YEARN LIQUITY STABILITY POOL SMART CONTRACT AUDIT

June 02, 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

Smart contract is a strategy to earn WETH by borrowing LUSD and depositing it into the Liquity stability pool.

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 main purpose of the project is to give users add additional ability to use the protocol managed by strategy.

1.5 PROJECT DASHBOARD

Client	Yearn
Audit name	Liquity stability pool
Initial version	c3fa76af0a4e2d5fd7132b8e24361d5b7439a75d e14ddae794b3be8f6c5cb39b8cac4e168c366bb1
Final version	e14ddae794b3be8f6c5cb39b8cac4e168c366bb1`
SLOC	329
Date	2021-04-28 - 2021-06-02
Auditors engaged	2 auditors

FILES LISTING

Strategy.sol	Strategy.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	2
Warning	3
Comment	2

CONCLUSION

Smart contract has 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 discussed with 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:
`e14ddae794b3be8f6c5cb39b8cac4e168c366bb1`

2. FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

MJR-1	Losses are not taken into account in the strategy
File	Strategy.sol
Severity	Major
Status	Fixed at Strategy.sol

DESCRIPTION

The withdrawer from vault should incur the losses from liquidation caused by his own withdrawal. However, the Strategy.sol#L307 is ignoring possible trove liquidation. Currenly liquidation will cause revert, but even if not, currently strategy is ignoring the case of trove liquidation. This may lead to improper accounting of user balances and possible locking of vault withdrawals.

RECOMMENDATION

It is recommended to rewrite logic of `liquidatePosition()` considering the liquidation.

MJR-2	Malfunction of strategy and entire vault on unexpected trove status
File	Strategy.sol
Severity	Major
Status	Fixed at Strategy.sol

DESCRIPTION

A *liquity trove* can be in one of several states: *nonExistent*, *active*, *closedByOwner*, *closedByLiquidation*, *closedByRedemption*. When the *trove* is in state different than *active*, any call to *ajdustTrove()* will fail. Unfortunately, the strategy does not implement proper handling of trove state. There is some scenarios that will cause trove to be in unexpected state: trove liquidation, full collaterial redemption and trove manual *Strategy.sol#L374* by priveleged user.

The Strategy.sol#L216 function is expected only two states: *zero* (which is *nonExistent*) and other (which is *active*, *closedByOwner*, *closedByLiquidation*, *closedByRedemption*). However, only *active* state is valid for *Strategy.sol#L237*, other three states will cause revert.

The *Strategy.sol#L307* function does not handle trove state. When trove is in any state except *active*, *liquidatePosition()* will revert on *Strategy.sol#L322* call. This will break harvest() and withdraw() functions so strategy will become broken and should be manually removed from the vault to prevent blocking of any withdrawal from it.

RECOMMENDATION

It is recommended to handle state of the trove properly.

2.3 WARNING

WRN-1	There is no check on the result of the function
File	Strategy.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

According to the ERC20 standard, the `approve()` function returns a boolean value. But in the contract on lines Strategy.sol#L120-L123, after the call to the `_approveAll()` function, this values are not processed. A situation may arise that a False will return.

RECOMMENDATION

It is recommended to add a check of the return value.

WRN-2	The approval value obtained in the constructor may not be enough for the long term of the smart contract
File	Strategy.sol
Severity	Warning
Status	Fixed at **e14ddae7**

DESCRIPTION

At line: Strategy.sol#L115
the smart contract constructor call `_approveAll()` function for different tokens. But in the process of work, the obtained value will only decrease. If this value decreases to zero, then the tokens will remain locked in the contract forever.

RECOMMENDATION

It is recommended to add a function to increase the value of approvals.

WRN-3	Overreportion of the losses
File	Strategy.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

In some rare conditions the liquidatePosition may report that all requested liquidity is lost even if no losses are actually suffered. Such liquidation may occur during harvest() and do a major overreporting to the vault and break its accounting. As no losses are suffered actually, this state can be fixed by migrating to a "fixing" strategy. However, the vault accounting will be invalid until manual interaction of the vault governance, and the vault shares will be underpriced.

RECOMMENDATION

To fix overreporting

CLIENT'S COMMENTARY

We concluded that we could avoid that with monitoring.

2.4 COMMENTS

CMT-1	Unused variable
File	Strategy.sol
Severity	Comment
Status	Fixed at e14ddae7

DESCRIPTION

Variable Strategy.sol#L386 is unused.

RECOMMENDATION

It is recommended to remove variable.

CMT-2	Repeated `provideToSP`
File	Strategy.sol
Severity	Comment
Status	Fixed at e14ddae7

DESCRIPTION

`spool.provideToSP` is executing in any condition Strategy.sol#L234-L238

RECOMMENDATION

Put Strategy.sol#L234 out of if-else statement.

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