PunkBid Security Review

Pashov Audit Group

Conducted by: pashov

March 27th, 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 **PunkBid** protocol was done by **pashov**, with a focus on the security aspects of the application's smart contracts implementation.

4. About PunkBid

The **PunkBid** protocol is an on-chain bid-side order book for CryptoPunk NFTs built on top of CryptoPunksMarket. CryptoPunks are currently sold on their native on-chain market but it has some downsides (it is custodial) and security issues (single active bid per punk, accepting a bid can be front-ran with a new bid of `bid.amount + 1 wei`). This protocol solves those problems by allowing multiple users to bid simultaneously on a single Punk as well as allowing a user to bid on multiple Punks at the same time since the protocol has non-custodial bids.

Currently bidders give the smart contract unlimited spending allowance on their `WETH`, while sellers have to call `CryptoPunksMarket::offerPunkForSaleToAddress` just before calling `PunkBidMarketV1::acceptBid`, allowing the `PunkBidMarketV1` contract to buy the Punk for 0 `wei`. Both flows are done so that each operation (bidding and selling) can happen in a non-custodial way.

Entering a new bid accepts a `bytes cartMetadata` argument, which holds data about the Punks you are bidding on (more info here). An off-chain indexer builds a merkle tree based on the `cartMetadata`, where the IDs of each CryptoPunk you'd like to bid on are the leafs in the tree. If the computed root does not match the `itemsChecksum` argument, the bid is discarded on the front-end. Accepting a bid requires a `bytes32[] proof` argument that shows the bid to be accepted was for the Punk that is being sold.

Threat Model

Roles & Actors

- Bidder - enters new bids to buy Punks and approves the protocol to spend its `WETH`
- Seller - is selling his Punks, so accepts bids by allowing the protocol to "spend" his Punk
- Indexer - off-chain component that discards invalid bids and indexes valid ones
- Protocol admin - can withdraw the fees accrued in the protocol

Security Interview

Q: What in the protocol has value in the market?

A: The protocol handles both `WETH` and CryptoPunk NFTs, having allowance to "spend" WETH any time and to "spend" a CryptoPunk just before a seller calls `acceptBid`.

Q: What is the worst thing that can happen to the protocol?

A: A seller accepting the wrong bid or a bidder bidding on the wrong CryptoPunk NFT.

Q: In what case can the protocol/users lose money?

A: The protocol can lose money if an attacker manages to steal the fees contained in the contract, while the users can lose money if the accept the wrong bid or bid on the wrong NFT, or if an attacker exploits the bidders' allowance for `WETH` to the contract or the sellers' allowance for Punk NFT 0 `wei` sale to the contract.

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 **- ff349a602259792b7b7601eaacbb4118b0453a58**

Scope

The following smart contracts were in scope of the audit:

- `PunkBidMarketV1`
- `interfaces/**`

7. Executive Summary

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

Protocol Summary

Protocol Name	PunkBid
Date	March 27th, 2023

Findings Count

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

Summary of Findings

ID	Title	Severity	Status
[L-01]	Protocol expecting a 0 wei Punk sale offer is risky	Low	Resolved
[L-02]	Bid expiration is not constrained	Low	Resolved
[QA-01]	The withdraw method can be permissionless	QA	Resolved
[QA-02]	Change FEE to a constant	QA	Resolved

8. Findings

8.1. Low Findings

[L-01] Protocol expecting a 0 `wei` Punk sale offer is risky

The `acceptBid` function expects that the Punk for sale has a market sell offer for 0 `wei`. In CryptoPunksMarket there are two methods to create a sell offer for your Punk: `offerPunkForSale` and `offerPunkForSaleToAddress`. By using the former your offer can be accepted by anyone on the market and by using the latter your offer can be accepted only by the `toAddress` argument you sent.

It is crucial here that the front-end and the users who interact with the protocol directly on-chain should use the `offerPunkForSaleToAddress` method and the `toAddress` argument should be the address of the `PunkBidMarketV1` contract. If `offerPunkForSale` is used with a 0 `wei` offer then bots will immediately back-run the offer and snipe the Punk. This came to my attention because actually the `ICryptoPunksMarket` interface as well as the unit tests of `PunkBidMarketV1` use the `offerPunkForSale` method which is incorrect. Update the interface and tests to use the `offerPunkForSaleToAddress` method and also make sure this is well documented in the NatSpec of `acceptBid` and for the front-end to use the `offerPunkForSaleToAddress` method.

[L-02] Bid expiration is not constrained

The `enterBid` method is missing input validation on the `expiration` argument so it is possible to enter an already expired bid or a bid that is never expiring. Another thing is that the `updateBids` method allows an already expired bid to be updated. While currently this results only in some used up storage and events emitted, it is recommended to do input sanitization so that only valid bids are accepted in the system. Consider adding lower and upper constraints, for example

```
require
  (expiration > block.timestamp && expiration < block.timestamp + 365 days);
```

8.2. QA Findings

[QA-01] The `withdraw` method can be permissionless

Instead of using the `onlyOwner` modifier on `withdraw`, just replace `msg.sender` with `owner` as the callee and remove the modifier. This will save some gas and remove any access controls in the protocol, making it completely permissionless.

[QA-02] Change `FEE` to a constant

Currently the `FEE` variable is `immutable` even though its value is known before deployment. It is more correct to use a `constant` here and you can also make it a `private` one since its value is not expected to be read on-chain and is easily visible off-chain by looking at the source code.