Pump Security Review

Pashov Audit Group

Conducted by: pashov

November 8th, 2023

Contents

1. About pashov

Krum Pashov, or **pashov**, is an independent smart contract security researcher. Having found numerous security vulnerabilities in various protocols, he does his best to contribute to the blockchain ecosystem and its protocols by putting time and effort into security research & reviews. Check his 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 I try to find as many vulnerabilities as possible. I 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** protocol was done by **pashov**, with a focus on the security aspects of the application's smart contracts implementation.

4. About Pump

Pump is an ERC20 token launch platform. Selling and buying tokens can happen through the platform which itself is using an off-chain orderbook and on-chain settlements. Out of a new token's supply 5% go to an admin-controlled address and 95% are vested over a year to the creator.

Observations

Tokens created in the platform don't need allowance to be transferred out of any wallet if the caller of `transferFrom` is the `PumpV1` contract.

Privileged Roles & Actors

- Token Factory - can transfer project-launched tokens out of any wallet without allowance, should be held by the `PumpV1` contract
- Pump Owner - receives fees on each order fulfillment, can update the `marketMaker` address, the status of the liveness of the orderbook, change the `feeRate` and the `vestingDuration`
- Market Maker - receives 5% of the supply of each new token created
- Token Creator - gets vested 95% of the supply of the token over time through the `vest` method, also the token name and symbol are based off of his address
- Order maker - signs an `Order` payload, gets tokens transferred out of on order fulfillment, can cancel an order
- Order recipient - receives tokens on order fulfillment
- Order taker - calls `fulfill` to execute a buy/sell order

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 hash* - [3f6454471a27ccde967e9b73a60006ce8af7267f]**

***fixes review commit hash* - [93753080ad8e9dded5d741413e1de4a1e7105fc7]**

Scope

The following smart contracts were in scope of the audit:

- `Token`
- `PumpV1`

7. Executive Summary

Over the course of the security review, pashov engaged with Pump to review Pump. In this period of time a total of **4** issues were uncovered.

Protocol Summary

Protocol Name	Pump
Date	November 8th, 2023

Findings Count

Severity	**Amount**
Medium	2
Low	2
Total Findings	**4**

Summary of Findings

ID	Title	Severity	Status
[M-01]	Protocol will not be compatible with commonly used ERC20 tokens	Medium	Resolved
[M-02]	Fee can be set to 100% before order fulfillment	Medium	Resolved
[L-01]	No input validation on the vesting duration setter	Low	Resolved
[L-02]	Cancellation can be front-run	Low	Resolved

8. Findings

8.1. Medium Findings

[M-01] Protocol will not be compatible with commonly used ERC20 tokens

Severity

Impact: Medium, as it most probably won't result in value loss but it limits the protocol usability

Likelihood: Medium, as it will not work with some commonly used tokens, but will work with others

Description

Here is a code snippet from `PumpV1::fulfill`:

```
if (order.isBuy) {
    ERC20(order.baseToken).transferFrom
      (order.maker, msg.sender, baseTokenAmount - fee);
    ERC20(order.baseToken).transferFrom(order.maker, owner(), fee);
    Token(order.assetToken).transferFrom(msg.sender, order.recipient, amount);
} else {
    ERC20(order.baseToken).transferFrom
      (msg.sender, order.recipient, baseTokenAmount - fee);
    ERC20(order.baseToken).transferFrom(msg.sender, owner(), fee);
    Token(order.assetToken).transferFrom(order.maker, msg.sender, amount);
}
```

The code is calling the `transferFrom` method of the `ERC20` implementation (`ERC20` imported from the `solady` dependency), which expects a `bool` return value. Tokens like `USDT`, `BNB` and others are missing a return value on `transfer` and `transferFrom`, which would break integration with the application. There are also tokens that do not revert on failure in transfer but return a `false` boolean value like `EURS`. You can read more about such tokens here.

Another issue is that the math to compute `baseTokenAmount` and `fee` in the `fulfill` method of `PumpV1` rely on the `baseToken` decimals to be exactly 18, since the code is using wad-based math when calculating `baseTokenAmount` and `fee`. Many commonly used tokens have lower than 18 decimals (`USDC` and `USDT` have 6 decimals) and some tokens have more than 18 decimals (`YAM-V2` has 24 decimals). While lower decimal tokens transactions can at most revert, the high decimal token transactions can possibly lead to more tokens spent from a user than what he intended.

Recommendations

Use OpenZeppelin's `SafeERC20` library and its `safeTransferFrom` method to handle such tokens. Also scale all token amounts to 18 decimals or forbid using non-18 decimals tokens in the application with an explicit check in `fulfill`.

[M-02] Fee can be set to 100% before order fulfillment

Severity

Impact: High, as it can mean the order recipient will receive nothing in exchange for his tokens

Likelihood: Low, as it requires a malicious or compromised admin

Description

The `setFeeRate` method in `PumpV1` currently has no input validation on the `_feeRate` parameter. If the value given is `1e18` this would set the fee to 100%. An admin can see a call to `fulfill` by monitoring the blockchain's pending transaction pool and front-run it by setting the fee to 100%, essentially stealing all of the tokens that the order recipient should have gotten.

Recommendations

Limit the fee rate to have a maximum value, for example 3%.

8.2. Low Findings

[L-01] No input validation on the vesting duration setter

The `setVestingDuration` method in `PumpV1` has no input validation, meaning it allows the protocol owner to set any value as `vestingDuration`. There are two potential problems with this - using the value of 0, which would lead to DoS of the protocol, because division by zero reverts the transaction, and using a too big of a value which would make the vesting rate of a user to be 0. Implement lower and upper bounds for the `vestingDuration` value.

[L-02] Cancellation can be front-run

The `PumpV1` contract implements a `cancel` method which allows an order maker to cancel his order. The problem is that anyone can see a call to `cancel` and front-run the transaction with a call to `fulfill`, essentially executing the order and not allowing the order maker to cancel it. Consider adding this to your documentation or allowing some other approach for order cancellation.