YEARN FINANCE YETH-BOOTSTRAP SECURITY AUDIT REPORT

July 11, 2023

MixBytes()

TABLE OF 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 the 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 Security Assessment Methodology

A group of auditors are involved in the work on the audit. The security engineers check the provided source code independently of each other in accordance with the methodology described below:

1. Project architecture review:

- Project documentation review.
- General code review.
- Reverse research and study of the project architecture on the source code alone.

Stage goals
- Build an independent view of the project's architecture.
- Identifying logical flaws.

2. Checking the code in accordance with the vulnerabilities checklist:

- Manual code check for vulnerabilities listed on the Contractor's internal checklist. The Contractor's checklist is constantly updated based on the analysis of hacks, research, and audit of the clients' codes.
- Code check with the use of static analyzers (i.e Slither, Mythril, etc).

Stage goal

Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flash loan attacks etc.).

3. Checking the code for compliance with the desired security model:

- Detailed study of the project documentation.
- Examination of contracts tests.
- Examination of comments in code.
- Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit.
- Exploits PoC development with the use of such programs as Brownie and Hardhat.

Stage goal

Detect inconsistencies with the desired model.

4. Consolidation of the auditors' interim reports into one:

- Cross check: each auditor reviews the reports of the others.
- Discussion of the issues found by the auditors.
- Issuance of an interim audit report.

Stage goals

- Double-check all the found issues to make sure they are relevant and the determined threat level is correct.
- Provide the Client with an interim report.

5. Bug fixing & re-audit:

- The Client either fixes the issues or provides comments on the issues found by the auditors. Feedback from the Customer must be received on every issue/bug so that the Contractor can assign them a status (either "fixed" or "acknowledged").
- Upon completion of the bug fixing, the auditors double-check each fix and assign it a specific status, providing a proof link to the fix.
- A re-audited report is issued.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Client with a re-audited report.

6. Final code verification and issuance of a public audit report:

- The Customer deploys the re-audited source code on the mainnet.
- The Contractor verifies the deployed code with the re-audited version and checks them for compliance.
- If the versions of the code match, the Contractor issues a public audit report.

Stage goals

- Conduct the final check of the code deployed on the mainnet.
- Provide the Customer with a public audit report.

Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss of funds.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Low	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer 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 Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

1.3 Project Overview

Launch yETH is a permissionless and self-governing representation of a basket of ETH Liquid Staking Tokens (LSDs).

1.4 Project Dashboard

Project Summary

Title	Description
Client	Yearn Finance
Project name	yETH-bootstrap
Timeline	May 22 2023 - June 12 2023
Number of Auditors	3

Project Log

Date	Commit Hash	Note
22.05.2023	2dd219d3af49952275934638e8c9d50d0fef0d8f	Commit for the audit
05.06.2023	210388227165577ac4a2a1d22c34bf2a88278cda	Commit for the re-audit
07.06.2023	0dd2b1925f6a060dc505f52565b540ad345f4a51	Commit for the re-audit 2

Project Scope

The audit covered the following files:

File name	Link
Bootstrap.vy	Bootstrap.vy
MerkleIncentives.vy	MerkleIncentives.vy
POL.vy	POL.vy
Token.vy	Token.vy
CurveLP.vy	CurveLP.vy
Shutdown.vy	Shutdown.vy
Stake.vy	Stake.vy

Deployments

File name	Contract deployed on mainnet	Comment
Token	0x1BED97CBC3c24A4fb5C069C6E311a967386131f7	yETH token

File name	Contract deployed on mainnet	Comment
POL	0x929401e30Aab6bd648dEf2d30FF44952BaB04478	Protocol Owned Liquidity
Bootstrap	0x41B994C192183793bB9cc35bAAb8bD9C6885c6bf	yETH bootstrap
CurveLP	0x26244DA5916BaFb59A00Ff6aFB7B9d30db882433	Curve LP Module
Shutdown	0xeD26022388281E12ff7706f82382aeAb4da2695a	Shutdown module
Stake	0x27a2FC2D47c5063A551C8fE81F580b84Ab78aC52	Staking Module
MerkleIncentives	0xAE9De8A3e62e8E2f1e3800d142D23527680a5179	Incentives for Snapshot votes

1.5 Summary of findings

Severity	# of Findings
Critical	0
High	0
Medium	3
Low	8

ID	Name	Severity	Status
M-1	Double voting in `Bootstrap.vy` in case of violation of time intervals	Medium	Fixed
M-2	Votes for cancelled protocols are lost	Medium	Fixed

M-3	CurveLP setters should not be used to update pools if the previous liquidity is not withdrawn yet	Medium	Fixed
L-1	Multiple ways for the management to mint any amount of yETH not backed by real ETH	Low	Acknowledged
L-2	An economic attack allowing protocols to bribe voters more effectively	Low	Acknowledged
L-3	A variable is not used	Low	Fixed
L-4	The yETH contract has one step for management role transferring and allows zero address as management	Low	Acknowledged
L-5	The Bootstrap.repay() function is likely supposed to be called by protocol addresses only, but it is public and just burns tokens	Low	Fixed
L-6	Rebase and deflationary tokens used as incentive can get stuck on the Bootstrap contract	Low	Acknowledged
L-7	The `operator` can make an approve to the zero address	Low	Fixed
L-8	Add an additional event	Low	Fixed

1.6 Conclusion

During the audit process 3 MEDIUM and 8 LOW severity findings were spotted. After working through the reported findings, all of them were acknowledged or fixed by the client.

2.FINDINGS REPORT

2.1 Critical

Not Found

2.2 High

Not Found

2.3 Medium

M-1	Double voting in `Bootstrap.vy` in case of violation of time intervals
Severity	Medium
Status	Fixed in 0dd2b192

Description

- Bootstrap.vy#L423

The Bootstrap contract is deployed by the owner setting key periods using the following functions:

- set_whitelist_period()
- set_incentive_period()
- set_deposit_period()
- set_vote_period()
- set_lock_end()

There is a scenario in which the `deposit_end` and `vote_end` parameters in the `Bootstrap` contract end up being higher than `lock_end`. This can happen, for example, if the management initially sets the correct values for the time intervals but later decides to manually extend the voting by calling `set_deposit_period()` and `set_vote_period()` with increased values, forgetting to also call `set_lock_end()`.

Calling `set_deposit_period()` with increased values will work without an error because requirement `_begin >= self.whitelist_begin` will be satisfied. Calling `set_vote_period()` with increased values will also work without an error because condition `_begin >= self.deposit_begin` will be satisfied.

If `deposit_end` and `vote_end` are increased enough to be higher than `lock_end`, a hacker can call `claim()` to withdraw their funds in styETH, sell them for ETH, and vote again repeating this process multiple times. Potentially, it can be done with a flash loan if the hacker finds a place to sell styETH or yETH, for example, via a curve stable pool. By accumulating votes, the hacker can claim 99.99% of all incentives from winning projects.

Recommendation

It is recommended to check invariant `vote_end < lock_end` in the `set_vote_period()` function.

Client's commentary

> As suggested, we have added a check in the `set_vote_period` function. Fixed in commit `78f08280b875531fb648acb0e1e80fa769c4530f`.

M-2	Votes for cancelled protocols are lost
Severity	Medium
Status	Fixed in 0dd2b192

Description

- Bootstrap.vy#L453

If the management calls `undo_whitelist()` for a protocol in `Bootstrap.vy` that users have already voted for, these users will not be able to reallocate their canceled votes to other projects.

Recommendation

It is recommended to add a function that allows users to call off their votes for a canceled project.

Client's commentary

> In commit `8bc8ae01cab8c1248f1dab406edc45c5ef13356a` we have added a function that allows anyone to undo anyone's votes for protocols that have had their whitelist retracted. Removing a whitelist after it has been granted is only done in exceptional situations, in the unlikely event of it happening management would most likely call this function for everyone that has voted for the protocol in question.

M-3	CurveLP setters should not be used to update pools if the previous liquidity is not withdrawn yet
Severity	Medium
Status	Fixed in 0dd2b192

Description

CurveLP stores key pool addresses - deposits, withdrawals and approvals are deterministic to these pool. But, for example, if the management calls `set_pool()` with the new pool, the operator will not be able to withdraw liquidity from the previous pool.

The are multiple of such setters:

- CurveLP.vy#L216
- CurveLP.vy#L270
- CurveLP.vy#L320
- CurveLP.vy#L349
- CurveLP.vy#L428

Recommendation

We recommend checking that the previous pool does not have liquidity or allowing withdrawal from any pool and setting restriction only for new deposits.

Client's commentary

> In commit `b018442ee6d85b438f58e4a0a31ac58da3ea6c17` we have added optional arguments to the withdraw function. In addition in commit `f8ac3c30da0acd7329b048ab2c28eeac452fe613` we have added a function to reset any token allowance of the Curve LP module, in case any of the addresses get changed by management before its allowance is set to 0.

2.4 Low

L-1	Multiple ways for the management to mint any amount of yETH not backed by real ETH
Severity	Low
Status	Acknowledged

Description

The POL contract seems to manage yETH minting, even for management and modules. But current limitations are easily avoidable.

First option. The management can:

1. send 10 ETH to `POL.__default__()`.
   `self.available` is 10 ETH now.
2. call send back 10 ETH to management calling `POL.send_native()`.
   `self.available` is not updated and remain 10 ETH.
3. send received 10 ETH to `POL.__default__()`.
   `self.available` is 20 ETH now.

As a result, 10 ETH was used to set `self.available` as 20 ETH. The management can mint 20 yETH backed by only 10 ETH.

Second option, an easier one. The management can:

1. flashloan X ETH
2. send X ETH to `POL.__default__()`
   `self.available` is set to X ETH.
3. call `POL.send_native()` to withdraw X ETH
4. repay X ETH flashloan

As a result, `self.available` is set to any amount X with little cost and in one transaction.

Recommendation

The current version of `self.available` does not set any restrictions and the whole contract does not control the backing of minted yETH.

We recommend some options:

1. checking that this behavior is intended (`self.available` can be dropped in this case)
2. adding the `self.available` decrease on ETH withdrawal
3. redesigning the control of yETH backing.

Client's commentary

> On deployment, only the existing modules will be given permissions. The modules are created in such a way that they guarantee that yETH remains fully backed. We acknowledge that if additional addresses are given arbitrary minting and transfer powers this guarantee will no longer hold. As such we are not giving these powers out, unless expicit prior approval has been given by yETH stakers. In the near future the management role of the POL will be controlled by an on-chain governance process.

L-2	An economic attack allowing protocols to bribe voters more effectively
Severity	Low
Status	Acknowledged

Description

All voters receive the incentive from the winner protocol despite the direction of their votes. Voters can vote for random protocols and still receive rewards from winners. Some economic attacks are allowed:

1. Protocol A bribes Voters A outside the Bootstrap. 10k USD are put on some external contract and distributed among Voters A only if Protocol A is a winner.
2. Protocol A does not put 10k USD as incentive to Bootstrap. But Protocol B gives 10k USD to Bootstrap.
3. Voters A vote their 50% for Protocol A. Voters B vote their 50% for Protocol B. All protocols are winners.
4. Claiming happens:

Voters A receive 10k USD from Protocol A on the external contract + 5k USD from Protocol B = 15k USD
Voters B receive 5k USD from Protocol B.

As a result, Protocol A arranged additional profits for their Voters A stealing rewards from Voters B.

Recommendation

Only management decides which protocols are the winners and takes into account potential vote bypasses. Therefore, with monitoring, such attacks won't work.

Client's commentary

> Acknowledged, we will monitor for out of band bribes.

L-3	A variable is not used
Severity	Low
Status	Fixed in 0dd2b192

Description

`POL_SPLIT` is not used.

- Bootstrap.vy#LL125

It was likely designed to calculate the funds split between treasury and POL during `split()`.

Recommendation

We recommend removing this constant.

Client's commentary

> Removed in `e8fa187c30d82c9b3b5056388b42d48fb9f2955c`.

L-4	The yETH contract has one step for management role transferring and allows zero address as management
Severity	Low
Status	Acknowledged

Description

The function does not use a two-step process for management role transferring.

- Token.vy#L125-L131

But other contracts use a two-step process with the pending_management role.

- POL.vy#L125-L147

Recommendation

We recommend using the same process of management role transferring as on other contracts (with the pending_management).

Client's commentary

> The management role on the yETH token is intended to be burned, as such we can't introduce a two step transfer.

L-5	The Bootstrap.repay() function is likely supposed to be called by protocol addresses only, but it is public and just burns tokens
Severity	Low
Status	Fixed in 0dd2b192

Description

repay() takes yETH tokens and burn them, giving nothing in return.

- Bootstrap.vy#L262-L271

It is used in the Shutdown contract as a part of a multi-step logic.

- Shutdown.vy#L47-L58

This function is available for all users and it is not obvious that it results in just losing funds.

Recommendation

We recommend shortlisting addresses allowed to call this function.

Client's commentary

> Made the repay call permissioned in commit `24de52bf432f9adf6470f218f41a9c9babb316f7`.

L-6	Rebase and deflationary tokens used as incentive can get stuck on the Bootstrap contract
Severity	Low
Status	Acknowledged

Description

The Bootstrap contract for incentives stores the amount of tokens "to be transferred", but not the amount which can be sent later.

- Bootstrap.vy#L168-L183

As a result, claim_incentive() and refund_incentive() can leave some tokens stuck on the contract if Bootstrap does not have enough balance for transfers.

- Bootstrap.vy#L300-L338

Recommendation

We recommend adding the list of allowed tokens in the documentation or accept the risk that some tokens can get stuck.

Client's commentary

> Acknowledged, the incentive mechanism is not compatible with rebasing or fee-on-transfer tokens, we will discourage users from depositing such tokens.

L-7	The `operator` can make an approve to the zero address
Severity	Low
Status	Fixed in 0dd2b192

Description

- CurveLP.vy#L232

The operator can make an extra approve to the zero address.

Recommendation

We recommend adding the following check:

```
assert self.pool != empty(address)
```

Client's commentary

> Added assertion in commit `7d392e9be4439241ae10b3f46f5cc2a837d31f73`.

L-8	Add an additional event
Severity	Low
Status	Fixed in 0dd2b192

Description

There is no event for the change of `convex_pool_id`:

- CurveLP.vy#L336

Recommendation

We recommend adding a new event for `convex_pool_id`.

Client's commentary

> Added event in commit `e90ff74e890da0444dc8837bcbf1615ece742d0d`.

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build opensource 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://mixbytes.io/


hello@mixbytes.io


https://twitter.com/mixbytes