PunksBids Security Review

Pashov Audit Group

Conducted by: pashov

July 11th, 2023

Contents

1. About pashov 2
2. Disclaimer 2
3. Introduction 2
4. About PunksBids 3
5. Risk Classification 4
   5.1. Impact 4
   5.2. Likelihood 4
   5.3. Action required for severity levels 5
6. Security Assessment Summary 5
7. Executive Summary 6
8. Findings 7
   8.1. Medium Findings 7
      [M-01] Malicious owner could arbitrage sales 7
   8.2. Low Findings 8
      [L-01] The chainId is cached but might change 8
      [L-02] The ecrecover precompile is vulnerable to signature malleability 8

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 **PunksBids** protocol was done by **pashov**, with a focus on the security aspects of the application's smart contracts implementation.

4. About PunksBids

The PunkBids protocol is a bidding platform for the CryptoPunks NFT collection. Anyone can bid on specific attributes or a given set of NFT IDs. The protocol is built on top of the CryptoPunks Marketplace, but it solves an important problem of it - it's non-custodial. While the original Marketplace contract forces bidders to submit their ETH into the contract in a custodial manner, the PunksBids protocol uses off-chain signed bids with which you can bid for multiple punks while also choosing concrete attributes that you'd like.

Observations

The protocol does string manipulation and comparisons on-chain, using a `StringUtils` library. It is used for comparing and checking CryptoPunks attributes. This is in contrast to other protocols built on top of the CryptoPunks Marketplace, who usually use a Merkle tree and proofs to check attributes on-chain.

The matching of bids and sellers is done via a relayer, who will pay for the gas of the sale transaction.

Bidders should give the `PunksBids` contract allowance to spend their `WETH`.

Privileged Roles & Actors

- PunksBids owner - can pause bid matching, change the `feeRate` and `localFeeRate` and withdraw the fees accrued
- Bidder - signs bids off-chain and gives `WETH` allowance to the `PunksBids` contract
- Bid matching relayer - calls `executeMatch` with a signed bid, pays for the gas for the sale

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 **- c783b2aa8d4a9e9efd631e921e2c3b21a2c26f18**

fixes review commit hash **- ba24b1f9e51091341e1775bcd7f5fd6d31892615**

Scope

The following smart contracts were in scope of the audit:

- `interfaces/**`
- `lib/**`
- `PunksBids`

7. Executive Summary

Over the course of the security review, pashov engaged with PunksBids to review PunksBids. In this period of time a total of **3** issues were uncovered.

Protocol Summary

Protocol Name	PunksBids
Date	July 11th, 2023

Findings Count

Severity	**Amount**
Medium	1
Low	2
Total Findings	**3**

Summary of Findings

ID	**Title**	**Severity**	**Status**
[M-01]	Malicious owner could arbitrage sales	Medium	Resolved
[L-01]	The chainId is cached but might change	Low	Resolved
[L-02]	The ecrecover precompile is vulnerable to signature malleability	Low	Resolved

8. Findings

8.1. Medium Findings

[M-01] Malicious owner could arbitrage sales

Severity

Impact: High, as it will charge users more than the should be charged

Likelihood: Low, as it requires a malicious/compromised owner

Description

Currently, the `setFeeRate` and `setLocalFeeRate` methods do not have an upper bound on the fee rate being set by the owner. This opens up a centralization attack vector, where the owner can front-run trades by setting a bigger fee. Consider the following scenario:

1. Alice puts a 100 ETH bid for an Alien Punk, considering fee is 1% and she actually is bidding 99 ETH
2. Bob puts an Alien Punk for sale for 98 ETH
3. Now instead of Alice paying 99 ETH (giving 1 or 0.9 to the protocol as fee) and being left with the punk + 1 ETH, the admin can set the fee to 2% and then execute the trade, essentially taking 1 ETH more from Alice.

Recommendations

Set upper bounds (limits) to both `setFeeRate` and `setLocalFeeRate` methods and revert if the value getting set is higher. This way users will know that fees can maximally go up to a particular number.

Discussion

pashov: Resolved.

8.2. Low Findings

[L-01] The `chainId` is cached but might change

Caching the `chainId` value is not a good practice as hard forks might change the chainId for a network. The better solution is to always check if the current `block.chainid` is the same as the cached one and if not, to update it. Follow the approach in OpenZeppelin's EIP712 implementation.

Discussion

pashov: Acknowledged.

[L-02] The `ecrecover` precompile is vulnerable to signature malleability

By flipping `s` and `v` it is possible to create a different signature that will amount to the same hash & signer. This is fixed in OpenZeppelin's ECDSA library like this. While this is not a problem since there is the `canceledOrFilled` mapping, it is still highly recommended that problem is addressed by using ECDSA.

Discussion

pashov: Resolved.