Pump Security Review

Pashov Audit Group

Conducted by: defsec, FrankCastle, shaflow

February 4th 2025 - February 6th 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:

- d21fc4b8ef7a2ab341bf6464c79acc06ac6144e5
- 76d90512e4192c65c40432f144f6c92f644a7a2c

fixes review commit hashes:

- 132e33f5d32f56c1404194ad27d054fdc48c011d

Scope

The following smart contracts were in scope of the audit:

- `common`
- `create_config`
- `disable`
- `update_admin`
- `update_fee_config`
- `create_pool`
- `extend_account`
- `deposit`
- `withdraw`
- `but`
- `mod`
- `sell`
- `lib`
- `global_config`
- `pool`
- `token`
- `extend_account`
- `fee_recipient`
- `migrate`

7. Executive Summary

Over the course of the security review, defsec, FrankCastle, shaflow 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	February 4th 2025 - February 6th 2025
Protocol Type	AMM and Bonding Curve tokensale

Findings Count

Severity	**Amount**
Medium	1
Low	1
Total Findings	**2**

Summary of Findings

ID	**Title**	**Severity**	**Status**
[M-01]	Miss creator check in Create	Medium	Resolved
[L-01]	Invalid pool_migration_fee check	Low	Resolved

8. Findings

8.1. Medium Findings

[M-01] Miss creator check in Create

Severity

Impact: Medium

Likelihood: Medium

Description

When creating a bonding curve in the Create operation, a creator field will be input to set the creator in the metadata.

```
pub fn create(
        ctx: Context<Create>,
        name: String,
        symbol: String,
        uri: String,
        creator: Pubkey,
    ) -> Result<()> {
        ...
        // set the metadata for the token
        helpers::set_metadata(&ctx, name.clone(), symbol.clone(), uri.clone
          (), creator)?;
        ...
    }
```

A malicious pool can set the creator to `Pubkey::default()`. After the curve is completed, the `migrate` operation will attempt to transfer the `creator_fee` to the creator's address. However, this step will fail because the creator is set to an invalid address.

```
let creator = get_creator(&ctx)?;
    if creator.is_some() {
        transfer_from_pool_authority(&ctx, ctx.accounts.creator.to_account_info
          (), creator_fee)?;
    }
```

As a result, the `migrate` operation for this curve will be stuck.

Recommendations

Perform a validity check on the creator.

```
pub fn create(
        ctx: Context<Create>,
        name: String,
        symbol: String,
        uri: String,
        creator: Pubkey,
    ) -> Result<()> {
        ...
        // set the metadata for the token
+       require_keys_neq!(
+          creator,
+          Pubkey::default(),
+          PumpError::CreatorShouldNotBeZero
+       );
        helpers::set_metadata(&ctx, name.clone(), symbol.clone(), uri.clone
          (), creator)?;
        ...
    }
```

8.2. Low Findings

[L-01] Invalid `pool_migration_fee` check

`pool_migration_fee` is used to cover the cost of the `migrate` operation. Therefore, in the `set_params` function, it is necessary to check that the amount of SOL available at the completion of the bonding curve is sufficient to pay for the `pool_migration_fee`.

```
require_gt!(
            initial_virtual_sol_reserves,
            pool_migration_fee,
            PumpError::PoolMigrationFeeShouldBeLessThanInitialVirtualSolReserves
        );
```

The check in `set_params` seems to be invalid because `initial_virtual_sol_reserves` represents the amount of SOL in virtual liquidity, not the actual SOL present in the bonding curve.

Instead, the real SOL amount at the completion of the bonding curve should be estimated based on virtual liquidity and `initial_real_token_reserves` to ensure it is greater than `pool_migration_fee`.