

# YEARN VAULT V2 SMART CONTRACT AUDIT

(Vyper part)



December 2, 2020

**MixBytes()**

# TABLE OF CONTENTS

INTRODUCTION TO THE AUDIT.....	2
General provisions .....	2
Scope of audit .....	2
SECURITY ASSESSMENT PRINCIPLES .....	3
Classification of issues .....	3
Security assessment methodology .....	3
DETECTED ISSUES .....	4
Critical .....	4
Major .....	4
1. Potential withdrawal lock .....	4
Warnings .....	5
1. Code commentary doesn't comply with real implementation .....	5
2. Potential issue with re-entrancy .....	5
Comments .....	6
1. Adding <code>Approval</code> event in <code>transferFrom</code> .....	6
2. Optimize deposit amount check.....	6
3. Typo in commentary.....	7
4. Implicit loss calculation .....	7
5. Unoptimized <code>withdrawalQueue</code> updating.....	8
6. Strategy can report loss and gain at the same time.....	8
CONCLUSION AND RESULTS .....	9
ABOUT MIXBYTES .....	9
DISCLAIMER .....	10

# 01 | INTRODUCTION TO THE AUDIT

## General Provisions

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

The audited contract is a part of a new second version of Yearn vaults. Yearn vaults represent a user funds manager in Yearn ecosystem.

Smart contract provides an entry point for a user to deposit and withdraw funds and under the hood operates with linked strategies.

The code is written using new Vyper language that improved readability of the code and allowed auditors to consider mostly only on business logic.

## Scope of the Audit

The scope of the audit includes the following smart contracts at:

<https://github.com/iearn-finance/yearn-vaults/blob/054034304c7912d227d460feadc23177103de0b9/contracts/Vault.vy>

The audited commit identifiers are:

- `054034304c7912d227d460feadc23177103de0b9` (tag v0.2.0)

# 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:

- "Blind" manual check of the code and its model
- "Guided" manual code review
- Checking the code compliance with the customer requirements
- Discussion of independent audit results
- Report preparation

# 03 | DETECTED ISSUES

## CRITICAL

Not found

## MAJOR

### 1. Potential withdrawal lock

#### Description

At this line: <https://github.com/iearn-finance/yearn-vaults/blob/054034304c7912d227d460feadc23177103de0b9/contracts/Vault.vy#L787>

```
amountNeeded: uint256 = value - self.token.balanceOf(self)
```

In case if the result of withdrawals in the previous loop iteration `self.token.balanceOf(self)` becomes more than `value` that will cause a transaction revert. That scenario is possible because only for the first iteration we can exactly consider that `value` more than `self.token.balanceOf(self)`, but according to the withdrawal logic there is no check for the following iterations that the real withdrawn amount(after `Strategy(strategy).withdraw(amountNeeded)` call) is less or equal to what is desired `amountNeeded`.

#### Recommendation

It seems in normal/optimistic flow that `Strategy(strategy).withdraw(amountNeeded)` never withdraws more tokens than requested, but anyway we recommend properly handling a pessimistic case because strategy is an external contract and can be broken. Due to withdrawal queue, it can be changed only by governance (usually governance is msig or smth like that). Unexpected strategy behavior can lock withdrawals for undefined period that might be fatal in some cases.

#### Status

**Fixed at** <https://github.com/iearn-finance/yearn-vaults/pull/111/commits/b129fb3b669322640dbe98b05fd3b236848613fb>

## WARNINGS

### 1. Code commentary doesn't comply with real implementation

#### Description

At [line](#) it is defined that rate limit has “tokens per block” dimension:

```
“Increase/decrease per block”
```

```
But in rate limit checker code https://github.com/iearn-finance/yearn-vaults/blob/054034304c7912d227d460feadc23177103de0b9/contracts/Vault.vy#L1124 strategy_rateLimit assumed as variable with “tokens per second” dimension
```

#### Recommendation

We recommend keeping commentaries consistent with implementation

#### Status

**Fixed at** <https://github.com/iearn-finance/yearn-vaults/pull/111/commits/62258c98bfd98315672b5f73b31b825438bec439>

### 2. Potential issue with re-entrancy

#### Description

Method `report` at this [line](#) called by strategy makes an external call back to strategy in `_assessFees` method: <https://github.com/iearn-finance/yearn-vaults/blob/054034304c7912d227d460feadc23177103de0b9/contracts/Vault.vy#L1239>. So broken or hacked, strategy can call back vaults methods while the current state is not finalized.

#### Recommendation

We recommend adding re-entrancy checks to avoid potential problems. Especially when the code logic is complicated even if for now the code is safe, in future it's really easy to implicitly introduce some bugs.

#### Status

**Fixed at** <https://github.com/iearn-finance/yearn-vaults/pull/111/commits/669c7ff9125197c3dc684fb88caa4eb839c5c0f0>

## COMMENTS

### 1. Adding Approval event in transferFrom

#### Description

Here <https://github.com/iearn-finance/yearn-vaults/blob/054034304c7912d227d460feadc23177103de0b9/contracts/Vault.vy#L468> we have a decrease in allowance as a result of `transferFrom` call, but new `Approval` event isn't emitted.

#### Recommendation

That behavior is not required by EIP20, but it's good to allow the client-side apps to sync the actual allowance amount using only events(without fetch data from the node state). Example from openzeppelin's implementation: <https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol#L154>

#### Status

**Fixed at** <https://github.com/iearn-finance/yearn-vaults/pull/111/commits/e5f8bee60e9877ad4301b1d0e3fcf9ff13111350>

### 2. Optimize deposit amount check

#### Description

Following `check` of deposit, the amount is required only if the previous `condition` returns false, in another case `amount` value is already limited by `self.depositLimit - self._totalAssets()`.

#### Recommendation

We can move limit `check` into `else` branch of the condition above to save gas.

#### Status

**Fixed at** <https://github.com/iearn-finance/yearn-vaults/pull/111/commits/1e94dc598e61961954c254153592ea36937e1a54>

### 3. Typo in commentary

#### Description

At line: <https://github.com/iearn-finance/yearn-vaults/blob/054034304c7912d227d460feadc23177103de0b9/contracts/Vault.vy#L1270>

```
@return Amount of debt outstanding (iff totalDebt > debtLimit).
```

There is an extra 'f'

#### Recommendation

We suggest removing the excess character.

#### Status

**Fixed at** <https://github.com/iearn-finance/yearn-vaults/pull/111/commits/cc82e882dfe686cb3eaa623e574c0cd16d60774c>

### 4. Implicit loss calculation

#### Description

`_reportLoss` function defined [here](#) implicitly changes the passed value of the loss tokens amount. Implicit behavior might be wrongly missed and the caller can expect another result.

#### Recommendation

We recommend to reduce the implicit logic as much as possible

#### Status

**Acknowledged**



## 5. Unoptimized `withdrawalQueue` updating

### Description

At lines: <https://github.com/iearn-finance/yearn-vaults/blob/054034304c7912d227d460feadc23177103de0b9/contracts/Vault.vy#L1062-L1066>,  
<https://github.com/iearn-finance/yearn-vaults/blob/054034304c7912d227d460feadc23177103de0b9/contracts/Vault.vy#L1041-L1046>

There are some places when we need to remove or add strategy. For now, we first iterate over the array to exclude duplicates or find item idx to remove and after that we call `_organizeWithdrawalQueue` to normalize the array. With this approach operations complexity can reach up to  $O(n^2)$ .

### Recommendation

It seems easier and cheaper to add/remove elements with instant normalization in a single walk through the array.

### Status

Acknowledged

## 6. Strategy can report loss and gain at the same time

### Description

For now strategy can `report` loss and gain at the same time. It's not a problem according to the code logic but it's little bit weird within the meaning.

### Recommendation

We recommend to make sure that this behavior is correct.

### Status

No issue

# 04 | CONCLUSION AND RESULTS

Findings list

Level	Amount
CRITICAL	0
MAJOR	1
WARNINGS	2
COMMENTS	6

Final commit identifier with all fixes:

99dcc2a8ce495ac6c2ff08e633e5b475a3088255

Smart contracts have been audited. The code is clear and well written. Compared with the Solidity based code, current implementation mostly looks more strict in terms of allowed invariants and it is much better to read and understand contract logic because the code mostly contains business-logic related constructions. Several suspicious places were spotted and some improvements were proposed.

## 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.

## Contacts

 [https://github.com/mixbytes/audits\\_public](https://github.com/mixbytes/audits_public)

 <https://mixbytes.io/>

 [hello@mixbytes.io](mailto:hello@mixbytes.io)

 <https://t.me/MixBytes>

## 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. 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.