YEARN VAULTS V.3 SMART CONTRACT AUDIT

April 09, 2021

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. 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 Vaults contracts are used to create a simple way to generate high risk-adjusted returns for depositors of various assets via best- in-class lending protocols, liquidity pools, and community-made yield farming strategies on Ethereum.

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 contracts studied during this audit are designed to collect data about the Vaults for tokens in one place. All Vaults are collected in the Register smart contract. Also, the logic has been developed for the case if several Volts will be created for the same tokens. With ERC-20's operations, values are processed in all Vaults for this token at once.

1.5 PROJECT DASHBOARD

Client	Yearn
Audit name	Vaults v.3
Initial version	e390c2a6b2ba6e2ecc8f3a72a1ea4adfabd17544 faaefd55548a25ab9ef23e98cc09f9ebeb54428a
Final version	faaefd55548a25ab9ef23e98cc09f9ebeb54428a
SLOC	464
Date	2021-03-24 - 2021-04-09
Auditors engaged	2 auditors

FILES LISTING

BaseWrapper.sol	BaseWrapper.sol
Registry.vy	Registry.vy
yToken.sol	yToken.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	2
Warning	5
Comment	3

CONCLUSION

Smart contracts have been audited and several suspicious places have been spotted. During the audit no critical issues were found, two issues were marked as major because they could lead to some undesired behavior, also several warnings and comments were found and fixed by the client. After working on the reported findings all of them were resolved or acknowledged (if the problem was not critical). So, the contracts are assumed as secure to use according to our security criteria.Final commit identifier with all fixes: `faaefd55548a25ab9ef23e98cc09f9ebeb54428a`

2. FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

MJR-1	Incorrect use of a library function
File	BaseWrapper.sol
Severity	Major
Status	Fixed at faaefd55

DESCRIPTION

At the lines

- BaseWrapper.sol#L117
- BaseWrapper.sol#L198
  use the `safeApprove()` function from the `SafeERC20` library.
  But before giving permission for a certain amount of tokens, you first need to zero this value.
  Now the `safeApprove()` function is called only once with a specific value.
  After the first call, everything will work, but after the second call, the function will be blocked.

RECOMMENDATION

It is necessary to make a correct call to the `safeApprove()` function:

```
token.safeApprove(addressValue, 0);
token.safeApprove(addressValue, amount);
```

MJR-2	Outdated `_cachedVaults` after `registry` changing
File	BaseWrapper.sol
Severity	Major
Status	Acknowledged

DESCRIPTION

Each `registry` has own `vaults` variable. It is possible that new `registry` contains different set of Vaults. In `allVaults` function we always copy old `_cachedVaults` BaseWrapper.sol#L71-73.

The cache is updated with `allVaults` produce BaseWrapper.sol#L148-149.

So we will never update `_cachedVaults` prefix part.

RECOMMENDATION

It is recommended to update all elements of the `_cachedVaults` array in the `setRegistry()` function.

CLIENT'S COMMENTARY

Even though `registry` is update-able by Yearn, the intended behavior is that any future upgrades to the registry will replay the version history so that this cached value does not get out of date.

2.3 WARNING

WRN-1	Gas overflow during iteration (DoS)
File	yToken.sol BaseWrapper.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

Each iteration of the cycle requires a gas flow.
A moment may come when more gas is required than it is allocated to record one block. In this case, all iterations of the loop will fail.

At the lines
yToken.sol#L107
yToken.sol#L154
BaseWrapper.sol#L151

for the `vaults` address array is not checked for its length. In iterations, data is written to the blockchain, which requires gas consumption. The contract has logic for adding new elements to the `vaults` array, but no logic for removing them.

RECOMMENDATION

It is recommended adding a check for the maximum possible number of elements of the arrays.

CLIENT'S COMMENTARY

This loop will attempt to withdraw from each Vault in `allVaults` that `sender` is deposited in, up to `amount` tokens. The withdraw action can be >expensive, so it if there is a denial of service issue in withdrawing, the downstream usage of this wrapper contract must give an alternative method of >withdrawing using this function so that `amount` is less than the full amount requested to withdraw (e.g. "piece-wise withdrawals"), leading to less loop >iterations such that the DoS issue is mitigated (at a tradeoff of requiring more txns from the end user).

WRN-2	No validation of the address parameter value in contract constructor
File	BaseWrapper.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

The variable is assigned the value of the constructor input parameter. But this parameter is not checked before this. If the value turns out to be zero, then it will be necessary redeploy the contract, since there is no other functionality to set this variable.

- At the line BaseWrapper.sol#L45 the `token` variable is set to the value of the `_token` input parameter.

RECOMMENDATION

It is necessary to add a check of the input parameter to zero before initializing the variables.

WRN-3	No response check after function call to transfer tokens
File	BaseWrapper.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

According to the ERC-20 standard, the token transfer functions return a boolean variable.
It is necessary to process the function response after the token transfer.
But now there is no such functionality on the following lines:

- BaseWrapper.sol#L137
- BaseWrapper.sol#L171

RECOMMENDATION

It is recommended to add a return value check after the token transfer.

WRN-4	Check for a zero input value
File	BaseWrapper.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the line BaseWrapper.sol#L50 there is no check for a zero input value in method `setRegistry()`.

RECOMMENDATION

It is necessary to add a check of the input parameter to zero before initializing the variables.

WRN-5	An attack like `Reentrancy` is possible
File	yToken.sol
Severity	Warning
Status	Fixed at **faaefd55**

DESCRIPTION

At the line yToken.sol#L186
the sending of the entire ETH that is on the balance sheet of the contract is made. But the sender can be a smart contract in which the function for receiving ETH calls the function `withdrawETH()` in the contract.
In this case, the behavior of the contract will differ from what is expected.

RECOMMENDATION

When calling the `withdrawETH()` function, it is recommended to use the `nonReentrant` access modifier from the contract: ReentrancyGuard.sol.

2.4 COMMENTS

CMT-1	Event is probably missing
File	BaseWrapper.sol Registry.vy yToken.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the line BaseWrapper.sol#L50 in method `setRegistry()` should probably emit an event `SetRegistry`.
At the line BaseWrapper.sol#L104 in method `_deposit()` should probably emit an event `Deposit`.
At the line BaseWrapper.sol#L140 in method `_withdraw()` should probably emit an event `Withdraw`.
At the line BaseWrapper.sol#L218 in method `_migrate()` should probably emit an event `Migrate`.

At the line Registry.vy#L304 in method `setBanksy()` should probably emit an event `SetBanksy`.

At the line yToken.sol#L171 in method `depositETH()` should probably emit an event `DepositETH`.
At the line yToken.sol#L180 in method `withdrawETH` should probably emit an event `WithdrawETH`.

RECOMMENDATION

It is recommended to create new events.

CMT-2	The name of the function does not correspond to the logic of work
File	Registry.vy
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the line Registry.vy#L316 in method `setBanksy()` should probably emit an event `SetBanksy`.
Line 316 has a function called `tagVault()`. This is the name for the function for reading the value of a variable.
But in this function, a new value of the variable is being set.
The name of the function does not correspond to the logic of work.

RECOMMENDATION

It is recommended to name the function `setTagVault()`.

CMT-3	Dublicate code
File	BaseWrapper.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the line BaseWrapper.sol#L57 in method `setRegistry()`
the same code `require(msg.sender == registry.governance());` is repeated twice.

RECOMMENDATION

Remove this row.

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