DefiPlaza Radix Security Analysis

by Pessimistic

This report is public

November 23, 2023

Abstract

In this report, we consider the security of the codebase of DefiPlaza Radix project. Our task is to find and describe security issues in the codebase of the platform.

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of the code. Besides, a security audit is not investment advice.

Summary

In this report, we considered the security of DefiPlaza Radix project smart contracts available on a private GitHub repository. We performed our audit according to the procedure described below. The initial audit showed one medium-severity and several low-severity issues.

After the initial audit, the developers provided us with a new version of the code. Additional details on that can be found in the Codebase update #1 section.

After the recheck, the codebase was updated to fix the issue discovered by developers during the final field testing.

Project overview

Project description

For the audit, we were provided with DefiPlaza Radix project on a private GitHub repository, commit 8820576c3bb6250ad145d04b97f804b8d2003a73.

The scope of the audit included:

- **src/pair.rs**,
- **src/helpers.rs**,
- **src/curves.rs**,
- **src/dex.rs**.

The documentation for the project included `whitepaper.pdf` (sha1sum: **4618fe5314623f35724660ed38e08412cf0eca7f**).

All 68 tests pass successfully.

Codebase update #1

After the audit, we were provided with a commit 5b41a00013bbba7452f97315dbe951d64df77c2c. In that update, the developers fixed one medium and most of the low-severity issues, as well as added new tests for the project. Moreover, the developers fixed the issue discovered by them (L12). The update introduced a new discrepancy with the documentation, which we described in the L08 issue. In addition to this, we added a new low-severity issue L09.

All 94 tests pass successfully.

Codebase update #2

After the recheck, we were provided with a commit 34b3c746e54ce1f3fe036ccbaa3ad722b5b2f854. In this update, the developers resolved an issue (H01), which had been uncovered by the developers during testing. The number of tests decreased. Moreover, L08 was fixed. During the recheck, we discovered two new low-severity issues (L10, L11).

All 80 tests pass successfully.

Procedure

We perform the audit according to the following procedure:

- **Automated analysis**
  - We compile the contracts.
  - We run the provided tests and calculate code coverage using Cargo Tarpaulin.
  - We manually verify (reject or confirm) all issues reported by the following tools:
    - Cargo Geiger;
    - Cargo Audit.
- **Manual audit**
  - We manually review the code and assess its quality.
  - We check the code for known vulnerabilities.
  - We check whether the code logic complies with the provided documentation.
  - We suggest possible gas and storage optimizations.
- **Report**
  - We reflect all the gathered information in the report.

Issues

We are actively looking for:

- Access control issues (incorrect admin or user identification/authorization).
- Lost/stolen assets issues (assets being stuck on the contract or sent to nowhere or to a wrong account).
- DoS due to logical issues (deadlock, state machine error, etc).
- Contract interaction issues (reentrancy, insecure calls, caller assumptions).
- Arithmetic issues (overflow, underflow, rounding issues).
- Other issues.

Automated analysis

Automated analysis shows the following:

- **Auto-tests**
  All 68 tests pass successfully.

  After the initial audit, the number of tests increased to 94. They all pass successfully.

- **Tests coverage**
  For code coverage calculation, we used Cargo Tarpaulin.
  The code coverage could not be calculated due to the nature of wasm based test system in the Radix Scrypto engine. The manual review showed uncovered `view` and `write` external functions:
  - **src/dex.rs**:
    - blacklist
    - deblacklist
    - delist
    - remove_liquidity
    - quote
    - get_lp_tokens
    - set_min_dfp2
    - swap
    - update_lp_metadata
    - withdraw_owned_liquidity
  - **src/pair.rs**:
    - assess_pool
    - donate_to_pool
    - quote
    - select_pool

- **Check for unsafe Rust code**
  For unsafe Rust code test, we used Cargo Geiger. The tool terminates with an error. However, a manual check showed that unsafe blocks were not used.

- **Check for crates vulnerabilities**
  To test the codebase for crates vulnerabilities, we used Cargo Audit. The tool showed one vulnerability, but in a third-party crate (`scrypto-unit`). Note that this vulnerability is associated with Radix and cannot currently be addressed within the audited project.

Manual analysis

High-severity issues

High-severity issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

H01. Possible token loss (fixed)

The `add_liquidity` function in the `pair.rs` might result in a token loss if there is a shortage of base tokens.

The problem was resolved by eliminating complex calculations in instances of shortages and implementing the use of reserves in these scenarios.

Medium-severity issues

Medium-severity issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

M01. Pairs removed from the blacklist are not re-listed in DEX (fixed)

In **src/dex.rs**, when a previously blacklisted pair is removed from the blacklist (deblacklisted), it will not be listed in DEX. In order to make this pair functional again, it is required to create a new pair, force all users to withdraw liquidity from the old pair, and deposit to the new one.

The issue has been fixed and is not present in the latest version of the code.

Low-severity issues

Low-severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

L01. Missing check (fixed)

In the `fn create_pair()` function of the **src/dex.rs** file, the initial price `p0` for the pair is not checked for negative or zero value.

The issue has been fixed and is not present in the latest version of the code.

L02. High initial price

In the `fn create_pair()` function of the **src/dex.rs** file, the initial price `p0` may initially be set to a significantly high value and might not be modifiable at a later stage.

L03. Discrepancy with documentation (fixed)

In the `quote()` function of the **src/pair.rs** file, the description states two conditions for `panic` for the `input_amount` parameter:

- it should be greater than zero,
- it should be greater than the traded amount.

However, the second condition is not checked in the code.

The issue has been fixed and is not present in the latest version of the code.

L04. Additional roles (addressed)

Consider using additional (different from `Admin`) roles for royalty claiming and liquidity withdrawal (`withdraw_owned_liquidity()`).

> *Comment from the developers: Due to how Radix Scrypto works, there are already two separate roles here. The royalties are claimed by the package owner, while the owned liquidity can be withdrawn by the DEX owner (the entity that instantiated the DEX from the blueprint) which is not the same role / authorization. Anyone can spawn a new instance of the DEX from the blueprint, they would then own that instance and can withdraw liquidity. However, only the package owner (ie the dev team) can collect royalties.*

L05. Consistency with the whitepaper

For better clarity and consistency, we recommend ensuring a precise correspondence between the function descriptions and the CALM whitepaper.

L06. High fee parameter (fixed)

The user has the ability to set the fee parameter at an unreasonably high value, such as 0.99999... There are no code-based restrictions in place to verify the fee's appropriateness, such as ensuring it does not exceed 10%.

The issue has been fixed and is not present in the latest version of the code.

L07. Swapping between liquidity addition (commented)

In **src/pair.rs**, liquidity addition occurs in two sequential steps. Initially, one token's liquidity is added, followed by the addition of the second token's liquidity. However, if these additions are done in separate transactions, an attacker can execute a swap between these transactions at a price that deviates from the actual market rate.

> *Comment from the developers: This can be done in the same transaction using Radix transaction manifest, thereby preventing any attacker from going in between the two adds.*

L08. Discrepancy with the documentation (fixed)

In **dex.rs** file, lines 36 and 38 state that the variables `pair_to_lps` and `dex_reserves` work with the `ComponentAddress`. However, in reality, according to lines 46 and 47, these variables operate with the `Global<PlazaPair>` type.

L09. Freezable tokens

We recommend checking that used assets cannot be frozen. The respective asset manager can check whether the access rule for frozen permissions is different from the `rule!(deny_all)`.

L10. Discrepancy with function description (new)

The `add_liquidity` function, in lines 267-286 in the **dex.rs** file, is inconsistent with its description.

L11. Usage of constant (new)

We recommend declaring the value `0.9999` used in **pair.rs** as a constant variable in **constants.rs** and using that variable.

L12. Recallable tokens (fixed)

The resources sent to the pool are not checked if they are not recallable (created with revocation permission). Radix has a flexible permissions model, including the ability to revoke previously sent FTs and NFTs.

The issue was fixed by calling `assure_is_not_recallable()`.

Notes

N01. TWAP decay factor

The deployer configures pool settings, including the TWAP decay factor. Using a smaller value will result in inadequate protection against price manipulation. Make sure the decay factor is close enough to 1, so the TWAP functions properly.

N02. Overpowered role (addressed)

The `Admin` role can:

- Change the pair's properties (e.g., the token's name);
- Withdraw DEX owned tokens from the pair;
- Redeem fees;
- Blacklist the pair.

There are scenarios that can lead to undesirable consequences for the project and its users, e.g., if the private key for this role becomes compromised. We recommend designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig.

> *Comment from the developers: The intent is to use multisig to safeguard the admin role as soon as it becomes available on Radix. It's available in the ledger but not yet in the wallet/tooling.*

N03. Outdated function description (new)

The description of the `is_quote` parameter at line 166 in the **pair.rs** file remains from the previous function description. We recommend removing it.

This analysis was performed by Pessimistic:

Sergey Grigoriev, Security Engineer
Nikita Kuznetsov, Security Engineer
Evgeny Marchenko, Senior Security Engineer
Yhtyyar Sahatov, Security Engineer
Pavel Kondratenkov, Senior Security Engineer
Irina Vikhareva, Project Manager
Konstantin Zherebtsov, Business Development Lead
Alexander Seleznev, Founder

November 23, 2023