RISK AUDIT

for

NOVEL LABS

on

April 10, 2025

FIDESIUM

Executive Summary

Report

13
/100

TOTAL

Low risk

April 10, 2025

Abstract

Fidesium's automated risk assessment service was requested to perform a risk posture audit on Novel Labss contracts

Repository Link:

https://github.com/mutantcartel/mutant-hound-contracts.git

Initial Commit Hash:

`8860b38f1dab16fcdd5472e1df2ebe7b0ff2a1e1`

Issue Summary

Critical	High	Medium	Low	Info
0 Issues	2 Issues	1 Issues	2 Issue	2 Issues

Caveats

This audit was conducted at commit `8860b38f1dab16fcdd5472e1df2ebe7b0ff2a1e1`. Access to the deployed contracts was not provided, and onchain data was not analyzed.

Test Approach

Fidesium performed both Whitebox and Blackbox testing, as per the scope of the engagement, and relied on automated security testing.

Methodology

The assessment methodology covered a range of phases and employed various tools, including but not limited to the following:

- Mapping Content and Functionality of API
- Application Logic Flaws
- Access Handling
- Authentication/Authorization Flaws
- Brute Force Attempt
- Input Handling
- Source Code Review
- Fuzzing of all input parameter
- Dependency Analysis

Severity Definitions

Critical	The issue can cause large economic losses, large-scale data disorder or loss of control of authority management.
High	The issue puts users' sensitive information at risk or is likely to lead to catastrophic financial implications.
Medium	The issue puts a subset of users' sensitive information at risk, reputation damage or moderate financial impact.
Low	The risk is relatively small and could not be exploited on a recurring basis, or is low-impact to the client's business.
Informational	The issue does not pose an immediate risk but is relevant to security best practices or defence in Depth.

Risk Issues

Vunerability	Description	Risk	Probability	Status
Logic Error: Inverted Comparison	`appendAuctionTokens` guard comparison is backwards	High	High	Active
Missing ownership validation	`initializeAuction` does not validate `tokenId` ownership	High	High	Active
Logic Error: Price Update inconsistency	`settleSale` updated price inconsistently	Medium	High	Active
Block Gas Limit: Unbounded Loop	`batchRecoverERC721` implements no limit on the setllement size and potentially hit block gas limit and revert.	Low	Medium	Acknowledged
Missing Zero Amount check before transfer	Missing zero amount check before transfer in `_refund`	Low	Low	Active
Gas optimization: Storage packing	`AuctionParams` and `SaleParams` are packed inefficiently	Info	Info	Active
Gas optimization: Unnecessary Storage reads	`price` makes unnecessary storage reads	Info	Info	Active

Risk Overview

Team Risk

Low risk: 1

No issues found in founding team

Doxxing Status	Team Experience	Risk Summary
Public	Highly relevant	Low

Liquidity

Risk summary: N/A

As this is a Github assessment, liquidity risks have not been assessed

Whale Concentration

Risk summary: N/A

As this is a Github assessment, whale risks have not been assessed

Smart Contract Risks

Risk summary: 19

The contract is mostly well written, but has a handful of significant flaws that need to be carefuly managed, and expose the ecosystem to a variety of risks, including price manipulation attacks and loss of funds.

Vulnerabilities Critical

Current scan criticals Clear

During this scan no critical security vulnerabilities were identified. The assessment covered all key components of the project, including smart contract logic, access controls, and potential attack vectors. While no critical issues were found, we recommend ongoing security monitoring and best practices to maintain the integrity and resilience of the system.

Vulnerabilities High

Missing ownership validation

Vulnerability severity: **High**

`initializeAuction` does not validate `tokenId` ownership

An admin could add tokens to auction which the contract does not own. This could result in reverting transactions and/or user loss of funds

In an extreme case, a malicious admin could use this to sabotage auctions and steal bids

Recommendations:

Add explicit checks to verify the contract owns each specific tokenId, in both `initializeAuction` and `appendAuctionTokens`

```
for (uint256 i = 0; i < tokenIds_.length; i++) {
    for (uint256 j = i + 1; j < tokenIds_.length; j++) {
        if (tokenIds_[i] == tokenIds_[j]) {
            revert DutchAuction__duplicateTokenId(tokenIds_[i]);
        }
    }
}

for (uint256 i = 0; i < tokenIds_.length; i++) {
    address tokenOwner = collection.ownerOf(tokenIds_[i]);
    if (tokenOwner != address(this)) {
        revert DutchAuction__initializeAuction_tokenNotOwned(tokenIds_[i]);
    }
}
```

If the expected set of tokens is very large, consider implementing Merkle Proofs instead of Loops

Logic Error: Inverted Comparison

Vulnerability severity: **High**

`appendAuctionTokens` guard comparison is backwards

```
if (collection.balanceOf(address(this)) > auctionParams.auctionedAmount)
    revert DutchAuction__appendAuctionTokens_invalidAuctionedAmount();
```

Recommendations:

Invert the comparison to ensure function reverts if contract has fewer tokens than are being auctioned

Vulnerabilities Medium

Logic Error: Price Update inconsistency

Vulnerability severity: **Medium**

`settleSale` updated price inconsistently

For low price differences below dust amount, price is not updated

Recommendations:

Move the `sale.price = floorPrice_;` call above dust check

Vulnerabilities Low

Missing Zero Amount check before transfer

Vulnerability severity: **Low**

Missing zero amount check before transfer in `_refund`

Recommendations:

Ensure zero amounts are not transferred by `_refund`

Block Gas Limit: Unbounded Loop

Vulnerability severity: **Low**

`batchRecoverERC721` implements no limit on the tokenIds array size and potentially hit block gas limit and revert.

Recommendations:

Implement Pagination and batch processing limits

Vulnerabilities Info

Gas optimization: Unnecessary Storage reads

Vulnerability severity: **Info**

`price` makes unnecessary storage reads

`priceDiff is computed and stored but only used once`

`block.number` is read repeatedly

Recommendations:

- Inline `priceDiff` computation
- Cache `block.number`

Gas optimization: Storage packing

Vulnerability severity: **Info**

`AuctionParams` and `SaleParams` are packed inefficiently

Recommendations:

```
struct SaleParams {
    address bidder;
    uint96 price;
    uint64 amount;
    uint32 _reserved;
}

struct AuctionParams {
    // Slot 1 (32 bytes total)
    uint64 startBlock;
    uint64 endBlock;
    uint96 startPrice;
    uint32 _reserved;

    // Slot 2 (32 bytes total)
    uint96 endPrice;
    uint96 floorPrice;
    uint64 auctionedAmount;
}
```

Disclaimer

Disclaimer

This report is governed by the Fidesium terms and conditions.

This report does not constitute an endorsement or disapproval of any project or team, nor does it reflect the economic value or potential of any related product or asset. It is not investment advice and should not be used as the basis for investment decisions. Instead, this report provides an assessment intended to improve code quality and mitigate risks inherent in cryptographic tokens and blockchain technology.

Fidesium does not guarantee the absence of bugs or vulnerabilities in the technology assessed, nor does it comment on the business practices, models, or regulatory compliance of its creators. All services, reports, and materials are provided "as is" and "as available," without warranties of any kind, including but not limited to merchantability, fitness for a particular purpose, or non-infringement.

Cryptographic assets and blockchain technologies are novel and carry inherent technical risks, uncertainties, and the possibility of unpredictable outcomes. Assessment results may contain inaccuracies or depend on third-party systems, and reliance on them is solely at the Customer’s risk.

Fidesium assumes no liability for content inaccuracies, personal injuries, property damages, or losses related to the use of its services, reports, or materials. Third-party components are provided "as is," and any warranties are strictly between the Customer and the third-party provider.

These services and materials are intended solely for the Customer's use and benefit. No third party or their representatives may claim rights to or rely on these services, reports, or materials under any circumstances.