code4rena

Print World

Smart Contract Security Assessment

Version 1.0

Audit dates:	Nov 26 — Dec 06, 2024
Audited by:	Victor Magnum Koolex

Contents

1. Introduction

1.1 About Zenith

Zenith is an offering by Code4rena that provides consultative audits from the very best security researchers in the space. We focus on crafting a tailored security team specifically for the needs of your codebase.

Learn more about us at https://code4rena.com/zenith.

1.2 Disclaimer

This report reflects an analysis conducted within a defined scope and time frame, based on provided materials and documentation. It does not encompass all possible vulnerabilities and should not be considered exhaustive.

The review and accompanying report are presented on an "as-is" and "as-available" basis, without any express or implied warranties.

Furthermore, this report neither endorses any specific project or team nor assures the complete security of the project.

1.3 Risk Classification

SEVERITY LEVEL	IMPACT: HIGH	IMPACT: MEDIUM	IMPACT: LOW
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

2. Executive Summary

2.1 About Print World

Printworld "a theme park built on the solana blockchain" - a token launch pad, trading tool, and social-fi app creating the future of memefi.

2.2 Scope

Repository	PrintXD-INC/hm-server/
Commit Hash	1a2cbfa701eaccdee9b2b13544b0b6f7cf438e92

2.3 Audit Timeline

DATE	EVENT
Nov 26, 2024	Audit start
Dec 06, 2024	Audit end
Dec 13, 2024	Report published

2.4 Issues Found

SEVERITY	COUNT
Critical Risk	0
High Risk	1
Medium Risk	2
Low Risk	2
Informational	1
Total Issues	6

3. Findings Summary

ID	DESCRIPTION	STATUS
H-1	In certain cases the users will not be able to claim the remainder of their multi claimable amount	Resolved
M-1	Improper Rent-Exempt Validation in Market Pot Initialization	Resolved

M-2	Slippage control check doesn't account for `market_pot.lamports() == 0`	Resolved
L-1	The max multi bps should be 20%, but is 19.99%	Resolved
L-2	Traders can set themselves as referrals to pay less fees	Resolved
I-1	`increase_reserves` and `decrease_reservers` in the BondingCurveMath trait are unused	Resolved

4. Findings

4.1 High Risk

A total of 1 high risk findings were identified.

[H-1] In certain cases the users will not be able to claim the remainder of their multi claimable amount

Severity: High	Status: Resolved

Context:

- buy.rs

Description: In the buy_token function, there's a vulnerability in how multi-token assignments are handled when the requested amount exceeds the remaining assignable amount. Currently, if a user's requested multi amount plus the total already assigned would exceed the 20% cap (bps), the user receives nothing (user_multi.last_assigned = 0). This a issue where the remaining assignable tokens become effectively locked if all subsequent requests are larger than the remaining amount.

For example:

- Total supply: 1,000,000
- Max assignable (20%): 200,000
- Already assigned: 190,000
- Remaining: 10,000

If a user's multi-claim is 100,000, they get 0 instead of the available 10,000

Recommendation: Modify the multi-assignment logic to assign the maximum possible amount instead of nothing when the requested amount exceeds the remaining assignable amount:

Print World: Fixed here

Zenith: Verified

4.2 Medium Risk

A total of 2 medium risk findings were identified.

[M-1] Improper Rent-Exempt Validation in Market Pot Initialization

Severity: Medium	Status: Resolved

Context:

- state.rs#L237

Description: During the buy operation, the code checks for market pot account initialization with a simple zero-balance check:

```
if market_pot.lamports() == 0 {
    trade_fee = trade_fee
        .checked_add(Rent::default().minimum_balance(8))
        .unwrap(); }
```

This creates a vulnerability where if the market pot account is pre-funded with an amount less than the rent-exempt minimum:

- The zero-balance check fails (since lamports > 0)
- The rent amount is not added to the trade fee
- The account remains under the rent-exempt threshold
- The account could be purged by the Solana runtime

This issue could lead to system instability or failures as the market pot account is critical for protocol operations but could be removed from the network if not properly rent-exempt.

Recommendation: Replace the zero-balance check with a minimum rent-exempt threshold check:

```
if market_pot.lamports() < Rent::default().minimum_balance(8) {
    let additional_needed = Rent::default().minimum_balance(8) -
market_pot.lamports();
    trade_fee = trade_fee.checked_add(additional_needed).unwrap(); }
```

Print World: Fixed with the following commit

Zenith: Verified.

[M-2] Slippage control check doesn't account for `market_pot.lamports() == 0`

Severity: Medium	Status: Resolved

Context:

- state.rs

Description:

When buying tokens, there's a check to ensure the `sol_to_transfer` + `platform_fee` + `trade_fee` is less than or equal `max_sol_out`

```
        require!(
>>>             sol_to_transfer.add(platform_fee).add(trade_fee) <=
max_sol_out,
            TetraError::SlippageExceeded
        );
```

However, later there's a check here if the market_pot lamports are 0, the rent gets added to the trade fee, which can, in theory be above the maximum slippage.

Recommendation: Increment the trade fee before the minimum slippage check

Print World: Fixed with @5aff939ab754..by dynamically adding additional amount

Zenith: Verified

4.3 Low Risk

A total of 2 low risk findings were identified.

[L-1] The max multi bps should be 20%, but is 19.99%

Severity: Low	Status: Resolved

Context:

- buy.rs

Description:

There is a bug in the multi-token assignment logic where users can only claim up to 19.99% instead of the intended 20% of tokens. This occurs because the code uses a strict less than comparison (<) when checking if the new total assigned amount is within the BPS limit. The current check:

```
if bonding_curve.total_multi_assigned.add(assign_amount) < bps
```

Prevents assignments that would exactly equal the BPS limit (20%), effectively capping the total assignable amount at 19.99%.

Recommendation: Change the comparison operator from < to <= to allow assignments that exactly match the BPS limit:

Print World: Fixed here

Zenith: Verified

[L-2] Traders can set themselves as referrals to pay less fees

Severity: Low	Status: Resolved

Context:

- state.rs

Description:

When buying/selling, there are referral fees that are taken from the platform fee if there's a referral. However, as it stands right now, there are no constraints on what the referral account might be.

This allows for the scenario where buyer/seller can set themselves as referrals purely to save on platform fee costs

Recommendation:

Add a check that prevents the caller to be the referral

Print World: Fixed here

Zenith: Verified

4.4 Informational

A total of 1 informational findings were identified.

[I-1] `increase_reserves` and `decrease_reservers` in the BondingCurveMath trait are unused

Severity: Informational	Status: Resolved

Context:

- state.rs

Description: The BondingCurveMath trait's increase_reserves and decrease_reserves functions are currently unused in the codebase. Instead, the reserve updates are performed directly within the buy and sell functions using the checked_add and checked_sub operations.

Recommendation:

Either:

1. Remove the unused increase_reserves and decrease_reserves functions to eliminate dead code, or
2. Refactor the buy and sell functions to utilize these helper functions

Print World: @8715d4abeaa43... - Removed those functions as we do all of those updates in buy/sell

Zenith: Verified