NFT Capsule Security Review

Pashov Audit Group

Conducted by: pashov

November 20th, 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 **NFT Capsule** protocol was done by **pashov**, with a focus on the security aspects of the application's smart contracts implementation.

4. About NFT Capsule

The protocol allows users to deposit their NFTs (only from the supported collections) and earn real yield (ETH) in an actively-managed strategy. The main goal is solving the NFT illiquid problem and also just making the token holders' experience better overall. The contracts will be deployed on the Ethereum blockchain.

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* - 9f3021c05ad94eb2a2616c63ca23e7f97f82f0c1**

***fixes review commit hash* - 19c9b6dcd3b43fc28d3a9a117edd16a0826ee5c1**

7. Executive Summary

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

Protocol Summary

Protocol Name	NFT Capsule
Date	November 20th, 2023

Findings Count

Severity	**Amount**
Critical	1
High	1
Medium	1
Low	1
Total Findings	**4**

Summary of Findings

ID	Title	Severity	Status
[C-01]	Protocol assets can be stolen through sandwich attacks	Critical	Resolved
[H-01]	Incorrect IPETHNFTVault::borrow assumption breaks the protocol	High	Resolved
[M-01]	User supplying a fake vault can steal protocol assets as a grief attack	Medium	Resolved
[L-01]	Convex extra reward tokens are not handled	Low	Resolved

8. Findings

8.1. Critical Findings

[C-01] Protocol assets can be stolen through sandwich attacks

Severity

Impact: High, as protocol assets will be lost

Likelihood: High, as sandwich attacks are very common and easy to execute

Description

The `Capsule` contract is currently vulnerable to sandwich attack in many places. Every place which does a swap or provides/removes liquidity is done in a manner that is vulnerable. While the code has slippage checks, they are flawed. Take for example the `tradeCRVtoWETH` method, here is how a swap is done:

```
uint256 expectedAmount = curvePoolWETHCRV.get_dy
//(2, 1, crvBal); // Estimate WETH received for CRV:WETH exchange
// Exchange CRV for WETH within the slippage limit
curvePoolWETHCRV.exchange
  (2, 1, crvBal, expectedAmount * LPslippageNum / LPslippageDen);
```

The problem is that if an attacker sees you calling `tradeCRVtoWETH` he can imbalance the pool with a very big front-run transaction, which will give you a much smaller `expectedAmount`, and then after you do the swap he will execute a back-run transaction putting the price back to normal, essentially sandwiching your bad trade and profiting your loss. This is a problem in all methods that have an underlying call to `add_liquidity`, `remove_liquidity_one_coin`, `calc_withdraw_one_coin` and `calc_token_amount`.

Recommendations

Instead of doing on-chain same transaction calculations to calculate slippage tolerance, add an argument to all methods that use slippage calculations called "minAmountReceived" - it has to be calculated off-chain so it is not prone to on-chain manipulation.

8.2. High Findings

[H-01] Incorrect `IPETHNFTVault::borrow` assumption breaks the protocol

Severity

Impact: Medium, as the protocol will have to be redeployed

Likelihood: High, as it it certain to happen

Description

The `depositNft` method in `Capsule` calls `IPETHNFTVault::borrow` with a `_borrowAmount` argument. Later, the code actually tries using the `_borrowAmount` value as the amount of `PETH` to provide as liquidity to a Curve pool. The problem is that the `borrow` method of those vaults always takes a fee, so `Capsule` will have received less than `_borrowAmount` of `PETH`. Quoted from `IPETHNFTVault::borrow`'s NatSpec:

> /// @param _amount The amount of PUSD to be borrowed. Note that the user will receive less than the amount requested, /// the borrow fee and insurance automatically get removed from the amount borrowed

This means that the liquidity provision will always fail due to insufficient `PETH` balance, making the protocol unusable.

Even if the fee value is currently zero it can be changed and the protocol will be broken.

Recommendations

Use only the `PETH` received from borrowing for providing liquidity as well as for the `newPosition.amountBorrowed` value in `depositNft`. The issue is also present in `increaseBorrowAmount` and should be addressed there as well.

8.3. Medium Findings

[M-01] User supplying a fake vault can steal protocol assets as a grief attack

Severity

Impact: High, as a malicious actor can drain protocol assets

Likelihood: Low, as it is a common attack vector and it requires no preconditions

Description

The `_vault` argument in the `repay` method (and in all others) has no input validation, meaning a user can supply a fake vault which he himself created. By calling `repay` with a fake `_vault` argument for an NFT borrowing position he owns, when `amountBorrowed > repayment` he will go through this code:

```
PETH.approve(address(_vault), tokensWithdrawn); // Approve JPEGD to take PETH
_vault.repay
//(_id, tokensWithdrawn); // Repay desired amount of loan based on desired repayment s
ePosition.lpSize -= LPcostToRepay; // No need to adjust profit index here,
// previous check requires they either withdraw or forfeit profits beforehande the use
ePosition.amountBorrowed -= tokensWithdrawn; // Decrease the users amount
// borrowed
emit DecreaseBorrowAmount(owner, _nft, _id, tokensWithdrawn);
```

this would send the protocol's `PETH` to the user's fake vault, which would be a steal of protocol assets, even though his NFT wouldn't be withdrawable anymore.

Recommendations

Use a whitelisting approach to the `_vault` argument in all methods of the contract so that users can't provide fake vaults to them.

8.4. Low Findings

[L-01] Convex extra reward tokens are not handled

The `claimCVXReward` method makes a call to `getReward` from the Convex reward pool contract. The `getReward` method actually can possibly send other token rewards to the caller as well, but those are not handled in the `Capsule` contract as only `CRV` rewards are expected. Make sure to have a way to handle different token rewards to not miss on potential yield.