CODE SECURITY ASSESSMENT

ROOT FINANCE

Triwei Assessed on 24 September 2024

CODE SECURITY ASSESEMENT

ROOT FINANCE

Triwei Assessed on

24 September 2024

TABLE OF CONTENTS

PROJECT DESCRIPTION **4**

EXECUTIVE SUMMARY **4**

REVIEWS **4**

SECURITY FINDINGS **5**

C-01. ACCESS CONTROL VIOLATION 5

M-02. LACK OF LIQUIDATOR BADGE REVOCATION MECHANISM 6

M-03. CENTRALISED PRICE FEED 6

L-01. CREATION OF EMPTY COLLATERALIZED DEBT POSITIONS 7

L-02. LACK OF DECIMAL PRECISION HANDLING 7

DETAILED SCOPE 8

APPROACH AND METHODOLOGY **9**

LIMITATIONS AND USE OF REPORT **10**

PROJECT DESCRIPTION

Root is a decentralised money market protocol operating on the Radix DLT, where lenders and borrowers interact within an overcollateralized market environment.

It's a non-custodial liquidity protocol enabling users to participate either as depositors or borrowers.

EXECUTIVE SUMMARY

Type	DeFi	**Languages**	Scrypto/Rust
Methods	Architecture Review, Manual Review, Unit Testing, Functional Testing, Automated Review		
Documentation	https://docs.rootfinance.xyz/		
Repository	Detailed scope can be found at the end of the document		

REVIEWS

Review	Date	Commit
#1	29/08/2024	4d8ed16
#2	24/09/2024	40ea51d

SECURITY FINDINGS

▲ ▲ ▲ ▲ Critical

C-01. ACCESS CONTROL VIOLATION

Severity	Critical	**Impact**	High	**Likelihood**	High
Type	Access Control	**Commit**	4d8ed16	**Status**	Fixed
Target	./lending_market/src/lending_market.rs: fn take_batch_flashloan()				

Description: A critical vulnerability has been identified in the flashloan mechanism of the lending market system. The current implementation allows for unrestricted burning of `TransientResData` resources, which could be exploited to avoid repaying flashloans.

This vulnerability could lead to significant financial losses for the lending pool and its depositors. It undermines the entire flashloan mechanism and poses a severe risk to the stability and trustworthiness of the lending market.

A test to reproduce the issue has been pushed to the repository: './lending_merket/rests/blueprints/flashloan.rs: test_exploit_flashloan_by_burning_transient()'

Recommendation: Modify the `TransientResData` resource in `resources.rs` to restrict burning. An example implementation is provided in the audit commit.

▲▲▲ High

No high severity issues were found.

▲▲ Medium

M-02. LACK OF LIQUIDATOR BADGE REVOCATION MECHANISM

Severity	Medium	**Impact**	Medium	**Likelihood**	Medium
Type	Access Control	**Commit**	4d8ed16	**Status**	Mitigated
Target	./lending_market/src/lending_market.rs: fn mint_liquidator_badge()				

Description: The `mint_liquidator_badge` function currently allows for the minting of liquidator badges but lacks a mechanism for revoking them. While the function effectively creates new badges for users, the inability to revoke badges could present potential risks in scenarios where a user's liquidator rights need to be revoked, such as in the case of misuse or after a certain time period.

Recommendation: Implement a mechanism to revoke liquidator badges.

This would allow for better control over who holds liquidator privileges and ensure that badges can be deactivated or reassigned as necessary.

M-03. CENTRALISED PRICE FEED

Severity	Medium	**Impact**	High	**Likelihood**	**Low**
Type	Centralization Risk	**Commit**	4d8ed16	**Status**	Mitigated

Description: The current price feed mechanism relies on a single source of truth, which introduces a centralization risk.

If this single price feed is compromised or experiences downtime, it could affect the entire system's functionality.

Recommendation: Implement a more robust, decentralized price feed system.

Consider using a combination of multiple price feeds and implementing a median or weighted average mechanism.

Low

L-01. CREATION OF EMPTY COLLATERALIZED DEBT POSITIONS

Severity	Low	**Impact**	Low	**Likelihood**	Medium
Type	Logic Flaw	**Commit**	4d8ed16	**Status**	Fixed
Target	./lending_market/src/lending_market.rs: fn create_cdp()				

Description: A potential issue has been identified in the \`create_cdp\` function of the lending market system.

The current implementation allows the creation of a Collateralized Debt Position (CDP)

even when no collateral is provided (i.e., when the \`deposits\` vector is empty).

Although there is a note in the code indicating that the creation of empty CDPs should be forbidden,

this restriction is not enforced.

Recommendation: Implement a check within the \`create_cdp\` function to prevent the creation

of a CDP if the \`deposits\` vector is empty.

If the vector is empty, the function should return an error or halt the process

to ensure that all CDPs have collateral.

L-02. LACK OF DECIMAL PRECISION HANDLING

Severity	Low	**Impact**	Low	**Likelihood**	Medium
Type	Numeric Precision	**Commit**	4d8ed16	**Status**	Mitigated

Target	Throughout the codebase, especially in financial calculations

Description: The project extensively uses the `Decimal` type for financial calculations, but there's no consistent handling of decimal precision or rounding.

This could lead to small discrepancies in calculations, which may accumulate over time or in high-volume scenarios.

Recommendation: Implement a standardized approach to decimal precision and rounding throughout the codebase.

Consider creating helper functions for financial calculations that enforce consistent precision and rounding rules.

DETAILED SCOPE

Last revision - Commit 4d8ed164b52d58bd688209a0e5ae038428620d98

Full path	LOCs
lending_market\src\lending_market.rs	287
lending_market\src\lib.rs	3
lending_market\src\resources.rs	130
lending_market\src\modules\cdp_data.rs	183
lending_market\src\modules\cdp_health_checker.rs	344
lending_market\src\modules\interest_strategy.rs	52
lending_market\src\modules\liquidation_threshold.rs	106
lending_market\src\modules\market_config.rs	32
lending_market\src\modules\mod.rs	9
lending_market\src\modules\operation_status.rs	109
lending_market\src\modules\pool_config.rs	167
lending_market\src\modules\pool_state.rs	118
lending_market\src\modules\utils.rs	33

single_resource_pool\src\lib.rs	105
internal_price_feed\src\lib.rs	157

External Documentation

File name	SHA3
EXPLAINATION OF ROOT FINANCE MONEY MARKET.pdf	b35603c0e54f16b75b45fa0a8c29d95aabc241f5f52298fa8795ca533e82f90f

APPROACH AND METHODOLOGY

To establish a uniform evaluation, we define the following terminology in accordance with the OWASP Risk Rating Methodology:

Likelihood

indicates the probability of a specific vulnerability being discovered and exploited in real-world scenarios

Impact

measures the technical loss and business repercussions resulting from a successful attack

Severity

reflects the comprehensive magnitude of the risk, combining both the probability of occurrence (likelihood) and the extent of potential consequences (impact)

Likelihood and impact are divided into three levels: High (H), Medium (M), and Low (L). The severity of a risk is a blend of these two factors, leading to its classification into one of four tiers: Critical, High, Medium, or Low.

When we identify an issue, our approach may include deploying contracts on our private testnet for validation through testing. Where necessary, we might also create a Proof of Concept (PoC) to demonstrate potential exploitability.

In particular, we perform the audit according to the following procedure:

Security Analysis

The process begins with a comprehensive examination of the system to gain a deep understanding of its internal mechanisms, identifying any irregularities and potential weak spots.

Semantic Consistency Checks

We then manually check the logic of implemented smart contracts and compare with the description in the white paper.

Advanced DeFi Scrutiny

We further review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.

Additional Recommendations: We also provide additional suggestions regarding the coding and development of smart contracts from the perspective of proven programming practices.

To effectively classify each detected issue, we utilize the Common Weakness Enumeration (CWE-699), a community-curated catalogue of software weakness types. This helps in precisely defining and organizing software development weaknesses. While some CWE-699 categories may not directly apply to smart contracts, we adapt them in our classification process. Furthermore, for issues impacting active protocols, the public report version may temporarily exclude specific details, which will be fully disclosed once the protocol is updated with the necessary fixes.

LIMITATIONS AND USE OF REPORT

Security evaluations can't identify all vulnerabilities; a vulnerability-free assessment doesn't equate to a fully secure system. Nonetheless, code reviews are crucial for uncovering overlooked vulnerabilities and pinpointing areas needing enhanced security. Typically, applications are either entirely secure against a specific attack or wholly vulnerable. Some vulnerabilities may impact the whole application, whereas others only affect specific sections. Therefore, we conduct a thorough source code review to identify all areas requiring attention. Within the timeframe set by the client, Chain Security strives to detect as many vulnerabilities as possible.

Our analysis was confined to the code sections specified in the engagement letter. We examined if the project adhered to the given specifications, guided by the outlined threat model and trust assumptions. It's important to note that due to the inherent limitations of any software development process and product, there is always a risk of significant undetected errors or malfunctions. Additionally, uncertainties arise from any software or application used in development, as they too can harbor errors or failures. These factors can influence the system's code, functions, or operation. Our assessment did not cover the underlying third-party infrastructure, introducing additional inherent risks depending on the accurate functioning of the third-party technology stack. Readers of this report should also consider that software changes over its life cycle, as well as changes in its operating environment, can lead to operational behaviors that deviate from the original business specifications.

Check out our collection of articles exploring in-depth security options for your project!
TriWei.io/education