

// Security Assessment

03.21.2025 - 03.25.2025

---

# **AMM Contracts**

**HALBORN**

# AMM Contracts



Prepared by: **H HALBORN**

Last Updated 03/28/2025

Date of Engagement: March 21st, 2025 - March 25th, 2025

## Summary

NO REPORTED FINDINGS TO ADDRESS

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>

## TABLE OF CONTENTS

1. Introduction
2. Assessment summary
3. Test approach and methodology
4. Risk methodology
5. Scope
6. Assessment summary & findings overview
7. Findings & Tech Details
8. Automated Testing

## **1. Introduction**

Laminar engaged Halborn to conduct a security assessment on smart contracts beginning on March 21st, 2025 and ending on March 25th, 2025. The security assessment was scoped to the smart contracts provided to the Halborn team.

Commit hashes and further details can be found in the Scope section of this report.

## **2. Assessment Summary**

The team at Halborn dedicated 3 days for the engagement and assigned one full-time security engineers to evaluate the security of the smart contract.

The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this assessment is to:

- Ensure that Smart Contract functions operate as intended.
- Identify potential security issues with the Smart Contracts.

In summary, no security vulnerabilities were identified for the contracts in-scope.

### **3. Test Approach And Methodology**

Halborn performed a combination of manual, semi-automated and automated security testing to balance efficiency, timeliness, practicality, and accuracy regarding the scope of this assessment. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the assessment:

- Research into architecture and purpose.
- Smart contract manual code review and walk-through.
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any vulnerability classes
- Manual testing by custom scripts.
- Static Analysis of security for scoped contract, and imported functions. ([Slither](#))
- Local deployment and testing ([Foundry](#))

## 4. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets of Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

### 4.1 EXPLOITABILITY

#### ATTACK ORIGIN (AO):

Captures whether the attack requires compromising a specific account.

#### ATTACK COST (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

#### ATTACK COMPLEXITY (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

#### METRICS:

EXPLOITABILITY METRIC ( $M_E$ )	METRIC VALUE	NUMERICAL VALUE
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2

EXPLOITABILITY METRIC ( $M_E$ )	METRIC VALUE	NUMERICAL VALUE
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability  $E$  is calculated using the following formula:

$$E = \prod m_e$$

## 4.2 IMPACT

### CONFIDENTIALITY (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

### INTEGRITY (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

### AVAILABILITY (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

### DEPOSIT (D):

Measures the impact to the deposits made to the contract by either users or owners.

### YIELD (Y):

Measures the impact to the yield generated by the contract for either users or owners.

### METRICS:

IMPACT METRIC ( $M_I$ )	METRIC VALUE	NUMERICAL VALUE
Confidentiality (C)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Integrity (I)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Availability (A)	None (A:N)	0
	Low (A:L)	0.25
	Medium (A:M)	0.5
	High (A:H)	0.75
	Critical (A:C)	1
Deposit (D)	None (D:N)	0
	Low (D:L)	0.25
	Medium (D:M)	0.5
	High (D:H)	0.75
	Critical (D:C)	1
Yield (Y)	None (Y:N)	0
	Low (Y:L)	0.25
	Medium (Y:M)	0.5
	High (Y:H)	0.75
	Critical (Y:C)	1

Impact  $I$  is calculated using the following formula:

$$I = \max(m_I) + \frac{\sum m_I - \max(m_I)}{4}$$

## 4.3 SEVERITY COEFFICIENT

### REVERSIBILITY (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

### SCOPE (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

### METRICS:

SEVERITY COEFFICIENT ( $C$ )	COEFFICIENT VALUE	NUMERICAL VALUE
Reversibility ( $r$ )	None (R:N) Partial (R:P) Full (R:F)	1 0.5 0.25
Scope ( $s$ )	Changed (S:C) Unchanged (S:U)	1.25 1

Severity Coefficient  $C$  is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score  $S$  is obtained by:

$$S = \min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

SEVERITY	SCORE VALUE RANGE
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4

SEVERITY	SCORE VALUE RANGE
Informational	0 - 1.9

## 5. SCOPE

### FILES AND REPOSITORY

^

(a) Repository: amm-contracts

(b) Assessed Commit ID: e7ed3b7

(c) Items in scope:

- contracts/v3/core/LaminarV3Pool.sol
- contracts/v3/core/LaminarV3PoolDeployer.sol
- contracts/v3/core/LaminarV3Factory.sol
- contracts/v3/periphery/WHYPE.sol
- contracts/v3/periphery/LaminarInterfaceMulticall.sol
- contracts/v3/periphery/Multicall3.sol
- contracts/v3/periphery/NonfungiblePositionManager.sol
- contracts/v3/periphery/NonfungibleTokenPositionDescriptor.sol
- contracts/v3/periphery/SwapRouter.sol
- contracts/v3/periphery/libraries/SafeERC20Namer.sol
- contracts/v3/periphery/libraries/PeripheryTransferHelper.sol
- contracts/v3/periphery/libraries/AddressStringUtil.sol
- contracts/v2/core/LaminarV2ERC20.sol
- contracts/v2/core/LaminarV2Factory.sol
- contracts/v2/core/LaminarV2Pair.sol
- contracts/router/V2SwapRouter.sol
- contracts/router/V3SwapRouter.sol
- contracts/router/SwapRouter.sol
- contracts/router/libraries/LaminarV2Library.sol

Out-of-Scope: Third party dependencies and economic attacks.

Out-of-Scope: New features/implementations after the remediation commit IDs.

## 6. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

**CRITICAL**

**0**

**HIGH**

**0**

**MEDIUM**

**0**

**LOW**

**0**

**INFORMATIONAL**

**0**

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
-------------------	------------	------------------

## **7. FINDINGS & TECH DETAILS**

### **8. AUTOMATED TESTING**

Halborn used automated testing techniques to enhance the coverage of certain areas of the smart contracts in scope. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified the smart contracts in the repository and was able to compile them correctly into their ABIs and binary format, Slither was run against the contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

The security team conducted a comprehensive review of findings generated by the Slither static analysis tool. All the issues identified by the Slither tool were false positives.



project's integrity and addressing potential vulnerabilities introduced by code modifications.

© Halborn 2025. All rights reserved.