Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Audit-Report 1inch AMM Smart Contracts 12.2020

Cure53, Dr.-Ing. M. Heiderich, Dr. N. Kobeissi

Index

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Introduction

"1inch offers the best rates by discovering the most efficient swapping routes across all leading DEXes."

From https://1inch.exchange/#/

This report describes the results of a thorough penetration test and source code audit against several Mooniswap v2 Smart Contracts and 1inch Token Smart Contracts. The work was requested by 1Inch and promptly executed by Cure53 in December 2020, precisely in CW51 and CW52.

White-box methodology was chosen and used, as is usual for auditing smart contracts. Cure53 was granted access to all relevant sources and background information as well. All information was shared via a private GitHub repository. The Cure53 team, which consisted of two members, invested a total of six days into the project to reach good coverage.

To optimize the structure of work, Cure53 proceeded with two different work packages (WPs), namely:

- **WP1**: Crypto Reviews and Audits against Mooniswap v2 Smart Contracts
- **WP2**: Crypto Reviews and Audits against 1Inch Token Smart Contracts

In connection to the scope objects, it can be clarified that Mooniswap is a simple and efficient design for an Automatic Market Maker (AMM). It aims to preserve high profit for liquidity providers by employing simple formulas based on constant-product invariant equations. This is done to minimize losses in the case of arbitrage or extreme price fluctuations.

The 1inch exchange employs Mooniswap as a core component of its service. As such, the Mooniswap contract-suite is responsible for token swaps and exchanges as well as pool liquidity management operations. This is done through four main *accessor* functions of *Swap*, *Deposit*, *Withdraw* and *Deploy*. They access a number of internal functions that deal with ancillary AMM operations such as the setting of governance fees and percentages. Cure53 was also supplied with documentation, background info and audit reports that were created by other firms in the past.

The project started on schedule and progressed efficiently. Communications between Cure53 and 1inch were done over email for initial preparations and then in a dedicated private Telegram channel into which all relevant personnel were invited. Discussions were productive and no roadblocks were encountered. The 1Inch team was very helpful

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

in aiding test preparations and getting Cure53 to be ready for this test and audit exercise. All questions posed by Cure53 were swiftly answered and prompt feedback was given. Cure53 faced no problems understanding specifics of the scope. Expected coverage levels were reached thanks to the assistance of the 1Inch team.

The initial version of this report was delivered in December 21st. On December 22nd, the 1Inch team shared an updated git commit tag containing a codebase that addressed three issues that were not documented in the initial version of this report. This led to a decision to produce a major second draft of this audit report which contains the following new elements:

- Analysis and confirmation of the fixes to the three issues identified by the 1Inch team, documented in a new subsection, Security Improvements Implemented by 1Inch.
- In-depth comparison of the git commit tag shared for the initial version of this report (*ca55668d1fe0abe7d9368a4ac3ac09affdf04aca*) and the git commit tag shared on December 23rd (*af6582d2e2a5ffd2677a51a0a80f2988932a8a5d*), including security-sensitive and non security-sensitive changes, documented in a new subsection, Smart Contract Change and Additions.

The report will now shed more light on the scope and the test-parameters. This section will be followed by a chapter detailing on the test-coverage reached in this exercise, so as to offer transparency on what areas Cure53 covered in full and what kinds of attacks were attempted. The rationale is quite clearly connected to the lack of findings across the WPs. Conclusions ensue, detailing the impressions gained by Cure53. The security properties of the smart contracts in scope are discussed, while Cure53 equips 1inch team with additional hardening advice as applicable.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Scope

- **Cryptographic Reviews & Security Audits against 1Inch SCs & Token Code**
  - **WP1**: Crypto reviews and audits of the Mooniswap v2 smart contracts
    - The following files were in scope
      - *Mooniswap.sol*
      - *MooniswapConstants.sol*
      - *MooniswapDeployer.sol*
      - *MooniswapFactory.sol*
      - *ReferralFeeReceiver.sol*
      - *governance/BaseGovernanceModule.sol*
      - *governance/GovernanceFeeReceiver.sol*
      - *governance/MooniswapFactoryGovernance.sol*
      - *governance/MooniswapGovernance.sol*
      - *governance/rewards/IRewardDistributionRecipient.sol*
      - *governance/rewards/Rewards.sol*
      - *inch/GovernanceMothership.sol*
      - *interfaces/IGovernanceModule.sol*
      - *interfaces/IMooniswapDeployer.sol*
      - *interfaces/IMooniswapFactoryGovernance.sol*
      - *interfaces/IReferralFeeReceiver.sol*
      - *libraries/LiquidVoting.sol*
      - *libraries/Sqrt.sol*
      - *libraries/UniERC20.sol*
      - *libraries/VirtualBalance.sol*
      - *libraries/Vote.sol*
      - *utils/BalanceAccounting.sol*
      - *utils/Converter.sol*
  - **WP2**: Crypto reviews and audits against 1Inch token smart contracts
    - The following file was in scope:
      - *contracts/inch/GovernanceMothership.sol*
  - **Sources were shared with Cure53 using GitHub**
  - **Audited GitHub commit**
    - https://github.com/1inch-exchange/mooniswap-v2/commit/ca55668d1fe0abe7d9368a4ac3ac09affdf04aca
  - **Audited Git tag**
    - https://github.com/1inch-exchange/mooniswap-v2/releases/tag/audit-4
  - **Test-supporting material was shared with Cure53**
  - **Reports from previous audits**
    - https://github.com/1inch-exchange/1inch-v2-audits

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Test Methodology

As a supplement to the coverage section, this chapter lists the types of standard checks that were performed on this set of smart contracts. This was based on a list of common security issues and standard auditing practices.

Compiler version

Some Solidity contracts may be pegged to an outdated version of the Solidity compiler. Most Mooniswap smart contracts specify Solidity 0.6.0. Solidity is currently at 0.8.x - therefore, a version bump for the specified Solidity compilers may be in order.

Integer arithmetic

All shared smart contracts were checked for safe integer arithmetic. Two exceptions were found. Both were discussed with the client:

- The *swapFor* function in *Mooniswap.sol* employs unsafe overflow arithmetic for type *uint128* values (on lines 240 and 241).
- *utils/Converter.sol* employs *for loop* with regular integer arithmetic. However, the iterations concern data structures that are practically impossible for an attacker to exploit.

Floating point arithmetic

Floating point arithmetic support in Solidity is limited and may sometimes lead to unexpected behaviors. No issues with floating point arithmetic were identified over the course of this audit.

Reentrancy attacks

When a contract calls an external function, said external function may itself call the calling function. This can have unexpected effects. To prevent problems, a contract can implement a lock-in storage to prevent re-entry calls. No re-entering-type issues were observed in the Mooniswap contract stack.

Access control

Especially in AMM-type contracts, proper access control across the entire smart contract interface is mandatory. The following items were checked for the Mooniswap smart contract stack:

- *Default visibility*: external and internal functions are correctly specified throughout the smart contract.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

- *Authentication with tx.origin*: explicitly forbidden within the smart contract stack.
- *Signature verification*: not applicable (no usage of signatures).
- *Unprotected functions*: no issues identified in any of the smart contracts (correct ACL function access).

Signature replay attacks

No issues identified in any smart contracts since the Mooniswap stack does not employ signatures.

Unchecked external calls

In Solidity, there are multiple ways to call an external contract and send ether. The function *transfer* reverts if the transfer fails. However, the functions *call* and *send* return *false*. Programmers may mistakenly expect a *call* and *send* to revert whilst failing to check their return value.

No issues relating to unchecked external calls were identified in the Mooniswap smart contract.

Denial-of-Service

Smart contracts may be manipulated to cause a broad Denial-of-Service to future contract callers through a variety of different techniques:

- *Unexpected revert*: no issues identified.
- *Block gas limit*: no issues identified.
- *External calls without gas stipends*: not applicable.
- *Offline owner*: no issues identified.

Privacy illusion

The Mooniswap smart contracts do not adopt false expectations of privacy regarding data stored on-chain.

Potential attacks from miners

Miners have some level of control over the ordering and timestamping of transactions on the chain. *Transaction ordering* and *timestamp manipulation* by a miner were deemed to be within bounds that are too small to have any significant impact on Mooniswap functionality. This was confirmed despite the usage of *block.timestamp* by Mooniswap in *Rewards.sol*, *LiquidVoting.sol* and *VirtualBalance.sol.*

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

External contract referencing

The Mooniswap smart contract stack did not reference any unchecked external contracts.

Uninitialized storage pointers

The Mooniswap smart contract stack did not use uninitialized storage pointers.

Incorrect interfaces

All Mooniswap-exposed interfaces were documented and called appropriately.

Variable shadowing

No issues were identified with variable name-spacing. Proper name-spacing was followed across all smart contracts with no overlap.

Assert violations

Calls to *assert* were not present in the Mooniswap smart contract stack, which instead relies entirely on *require*.

Deprecated/historic functionality

As documented by the Smart Contract Weakness Classification Registry:[1]

> *"Several functions and operators in Solidity are deprecated. Using them leads to reduced code quality. With new major versions of the Solidity compiler, deprecated functions and operators may result in side effects and compile errors."*

No deprecated functionality was identified in the Mooniswap smart contract stack.

[1] https://swcregistry.io/docs/SWC-111

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Report Revisions for Updated Codebase

On December 24th, 2020, 1Inch shared an updated set of smart contracts (with commit tag *af6582d2e2a5ffd2677a51a0a80f2988932a8a5d*) which contained substantial differences from the code shared at the outset of this audit. These differences can be split into two categories:

- Security improvements implemented by 1Inch, which resolved issues that were not included in the initial audit report targeting the initial shared codebase.
- Substantial changes to the smart contract stack including performance/gas improvements, namespacing improvements, stricter checks and new features.

This section covers both of the above classes of elements introduced in the updated smart contract stack.

Security Improvements Implemented by 1Inch

A set of three security fixes were implemented by 1Inch:

Correct Balance Calculation in Edge Case

1inch contributed a fix to the *_beforeTokenTransfer* function within *MooniswapGovernance.sol* which could lead to the *balanceTo* value to be calculated incorrectly in some cases.[2] The original code for calculating *balanceTo* was the following:

```
uint256 balanceTo = (from != address(0)) ? balanceOf(to) : 0;
```

This rendered the value identical to the preceding *balanceFrom* value, thereby corrupting the ternary check logic. The fix replaces the *from* variable with the correct *to* variable.

Balance Notifications Sendable to Arbitrary Addresses

In *GovernanceMothership.sol*, the *notifyFor* function could be used in order to notify an account holder of an arbitrary sender's balance instead of their own, leading to potential governance issues.[3]

```
function notify() external {
_notifyFor(msg.sender, balanceOf(msg.sender));
}

function notifyFor(address account) external {
```

[2] https://github.com/1inch-exchange/mooniswap-v2/blob/ca55668d1fe0abe7d9368a4ac3ac0...sol#L102
[3] https://github.com/1inch-exchange/mooniswap-v2/blob/ca55668d1fe0abe7d9368a4ac3ac0...sol#L49

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

```
_notifyFor(account, balanceOf(msg.sender));
}
```

Additional Validation for Mooniswap Arguments in Referral Logic

An additional requirement, *validPool*, was added to all external functionality in *ReferralFeeReceiver.sol*.[4]

Smart Contract Changes and Additions

- *Governance*
  - New: ExchangeGovernance.sol
    - New master class for exchange governance.
    - *_notifyStakeChanged*
  - New: GovernanceRewards.sol
    - Constructor
  - Modified: GovernanceFeeReceiver
    - Namespace changes only
  - Modified: MooniswapFactoryGovernance.sol
    - Namespace changes
    - New *shutdown()* function
    - New *isActive()* function
    - Fee receiver management functions and one-way *isFeeReceiver* mapping
  - Modified: MooniswapGovernance.sol
    - Correct Balance Calculation in Edge Case
    - Namespace changes
    - Gas improvements to *_beforeTokenTransfer*
- *Inch*
  - New: FarmingRewards.sol
  - Modified: GovernanceMothership.sol
    - Balance Notifications Sendable to Arbitrary Addresses
    - Changes to *_notifyFor* focusing on gas optimization
- *Libraries*
  - New: ExchangeConstants.sol
    - Namespacing
  - New: ExplicitLiquidVoting.sol
  - New: SafeCast.sol
    - Safe uint casting library
  - New: VirtualVote.sol
    - Vote utility function
  - New: Voting.sol
  - Modified: MooniswapConstants.col

[4] https://github.com/1inch-exchange/mooniswap-v2/commit/6340a4844dbb65f38706c1e78f83d4a...42

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

        - Namespacing
    - Modified: LiquidVoting.sol
        - Minor changes
    - Modified: UniERC20.sol
    - Modified: VirtualBalance.sol
- *Utils*
    - New: BaseRewards.sol
    - Modified: BalanceAccounting.sol
    - Modified: Converter.sol
    - Modified: Mooniswap.sol
        - Minor changes
    - Modified: MooniswapFactory.sol
        - *transferOwnership(_poolOwner)* is part of a “new shutdown functionality that disables swaps” according to client
    - Modified: ReferralFeeReceiver.sol
        - Additional Validation for Mooniswap Arguments in Referral Logic
        - Reentrancy guard added
        - Elaborated spread calculation algorithm

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Conclusions

Immediately after the conclusion of the initial version of this report, the 1inch team reported three security-relevant issues in the original shared codebase that were not identified in the original report. This led to this second report which documents the issues shared by 1inch, confirms fixes, and includes a new overview of all changes between the original shared codebase and the new git commit tag shared by 1inch on December 23rd. As of this second report, no additional security-relevant flaws were spotted on the 1inch scope during this 2020 project. Cure53 concludes this work with a positive impression about the examined security goals of the Mooniswap smart contract stack. After spending six days on the scope, two members of the Cure53 team can confirm that the Mooniswap smart contract stack appears to provide a simple and efficient design for an Automatic Market Maker (AMM).

As noted in this report, Mooniswap's AMM is capable of aptly preserving high profit for liquidity providers via the employment of simple formulas based on constant-product invariant equations. The aim, just to reiterate, is to minimize losses in the case of arbitrage or extreme price fluctuations. The 1inch exchange employs Mooniswap as a core component of its service. As such, the Mooniswap contract suite is responsible for token swaps and exchanges, as well as pool liquidity management operations with four functions (*Swap*, *Deposit*, *Withdraw* and *Deploy*) which access a number of internal functions that deal with ancillary AMM operations.

This audit comprised the entirety of the core Mooniswap functionality and began after a review of the Mooniswap's AMM design whitepaper. The functionality of the Mooniswap contracts was found to match that which was described in the provided whitepaper. The Mooniswap contracts were manually checked for security and in terms of correctly exposing interfaces, closely following the provided documentation. No security issues were identified in any of the contracts placed in scope.

Minor potential issues were identified and discussed with the Mooniswap team. This led to the discussions that ultimately ruled out these flaws. The results of these discussions were included in the Test Methodology section of this report. Overall, the 1inch Mooniswap smart contract stack appears to be a mature set of smart contracts. In light of this Cure53 2020 project, the smart contracts fit with a well-documented use-cases and design. From a security perspective, they should be considered ready for deployment.

Cure53 would like to thank Daria Melnik, Natalia Toporkova, Mikhail Melnik, Anton Bukov and Sergej Kunz from the 1inch team for their excellent project coordination, support and assistance, both before and during this assignment.