VIBRANIUM AUDITS

Security Assessment

AURA DEX (CPMM)

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 (CPMM)

The security assessment was prepared by Vibranium Audits.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DEFI/NFT	Anchor	Manual Review, penetration testing
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Rust	Delivered on 03/12/2024	N/A
CODEBASE		**COMMITS**
Privately Shared Codebase		N/A

Vulnerability Summary

6	5	0	0	6	1	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

1 Critical	1 Resolved	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.
2 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.
2 Medium	2 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.
0 Low	0 Resolved	Low vulnerabilities are related to outdated and unused code or minor gas optimization.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

FINDINGS | AURADEX

6	1	2	2	0	1
Total Findings	Critical	High	Medium	Low	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	Partial Transaction Failures Leading to Inconsistent State	Logical Issue	CRITICAL	Resolved
VAD-02	Use of .unwrap() Leading to Panics	Logical Issue	HIGH	Resolved
VAD-03	Arithmetic Overflows and Underflows	Logical Issue	HIGH	Resolved
VAD-06	Lack of Documentation and Comments	Best Practices	Informational	Rejected
VAD-05	Event Emission Before Successful Execution	Logical Issue	Medium	Resolved
VAD-04	Inconsistent Token Program Selection	Logical Issue	Medium	Resolved

VAD-01 | Partial Transaction Failures Leading to Inconsistent State

Category	Severity	Location	Status
Logical Issue	CRITICAL	collect_protocol_fee deposit swap_base_input swap_base_output withdraw	Resolved

Description

- The code updates the program's state before performing external calls, such as token transfers. If an external call fails after the state has been updated, the transaction will fail, but the prior state updates may have already been applied. This can lead to an inconsistent state where the program's state reflects changes that did not fully occur due to the failed external call.
- State updates (e.g., updating the pool_state) are performed before external calls (e.g., transferring tokens). If an external call fails after the state update, the state remains changed, but the external action did not occur.
  - If transfer_from_user_to_pool_vault fails, the lp_supply has already been incremented, but the user's tokens were not transferred to the pool.
  - The lp_supply reflects an increased supply of LP tokens, but the pool did not receive the corresponding tokens from the user, leading to an imbalance.

Recommendation

- Reorder Operations to Ensure Atomicity:
  - Perform all checks and external calls before updating the program's state.
  - Only update the state after all external calls have succeeded.
- Use Temporary Variables:
  - Calculate all changes and store them in temporary variables before applying them to the state.

```
// Perform external calls
transfer_from_user_to_pool_vault(...)?;
transfer_from_pool_vault_to_user(...)?;

// Update state after successful operations
pool_state.lp_supply = pool_state.lp_supply.checked_add(lp_token_amount)
    .ok_or(ErrorCode::ArithmeticOverflow)?;
```

Revision

- The AuraDex team implemented the necessary measures to mitigate against this issue.

VAD-02 | Use of .unwrap() Leading to Panics

Category	Severity	Location	Status
Logical Issue	HIGH	General	Resolved

Description

- Issue: Use of .unwrap() can cause panics if an operation fails.
- Panics lead to unexpected program termination, which can be exploited or cause denial of service.

Recommendation

Use Proper Error Handling:

```
let amount = u64::try_from(results.amount)
    .map_err(|_| ErrorCode::ConversionError)?;
```

Define Custom Error Codes:

```
#[error_code]
pub enum ErrorCode {
    ConversionError,
    ArithmeticOverflow,
    // ... other errors
}
```

Revision

The proper error handling has been implemented.

VAD-03 | Arithmetic Overflows and Underflows

Category	Severity	Location	Status
Logical Issue	HIGH	General	Resolved

Description

- The code frequently performs arithmetic operations (addition, subtraction, multiplication, division) without checking for overflows or underflows. For example, using methods like .unwrap() after checked_ arithmetic operations can lead to panics if the operation overflows or underflows.
- Unchecked Arithmetic Operations: In Rust, when using methods like checked_add, checked_sub, checked_mul, if the operation overflows, these methods return None. Calling .unwrap() on the result without checking can cause the program to panic.
- Example:

```
// Potential overflow if lp_supply + lp_token_amount exceeds u64::MAX
pool_state.lp_supply = pool_state.lp_supply.checked_add(lp_token_amount).unwrap();
```

- Unchecked overflows can cause the program to panic, leading to unexpected termination. In the context of Solana programs, panics are highly discouraged as they can be exploited to cause denial of service.
- Incorrect State Updates: Overflows and underflows can result in incorrect calculations, leading to incorrect balances, incorrect fee calculations, or incorrect token amounts being transferred.

Recommendation

- Use Safe Arithmetic Methods:

```
let total = amount.checked_add(fee)
    .ok_or(ErrorCode::ArithmeticOverflow)?;
```

Replace .unwrap() with proper error handling using ok_or and the ? operator.

Use Rust's built-in checked arithmetic functions (checked_add, checked_sub, checked_mul, checked_div) and handle None results appropriately.

Revision

- The necessary error handling for arithmetic operations has been mostly covered.

VAD-04 | Event Emission Before Successful Execution

Category	Severity	Location	Status
Logical Issue	MEDIUM	• deposit • swap_base_input • swap_base_output • withdraw	Resolved

Description

- Events are emitted before all operations in the transaction have successfully completed. If an error occurs after the event is emitted but before the transaction finishes, the event will still be recorded on the blockchain, even though the transaction ultimately failed.
- The code emits events immediately after calculating values but before performing token transfers or updating state.
- If the transaction fails after the event is emitted, external observers relying on events (e.g., indexers, analytics tools) may believe that the deposit succeeded, leading to inaccurate data and potential user confusion.

Recommendation

- Emit Events Only After Successful Execution:
  - Reorder the code to perform all critical operations (token transfers, state updates) before emitting events.
  - Ensure that the event accurately reflects the final state after all operations have succeeded.

```
// Perform token transfers and state updates
transfer_from_user_to_pool_vault(...)?;
// Update pool_state
pool_state.lp_supply = pool_state.lp_supply.checked_add(lp_token_amount)
    .ok_or(ErrorCode::ArithmeticOverflow)?;

// Emit event after successful execution
emit!(LpChangeEvent { ... });
```

Revision

- The necessary correct event emitting has been implemented.

VAD-05 | Inconsistent Token Program Selection

Category	Severity	Location	Status
Logical Issue	MEDIUM	• collect_protocol_fee • deposit • withdraw	Resolved

Description

The code selects the token program (e.g., SPL Token v1 or Token2022) based on the owner of the mint account but lacks explicit checks or error handling when the mint's owner does not match any expected token program.

- If the mint's owner does not match any known token program, the code may proceed with an incorrect token program, leading to failed transactions or security vulnerabilities.
- An attacker could create a malicious token program and set it as the owner of a mint, potentially causing the program to interact with the malicious token program without proper validation.

Recommendation

- Add Explicit Checks and Error Handling:
  - Check for Known Token Programs:

```
let token_program_info = if ctx.accounts.vault_0_mint.owner == ctx.accounts.token_program.key() {
    ctx.accounts.token_program.to_account_info()
} else if ctx.accounts.vault_0_mint.owner == ctx.accounts.token_program_2022.key() {
    ctx.accounts.token_program_2022.to_account_info()
} else {
    return Err(ErrorCode::InvalidTokenProgram.into());
};
```

Validate Mints and Token Programs Together:

- Ensure that for each mint, the corresponding token program is used.
- For token transfers, verify that the token account's owner matches the expected token program.

Revision

- The necessary error handling for arithmetic operations has been mostly covered.

VAD-06 | Lack of Documentation and Comments

Categor y	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 KINDTHAT 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 ORCURRENCY 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 OWNEROR 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 ORMATERIALS 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.