MixBytes()

PICKLE.FINANCE

SMART CONTRACT

AUDIT REPORT

OCTOBER 09
2020

FOREWORD TO REPORT

A small bug can cost you millions. **MixBytes** is a team of experienced blockchain engineers that reviews your codebase and helps you avoid potential heavy losses. More than 10 years of expertise in information security and high-load services and 18 000+ lines of audited code speak for themselves. This document outlines our methodology, scope of work, and results. We would like to thank **Pickle.Finance** for their trust and opportunity to audit their smart contracts.

CONTENT DISCLAIMER

TABLE OF CONTENTS

01 INTRODUCTION TO THE AUDIT

GENERAL PROVISIONS

Pickle.Finance is an experiment in using farming to bring stablecoins closer to their peg.

SCOPE OF THE AUDIT

The scope of the audit includes following smart contract at:
https://github.com/pickle-finance/protocol/blob/master/src/strategies/curve/strategy-curve-scrv-v4_1.sol

The audited commit identifier is: **8d2a96ced740cb5dda4381e70d476760ce4b13e1**

02 SECURITY ASSESSMENT PRINCIPLES

CLASSIFICATION OF ISSUES

CRITICAL

Bugs leading to Ether or token theft, fund access locking or any other loss of Ether/tokens to be transferred to any party (for example, dividends).

MAJOR

Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.

WARNINGS

Bugs that can break the intended contract logic or expose it to DoS attacks.

COMMENTS

Other issues and recommendations reported to/acknowledged by the team.

SECURITY ASSESSMENT METHODOLOGY

Two auditors independently verified the code.

Stages of the audit were as follows:

1. “Blind” manual check of the code and its model
2. “Guided” manual code review
3. Checking the code compliance to customer requirements
4. Discussion of independent audit results
5. Report preparation

03 DETECTED ISSUES

CRITICAL

Not found.

MAJOR

Not found.

WARNINGS

1. Harvesting Potential Front-Run

This particular issue is about `harvest()` function having no limitations on caller identifier, not having “commit and reveal” scheme usage or any other mechanism preventing front-running in here: **https://github.com/pickle-finance/protocol/blob/master/src/strategies/curve/strategy-curve-scrv-v4_1.sol#L153**

Calling this particular function wrongly (implementing a displacement front-running pattern) brings risks Alice would not being able to rely on transaction execution result in case Bob front-runs harvesting transaction with higher gas fee bid.

This issue does not break the state, but it breaks the logic contract works from Alice’s point of view.

This particular issue was classified as a warning because the author is clearly aware about it, from what we see from the comments.

Also, since there is no way to benefit from front-running this particular function call, this issue makes no significant damage.

Since this function is a public function, the proof of concept would be a node mempool listener, which would transact with enormous gas amount right after gets the transaction spotted.

Update: The issue was resolved with restricting the calling of this function to only EOAs and the strategist or governance addresses along with `onlyBenevolent` modifier being appended to the `harvest` function signature **here**. The `onlyBenevolent` modifier is defined **here**.

2. Potentially Null Swap Destination Address

This particular issue is about `_swapUniswap` function not being used carefully having a possibility to swap tokens with `UniswapRouterV2` to `address(0)` destination.

In here:
https://github.com/pickle-finance/protocol/blob/a808366db95b07c0b8940e919ec72d80e2deaca7/src/strategies/strategy-base.sol#L214

Since this function is an internal one and the actual destination address is still being checked later, this is not a major issue, but just a warning.

It is recommended to append `require(_to != address(0))` requirement right after the function execution starts.

Update: This issue was resolved with the following PR:
https://github.com/pickle-finance/protocol/pull/5.

3. Non-Null Addresses

https://github.com/pickle-finance/protocol/blob/master/src/strategies/curve/strategy-curve-scrv-v4_1.sol#L49
https://github.com/pickle-finance/protocol/blob/a808366db95b07c0b8940e919ec72d80e2deaca7/src/strategies/strategy-base.sol#L44

Since all the operations with `address`-typed `StrategyCurveSCRVv4_1` and `StrategyBase` contract fields are being performed with `safeTransfer` or `_approve` functions, checking if an `address`-typed argument is not `0`, it is recommended to implement additional `require`s in contract constructor, checking if each of input arguments `!= address(0)`, to avoid potential issues in newcoming contract updates (according to `harvest()` functions comment).

Update: This issue was resolved with the following PR:
https://github.com/pickle-finance/protocol/pull/7.

COMMENTS

1. Redundant memory allocation for asset index type

`getMostPremium()` function (in here: **https://github.com/pickle-finance/protocol/blob/master/src/strategies/curve/strategy-curve-scrv-v4_1.sol#L81**)

returns tuple `(address, uint256)`, which contains unnecessarily huge asset index type `uint256`. Since the function structure would require to update the contract anyway in case new stablecoins would be added, there is no need in reserving index type sized for enormously huge amount of assets.

Storing asset index in a type like `uint8` would be more appropriate for now.

Update: This issue was resolved with the following PR: **https://github.com/pickle-finance/protocol/pull/6**.

04 CONCLUSION AND RESULTS

The smart contract was audited and no critical or major issues were found.

Two suspicious places were spotted (marked as a warning), but, as it has been understood, the developer is already acknowledged about one of them. According to the implementation mostly consists of well-reviewed OpenZeppelin libraries (or modules implemented with following OpenZeppelin's manuals step by step) along with Uniswap interface modules, using extensive overflow prevention techniques with OpenZeppelin's `SafeMath` library, checking destination addresses along with message senders all the way, using reentrancy-safe implementation techniques, the contract itself is mostly safe to use.

ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build open-source solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, consult universities and enterprises, do research, publish articles and documentation.

Stack	Blockchains

JOIN US