RISK AUDIT

for

NOVEL LABS

on

April 10, 2025

FIDESIUM

Executive Summary

Report

7
/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	1 Issues	1 Issues	1 Issues	0 Issues

Caveats

Novel Lab's codebase is well written, but does incur a handful of high value flaws.

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
`EIP-712` signature replay	The `FusedMinterUpgradable` relies on `EIP-712` without signature expiration or nonces	High	Low	Active
Addresses presumed to be contracts	The `FusedMinterUpgradable.constructor` function assumes multiple addresses are contracts	Medium	Low	Active
Missing null signature validation	The `FusedMinterUpgradable._validateSigner` function does not validate against null signatures	Low	Low	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: 9

The contracts are mostly well written, but have a handful of flaws that should to be carefuly managed.

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

EIP-712 signature replay

Vulnerability severity: **High**

Vulnerability probability: **Low**

The `FusedMinterUpgradable` relies on `EIP-712` without signature expiration or nonces

A malicious attacker could identify duplicate allocation requirements.

Additionally, if `resetUsedTokens` were to be called in error, this could open the contract up to damaging replay and value extraction

Recommendations:

- Correlate the `ALLOCATION_TYPEHASH` to specific collections and tokenIds in addition to minter/signer
- Generate and track per signature nonces
- Enforce expiration timestamps and block numbers
- Add global state change counter and track state changes (especially `resetUsedTokens` invocations), include counter in signature

Vulnerabilities Medium

Addresses presumed to be contracts

Vulnerability severity: **Medium**

Vulnerability probability: **Low**

The `FusedMinterUpgradable.constructor` function assumes multiple addresses are contracts

This could lead to silent transaction failures, or, in the event of malicious misconfiguration, the injection of malicious contracts and protocol failure

Recommendations:

- Validate `codesize` in initialize

```
uint256 codeSize;
assembly {
    codeSize := extcodesize(oathCollectionAddress_)
}
require(codeSize > 0, "Governor::initialize: oathCollectionAddress_ is not a contract");
```

- Validate ABI conformity

```
try IERC20(warmRegistry_).getColdWallets(known_test_address) returns (address[] list) {
    require(list.length == known_value, "List length invalid");
    require(list[0] == known_value, "list is incorrect")
}
```

Vulnerabilities Low

Missing null signature validation

The `FusedMinterUpgradable._validateSigner` function does not validate against null signatures Low Low Active

Vulnerability severity: **Low**

Vulnerability probability: **Low**

Missing null signature validation

Recommendations:

Validate the signer is non zero, the signature is non null, and the signature length conform to ECDSA

```
require(signer != address(0), "Invalid signer: zero address");

if (signer == msg.sender) {
    return;
}

require(signature.length == 65, "Invalid signature length");
bytes32 r;
bytes32 s;
uint8 v;
assembly {
    r := calldataload(signature.offset)
    s := calldataload(add(signature.offset, 32))
    v := byte(0, calldataload(add(signature.offset, 64)))
}
require(r != 0 && s != 0 && (v == 27 || v == 28), "Invalid signature format");

bytes32 digest = getDigest(msg.sender, signer);
address recoveredSigner = ECDSA.recover(digest, signature);

require(recoveredSigner != address(0), "Invalid signature: recovers to zero address");
```

Vulnerabilities Info

Current scan info Clear

During this scan no informational 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.

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.