Pump Security Review

Pashov Audit Group

Conducted by: defsec, mahdiRostami, krikoeth, 0xhuy0512, llill, Johny

April 28th 2025 - May 1st 2025

Contents

1. About Pashov Audit Group

Pashov Audit Group consists of multiple teams of some of the best smart contract security researchers in the space. Having a combined reported security vulnerabilities count of over 1000, the group strives to create the absolute very best audit journey possible - although 100% security can never be guaranteed, we do guarantee the best efforts of our experienced researchers for your blockchain protocol. Check our previous work here or reach out on Twitter @pashovkrum.

2. Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where we try to find as many vulnerabilities as possible. We can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

3. Introduction

A time-boxed security review of the **pump-fun/pump-contracts-solana** and **pump-fun/pump-amm-2** repositories was done by **Pashov Audit Group**, with a focus on the security aspects of the application's smart contracts implementation.

4. About Pump AMM and Pump Solana

Pump on Solana is a platform for launching SPL coins that can be traded on a bonding curve without needing to provide initial liquidity. Once the coin reaches a particular market cap, liquidity is deposited from the bonding curve to Raydium, and the received LP tokens are burnt.

Pump AMM is an AMM on the Solana blockchain, built with the Anchor framework.

5. Risk Classification

Severity	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

5.1. Impact

- High - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- Medium - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- Low - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

5.2. Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

5.3. Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

6. Security Assessment Summary

review commit hashes:

- d38d4896c1e29e4f66a39c07caaadae0961ba17f
- c564cdcbc515919b20836679932cc88af08bb18f

fixes review commit hashes:

- 24587fd504c1ce7a11b7e214462332abdf79e323
- f42ced72c159b31bcd8fe3efe7159563e900a5bf

Scope

The following smart contracts were in scope of the audit:

- `collect_creator_fee`
- `mod`
- `set_creator`
- `set_metaplex_creator`
- `lib`
- `migrate`
- `fee`
- `create_config`
- `update_fee_config`
- `collect_coin_creator_fee`
- `set_coin_creator`
- `create_pool`
- `buy`
- `sell`
- `global_config`
- `token`
- `pool`

7. Executive Summary

Over the course of the security review, defsec, mahdiRostami, krikoeth, 0xhuy0512, llill, Johny engaged with Pump to review Pump AMM and Pump Solana. In this period of time a total of **2** issues were uncovered.

Protocol Summary

Protocol Name	Pump AMM and Pump Solana
Repository	https://github.com/pump-fun/pump-contracts-solana
Date	April 28th 2025 - May 1st 2025
Protocol Type	AMM and Bonding Curve tokensale

Findings Count

Severity	**Amount**
Low	2
Total Findings	**2**

Summary of Findings

ID	**Title**	**Severity**	**Status**
[L-01]	Inefficient size allocation pattern	Low	Resolved
[L-02]	Fee vault is made rent-exempt even if no fee shall be collected	Low	Resolved

8. Findings

8.1. Low Findings

[L-01] Inefficient size allocation pattern

The program implements an inefficient storage allocation strategy in the BondingCurve account structure. The code allocates `SIZE+175` bytes for each account when post-upgrade requirements with creator fee updates, leading to unnecessary resource consumption and cost inefficiencies.

This pattern of excessive pre-allocation creates several economic and performance issues:

1. Increased user costs: Users must pay higher rent fees for the over-allocated space, making transactions more expensive than necessary.

2. Validator storage burden: With large numbers of accounts being created, the excessive space allocation multiplies across the ecosystem, potentially adding gigabytes of unnecessary data to the validator storage requirements.

```
impl BondingCurve {
    pub const SIZE: usize = 8 + BondingCurve::INIT_SPACE;
+   pub const NEW_SIZE: usize = BondingCurve::SIZE + 175;

    pub fn buy_quote(&self, amount: u64) -> Result<u128> {
        let amount = u128::from(amount);
        let virtual_sol_reserves = u128::from(self.virtual_sol_reserves);
        let virtual_token_reserves = u128::from(self.virtual_token_reserves);

        let sol_cost = amount
            .checked_mul(virtual_sol_reserves)
            .ok_or_else(|| error!(PumpError::Overflow))?
            .checked_div(
                virtual_token_reserves
                    .checked_sub(amount)
                    .ok_or_else(|| error!(PumpError::Overflow))?,
            )
            .ok_or_else(|| error!(PumpError::DivisionByZero))?;

        sol_cost
            .checked_add(1)
            .ok_or_else(|| error!(PumpError::Overflow)) // always round up
    }
```

Recommendation: Implement a more efficient account sizing model that allocates space closer to actual requirements, with a reasonable buffer for future expansion (e.g., 50-100% additional space rather than several hundred percent).

[L-02] Fee vault is made rent-exempt even if no fee shall be collected

The Pump program now collects creator fees that will be sent to the creator vault, from where the creators can withdraw the fees afterward. In order for the fee transfer not to fail, the fee vault account must be rent-exempt. That's why during `buy` and `sell` instructions the vaults are made rent-exempt if the creator is present:

```
if has_creator {
    make_creator_vault_rent_exempt(system_program, user, creator_vault)?;

    // transfer the sol creator_fee to creator_vault
    transfer_sol_from_user(system_program, user, creator_vault, creator_fee)?;
}
```

However, the fee can be zero, and the creator vault will be made rent-exempt anyway, along with the transfer of 0 lamports. This means an additional unnecessary cost to the transaction fee. Although the cost is low (`0.00089088` SOL which is around $0.13 at the time of writing), it is a significant amount in comparison to a regular Solana transaction.

Consider expanding the condition to check if the fee to be transferred is non-zero:

```
-       if has_creator {
+       if has_creator && creator_fee > 0 {
            make_creator_vault_rent_exempt
              (system_program, user, creator_vault)?;

            // transfer the sol creator_fee to creator_vault
            transfer_sol_from_user
              (system_program, user, creator_vault, creator_fee)?;
        }
```