YEARN FINANCE YORACLE.LINK SMART CONTRACT AUDIT

December 28, 2020

yearn.finance

MixBytes()

CONTENTS

1. INTRODUCTION

1.1 DISCLAIMER

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Yearn Finance (name of Client). If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 PROJECT OVERVIEW

Yearn Finance is a decentralized investment aggregator that leverages composability and uses automated strategies to earn high yield on crypto assets.

1.3 SECURITY ASSESSMENT METHODOLOGY

At least 2 auditors are involved in the work on the audit who check the provided source code independently of each other in accordance with the methodology described below:

01 "Blind" audit includes:

> Manual code study

> "Reverse" research and study of the architecture of the code based on the source code only

Stage goal:
Building an independent view of the project's architecture
Finding logical flaws

02 Checking the code against the checklist of known vulnerabilities includes:

> Manual code check for vulnerabilities from the company's internal checklist

> The company's checklist is constantly updated based on the analysis of hacks, research and audit of the clients' code

Stage goal:
Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flashloan attacks, etc.)

03 Checking the logic, architecture of the security model for compliance with the desired model, which includes:

> Detailed study of the project documentation

> Examining contracts tests

> Examining comments in code

> Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of the reports from all auditors into one common interim report document

> Cross check: each auditor reviews the reports of the others

> Discussion of the found issues by the auditors

> Formation of a general (merged) report

Stage goal:
Re-check all the problems for relevance and correctness of the threat level
Provide the client with an interim report

05 Bug fixing & re-check.

> Client fixes or comments on every issue

> Upon completion of the bug fixing, the auditors double-check each fix and set the statuses with a link to the fix

Stage goal:
Preparation of the final code version with all the fixes

06 Preparation of the final audit report and delivery to the customer.

Findings discovered during the audit are classified as follows:

FINDINGS SEVERITY BREAKDOWN

Level	Description	Required action
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party	Immediate action to fix issue
Major	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.	Implement fix as soon as possible
Warning	Bugs that can break the intended contract logic or expose it to DoS attacks	Take into consideration and implement fix in certain period
Comment	Other issues and recommendations reported to/acknowledged by the team	Take into consideration

Based on the feedback received from the Customer's team regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The project team is aware of this finding. Recommendations for this finding are planned to be resolved in the future. This finding does not affect the overall safety of the project.
No issue	Finding does not affect the overall safety of the project and does not violate the logic of its work.

1.4 EXECUTIVE SUMMARY

The volume checked includes 2 smart contracts that are part of the oracle on-chain mechanism for UniswapV2 pairs. The project also uses other smart contracts. But the smart contracts we tested are among the main ones.

1.5 PROJECT DASHBOARD

Client	Yearn Finance
Audit name	Yoracle.Link
Initial version	faf1309cbe7a05f70b338351315039eb8e5b9c09
Final version	22a438fcce04ea08be383e1a3e757b49af765ed4
SLOC	702
Date	2020-11-19 - 2020-12-28
Auditors engaged	2 auditors

FILES LISTING

Keep3rV1Oracle.sol	Keep3rV1Oracle.sol
Keep3rV1Volatility.sol	Keep3rV1Volatility.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	0
Warning	8
Comment	8

CONCLUSION

Smart contracts were audited and several suspicious places were spotted. During the audit no critical and major issues were found, eight issues were marked as warnings and eight comments were found and discussed with the client. After working on the reported findings all of them were resolved or acknowledged. So, the contracts are assumed as secure to use according to our security criteria.

2.FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

Not Found

2.3 WARNING

WRN-1	Safe math library isn't used
File	Keep3rV1Oracle.sol Keep3rV1Volatility.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the lines 101, 108, 116, 151, 154, 156, 377, 387, 396, 621, 627, 641, 664, 667, 672, 675, 691, 696, 697, 701, 705, 706, 710, 770, 772 , 776, 778, 796 in Keep3rV1Oracle.sol and at the lines 33, 40, 53, 55, 61, 64, 69, 90, 92, 94, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124 , 126, 128, 131, 133, 135, 137, 139, 141, 143, 189, 196, 209, 211, 217, 220, 225, 246-284, 287, 289, 291, 293, 295, 297, 299 , 320, 323, 330-346, 350-354, 363, 365, 371, 374, 379, 393-398, 435, 438, 445, 450-461, 465-469, 478, 480, 486, 489, 494 , 508-513 in Keep3rV1Volatility.sol if you do not use a library for safe math, then an arithmetic overflow may occur, which will lead to incorrect operation of smart contracts.

RECOMMENDATION

All arithmetic operations need to be redone using the safe math library. Moreover, this library is already in the contracts Keep3rV1Oracle.sol#L176.

WRN-2	The number of loop iterations should be limited
File	Keep3rV1Oracle.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

In Keep3rV1Oracle.sol there are internal arrays `_pairs` for addresses and an `observations` map for structures `Observation[]`. The number of elements of these arrays in the logic of smart contracts only increases, but there is no functionality to decrease. Theoretically, the number of their elements can be very large, since it is not limited anywhere in the program.
In loops, the number of elements of this array is used as the upper bound on the number of iterations. Any iteration in the loop uses gas. But the amount of gas in one block is limited. It means that a situation may arise when there is not enough gas to perform all the iterations of the cycle and the function will stop working.
There are such loops here:
line 575, 595, 662, 670, 696, 705,

RECOMMENDATION

It is necessary in loops to introduce a limit on the number of iterations or on the possible number of array elements in loops.

WRN-3	Possible incorrect operation of the function
File	Keep3rV1Oracle.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the line Keep3rV1Oracle.sol#L626 the result of the `_valid` function depends on the value of `block.timestamp`. But the `block.timestamp` value is set by the miner. This may result in an incorrect result. According to the specification Block-Protocol-2.0.md, the miner can shift `block.timestamp` up to 900 seconds. If the range is greater, then you are safe.

RECOMMENDATION

We recommend adding a restriction on the value of variable `age`. For the same reason, we do not recommend setting the value of the `periodSize` constant less than 900.

WRN-4	It is possible to go beyond the boundaries of the array
File	Keep3rV1Oracle.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the line Keep3rV1Oracle.sol#L685 the `sample` function does not check for a valid `observations[pair]` array element number.
On lines 700, 701, 709, 710 there is a call to the array element with the `nextIndex` index. But there is nowhere a check that the `nextIndex` value is less than and equal to the value of the variable `length`.

RECOMMENDATION

We recommend to add such a check.

WRN-5	There is no check of the ether balance on the contract before it is transferred
File	Keep3rV1Oracle.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the line Keep3rV1Oracle.sol#L479 before transferring ether from the contract, you need to check that its amount is greater than 0.

RECOMMENDATION

Add checking the ether balance on the contract before transferring it.

WRN-6	Function calculation result is not processed
File	Keep3rV1Oracle.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

Below there will be a case when the code does not process the result of calling `approve`.

Keep3rV1Oracle.sol#L819

`approve` method may return `false`.

RECOMMENDATION

Check the return value of the function.

WRN-7	Potential `Out of range` error
File	Keep3rV1Oracle.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

Code line Keep3rV1Oracle.sol#L638 may return an exception because `observations[pair]` cannot guarantee the length more than 1.

RECOMMENDATION

It is recommended to check it.

WRN-8	Contract cannot be compiled
File	Keep3rV1Volatility.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

The contact `Keep3rV1Volatility.sol` has the following syntax errors:
Keep3rV1Volatility.sol#L157
Keep3rV1Volatility.sol#L428

RECOMMENDATION

It is recommended to fix the errors above.

2.4 COMMENTS

CMT-1	We recommend changing the scope of functions
File	Keep3rV1Oracle.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

Changing the scope of functions will allow making transactions with a lower cost of gas.

RECOMMENDATION

Keep3rV1Oracle.sol
On line 560, the `work` function can be changed from `public` to `external`.
On line 565, the `workForFree` function can be changed from `public` to `external`.
On line 574, function `_updateAll` can be changed from `internal` to `private`.
On line 603, the `_update` function can be changed from `internal` to `private`.
On line 626, the `_valid` function can be scoped from `"internal` to `private`.
On line 807, function `retBasedBlackScholesEstimate` can be changed from `public` to `external`.
On line 818, the `_swap` function can be changed from `internal` to `private`.

Keep3rV1Volatility.sol
On line 26, the `floorLog2` function can be changed from `internal` to `private`.
On line 48, the `ln` function can be changed from `internal` to `private`.
On line 83, for the `optimalExp` function, the scope can be changed from `internal` to `private`.

These steps will save you some gas cost for transactions.

CMT-2	We recommend moving the functions to a separate library
File	Keep3rV1Oracle.sol Keep3rV1Volatility.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

In smart contract Keep3rV1Oracle.sol#L751 there is the `sqrt` function and in smart contracts
Keep3rV1Volatility.sol#L382 and Keep3rV1Volatility.sol#L497 this function is again described.

RECOMMENDATION

We recommend that you describe it once in the library file.
We also recommend transferring the following functions to this library:
`floorLog2` located here Keep3rV1Volatility.sol#L182
`ln` located here Keep3rV1Volatility.sol#L204
`optimalExp` located here Keep3rV1Volatility.sol#L239
`C` located here Keep3rV1Volatility.sol#L328
The `C` function can be named more meaningfully and clearly.

It is possible that you can take other functions. But the main thing is that the mathematical functions are located in a separate library.
Then the source code will be easier to test and understand.

CMT-3	Duplicate code
File	Keep3rV1Volatility.sol Keep3rV1Oracle.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

The `quote` function is completely repeated here Keep3rV1Volatility.sol#L148 and here Keep3rV1Volatility.sol#L304
It needs to be described only once.

The `IERC20` interface is completely repeated here Keep3rV1Volatility.sol#L10 and here Keep3rV1Volatility.sol#L166
It needs to be described only once.

Keep3rV1Oracle.sol#L486
On lines 486, 495, 560 the code is repeated:
`require (msg.sender == governance," setGovernance:! Gov ");`
It is necessary to transfer this code to a separate access modifier.

Keep3rV1Oracle.sol#L468
On lines 468 and 474 the code is repeated:
`require (KP3R.isMinKeeper (msg.sender, minKeep, 0, 0)," :: isKeeper: keeper is not registered ");`
It is necessary to transfer this code to a separate access modifier.

Keep3rV1Oracle.sol#L562
On lines 562 and 567 the code is repeated:
`require (worked," UniswapV2Oracle:! Work ");`
It is necessary to transfer this code to a separate access modifier.

Keep3rV1Oracle.sol#L632
On lines 632 and 652 the code is repeated:
`require (_valid (pair, periodSize.mul (2))," UniswapV2Oracle :: quote: stale prices ");`
It is necessary to transfer this code to a separate access modifier.

RECOMMENDATION

Duplicate code is a very bad programming practice and is against the principles of SOLID (single responsibility, open-closed, Liskov substitution, interface segregation и dependency inversion). We recommend refactoring the source code.

CMT-4	We recommend removing additional functionality from the access modifier
File	Keep3rV1Oracle.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the line Keep3rV1Oracle.sol#L472 the `upkeep` access modifier has additional functionality designed to calculate the amount of ETH and send it from the contract.
In accordance with the principles of SOLID (single responsibility, open-closed, Liskov substitution, interface segregation и dependency inversion) software development, each module is responsible for only one thing. An access modifier should only be responsible for access.

RECOMMENDATION

It is necessary to transfer all additional functionality from this modifier to a separate function.

CMT-5	We recommend caching the variable
File	Keep3rV1Oracle.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the line Keep3rV1Oracle.sol#L411 in the `getAmountsIn` function, the length of the `path.length` array is calculated 3 times: on lines 412, 413, 415.

RECOMMENDATION

We recommend storing the value of the array length in a separate variable and referring to this variable.
This will slightly reduce the gas consumption.

CMT-6	Mixed formatting
File	Keep3rV1Volatility.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

`Tabs` are used instead of `Spaces`:
Keep3rV1Volatility.sol#L443

RECOMMENDATION

Make corrections to the source code.

CMT-7	Potential `Out of range` error
File	Keep3rV1Oracle.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the lines Keep3rV1Oracle.sol#L656, Keep3rV1Oracle.sol#L690 in `observations[pair]` there is always more than 1 element (Keep3rV1Oracle.sol#L557).

RECOMMENDATION

However, it is recommended to check it.

CMT-8	Lack of event
File	Keep3rV1Oracle.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

Keep3rV1Oracle.sol#L479
Logging events while a smart contract is running is a good practice.

RECOMMENDATION

It is recommended to emit an event when ethers are transferred.

3.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, do research and tech consultancy.

BLOCKCHAINS

- Ethereum
- Cosmos
- EOS
- Substrate

TECH STACK

- Python
- Solidity
- Rust
- C++

CONTACTS

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://t.me/MixBytes

https://twitter.com/mixbytes