Ascendant Security Audit

Report Version 1.0

January 7, 2025

Conducted by **Hunter Security**

Table of Contents

1 About Hunter Security

Hunter Security is an industry-leading smart contract security auditing firm. Having conducted over 100 security audits protecting over $1B of TVL, our team always strives to deliver top-notch security services to the best DeFi protocols. For security audit inquiries, you can reach out on Telegram or Twitter at *@georgehntr*.

2 Disclaimer

Audits are a time-, resource-, and expertise-bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can reveal the presence of vulnerabilities, but cannot guarantee their absence.

3 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	High	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

3.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - involves a small loss of funds or affects a core functionality of the protocol.
- **Low** - encompasses any unexpected behavior that is non-critical.

3.2 Likelihood

- **High** - a direct attack vector; the cost is relatively low compared to the potential loss of funds.
- **Medium** - only a conditionally incentivized attack vector, with a moderate likelihood.
- **Low** - involves too many or unlikely assumptions; offers little to no incentive.

3.3 Actions required by severity level

- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

4 Executive summary

Overview

Project Name	Ascendant
Repository	https://github.com/De-centraX/ascendant-contracts
Commit hash	8d6f92609d748759061328c7c1fbdb3d6b02bd7e
Resolution	2ae7aa5ee0265d7e2df5f7dacc151b8fde13c246
Methods	Manual review & testing

Scope

src/core/Ascendant.sol
src/core/AscendantAuction.sol
src/core/AscendantBuyAndBurn.sol
src/core/AscendantNFTMarketplace.sol
src/core/AscendantNFTMinting.sol
src/core/AscendantPride.sol

Issues Found

High risk	1
Medium risk	0
Low risk	0
Informational	12

5 Findings

5.1 High

5.1.1 Unable to collect fees from liquidity provision in auction

Severity: High

Context: src/*

Description: The Ascendant token creates a UniswapV3 pool of Ascendant:DragonX pair upon construction. The initial liquidity for this pair is provided by minting a full-range position in the Ascendant auction once enough tokens have been collected.

The problem is that later, when attempting to collect the liquidity provision fees, instead of transferring the collected dragonX tokens to the genesis wallet, titanX tokens are attempted to be sent treating the dragonX amount as titanX amount.

This leads to the following problems:
- there is no way to withdraw the dragonX fees collected
- collecting the fees would cause a revert as there are no titanX tokens in the balance of the contract

Recommendation: Consider replacing the *titanX* token transfer with *dragonX*.

Resolution: Resolved.

5.2 Informational

5.2.1 Typographical mistakes, non-critical issues and code-style suggestions

Severity: Informational

Context: src/*

Description: The contracts contains one or more typographical mistakes, non-critical issues and code-style suggestions. In an effort to keep the report size reasonable, we enumerate these below:

1. Unused imports, libraries, custom errors and storage variables.
2. Explicit visibility not set to some variables and constants.
3. *collection* could be marked *immutable*.
4. Not reverting in *getTwapAmount* and *getTwapPrice* when *oldestObservation* < *secondsAgo*.
5. Consider adding *totalShares == 0* check in *batchClaimableAmount*.
6. Rounding down when calculating *_amountPerInterval* in AscendantBuyAndBurn.
7. The converted titanx amount should be checked instead of *msg.value* in *depositETH*.
8. The *noContract* modifier does not prevent from contracts in construction. It would prevent multi-signature wallets from using the contracts.
9. Consider better adhering to the CEI pattern in *buyItemWithETH* by making the untrusted external call to the *msg.sender* at the end.
10. Consider enforcing better constraints in *setSecondsAgo* such as 5 minutes minimum and 30 minutes maximum.

11. There is no need to have a separate variable for *ascendantNFTMinting*. Consider using only *collection*.
12. Inconsistency when using 14 hours and 16 hours as a start of the day.

Recommendation: Consider fixing the above typographical mistakes, non-critical issues and code-style suggestions.

Resolution: Partially resolved.