or technological innovation utilized. This may include, but is not limited to:

- Reviewing previous work in the area including academic papers
- Reviewing programming language constructs for specific languages
- Researching common flaws and recent technological advancements

Review

The review phase is where most of the work on the engagement is completed. This is the phase where we analyze the project for flaws and issues that impact the security posture. Depending on the project this may include an analysis of the architecture, a review of the code, and a specification matching to match the architecture to the implemented code.

In this code audit, we performed the following tasks:

1. Security analysis and architecture review of the original protocol
2. Review of the code written for the project
3. Compliance of the code with the provided technical documentation

The review for this project was performed using manual methods and utilizing the experience of the reviewer. No dynamic testing was performed, only the use of custom-built scripts and tools were used to assist the reviewer during the testing. We discuss our methodology in more detail in the following sections.

Code Safety

We analyzed the provided code, checking for issues related to the following categories:

- General code safety and susceptibility to known issues
- Poor coding practices and unsafe behavior
- Leakage of secrets or other sensitive data through memory mismanagement
- Susceptibility to misuse and system errors
- Error management and logging

This list is general and not comprehensive, meant only to give an understanding of the issues we are looking for.

Technical Specification Matching

We analyzed the provided documentation and checked that the code matches the specification. We checked for things such as:

- Proper implementation of the documented protocol phases
- Proper error handling
- Adherence to the protocol logical description

Reporting

FYEO Inc. delivers a draft report that contains an executive summary, technical details, and observations about the project.

The executive summary contains an overview of the engagement including the number of findings as well as a statement about our general risk assessment of the project. We may conclude that the overall risk is low but depending on what was assessed we may conclude that more scrutiny of the project is needed.

We report security issues identified, as well as informational findings for improvement, categorized by the following labels:

- Critical
- High
- Medium
- Low
- Informational

The technical details are aimed more at developers, describing the issues, the severity ranking and recommendations for mitigation.

As we perform the audit, we may identify issues that aren't security related, but are general best practices and steps that can be taken to lower the attack surface of the project. We will call those out as we encounter them and as time permits.

As an optional step, we can agree on the creation of a public report that can be shared and distributed with a larger audience.

Verify

After the preliminary findings have been delivered, this could be in the form of the approved communication channel or delivery of the draft report, we will verify any fixes within a window of time specified in the project. After the fixes have been verified, we will change the status of the finding in the report from open to remediated.

The output of this phase will be a final report with any mitigated findings noted.

Additional Note

It is important to note that, although we did our best in our analysis, no code audit or assessment is a guarantee of the absence of flaws. Our effort was constrained by resource and time limits along with the scope of the agreement.

While assessing the severity of the findings, we considered the impact, ease of exploitability, and the probability of attack. This is a solid baseline for severity determination.