SALUS SECURITY

MAY 2025

CODE SECURITY ASSESSMENT

ASPECTA-AI

Overview

Project Summary

- Name: Aspecta-ai
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
    - https://github.com/aspecta-ai/contract-v1-core
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Aspecta-ai
Version	v2
Type	Solidity
Dates	May 20 2025
Logs	May 20 2025, May 20 2025

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	1
Total Low-Severity issues	0
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	Slippage control does not take fees into account	Medium	Business Logic	Resolved
2	A user may be added to the userList multiple times	Informational	Business Logic	Acknowledged

2.2 Notable Findings

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

1. Slippage control does not take fees into account

Severity: Medium	Category: Business Logic
Target: - contracts/AspectaKeyPool/AspectaKeyPool.sol	

Description

In the `sell` function, there is a `minPrice` parameter used for slippage control. However, this slippage parameter does not take fees into account. Fee amounts may be affected by transaction volatility — for example, fluctuations in `projectFeePercentage` and `protocolFeePercentage`, or changes in the user's discount data.

If the volatility causes the user to pay more fees than expected, the amount of BNB received could be less than anticipated.

contracts/AspectaFairLaunch/AspectaKeyPool.sol L313-L316

```
function _sell(
    address seller,
    uint256 amount,
    uint256 minPrice
) internal nonReentrant {
    ...
    if (totalPrice < minPrice) {
        revert MinPriceNotMet(minPrice, totalPrice);
    }
    uint256 userDiscount = IAspectaMembership(membershipAddress)
        .getUserDiscount(seller);
    uint256 projectFeePercentageWithDiscount = projectFeePercentage -
        (projectFeePercentage * userDiscount) /
        1 ether;
    uint256 protocolFeePercentageWithDiscount = protocolFeePercentage -
        (protocolFeePercentage * userDiscount) /
        1 ether;
    projectFee = (totalPrice * projectFeePercentageWithDiscount) / 1 ether;
    protocolFee =
        (totalPrice * protocolFeePercentageWithDiscount) /
        1 ether;
    ...
```

Recommendation

It is recommended that the slippage parameter take fees into consideration.

Status

This issue has been resolved by the team with commit f8efd00.

2.3 Informational Findings

2. A user may be added to the userList multiple times

Severity: Informational	Category: Business Logic
Target: - contracts/AspectaMembership/AspectaMembership.sol	

Description

When `updateUserLevel` is called, if a user's level becomes non-zero, the user is pushed to the `userList`.

However, if the user's level later becomes zero, the user is not removed from `userList`. Furthermore, if the same user's level becomes non-zero again in the future, they will be pushed to `userList` once more.

This can result in users who are already in the list not being removed when their level becomes zero, and the same user being added to the list multiple times.

contracts/AspectaMembership/AspectaMembership.sol:L276-L298

```
function updateUserLevel(
    address[] memory users,
    uint8[] memory levels
) external onlyRole(ASPECTA_ADMIN_ROLE) {
    if (users.length != levels.length) {
        revert InvalidInput();
    }
    for (uint64 i = 0; i < users.length; i++) {
        address user = users[i];
        uint8 level = levels[i];
        if (levelDiscountMap[level] == 0) {
            revert InvalidLevel();
        }
        if (userLevelMap[user] == 0 && level > 0) {
            userList.push(user);
        }
        userLevelMap[user] = level;
    }
}
```

Recommendation

It is recommended to remove the user from `userList` when their level changes from a non-zero value to zero.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit a35618c:

File	SHA-1 hash
AspectaKeyPoolFactory.sol	47fc44bfdc914e41b24f43a7e4a3d144ddcd8a50
AspectaKeyPoolFactoryStorage.sol	3723dc546a0a5e64eb0164cbc947496af7ff67e5
AspectaAssetStorage.sol	6ef27b9bd98cfef718cfa36f3d7c4a34987c018f
AspectaAsset.sol	f539532184b39e12b8d34046e7d7bd24d1cc53f8
BondingCurve.sol	1742bdd3e58d56a5f1735910cfaf3e0f7367b433
Piecewise.sol	0c7746ca43511fa581ad69d07260c35aff48a9d5
AspectaAssetStorage.sol	9b16dcde8827b8c41dde727b7f93e451902672f7
AspectaAsset.sol	ff0f0b90c947baa27b42f435e1bfdb618d2c199c
AspectaKeyPoolStorage.sol	a35a39b4009c9617d49d3db2e069faeef6b1ccac
AspectaKeyPool.sol	1b19048d492c01983e0e34fe0c3ca4c7d813e2f9
AspectaMembershipStorage.sol	73af917c3837f6624a2034644be29700f6615b99
AspectaMembership.sol	b4671143a51f8b434984ac53efca29a70f49ec0a