QuillAudits

Audit Report
December, 2023

For

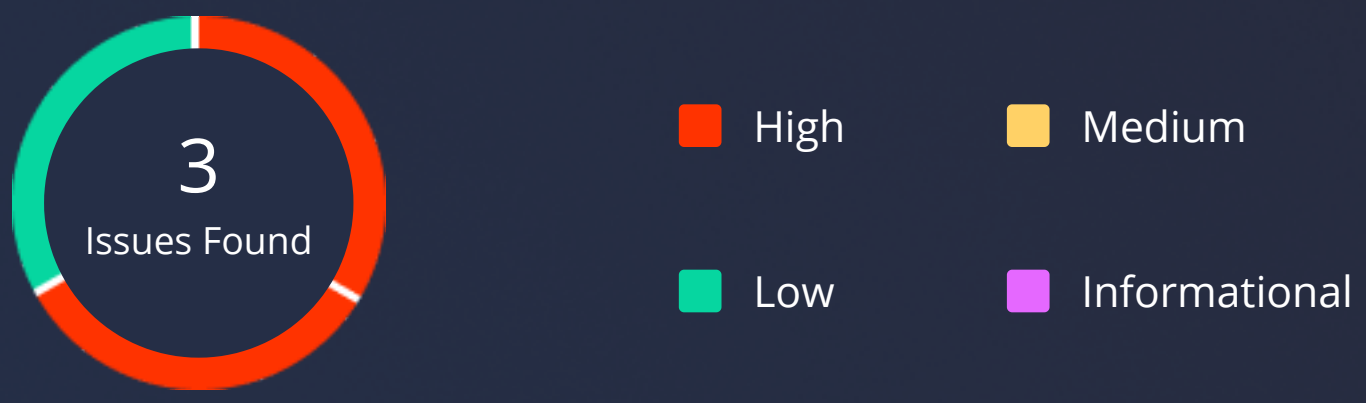
NovaDEX

Table of Content

Executive Summary

Project Name	NovaDEX-CLMM
Project URL	*https://novadex.io/*
Overview	NovaDEX is a community-driven concentrated liquidity DEX (decentralized exchange) built on the Solana blockchain. It aims to facilitate lightning-fast trades and offers features such as trading, swapping, and earning yield through fees and yield farms.
Audit Scope	The scope of this audit was to analyze and document the NovaDEX-CLMM Token smart contract codebase for quality, security, and correctness. *https://github.com/NovaDexDev/novadex-clmm*
Commit Hash	5f1a8739ce3f28b24877e9f1b64dbdf5781e4a0c
Language	Rust
Blockchain	Solana
Method	Manual Testing, Automated Tests, Functional Testing
Review 1	21st November 2023 - 29th November 2023
Updated Code Received	4th December 2023
Review 2	5th December 2023
Fixed In	628d440295e6270b402030a19e21505ab2f790e9

Number of Security Issues per Severity



	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	1	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	2	0	0	0

Checked Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Transaction-Ordering Dependence
- Exception Disorder
- Balance Equality
- Malicious Libraries
- Compiler Version Not Fixed
- Redundant Fallback Function
- Style Guide Violation
- Unchecked Math
- Unsafe Type Inference
- Implicit Visibility Level
- Correct Liquidity on Ticks

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code
- Use of best practices
- Code documentation and comments match logic and expected behaviour
- Liquidity computation on ranged ticks is correct

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Code base were completely manually analyzed, and their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

A high-severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

A. NovaDEX-CLMM

Issues Found – Code Review/Manual Testing

High Severity Issues

A.1 Potential for overflow data when converted to form u64 to i64

Line	Code/Function
decrease_liquidity.rs:L526	let (amount_0_int, amount_1_int, flip_tick_lower, flip_tick_upper) = modify_position(
open_position.rs: L665	let (amount_0_int, amount_1_int, flip_tick_lower, flip_tick_upper) = modify_position(
liquidity_math.rs:L234	-(i64::try_from(get_delta_amount_0_unsigned(

Description

In Rust, converting from u64 to i64 carries a risk of overflow. Such operations can be intricate and prone to causing bugs or potentially compromising the system's stability.

Remediation

To address this concern, it's advisable to avoid down-typecasting where possible. If unavoidable, it's crucial to implement robust measures that effectively mitigate the risks associated with such conversions.

Status

Resolved

A.2 Usage of crage::ID instead of poolID

Line	Code/Function
Create_pool.rs: L115	let observation_state_loader = initialize_observation_account(ctx.accounts.observation_state.to_account_info(), &crate::id(),)?;

Description

The retrieval of the **poolId** is being performed, yet it remains unused in the code. Instead, the code relies on the use of **crate::id**.

Remediation

To rectify this issue, consider implementing the usage of **poolID** where applicable in the code.

Status

Resolved

Low Severity Issues

A.3 Test Suite is failing and has no coverage report

Description

The test suite exhibits a failure rate of approximately 6 to 7 tests, indicating significant issues. Additionally, the absence of a coverage report implies a lack of information regarding test coverage.

Remediation

To rectify this issue, the recommended steps involve resolving the failing test cases and integrating a library such as **cargo-llvm-cov** to generate a comprehensive coverage report.

Status

Acknowledged

Automated Tests

Dylint:

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorised above according to their level of severity.

Closing Summary

Overall, in the initial audit, there are two high-severity issues associated with overflow when performing down typecasting and inconsistent pool id.

No instances of Integer Overflow and Underflow vulnerabilities are found in the contract.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in NovaDEX-CLMM smart contracts. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of NovaDEX-CLMM smart contracts. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of the NovaDEX-CLMM to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

850+
Audits Completed

$30B
Secured

$30B
Lines of Code Audited

Follow Our Journey

Audit Report
December, 2023

For

NovaDEX

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com