YEARN STRATEGY SSB SMART CONTRACT AUDIT

December 23, 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 SECURITY ASSESSMENT METHODOLOGY

A group of 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 Project architecture review:
> Reviewing project documentation
> General code review
> Reverse research and study of the architecture of the code based on the source code only
> Mockup prototyping

Stage goal:
Building an independent view of the project's architecture and identifying logical flaws in the code.

02 Checking the code against the checklist of known vulnerabilities:
> 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
> Checking with static analyzers (i.e Slither, Mythril, etc.)

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

03 Checking the code for compliance with the desired security model:
> 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
> Exploits PoC development using Brownie

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of interim auditor reports into a general one:
> 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 and 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.

1.3 PROJECT OVERVIEW

`Yearn Finance` is a suite of products in Decentralized Finance (DeFi) that provides lending aggregation, yield generation, and insurance on the Ethereum blockchain. The protocol is maintained by various independent developers and is governed by YFI holders.Tonkers Kuma Strategy SSB is `Yearn Finance` strategy for gaining fee and airdrop rewards for providing liquidity to the `Balancer` project. The project consists of a single smart contract `Strategy.sol`.Most important functions are: `prepareReturn` - called each time as the strategy `keeper` is executing `harvest` on the strategy. This function is collecting trading fees and sells any airdrops.`adjustPosition` - called to (re)invest tokens`liquidatePosition` - called to immediately release tokens to be available for the vault

1.4 PROJECT DASHBOARD

Client	Yearn
Audit name	Yearn Strategy SSB
Initial version	e49d07a64ea0eb4f5a199c2bf9ea4c8aee2b313f
Final version	edbac164b49713a6c32fd7f318b5232c4a69d289
Date	December 06, 2021 - December 23, 2021
Auditors engaged	5 auditors

FILES LISTING

Strategy.sol	https://github.com/tonkers-kuma/strategy-ssb/blob/e49d07a64ea0eb4f5a199c2bf9ea4c8aee2b313f/contracts/Strategy.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	0
Warning	0
Comment	10

CONCLUSION

During the audit process, some comments were detected, 4 of them were FIXED by developer and 6 issues were ACKNOWLEDGED. Issues marked ACKNOWLEDGED make no significant impact on overall security of the project.Final commit identifier with all fixes: `edbac164b49713a6c32fd7f318b5232c4a69d289`

CONTRACT DEPLOYMENT

The following addresses contain deployed to the Ethereum mainnet and verified smart contracts code that matches audited scope:

- https://etherscan.io/address/0x9cfF0533972da48Ac05a00a375CC1a65e87Da7eC#code
- https://etherscan.io/address/0x3ef6Ec70D4D8fE69365C92086d470bb7D5fC92Eb#code
- https://etherscan.io/address/0x7A32aA9a16A59CB335ffdEe3dC94024b7F8A9a47#code
- https://etherscan.io/address/0x7c1612476D235c8054253c83B98f7Ca6f7F2E9D0#code

2. FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

Not Found

2.3 WARNING

Not Found

2.4 COMMENT

CMT-1	Unused "payable"
File	Strategy.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At line:
Strategy.sol#L133

RECOMMENDATION

We recommend removing "payable".

CLIENT'S COMMENTARY

This is required due to balancer swap.
AUDITOR'S COMMENT:
Removing `receive()` will not affect interaction with `Balancer` because `ether` is not transferred.

CMT-2	Unused "Address"
File	Strategy.sol
Severity	Comment
Status	Fixed at edbac164

DESCRIPTION

At line:
Strategy.sol#L21

RECOMMENDATION

We recommend removing "using Address for address".

CMT-3	Unused internal constant `weth`
File	Strategy.sol
Severity	Comment
Status	Fixed at edbac164

DESCRIPTION

The internal constant `weth` is unused in the contract and inaccessible outside of the contract. Probably it can be removed.

At line:
Strategy.sol#L24

RECOMMENDATION

We recommend to remove unused `weth`.

CMT-4	`sellRewards()` threshold values
File	Strategy.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At lines
Strategy.sol#L295
Strategy.sol#L277

The threshold values for selling reward tokens is probably unreasonably lower than gas amount required for token swap operation.

RECOMMENDATION

We recommend to use reasonable threshold for amount of reward tokens.

CLIENT'S COMMENTARY

This is a decimal precision concern rather than gas. Gas considered is mitigated by doing infrequent harvests manually.

CMT-5	Same value is calculated in every loop iteration
File	Strategy.sol
Severity	Comment
Status	Fixed at edbac164

DESCRIPTION

Value of `decWant` is persistent during each iteration of loops at Strategy.sol#L276 and at Strategy.sol#L293 and therefore could be calculated outside the loop to save gas.

RECOMMENDATION

We recommend to calculate value of `decWant` outside of loops in methods `harvestTrigger()` and `sellRewards()`.

CMT-6	Value of memory variable could be used instead of storage variable
File	Strategy.sol
Severity	Comment
Status	Fixed at edbac164

DESCRIPTION

Value of `maxAmountsIn[tokenIndex]` was set to a value of a memory variable `amountIn` at Strategy.sol#L199. Therefore it's more gas-efficient to use value of a memory variable instead of storage varialbe.

RECOMMENDATION

We recommend to use value of `amountIn` to evaluate if condition.

CMT-7	Malfunction on managed pools
File	Strategy.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

Once initialized, the strategy will not proceed further changes in tokens set of balancer pool. However, https://docs.balancer.fi/products/balancer-pools/managed-pools can change its set of tokens. It can lead to undesired behaviour of the strategy.

RECOMMENDATION

We recommend to perform check that pool is not `managed type` during strategy initialization, or at least describe this requirement.

CLIENT'S COMMENTARY

This strategy only applies to stable and metastable pools, which don't have the same managed functionality.

CMT-8	Unused `receive()` function
File	Strategy.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At line
Strategy.sol#L422

The contract implements function to receive ether, however, the contract's logic does not implement any further use of ether. Any ether transferred to the contract will be frozen (unaccessible).

RECOMMENDATION

We recommend to disable receiving of ether to the contract.

CLIENT'S COMMENTARY

This function is required because `Balancer` swaps requires the receiving address to be payable.

AUDITOR'S COMMENT:
Removing `receive()` will not affect interaction with `Balancer` because `ether` is not transferred.

CMT-9	Strategy's outstanding debt is not accounted in `adjustPosition()`
File	Strategy.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

Argument `_debtOutstanding` is unused in method `adjustPosition()` at Strategy.sol#L189. As a result, strategy joins pool with `amountIn` of want token that does not account strategy's outstanding debt.

RECOMMENDATION

We recommend to calculate `amountIn` as following:

```
uint256 amountIn = Math.min(maxSingleDeposit, balanceOfWant().sub(_debtOutstanding));
```

CLIENT'S COMMENTARY

Since `adjustPosition()` does not exit any funds, `_debtOutstanding` is intentionally ignored here so that any idle funds can be utilized until the next harvest.

AUDITOR'S COMMENT:
We recommend to avoid any utilization of outstanding debt, because this debt is intended to be returned to the vault ASAP.

CMT-10	`Balancer` LP token is not in `protected tokens` list
File	Strategy.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

`BaseStrategy` implements `sweep()` function that can rescue accidently transferred tokens from being locked at the strategy. Tokens used by the strategy should be protected from being `sweep()`-ed. Such tokens should be enumerated by `protected tokens`. However, `balancer` LP tokens `bpt` are not in the `protected tokens` list Strategy.sol#L262

RECOMMENDATION

We recommend to add `bpt` token to the `protected tokens`

CLIENT'S COMMENTARY

This is intentional. `protectedTokens()` is left empty so that during emergencies, governance can step in to rescue any positions
AUDITOR'S COMMENT:
We recommend to avoid using `sweep()` function for emergency rescue of strategy's tokens, because `sweep()` is intended for rescue of accidently transfered funds, not for emergencies. The `bpt` can be rescued by migration.

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