YEARN YSWAPS SECURITY AUDIT REPORT

10.01.2022-28.02.2022

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 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.

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 cients' 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 reoprts 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 Customer with an interim report.

5. Bug fixing & re-audit:

- The Customer 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 Customer 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

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Customer with a re-audited 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 funds to be transferred to any party.
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

`Yearn swaps` is a wrapper for several DEX (currently `1inch`, `Bancor`, `Uniswap`, `ZRX`) intended to be used by `Yearn Vaults` strategies. It provides some common security checks, some centralized management to allow centralized emergency disable of selected swap paths and centralized asyncronous swapping by a trusted account.

Project consists of smart contracts:

- `TradeFactory.sol` - entry point for use by `Yearn Vaults` strategies, management and off-chain bots so-called `Mechanics`. Inherits `TradeFactoryAccessManager.sol` for ACL, `TradeFactoryExecutor.sol` for implementation of swapper interaction, `TradeFactoryPositionsHandler.sol` for management of swap paths and `TradeFactorySwapperHandler.sol` for management of swappers per strategy.

- Directory `swappers/sync` contains several contracts for interaction with specific DEX swappers: `BancorSwapper.sol`, `UniswapV2Swapper.sol`. Additionally, it contains base contract `SyncSwapper.sol` for common syncronous swapper logic.

- Directory `swappers/async` contains similar contracts as above: `AsyncSwappers.sol` and `MultipleAsyncSwapper` as base contracts, `BancorSwapper.sol`, `OneInchAggregatorSwapper.sol`, `UniswapV2Swapper.sol` and `ZRXSwapper.sol` for specific DEX swappers. Async swappers are intended to be called by trusted account, so-called `Mechanics`.

Additionally, project contains some common libraries and base contracts for implementation of access control and utility function like collecting `dust` tokens: `Governable.sol`, `Machinery.sol` and `CollectableDust.sol`

1.4 Project Dashboard

Project Summary

Title	Description
Client	Yearn Finance
Project name	yswaps
Timeline	10.01.2022-28.02.2022
Number of Auditors	5

Project Log

Date	Commit Hash	Note
10.01.2022	ecc0b5147992b34c315e08af170ceb4a5fe071ee	Initial commit
31.01.2022	e933fcb4b64548200e8090bc9cbbf54c7119bc7b	Updated commit with fixes
23.02.2022	a1b210e78f6e7936ca0d7d79d512800b794fff2a	Final commit including issue found by client

Project Scope

The audit covered the following files:

File name	Link
TradeFactory.sol	TradeFactory.sol
TradeFactoryAccessManager.sol	TradeFactoryAccessManager.sol

File name	Link
TradeFactoryExecutor.sol	TradeFactoryExecutor.sol
TradeFactoryPositionsHandler.sol	TradeFactoryPositionsHandler.sol
TradeFactorySwapperHandler.sol	TradeFactorySwapperHandler.sol
CommonErrors.sol	CommonErrors.sol
AsyncSwapper.sol	AsyncSwapper.sol
BancorSwapper.sol	BancorSwapper.sol
MultiCallOptimizedSwapper.sol	MultiCallOptimizedSwapper.sol
OneInchAggregatorSwapper.sol	OneInchAggregatorSwapper.sol
UniswapV2Swapper.sol	UniswapV2Swapper.sol
ZRXSwapper.sol	ZRXSwapper.sol
BancorSwapper.sol	BancorSwapper.sol
SyncSwapper.sol	SyncSwapper.sol
UniswapV2AnchorSwapper.sol	UniswapV2AnchorSwapper.sol
UniswapV2Swapper.sol	UniswapV2Swapper.sol
Swapper.sol	Swapper.sol
SwapperEnabled.sol	SwapperEnabled.sol
CollectableDust.sol	CollectableDust.sol
Governable.sol	Governable.sol
Machinery.sol	Machinery.sol

1.5 Summary of findings

Severity	# of Findings
Critical	0
High	0
Medium	4
Low	5

ID	Name	Severity	Status
M-1	Security and sanity check bypass in `MultiCallOptimizedSwapper`	Medium	Acknowledged
M-2	Unnecessary and unrestricted `trade()` at `sync/BancorSwapper`	Medium	Fixed
M-3	Incompatibility with fee-on-transfer tokens	Medium	Acknowledged
M-4	Incorrect slippage check	Medium	Fixed
L-1	Unrestricted setter in `Machinery` contract	Low	Fixed
L-2	Unnecessary assembly in `MultiCallOptimizedSwapper.sol`	Low	Acknowledged
L-3	Variables can be declared as immutable	Low	Fixed
L-4	Duplicating sanity check	Low	Fixed
L-5	Redundant operation on return values	Low	Fixed

1.6 Conclusion

During the audit process, 3 medium and 5 low issues were found by the team of auditors. Also, the developers found and fixed 1 medium issue.

One of the issues, described as Medium 1, grants too much access on `Yearn Vaults` strategies funds to a single centralized account key known as `Mechanic`. To prevent internal security threat, this key should be properly secured, however, it will be permanently accessible by the off-chain technical infrastructure, thus it may be compromised by a persons who performs infrastructure maintenance. This issue was acknowledged by the developers but was not fixed.

Other issues were fixed or acknowledged by the developers. Acknowledged issues make no significant impact on the contract security and perfomance.

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

Contract name	Deployed link
TradeFactory.sol	0x7BAF843e06095f68F4990Ca50161C2C4E4e01ec6
TradeFactoryPositionsHandler.sol	0x7BAF843e06095f68F4990Ca50161C2C4E4e01ec6
TradeFactoryExecutor.sol	0x7BAF843e06095f68F4990Ca50161C2C4E4e01ec6
MultiCallOptimizedSwapper.sol	0x711d1D8E8B2b468c92c202127A2BBFEFC14bf105
MultipleAsyncSwapper.sol	0x711d1D8E8B2b468c92c202127A2BBFEFC14bf105
AsyncSwapper.sol	0x711d1D8E8B2b468c92c202127A2BBFEFC14bf105
Swapper.sol	0x711d1D8E8B2b468c92c202127A2BBFEFC14bf105
Governable.sol	0x711d1D8E8B2b468c92c202127A2BBFEFC14bf105
CollectableDust.sol	0x711d1D8E8B2b468c92c202127A2BBFEFC14bf105

2. FINDINGS REPORT

2.1 Critical

Not Found

2.2 High

Not Found

2.3 Medium

M-1	Security and sanity check bypass in `MultiCallOptimizedSwapper`
Files	TradeFactoryExecutor.sol#L126 MultiCallOptimizedSwapper.sol#L27
Severity	Medium
Status	Acknowledged

Description

As well as similar functions, TradeFactoryExecutor.sol#L126 performs several sanity and security checks:

- provided swapper is required to be in `_swappers` enumerable set
- exchange path `tokenIn` and `tokenOut` is required to be enabled for given `strategy`
- output token amount is required to be greater or equal to provided `minAmountOut`.

However, the actual code executed by MultiCallOptimizedSwapper.sol#L27 is arbitrary and can accidentally or intentionally bypass these checks. Although `MultiCallOptimizedSwapper` can be only invoked by the trusted actor `onlyMechanics`, arbitrary code execution can probably lead to loss of funds in case of accidental bug at off-chain bot. Also, single compromised `Mechanic` account key can lead to "rug pull" from all connected strategies.

Recommendation

We recommend to disallow execution of arbitrary code and keep the best practice of white-listing of allowed actions and calls, similar to other swappers at `contracts/swappers/async` directory.

Client's commentary

We decided to maintain the MultiCallOptimizedSwapper as is.

M-2	Unnecessary and unrestricted `trade()` at `sync/BancorSwapper`
File	BancorSwapper.sol#L52
Severity	Medium
Status	Fixed at 442934a4

Description

BancorSwapper.sol#L52 function has no use and most likely remains in contract by an accident. However, it is unrestricted and can be called by anyone.

Recommendation

Although we have not found any attack vector for this issue, we recommend to remove this code to improve security.

M-3	Incompatibility with fee-on-transfer tokens
File	TradeFactoryExecutor.sol#L82-L88
Severity	Medium
Status	Acknowledged

Description

Several parts of code have assumption that `transferFrom(..., amount)` will result in receiving of exact `amount` of tokens:

- TradeFactoryExecutor.sol#L82-L88
- TradeFactoryExecutor.sol#L111-L117

However, actual amount of received tokens can be less for fee-on-transfer tokens.

Recommendation

For compatibility with fee-on-transfer tokens we recommend to use `balanceOf` to obtain actual amounts of tokens received by `transferFrom`.

Client's commentary

We will not support fee on trasfer tokens.

M-4	Incorrect slippage check
File	TradeFactoryExecutor.sol#L150
Severity	Medium
Status	Fixed at PR-62

Description

Condition in slippage check is logically reversed:
TradeFactoryExecutor.sol#L150

Recommendation

Note: this issue was found by the developers

2.4 Low

L-1	Unrestricted setter in `Machinery` contract
File	Machinery.sol#L22
Severity	Low
Status	Fixed at ad915949

Description

Setter for the mechanics registry Machinery.sol#L22 has no access restriction. It can cause security breach in inherited contracts that lack security-aware override of `setMechanicsRegistry` function.

Recommendation

Although currently all of the inherited contracts from audited scope override `setMechanicsRegistry` properly, we recommend to prevent accidental inheritance of insecure code by modifying or removing `setMechanicsRegistry` function from `Machinery` contract. Additionally, this contract is most likely has been planned to be `abstract`.

Client's commentary

Took out biz logic of setMechanicsRegistry contract, and converted it to an abstract, so it must be implemented by those who inherit it.

L-2	Unnecessary assembly in `MultiCallOptimizedSwapper.sol`
File	MultiCallOptimizedSwapper.sol
Severity	Low
Status	Acknowledged

Description

MultiCallOptimizedSwapper.sol contract is using assembly most likely for gas optimization purposes. However, this approach does not make significant impact to overall gas usage while it makes the code more complex and unsafe.

Recommendation

In favour of readability and safety of code. we recommend to avoid unnecessary usage of assembly in Solidity.

Client's commentary

We decided to maintain the MultiCallOptimizedSwapper as is.

L-3	Variables can be declared as immutable
Files	BancorSwapper.sol#L40-L41 BancorSwapper.sol#L38-L39
Severity	Low
Status	Fixed at 635ce3db

Description

Following variables of both sync and async Bancor swappers can be declared as immutable: `contractRegistry`, `bancorNetworkName` at BancorSwapper.sol#L40-L41 and `contractRegistry`, `bancorNetworkName` at BancorSwapper.sol#L38-L39

Recommendation

We recommend to declare mentioned variables as immutable.

L-4	Duplicating sanity check
Files	TradeFactoryExecutor.sol#L81 SyncSwapper.sol#L38
Severity	Low
Status	Fixed at 5a99d094

Description

Sanity check `maxSlippage != 0` at TradeFactoryExecutor.sol#L81 duplicates sanity check at SyncSwapper.sol#L38

Recommendation

We recommend to remove redundant check

L-5	Redundant operation on return values
Files	UniswapV2Swapper.sol#L52 UniswapV2AnchorSwapper.sol#L74
Severity	Low
Status	Fixed at 09d1bcfc, e933fcb4

Description

Here are performed some redundant operation on return values of `swapExactTokensForTokens()`:

- UniswapV2Swapper.sol#L52
- UniswapV2AnchorSwapper.sol#L74

It makes no impact on contract security or perfomance and only noted in favour of graceful code.

Recommendation

We recommend to remove redundant operations

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