Three Sigma

Code Audit

Ordiswap Bitcoin AMM

Disclaimer

Code Audit

Ordiswap Bitcoin AMM

Disclaimer

The ensuing audit offers no assertions or assurances about the code's security. It cannot be deemed an adequate judgment of the contract's correctness on its own. The authors of this audit present it solely as an informational exercise, reporting the thorough research involved in the secure development of the intended contracts, and make no material claims or guarantees regarding the contract's post-deployment operation. The authors of this report disclaim all liability for all kinds of potential consequences of the contract's deployment or use. Due to the possibility of human error occurring during the code's manual review process, we advise the client team to commission several independent audits in addition to a public bug bounty program.

Table of Contents

Summary

Code Audit

Ordiswap Bitcoin AMM

Summary

Three Sigma audited Ordiswap in a 4 person week engagement. The audit was conducted from 26-12-2023 to 10-01-2024.

The codebase is still actively being developed and the Three Sigma team recommended the Ordiswap team to keep improving the protocol before launch. The Ordiswap team stated that the codebase will be released with a disclaimer and commentary that the audit will continue after the experimental mainnet launch, and this will all be accurately communicated to its users.

Protocol Description

Ordiswaps is a Bitcoin exchange which uses an off-chain server to manage the logic and parameters of the Automated Market Maker (AMM). This includes handling trades, liquidity provision, and setting dynamic AMM parameters.

Scope

Code Audit

Ordiswap Bitcoin AMM

Scope

The scope of the engagement includes the entire backend code for the Ordiswap server.

Assumptions

Since Ordiswap is a centralized custodial service, it is an assumption that the Ordiswap team is a trusted actor that will not act maliciously since they hold the custody of all the funds.

Methodology

Code Audit

Ordiswap Bitcoin AMM

Methodology

To begin, we reasoned meticulously about the backend's business logic, checking security-critical features to ensure that there were no gaps in the business logic and/or inconsistencies between the aforementioned logic and the implementation. Second, we thoroughly examined the code for known security flaws and attack vectors. Finally, we discussed the most catastrophic situations with the team and reasoned backwards to ensure they are not reachable in any unintentional form.

Taxonomy

In this audit we report our findings using as a guideline Immunefi's vulnerability taxonomy, which can be found at immunefi.com/severity-updated/. The final classification takes into account the severity, according to the previous link, and likelihood of the exploit. The following table summarizes the general expected classification according to severity and likelihood; however, each issue will be evaluated on a case-by-case basis and may not strictly follow it.

Severity / Likelihood	LOW	MEDIUM	HIGH
NONE	None		
LOW	Low		
MEDIUM	Low	Medium	Medium
HIGH	Medium	High	High
CRITICAL	High	Critical	Critical

Project Dashboard

Code Audit

Ordiswap Bitcoin AMM

Project Dashboard

Application Summary

Name	Ordiswap AMM
Commit	0c7e1081cb20a6cb69671e55028642cd4d6c444d
Language	Javascript
Platform	Bitcoin

Engagement Summary

Timeline	26-12-2023 to 10-01-2024
Nº of Auditors	2
Review Time	4 person weeks

Vulnerability Summary

Issue Classification	Found	Addressed	Acknowledged
Critical	3	1	0
High	3	1	0
Medium	3	1	0
Low	14	8	0
None	47	34	1

Category Breakdown

Suggestion	30
Documentation	1
Bug	24
Optimization	11
Good Code Practices	53

Code Maturity Evaluation

Code Audit

Ordiswap Bitcoin AMM

Code Maturity Evaluation

Code Maturity Evaluation Guidelines

Category	Evaluation
Access Controls	The use of robust access controls to handle identification and authorization and to ensure safe interactions with the system.
Arithmetic	The proper use of mathematical operations and semantics.
Centralization	The presence of a decentralized governance structure for mitigating insider threats and managing risks posed by contract upgrades
Code Stability	The extent to which the code was altered during the audit.
Upgradeability	The presence of parameterizations of the system that allow modifications after deployment.
Function Composition	The functions are generally small and have clear purposes.
Front-Running	The system's resistance to front-running attacks.
Monitoring	All operations that change the state of the system emit events, making it simple to monitor the state of the system. These events need to be correctly emitted.
Specification	The presence of comprehensive and readable codebase documentation.
Testing and Verification	The presence of robust testing procedures (e.g., unit tests, integration tests, and verification methods) and sufficient test coverage.

Code Maturity Evaluation Results

Category	Evaluation
Access Controls	Not in the scope of the audit. The codebase is a centralized closed-source server that can be updated at any time.
Arithmetic	Unknown. The team did not have enough time to thoroughly check all the arithmetics operations.
Centralization	Weak. The team has full control over the deposited funds and application functionality.
Code Stability	Satisfactory. The code was stable during the audit.
Upgradeability	Not in the scope of the audit. The codebase is a centralized closed-source server that can be updated at any time.
Function Composition	Satisfactory. Functionalities are well split into different contracts and helpers.
Monitoring	Weak. The codebase relies on external indexing tools to properly function.
Specification	Weak. The codebase does not have documentation or specification.
Testing and Verification	Weak. The codebase does not have testing or verification.