STRONGHOLD
SECURITY

RabbitX Security Audit Report

August 25, 2023

Contents

Executive Summary

Title	Description
Client	RabbitX
Project	Rabbitx-contracts
Platform	Ethereum
Language	Solidity
Repository	https://github.com/rabbitx-io/rabbitx-contracts
Initial commit	af4635a7b96df0db1b17b8ab849c3e7120a94688
Final commit	af4635a7b96df0db1b17b8ab849c3e7120a94688
Timeline	August 15 2023 - August 17 2023

Project Overview

RabbitX is a global permissionless perpetuals exchange powered on Starknet. RabbitX transactions are settled on the layer-2 Starknet before being broadcasted on layer-1 Ethereum.

Audit Scope

File	SHA
Rabbit.sol	b06ab065a4d2f0d9e53d95876f3b4d282f874be5
RabbitDeposit.sol	fbfdee366791d5e4e04000b0b8a20cc359a0969d

Audit Methodology

General Code Assessment

The code is reviewed for clarity, consistency, style, and whether it follows code best practices applicable to the particular programming language used, such as indentation, naming convention, commented code blocks, code duplication, confusing names, irrelevant
or missing comments, etc. This part is aimed at understanding the overall code structure and protocol architecture. Also, it seeks to learn overall system architecture and business logic and how different parts of the code are related to each other.

Code Logic Analysis

The code logic of particular functions is analyzed for correctness and efficiency. The code is checked for what it is intended for, the algorithms are optimal and valid, and the correct data types are used. The external libraries are checked for relevance and correspond to the tasks they solve in the code. This part is needed to understand the data structures used and the purposes for which they are used. At this stage, various public checklists are applied in order to ensure that logical flaws are detected.

Entities and Dependencies Usage Analysis

The usages of various entities defined in the code are analyzed. This includes both: internal usage from other parts of the code as well as possible dependencies and integration usage. This part aims to understand and spot overall system architecture flaws and bugs in integrations with other protocols.

Access Control Analysis

Access control measures are analyzed for those entities that can be accessed from outside. This part focuses on understanding user roles and permissions, as well as which assets should be protected and how.

Use of checklists and auditor tools

Auditors can perform a more thorough check by using multiple public checklists to look at the code from different angles. Static analysis tools (Slither) help identify simple errors and highlight potentially hazardous areas. While using Echidna for fuzz testing will speed up the testing of many invariants, if necessary.

Vulnerabilities

The audit is directed at identifying possible vulnerabilities in the project's code. The result of the audit is a report with a list of detected vulnerabilities ranked by severity level:

Severity	Description
Critical	Vulnerabilities leading to the theft of assets, blocking access to funds, or any other loss of funds.
High	Vulnerabilities that cause the contract to fail and that can only be fixed b y modifying or completely replacing the contract code.
Medium	Vulnerabilities breaking the intended contract logic but without loss of fun ds and need for contract replacement.
Low	Minor bugs that can be taken into account in order to improve the overall qu ality of the code

After the stage of bug fixing by the Customer, the findings can be assigned t he following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect it s security.
Acknowledged	The Customer took into account the finding. However, the recommendations wer e not implemented since they did not affect the project's safety.

Findings Summary

Severity	# of Findings
Critical	0
High	0
Medium	0
Low	3

ID	Severity	Title	Status
L-1	Low	Null checks	Acknowledged
L-2	Low	Fee Tokens	Acknowledged
L-3	Low	Centralization Risk	Acknowledged

Findings

Critical

Not Found

High

Not Found

Medium

Not Found

Low

L-1 Low Null checks Acknowledged

Description

Some parameters have no null checks:

RabbitDeposit.sol#L129

Rabbit.sol#L900

Recommendation

We recommend adding a null check for `_paymentToken`.

Client's commentary

> Acknowledged

L-2 Low Fee Tokens Acknowledged

Description

Some ERC20 tokens charge a transaction fee for every transfer (for example, USDT is a fee token with a null commission now). Thus, the amount of the tokens received using `transferFrom` may differ from the transfer amount.

RabbitDeposit.sol#L140

RabbitDeposit.sol#L144

Rabbit.sol#L945

Rabbit.sol#L948

Recommendation

We recommend checking token balances before and after `transferFrom`.

Client's commentary

> Acknowledged

L-3 Low Centralization Risk Acknowledged

Description

The owner of the Rabbit contract can withdraw all funds with the `withdrawTokensTo` function (or using the `withdraw` function with the `external_signer` param).

Recommendation

We recommend using governance module.

Client's commentary

> Acknowledged

Conclusion

During the audit process 3 LOW severity findings have been spotted. After working through the reported findings, all of them have been acknowledged or fixed by the client.

Disclaimer

The Stronghold audit makes no statements or warranties about the utility of the code, the safety of the code, the suitability of the business model, investment advice, endorsement of the platform or its products, the regulatory regime for the business model, or any other statements about the fitness of the contracts to purpose, or their bug-free status. The audit documentation is for discussion purposes only.