HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Caviar
Date: October 17th, 2023

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for Caviar
Approved By	
Type	DeFi, Exchange
Platform	Radix
Language	Rust
Methods	Manual Review, Architecture review
Website	https://www.caviarnine.com/
Timeline	28.09.2023 - 13.10.2023
Changelog	17.10.2023 - Initial Review 26.10.2023 - Second Review

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by Caviar (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

Scope

The scope of the project is smart contracts in the repository:

Initial review scope
Repository:
https://github.com/InvariancePte/scrypto-caviar
Commit:
c7981c9a6b1b34080cb08841e431e8ae7419916d

Integration and Unit Tests: Yes

Contracts:
File: ./quantaswap/src/bin.rs
SHA2: a3cfbb0375acc916a1328a621a23724a7a0ac189ad70b2f44b975c3583340c72

File: ./quantaswap/src/const.rs
SHA2: 33890c01b4dd79e7682813841044d2cadb3fd18e1ce364b5fe38cf35d75fe70b

File: ./quantaswap/src/events.rs
SHA2: 3ba96a3c29fdd823ff7c52a1bf2b95c8e80e3e3e8dc8f52d655451df3934a03d

File: ./quantaswap/src/lib.rs
SHA2: a8a8ae76869db514fffd34cb705c48c2ccfe102f2dcbd702f5ec3257e8d8a346

File: ./quantaswap/src/liquidity_receipt.rs
SHA2: 6d268fe5e30c54cc5102cc94ab8f5150de299f34799a5ee2afb621b6760c7965

File: ./quantaswap/src/quantaswap.rs
SHA2: df35ff03883ac6de13839481c626087d09b267ca3ac7f2f9f5cecad45ca20044

File: ./quantaswap/src/swap_math.rs
SHA2: c0de0568016593f9e6794c3b4d2eca1e85359cc68935c69d0cc719ab134e736b

File: ./quantaswap/src/tick_index.rs
SHA2: 711a84c100fff312c10b57f812d7330fdae125ed40f36fa5bebd00c6989fd847

File: ./quantaswap/src/tick.rs
SHA2: 876f311752b3838624857a6dedca87d840a143e34847e8009219ef5cae8cb655

Second review scope

Repository:
https://github.com/InvariancePte/scrypto-caviar

Commit:
ef8cf80a21c9faa3ab5bfcc422ce6cc8f050dda0

Integration and Unit Tests: Yes

Contracts:

File: ./quantaswap/src/bin.rs
SHA2: a3cfbb0375acc916a1328a621a23724a7a0ac189ad70b2f44b975c3583340c72

File: ./quantaswap/src/const.rs
SHA2: 33890c01b4dd79e7682813841044d2cadb3fd18e1ce364b5fe38cf35d75fe70b

File: ./quantaswap/src/events.rs
SHA2: 147e6785d360520743c14b0d3b0949058c5f862129410b605774833822ccf3ef

File: ./quantaswap/src/lib.rs
SHA2: a8a8ae76869db514fffd34cb705c48c2ccfe102f2dcbd702f5ec3257e8d8a346

File: ./quantaswap/src/liquidity_receipt.rs
SHA2: a8a8ae76869db514fffd34cb705c48c2ccfe102f2dcbd702f5ec3257e8d8a346

File: ./quantaswap/src/quantaswap.rs
SHA2: 48f8c00b159ab6a32fbe10316a3efd74d292e1af61122c074cec7fc7d9cbfbe5

File: ./quantaswap/src/swap_math.rs
SHA2: c0de0568016593f9e6794c3b4d2eca1e85359cc68935c69d0cc719ab134e736b

File: ./quantaswap/src/tick_index.rs
SHA2: 711a84c100fff312c10b57f812d7330fdae125ed40f36fa5bebd00c6989fd847

File: ./quantaswap/src/tick.rs
SHA2: 876f311752b3838624857a6dedca87d840a143e34847e8009219ef5cae8cb655

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions.
Medium	Medium-level vulnerabilities are important to fix; however, they cannot lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution.

Executive Summary

The Score measurements details can be found in the corresponding section of the methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10.** The project boasts comprehensive and meticulously detailed documentation, reflected in a well-structured README.md file. This not only provides essential overviews and introductions but also serves as a reliable reference for users and developers alike.

One of the standout attributes of the project's documentation is the consistent commenting throughout the codebase. Each significant function is supplemented with clear and concise comments, shedding light on the purpose, functionality, and underlying mechanics. These annotations effectively bridge any potential gaps in understanding and offer valuable insights into the code's nuances.

Moreover, the code itself embodies a degree of clarity that makes it innately comprehensible. Designed with the developer's perspective in mind, most segments of the code are self-explanatory.

Code quality

The total Code Quality score is **10** out of **10**. The codebase is a clear example of high-quality coding standards. It's not just well-organized and clean, but it's also built using best practices, making it easy to read and understand. The added documentation in comments helps explain each part, ensuring anyone looking at the code can understand it easily.

Importantly, the testing approach is impressive. With nearly 100% coverage for both functions and lines, it's clear that a lot of effort has been put into making sure the code works as expected. This level of testing is not just good for this project but sets a great example for other projects in the Radix ecosystem. It's a clear indicator of the value the team places on quality and reliability.

The developers have also shown a deep understanding of the Radix ecosystem. Their expertise is evident in the way they've used the platform, ensuring that the code isn't just strong on its own, but also makes the most of what Radix protocol has to offer.

Architecture quality

The architecture quality score is **10** out of **10**. The project's architectural design is both efficient and well-organized. The quantaswap interacts with

other packages that match the same high quality as the audited package. These interactions are streamlined, with each package maintained as separate projects, ensuring clarity and ease of management.

A clear indicator of the project's well-thought-out architecture is the event system. Every action within the project triggers the appropriate events, ensuring transparency and facilitating easier debugging and monitoring.

Data storage within the project has been approached with efficiency in mind. The chosen methods ensure that data is stored compactly, minimizing overhead and ensuring fast access.

Moreover, when it comes to the functional implementation, the project doesn't compromise. Each function is designed to be gas-efficient, ensuring that operations are not just effective but also cost-effective. There's a clear emphasis on avoiding any wastage, optimizing both in terms of performance and resource consumption.

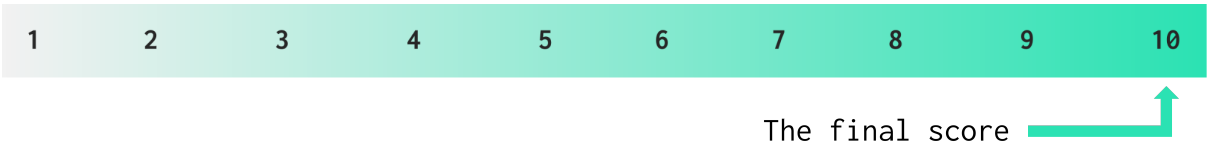
Security score

As a result of the audit, the code contains **1** low severity issue. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **10**.



Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Findings

Critical

1. Missing ownership validation

No critical severity issues were found.

High

No high severity issues were found.

Medium

No medium severity issues were found.

Low

1. Lack of bucket address validation in burn_liquidity_receipt

The function *burn_liquidity_receipt* takes as an argument Bucket, which contains Liquidity Receipt to be burned. The problem is that it lacks validation of Bucket address which allows the use of a Bucket which was not created by Quantaswap. However, this vulnerability can be only used to emit an event that a given bucket was burned, with current implementation it cannot lead to any other problem.

File: ./quantaswap/src/quantaswap.rs

Recommendation: an assertion *liquidity_receipt.resource_address() == self.liquidity_receipt_manager.address()* should be added to this function.

Status: Fixed in ef8cf80a

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on the security of the code. It also cannot be considered a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit cannot guarantee the explicit security of the audited smart contracts.