SALUS SECURITY

JUN 2025

CODE SECURITY ASSESSMENT

ASTERDEX

Overview

Project Summary

- Name: Asterdex - Apx exchange V2
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
    - https://github.com/asterdex/apx-exchange-contract
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Asterdex - Apx exchange V2
Version	v2
Type	Solidity
Dates	Jun 24 2025
Logs	Jun 19 2025; Jun 24 2025

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	1
Total informational issues	1
Total	2

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Implementation contract lacks _disableInitializers()	Low	Configuration	Acknowledged
2	Gas optimization suggestions	Informational	Gas Optimization	Acknowledged

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Implementation contract lacks _disableInitializers()

Severity: Low	Category: Configuration
Target: - contracts/ApxExchange.sol	

Description

The `ApxExchange` contract does not call OpenZeppelin's `_disableInitializers()` in its constructor.

As a result, anyone can send a direct transaction to the implementation address (not the proxy) and invoke `initialize()`, gaining `DEFAULT_ADMIN_ROLE` and `ADMIN_ROLE` within the implementation's own storage context.

Recommendation

Add the `_disableInitializers()` function in the constructor.

Status

This issue has been acknowledged by the team.

2.3 Informational Findings

2. Gas optimization suggestions

Severity: Informational	Category: Gas Optimization
Target: - contracts/ApxExchange.sol	

Description

Memory reading saves more gas than storage reading multiple times when the state is not changed. So caching the storage variables in memory and using the memory instead of storage reading is effective. Cache array length outside of the loop can save gas.

contracts/ApxExchange.sol:L92

```
for (uint i = 0; i < _exchangeInfos.length; i ++) {
```

contracts/ApxExchange.sol:L106

```
for (uint i = 0; i < exchangeInfos.length; i++) {
```

contracts/ApxExchange.sol:L157

```
for (uint i = 0; i < nftIds.length; i ++) {
```

contracts/ApxExchange.sol:L224

```
for (uint i = 0; i < lockOrderIds.length; i ++) {
```

Recommendation

Consider using the above suggestions to save gas.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit f5993b7:

File	SHA-1 hash
contracts/ApxExchange.sol	3ccb4731f1b26c8778947df42ac49707a7e64f41