MELLOW FINANCE SMART CONTRACT AUDIT

July 19, 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 Mellow 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 PROJECT OVERVIEW

Mellow protocol is UNI V3 liquidity providing optimization solution.

1.3 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.

1.4 EXECUTIVE SUMMARY

The audited scope of contracts arranges UNI V3 liquidity providing optimization solution.

1.5 PROJECT DASHBOARD

Client	Mellow Finance
Initial version	ec766bf844db0bce4b4adae63458fc08d02e21af
Final version	702dc5a2a8a6dd70a3faf45838934321cacf4b49 a58075bb16ad05cfc766d65cf68f8f27b8739ff0
Date	June 11, 2021 - July 19, 2021
Auditors engaged	2 auditors

FILES LISTING

LiquidityVault.sol	https://github.com/mellow-finance/mellow-contracts/blob/ec766bf844db0bce4b4adae63458fc08d02e21af/contracts/LiquidityVault.sol
StrategyManager.sol	https://github.com/mellow-finance/mellow-contracts/blob/ec766bf844db0bce4b4adae63458fc08d02e21af/contracts/StrategyManager.sol
LpToken.sol	https://github.com/mellow-finance/mellow-contracts/blob/ec766bf844db0bce4b4adae63458fc08d02e21af/contracts/LpToken.sol
VaultAccessControl.sol	https://github.com/mellow-finance/mellow-contracts/blob/ec766bf844db0bce4b4adae63458fc08d02e21af/contracts/VaultAccessControl.sol
VaultMath.sol	https://github.com/mellow-finance/mellow-contracts/blob/ec766bf844db0bce4b4adae63458fc08d02e21af/contracts/libraries/VaultMath.sol
ILiquidityVault.sol	https://github.com/mellow-finance/mellow-contracts/blob/ec766bf844db0bce4b4adae63458fc08d02e21af/contracts/interfaces/ILiquidityVault.sol

IStrategyManager.sol	https://github.com/mellow-finance/mellow-contracts/blob/ec766bf844db0bce4b4adae63458fc08d02e21af/contracts/interfaces/IStrategyManager.sol
IToken.sol	https://github.com/mellow-finance/mellow-contracts/blob/ec766bf844db0bce4b4adae63458fc08d02e21af/contracts/interfaces/IToken.sol

FINDINGS SUMMARY

Level	Amount
Critical	1
Major	1
Warning	4
Comment	1

CONCLUSION

Smart contracts have been audited and several suspicious places have been spotted. During the audit one critical issue was found and fixed by the client. One issue was marked as major because it could lead to 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. So, the contracts are assumed as secure to use according to our security criteria.Final commit identifier with all fixes: `a58075bb16ad05cfc766d65cf68f8f27b8739ff0`

2. FINDINGS REPORT

2.1 CRITICAL

CRT-1	Balance drain with deposit/withdraw repeat
File	LiquidityVault.sol
Severity	Critical
Status	Fixed at a58075bb

DESCRIPTION

The `deposit` and the `withdraw` are using different base value for conversion between vault shares into user deposits. The `withdraw` is based on LiquidityVault.sol#L198, however `deposit` is based on LiquidityVault.sol#L126. The attacker can drain vault balance by making deposit/withdraw repeatedly. The exploit is provided.

RECOMMENDATION

It is recommended to make accounting of deposit/withdraw operation properly

2.2 MAJOR

MJR-1	Possible withdraw unavailiability
File	LiquidityVault.sol
Severity	Major
Status	Fixed at a58075bb

DESCRIPTION

The amount of transferred out tokens are not accounted at the vault state while processing user withdrawals. This will cause attempt to withdraw incorrect (too large) amount and probable almost any withdrawals attempts will be reverted.

RECOMMENDATION

It is recommended to make accounting of deposit/withdraw operation properly

2.3 WARNING

WRN-1	Reversed expression of require() at `commitStrategyParams()`
File	LiquidityVault.sol
Severity	Warning
Status	Fixed at 702dc5a2

DESCRIPTION

This code is requiring to be called **before** specified deadline. However, it should require to be called **after** the deadline.

```
require(
    (pendingStrategyParams[strategyId].unlockDate != 0) &&
        (pendingStrategyParams[strategyId].unlockDate > block.timestamp),
    "L"
);
```

RECOMMENDATION

It is recommended to fix expression

WRN-2	Unability to withdraw profits from Position Manager
File	LiquidityVault.sol
Severity	Warning
Status	Fixed at a58075bb

DESCRIPTION

The call parameters of positionManager.collect() prohibits collecting of profit. Only liquidity value will be collected.

RECOMMENDATION

It is recommended to not use undesired limitation of amount to be collected

WRN-3	collectExitFees, collectPerformanceFees don't check transfer result
File	LiquidityVault.sol
Severity	Warning
Status	Fixed at a58075bb

DESCRIPTION

`freeTokens` value will be deceased even if token transfer fails

LiquidityVault.sol#L478
LiquidityVault.sol#L483
LiquidityVault.sol#L516
LiquidityVault.sol#L521

RECOMMENDATION

It is recommended to use `safeTransfer` instead of `transfer`

WRN-4	Inproper copy-paste in `rebalance()`
File	LiquidityVault.sol
Severity	Warning
Status	Fixed at 702dc5a2

DESCRIPTION

amount1Min value is copy-pasted from previous live without proper modification
LiquidityVault.sol#L433

```
ClosePositionArgs({
    strategyId: args.strategyId,
    amount0Min: args.token0MinClose,
    amount1Min: args.token0MinClose,
    deadline: args.deadline
})
```

RECOMMENDATION

It is recommended to fix amount1Min value

2.4 COMMENT

CMT-1	Unused variable
File	LiquidityVault.sol
Severity	Comment
Status	Fixed at a58075bb

DESCRIPTION

State variable `factory` of `LiquidityVault` is used only during initialization in constructor. It is never used anywhere else in the contract.

RECOMMENDATION

It is recommended to remove unused variable `factory` declaration and usage in constructor

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