Radiant Security Review

Pashov Audit Group

Conducted by: yttriumzz, Dan Ogurtsov, ubermensch

July 25th 2024 - July 26th 2024

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 **radiant-capital/v2-core** repository was done by **Pashov Audit Group**, with a focus on the security aspects of the application's smart contracts implementation.

4. About Radiant

UniV3TokenizedLp manages a tokenized liquidity position in a Uniswap V3-like pool. It allows users to deposit in exchange for ERC20 tokens that represent their share of the liquidity pool. The contract includes mechanisms for rebalancing liquidity positions and ensuring the accuracy of external oracle price feeds to prevent price manipulation.

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 - bb1680bea467e1032d0431ddc66ccc96cbd8fa33

fixes review commit hash - 2c30b8f76493ae9b6da993e384e5edf21b5d25e6

Scope

The following smart contracts were in scope of the audit:

- `UniV3PoolHelper`
- `UniV3TokenizedLP`

7. Executive Summary

Over the course of the security review, yttriumzz, Dan Ogurtsov, ubermensch engaged with Radiant to review Radiant. In this period of time a total of **4** issues were uncovered.

Protocol Summary

Protocol Name	Radiant
Repository	https://github.com/radiant-capital/v2-core
Date	July 25th 2024 - July 26th 2024
Protocol Type	Omnichain money market

Findings Count

Severity	**Amount**
High	1
Medium	3
Total Findings	**4**

Summary of Findings

ID	Title	Severity	Status
[H-01]	spotTimeWeightedPrice and withSwapping may conflict	High	Resolved
[M-01]	deposit may not mint liquidity	Medium	Resolved
[M-02]	swapIdleAndAddToLiquidity may be DoSed	Medium	Resolved
[M-03]	No price limit in zapWETH	Medium	Resolved

8. Findings

8.1. High Findings

[H-01] `spotTimeWeightedPrice` and `withSwapping` may conflict

Severity

Impact: High

Likelihood: Medium

Description

The `autoRebalance` function has two input parameters.

- `useOracleForNewBounds`: If `false`, use `spotTimeWeightedPrice` to determine the new `baseLower` and `baseUpper`
- `withSwapping`: If `true`, when the difference between spot and oracle prices is too large, some tokens will be swapped to make the spot price closer to the oracle price.

The following scenario occurs when `useOracleForNewBounds` is `false` and `withSwapping` is `true` and the price difference between spot and oracle is too large. There is a discrepancy between the spot price after the swap and the `spotTimeWeightedPrice` obtained before the swap. The `baseLower` and `baseUppe` are determined by the price before the swap. This will lead to an unreasonable final liquidity range.

In extreme cases, this may result in mint failure or uncompensated losses due to the use of unreasonable liquidity ranges.

Recommendations

Make sure the swap limit price is the same as `priceRefForBounds`.

8.2. Medium Findings

[M-01] `deposit` may not mint liquidity

Severity

Impact: Medium

Likelihood: Medium

Description

The `deposit` function will mint liquidity only if bounds are defined. However, the method to determine whether bounds are defined is wrong.

When `baseLower` and `baseUpper` are both `0`, the bounds are undefined. However, the `deposit` function does not mint liquidity as long as one of `baseLower` and `baseUpper` is `0`. For uniswap pool v3, if the price is `1`, its tick is `0`. Therefore, this results in the `deposit` function not minting liquidity when bounds are already defined.

Recommendations

If one of `baseLower` and `baseUpper` is not `0`, mint liquidity.

[M-02] `swapIdleAndAddToLiquidity` may be DoSed

Severity

Impact: Medium

Likelihood: Medium

Description

The `swapIdleAndAddToLiquidity` function allows the rebalancer to input `swapQuantity` to swap a certain amount of tokens. Assume that the rebalancer uses the balance as `swapQuantity`, that is, he wants to swap all tokens. The user can withdraw some tokens in advance, which will cause the contract balance to be less than `swapQuantity`, and then the swap will fail.

Recommendations

Take the smaller value of `swapQuantity` and balance as the swap input.

[M-03] No price limit in `zapWETH`

Severity

Impact: Medium

Likelihood: Medium

Description

The `zapWETH` function deposits some WETH to `UniV3TokenizedLp`. If there is too much WETH in `UniV3TokenizedLp`, some WETH will be swapped without a price limit. Furthermore, its caller, `LockZap` contract, does not check slippage. `LockZap` only checks the number of LPs received from the deposit. However, this problem results in a smaller value for LPs rather than a smaller number.

Recommendations

Add slippage check.