



Security Assessment **AURA DEX (CLMM)**

Verified by Vibranium Audits on 30 November 2024

Revised by Vibranium Audits on 14 December 2024



Vibranium Audits Verified on November 30th, 2024

AURA DEX (CLMM)

The security assessment was prepared by Vibranium Audits.

Executive Summary

TYPES

DEFI/NFT

ECOSYSTEM

Anchor

METHODS

Manual Review, penetration testing

LANGUAGE

Rust

TIMELINE

Delivered on 03/12/2024

KEY COMPONENTS

N/A

CODEBASE

Privately Shared Codebase

COMMITTS

N/A

Vulnerability Summary

6

Total Findings

4

Resolved

1

Mitigated

0

Partially Resolved

6

Acknowledged

1

Declined

0

Unresolved



0

Critical

Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation by external or internal actors.



3

High

2 Resolved

High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation by external or internal actors.



1

Medium

1 Resolved

Medium vulnerabilities are usually limited to state manipulations, but cannot lead to assets loss. Major deviations from best practices are also in this category.



1

Low

1 Resolved

Low vulnerabilities are related to outdated and unused code or minor gas optimization. These issues won't have a significant impact on code execution, but affect the code quality.



1

Informational

0 Resolved

Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

CODEBASE | AURADEX

Repository

N/A

Commits

N/A




AUDIT SCOPE | AURADEX

1 repo audited ● 1 repo with Acknowledged findings ● 0 files with Resolved findings



ID		Branch	Commit Hash
----	--	--------	-------------

● VAD	 N/A	N/A
-------	---	-----

APPROACH & METHODS | AURADEX

This report has been prepared for AURADEX(2024) to discover issues and vulnerabilities in the source code of the AURADEX project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review, rigorous Penetration Testing and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Pen-Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the code base to ensure compliance with current best practices and industry standards
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire code base by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices.

We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors.
- Enhance general coding practices for better structures of source codes.
- Review unit tests to cover the possible use cases.
- Review functions for readability, especially for future development work.

TABLE OF CONTENTS | AURADEX

Summary

- Executive Summary
- Vulnerability Summary
- Codebase
- Audit Scope
- Approach & Methods

Findings

- VAD-01 Potential Panics Due to Unchecked unwrap() and assert!
- VAD-02 Improper Use of Dynamic Data in Account Constraints
- VAD-03 Missing or Insufficient Input Validation
- VAD-04 Missing Event Emissions or Incomplete Event Data
- VAD-05 Unchecked Type Conversions Leading to Potential Panics
- VAD-06 Lack of Documentation and Comments

Disclaimer

FINDINGS | AURADEX

6

Total Findings

3

High

1

Medium

1

Low

1

Informational

This report has been prepared to discover issues and vulnerabilities for AURADEX.

Through this audit, we have uncovered 6 issues ranging from different severity levels.

Utilizing the techniques of Manual Review & Penetration Testing, we discovered the following findings:

ID	Title	Category	Severity	Status
VAD-01	Potential Panics Due to Unchecked unwrap() and assert!	Logical Issue	HIGH	● Resolved
VAD-02	Improper Use of Dynamic Data in Account Constraints	Logical Issue	HIGH	● Resolved
VAD-03	Missing or Insufficient Input Validation	Logical Issue	HIGH	● Mitigated
VAD-06	Lack of Documentation and Comments	Best Practices	Informational	● Rejected
VAD-05	Unchecked Type Conversions Leading to Potential Panics	Logical Issue	Low	● Revised
VAD-04	Missing Event Emissions or Incomplete Event Data	Logical Issue	Medium	● Revised

VAD-01 | Potential Panics Due to Unchecked unwrap() and assert!

Category	Severity	Location	Status
Logical Issue	● HIGH	<ul style="list-style-type: none">◦ create_pool◦ increase_liquidity◦ initialize_reward◦ set_reward_params◦ swap_router_base_in◦ swap_v2	● Resolved

Description

- Functions use `unwrap()` or `assert!` on operations that can fail, which may cause the program to panic if the operation returns `None` or an error. Examples include unchecked arithmetic operations, unwrapping results from functions that may return errors, and using `assert!` instead of proper error handling.
- Denial of Service: Panics consume all remaining compute units and cannot be recovered within the transaction, leading to a denial of service.
- Security Vulnerability: Malicious users could exploit these panics to disrupt the program.

Recommendation

- Handle Errors Gracefully:
 - Replace `unwrap()` and `assert!`: Use `require!`, `ok_or()`, or the `?` operator to handle errors properly.
 - Check Arithmetic Operations: Use checked arithmetic and handle potential overflows or underflows.
 - Define Error Codes: Add specific error codes for different error scenarios.

```
let result = operation.checked_sub(value).ok_or(ErrorCode::ArithmeticUnderflow)?;
```

```
require!(  
    (reward_index as usize) < REWARD_NUM,  
    ErrorCode::InvalidRewardIndex  
);
```

Revision

- The AuraDex team implemented the necessary error handling.

VAD-02 | Improper Use of Dynamic Data in Account Constraints

Category	Severity	Location	Status
Logical Issue	● HIGH	<ul style="list-style-type: none">• initialize_reward• set_reward_params• swap_v2	● Resolved

Description

- Account constraints use dynamic data loaded at runtime, which is not recommended in Anchor. This can lead to incorrect validation or unexpected behavior.
- Incorrect Validation: Dynamic constraints may not validate accounts properly, allowing unauthorized access.
- Security Risk: Unauthorized accounts could be used, compromising the program's security.

Recommendation

- Remove Dynamic Constraints:
 - Validate at Runtime: Perform necessary checks within the function body using `require!` macros.
 - Static Constraints: Use static data in account constraints whenever possible.

initialize_reward:

```
// Remove dynamic constraint
#[account()]
pub amm_config: Box<Account<'info, AmmConfig>>;

// In the function body
let pool_state = ctx.accounts.pool_state.load()?;
require_keys_eq!(
    ctx.accounts.amm_config.key(),
    pool_state.amm_config,
    ErrorCode::InvalidAmmConfig
);
```

Revision

- Proper Validations have been implemented.

VAD-03 | Missing or Insufficient Input Validation

Category	Severity	Location	Status
Logical Issue	● HIGH	<ul style="list-style-type: none">• create_pool• increase_liquidity• initialize_reward• set_reward_params• swap_router_base_in• swap_v2	● Mitigated

Description

- Input parameters are not fully validated to ensure they are within acceptable and safe ranges. Parameters like `sqrt_price_x64`, `liquidity`, `amount_in`, and time-related parameters need validation.
- Invalid Operations: Improper inputs may lead to unexpected behavior, errors, or vulnerabilities.
- Overflow/Underflow: Unchecked parameters may cause arithmetic overflows or underflows.

Recommendation

- Validate Inputs:
 - Use `require!` Macros: Check that inputs are within acceptable ranges.
 - Define Limits: Establish maximum and minimum allowed values.
 - Error Codes: Add error codes like `ErrorCode::InvalidParameter`.

`create_pool`:

```
require!(
    sqrt_price_x64 >= MIN_SQRT_PRICE_X64 && sqrt_price_x64 <= MAX_SQRT_PRICE_X64,
    ErrorCode::InvalidSqrtPrice
);
```

`increase_liquidity`:

```
require!(liquidity > 0, ErrorCode::InvalidLiquidityAmount);
```

Revision

- Inputs were properly validated in most cases.

VAD-04 | Missing Event Emissions or Incomplete Event Data

Category	Severity	Location	Status
Logical Issue	● MEDIUM	<ul style="list-style-type: none">• initialize_reward• set_reward_params• swap_router_base_in• swap_v2• increase_liquidity	● Resolved

Description

- Some functions do not emit events when critical actions are performed, or the events lack important details. This hinders transparency and makes off-chain monitoring difficult.
- Lack of Transparency: Without events, tracking actions becomes challenging.
- Audit Difficulty: Monitoring and auditing activities are hindered.

Recommendation

- Emit Comprehensive Events:
 - Define Events: Create event structs with all relevant fields.
 - Emit Events: Use emit! macro to emit events after critical actions.
 - Include Important Details: Such as amounts, accounts involved, and timestamps.

initialize_reward:

```
#[event]
pub struct RewardInitializedEvent {
    pub pool_state: Pubkey,
    pub reward_funder: Pubkey,
    // Other fields...
}

emit!(RewardInitializedEvent {
    pool_state: ctx.accounts.pool_state.key(),
    reward_funder: ctx.accounts.reward_funder.key(),
    // Other fields...
});
```

Revision

- Proper event emission was implemented in most cases.

VAD-05 | Unchecked Type Conversions Leading to Potential Panics

Category	Severity	Location	Status
Logical Issue	● LOW	<ul style="list-style-type: none">• initialize_reward• set_reward_params• swap_v2	● Resolved

Description

- Casting `unix_timestamp` from `i64` to `u64` without handling negative values may cause panics or incorrect behavior.
- Panic Risk: Negative timestamps may cause panics during type conversion.
- Incorrect Behavior: Time-dependent logic may fail with negative timestamps.

Recommendation

- Handle Type Conversions Safely:
 - Check for Negative Values: Use `require!` to ensure timestamps are non-negative.
 - Proper Casting: Only cast to `u64` after validation.

```
let unix_timestamp = Clock::get()?.unix_timestamp;  
require!(unix_timestamp >= 0, ErrorCode::InvalidTimestamp);  
let current_timestamp = unix_timestamp as u64;
```

Revision

- Proper type conversion checks have been implemented correctly.

VAD-06 | Lack of Documentation and Comments

Category	Severity	Location	Status
Logical Issue	● INFORMATIONAL GENERAL		● Rejected

Description

- The code lacks detailed comments and documentation explaining the purpose and functionality of key components.
- Maintainability: Future developers may find it challenging to understand or modify the code.
- Potential Misuse: Without clear documentation, there's a higher risk of unintended usage or misconfiguration.

Recommendation

- Enhance Documentation:
 - Function Comments: Add doc comments explaining what the function does, its parameters, and return values.
 - Inline Comments: Include comments within the code to explain complex logic.
 - Documentation Standards: Follow Rust's documentation conventions.

DISCLAIMER | VIBRANIUM AUDITS

This report is subject to the terms and conditions (including without limitation description of services confidentiality disclaimer and limitation of liability) set forth in the Services Agreement or the scope of services and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement This report may not be transmitted disclosed referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Vibranium Audits prior written consent in each instance

This report is not nor should be considered an “endorsement” or “disapproval” of any particular project or team This report is not nor should be considered an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Vibranium Audits to perform a security assessment This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed nor do they provide any indication of the technologies proprietors business business model or legal compliance

This report should not be used in any way to make decisions around investment or involvement with any particular project This report in no way provides investment advice nor should be leveraged as investment advice of any sort This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology

Blockchain technology and cryptographic assets present a high level of ongoing risk Vibranium Audits position is that each company and individual are responsible for their own due diligence and continuous security Vibranium Audits goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze

The assessment services provided by Vibranium Audits is subject to dependencies and under continuing development You Agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty The assessment reports could include false positives false negatives and other unpredictable results The services may access and depend upon multiple layers of third-parties

ALL SERVICES THE LABELS THE ASSESSMENT REPORT WORK PRODUCT OR OTHER MATERIALS OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW VIBRANIUM AUDITS HEREBY DISCLAIMS ALL WARRANTIES WHETHER EXPRESS IMPLIED STATUTORY OR OTHERWISE WITH RESPECT TO THE SERVICES ASSESSMENT REPORT OR OTHER MATERIALS WITHOUT LIMITING THE FOREGOING VIBRANIUM AUDITS SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY FITNESS FOR A PARTICULAR PURPOSE TITLE AND NON-INFRINGEMENT AND ALL WARRANTIES ARISING FROM COURSE OF DEALING USAGE OR TRADE PRACTICE WITHOUT LIMITING THE FOREGOING VIBRANIUM AUDITS MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES THE LABELS THE ASSESSMENT REPORT WORK PRODUCT OR OTHER MATERIALS OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF WILL MEET CUSTOMER’S OR ANOTHER PERSON’S REQUIREMENTS ACHIEVE ANY INTENDED RESULT BE COMPATIBLE OR WORK WITH ANY SOFTWARE SYSTEM OR OTHER SERVICES OR BE SECURE ACCURATE COMPLETE FREE OF HARMFUL CODE

OR ERROR-FREE WITHOUT LIMITATION TO THE FORGOING, VIBRANIUM AUDITS PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT WILL MEET CUSTOMER'S REQUIREMENTS ACHIEVE ANY INTENDED RESULTS BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE APPLICATIONS SYSTEMS OR SERVICES OPERATE WITHOUT INTERRUPTION MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED

WITHOUT LIMITING THE FOREGOING NEITHER VIBRANIUM AUDITS NOR ANY OF VIBRANIUM AUDITS AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND EXPRESS OR IMPLIED AS TO THE ACCURACY RELIABILITY OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE VIBRANIUM AUDITS WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS MISTAKES OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE OF ANY NATURE WHATSOEVER RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES ASSESSMENT REPORT OR OTHER MATERIALS

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS

THE SERVICES ASSESSMENT REPORT AND ANY OTHER MATERIALS HERE UNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT NOR MAY COPIES BE DELIVERED TO ANY OTHER PERSON WITHOUT VIBRANIUM AUDITS PRIOR WRITTEN CONSENT IN EACH INSTANCE

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES ASSESSMENT REPORT AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST VIBRANIUM AUDITS WITH RESPECT TO SUCH SERVICES ASSESSMENT REPORT AND ANY ACCOMPANYING MATERIALS

THE REPRESENTATIONS AND WARRANTIES OF VIBRANIUM AUDITS CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER ACCORDINGLY NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST VIBRANIUM AUDITS WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE

FOR AVOIDANCE OF DOUBT THE SERVICES INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Vibranium | Securing the Web3 World

Vibranium Audits is a blockchain security company that was founded in 2021 by professors from the University of Greenwich and cyber-security engineers from ITI Capital. As pioneers in the field, Vibranium Audits utilizes best-in-class Formal Verification and AI technology to secure and monitor blockchains, smart contracts, and Web3 apps.

