1Inch Mooniswap V2 & 1INCH Token

SMART CONTRACT AUDIT

09.02.2020

Made in Germany by Chainsulting.de

Table of contents

1. Disclaimer

The audit makes no statements or warrantees 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 1Inch Exchange. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (25.11.2020)	Layout
0.5 (02.12.2020)	Automated Security Testing Manual Security Testing
0.8 (02.12.2020)	Testing SWC Checks
1.0 (03.12.2020)	Summary and Recommendation
2.0 (03.12.2020)	Final document
2.1 (10.12.2020)	Recheck of codebase
2.5 (15.12.2020)	Verify findings
2.9 (23.12.2020)	Acknowledge of 3th party findings
3.0 (25.12.2020)	Final document
3.5 (19.01.2021)	Recheck of codebase
3.8 (20.01.2021)	Final document
4.0 (09.02.2021)	Added deployed Mooniswap V2 Factory

2. About the Project and Company

Company address:

1Inch Limited
Quijano Chambers, P.O. Box 3159, Road Town
Tortola, British Virgin Islands

Sergej Kunz Co-Founder & Chief Executive Officer
Anton Bukov Co-Founder & Chief Technology Officer

Website: https://mooniswap.exchange/

GitHub: https://github.com/CryptoManiacsZone

Twitter: https://twitter.com/mooniswap

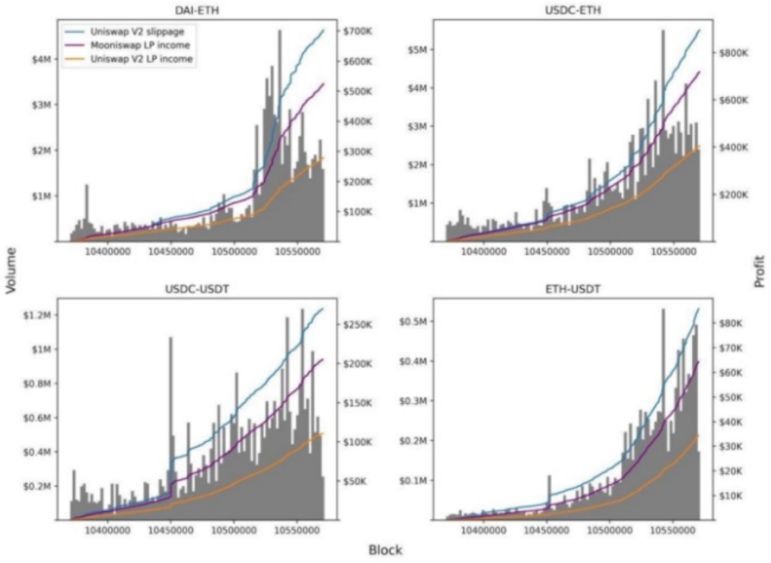
Discord: https://discord.gg/FZADkCZ

Youtube: https://www.youtube.com/channel/UCk0nvK4bHpteQXZKv7lkq5w

Medium: https://medium.com/@1inch.exchange

2.1 Project Overview

Mooniswap is an implementation of an “Automated Market Maker” (AMM), allowing exchange between two assets using an automated pricing mechanism based on balance ratios. While similar in approach to simple AMMs, such as Uniswap or Sushiswap, Mooniswap utilizes a „virtual“ pricing mechanism which limits the value arbitrageurs can extract from liquidity providers. Over the course of a fixed period of time, buy and sell prices continually converge on their prices based on the „real” underlying balance ratio. For example, a large buy order creates price movement which is not immediately reflected in the corresponding sell price. This new, better sell price is linearly converged on over the course of a fixed period of time, When a „sell” order is placed within this time window, the price will be between the price before and after the original buy transaction. After a fixed period of time with no additional trading, the price is based exclusively on the real underlying balance ratio.



Comparison of Uniswap V2 LP income with potential Mooniswap LP income on different pools

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

4.2 Used Code from other Frameworks/Smart Contracts (direct imports)

1. SafeMath.sol (0.6.0)
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/math/SafeMath.sol
2. Math.sol (0.6.0)
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/math/Math.sol
3. SafeERC20.sol (0.6.0)
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/SafeERC20.sol
4. ERC20.sol (0.6.0)
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol
5. IERC20.sol (0.6.0)
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/IERC20.sol
6. EnumerableSet.sol (0.6.0)
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/EnumerableSet.sol
7. Ownable.sol (0.6.0)
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol
8. ReentrancyGuard.sol (0.6.0)
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/ReentrancyGuard.sol

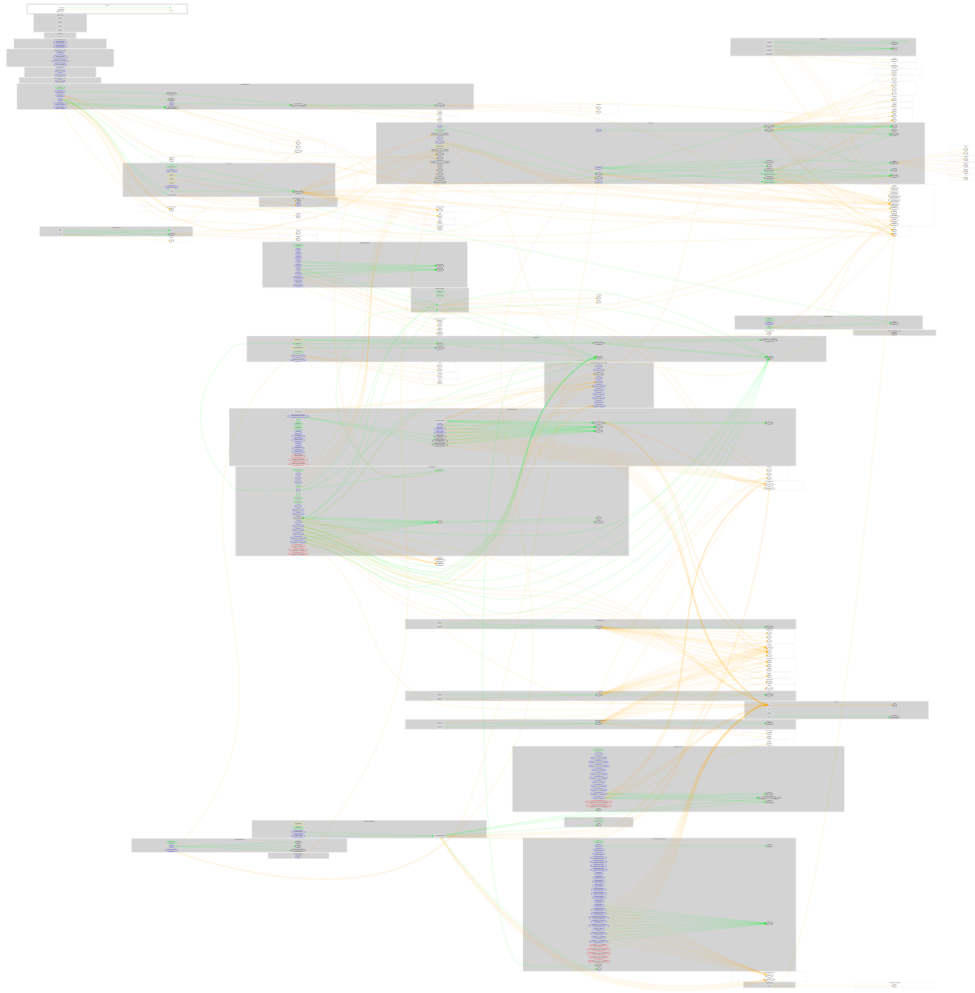
4.3 Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (SHA256)
governance/BaseGovernanceModule.sol	4ED1DEBD12D6573307B079974EBD48B4
governance/ExchangeGovernance.sol	13C1C2FCD1646B2B832240F77249D844
governance/GovernanceFeeReceiver.sol	B85CFD3A3BC0C141A16F3C5E7E91F7B9
governance/GovernanceRewards.sol	C97CCB659D83CB7D29B99399D9885C82
governance/MooniswapFactoryGovernance.sol	78BF7418E1558BE8D5C3C8E6A9108B58
governance/MooniswapGovernance.sol	CB5694E0B7690A9564C8C23DCA396467
farming/FarmingRewards.sol	509F21B85B9641C5F8C8847DBBE67220
inch/GovernanceMothership.sol	007D56E80B545EF47C357EE7D85C9D9D
interfaces/IExchangeGovernance.sol	B2C61C7F531C6748C9A7F282B37B632C
interfaces/IFeeCollector.sol	69C33052DED3F376B181C5207207AFE7
interfaces/IGovernanceModule.sol	5331C394F645331661AF1D2372F6E62B
interfaces/IMooniswapDeployer.sol	69C4AD756BDBBAB4611C473A7EBCD966
interfaces/IMooniswapFactory.sol	60965B31FD45F72093CF3B44AC569039
interfaces/IMooniswapFactoryGovernance.sol	B9F2AE13EF95F912C5C5F5AA8FF13621
interfaces/IReferralFeeReceiver.sol	75F922F7F5AA7F7F00B175C5F696317D
libaries/LiquidVoting.sol	B4BA774AA093B62685EEFB701E3ABDF1
libaries/ExchangeConstants.sol	878E3533BE8C03C300B52513200D4CCE
libaries/MooniswapConstants.sol	9DEF694FA410002FD1EB6562DAFB3596
libaries/ExplicitLiquidVoting.sol	C616934B732DFFA4A2DE87AFC01361D1
libaries/SafeCast.sol	61C41303A6A0FCB0AAEED06FD38ACDB1
libaries/Sqrt.sol	0DD38AA969B9E3DF73A05FB2D450CA03
libaries/UniERC20.sol	4984CA9FA5B8909BB8555988488790DA
libaries/VirtualBalance.sol	12FB84E4BB3DF0BB3222F1B2526408A4
libaries/VirtualVote.sol	357BC1150CE2A62D88BFE70DAB3A39CB

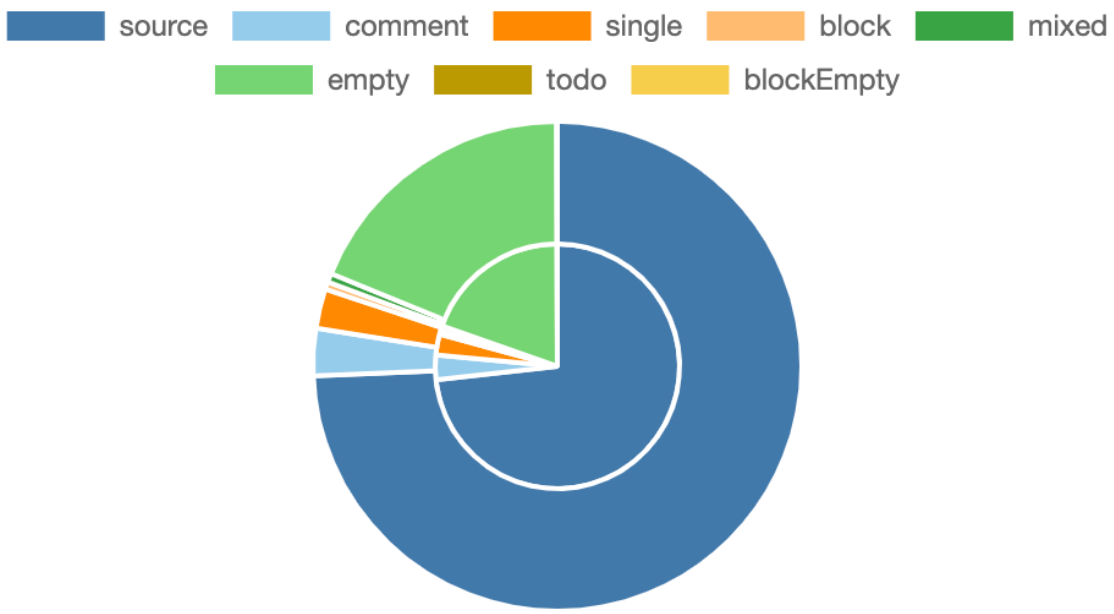
libaries/Vote.sol	4AED7A579C95004ABAA35636CA72E12C
libaries/Voting.sol	B9E713E9399E85D3600AB5912EB3F7CF
utils/BalanceAccounting.sol	298A4E024446FBEF2D1D215ADA472195
utils/Converter.sol	5298D343A9DFBD111FEC8494BE9D7A7A
utils/BaseRewards.sol	47042875DB3C467C4B69E63E328D0B3C
Mooniswap.sol	5E66E6A21B10F5D4C36908C248F7BB2E
MooniswapDeployer.sol	0BA27E5334C0D6445CB57737DB6DBF76
MooniswapFactory.sol	4A34D579B9A9738C3F18CE9064F7B52F
ReferralFeeReceiver.sol	ABE7C8C482F14E9DDA236C39AA7A1DEE

4.4 Metrics / CallGraph



Full Version: http://chainsulting.de/wp-content/uploads/2021/01/solidity-metrics-mooniswap.html

4.5 Metrics / Source Lines



4.6 Metrics / Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts
`^0.6.0` `^0.6.12`		`yes`	**** (0 asm blocks)	

📩 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧱 Uses Hash Functions	🪚 ECRecover	🌀 New/Create/Create2
`yes`					`yes` → `NewContract:Mooniswap`

4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/Mooniswap.sol	1		362	360	281	23	313	💰
🔍	contracts/interfaces/IMooniswapDeployer.sol		1	15	8	4	1	3	
🔍	contracts/interfaces/IMooniswapFactory.sol		1	10	8	4	1	7	
🔍	contracts/interfaces/IReferralFeeReceiver.sol		1	8	7	3	1	3	
🔍	contracts/interfaces/IFeeCollector.sol		1	9	7	3	1	5	
🔍	contracts/interfaces/IExchangeGovernance.sol		1	12	7	3	1	9	
🔍	contracts/interfaces/IGovernanceModule.sol		1	9	7	3	1	5	
🔍	contracts/interfaces/IMooniswapFactoryGovernance.sol		1	20	7	3	1	23	
🎨	contracts/governance/BaseGovernanceModule.sol	1		34	33	21	1	21	
📝	contracts/governance/ExchangeGovernance.sol	1		200	200	159	4	90	

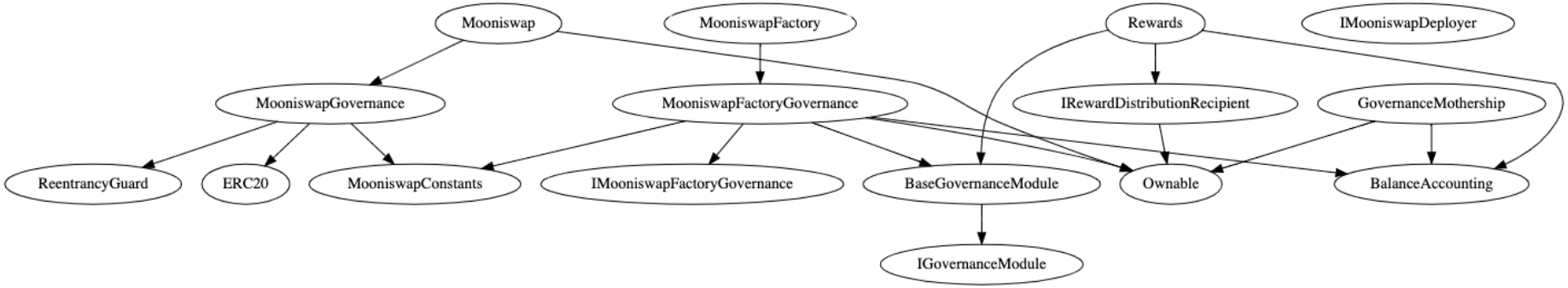
Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/governance/GovernanceFeeReceiver.sol	1		27	27	19	1	34	
📝	contracts/governance/GovernanceRewards.sol	1		19	19	11	2	11	
🎨	contracts/governance/MooniswapGovernance.sol	1		195	190	146	3	135	
📝	contracts/governance/MooniswapFactoryGovernance.sol	1		215	215	168	1	160	
📝	contracts/MooniswapDeployer.sol	1		25	19	14	1	16	🌀
📝	contracts/utils/Converter.sol	1		137	136	106	2	92	💰🔽
📝	contracts/utils/BaseRewards.sol	1		97	97	80	1	68	🔽
📝	contracts/utils/BalanceAccounting.sol	1		39	39	28	1	14	
📚	contracts/libraries/Sqrt.sol	1		23	23	18	2	4	
📚	contracts/libraries/LiquidVoting.sol	1		106	78	61	1	33	
📚	contracts/libraries/SafeCast.sol	1		25	25	19	1	9	
📚	contracts/libraries/VirtualBalance.sol	1		42	42	31	1	19	
📚	contracts/libraries/Voting.sol	1		95	67	50	1	29	

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📚	contracts/libraries/ExplicitLiquidVoting.sol	1		101	76	59	1	31	
📚	contracts/libraries/UniERC20.sol	1		109	109	90	2	83	🔹
📚	contracts/libraries/ExchangeConstants.sol	1		11	11	7	5	5	
📚	contracts/libraries/MooniswapConstants.sol	1		22	22	15	7	13	
📚	contracts/libraries/VirtualVote.sol	1		27	27	19	1	11	
📚	contracts/libraries/Vote.sol	1		44	44	34	1	6	
📝	contracts/ReferralFeeReceiver.sol	1		220	220	175	11	145	🔹
📝	contracts/MooniswapFactory.sol	1		70	70	54	1	36	
📝	contracts/inch/farming/FarmingRewards.sol	1		170	170	131	1	138	🔹
📝	contracts/inch/GovernanceMothership.sol	1		94	94	72	1	91	🔹
📝📚 🔍🎨	**Totals**	**26**	**7**	**2592**	**2464**	**1891**	**84**	**1662**	💰🔹🌀

5. Scope of Work

The 1inch exchange team provided us with the files that needs to be tested. The scope of the audit is the Mooniswap V2 contracts with instant governance and the 1INCH native token (ERC20).

Following contracts with the direct imports been tested
MooniswapFactory.sol
Mooniswap.sol
inch/GovernanceMothership.sol



5.1 Manual and Automated Vulnerability Test

CRITICAL ISSUES
During the audit, Chainsulting's experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES
During the audit, Chainsulting's experts found **no High issues** in the code of the smart contract.

MEDIUM ISSUES
5.1.1 Cannot pause / shutdown Mooniswap Factory
Severity: MEDIUM
Status: FIXED (https://github.com/1inch-exchange/mooniswap-v2/commit/648157bfbe3ebaad1acf5de6df9a9add9a1ef448)
File(s) affected: Mooniswap.sol
Author: k06a

Attack / Description	Code Snippet	Result/Recommendation
The Mooniswap.sol contract needs a mechanism for the owner to pause the contract. This disables the swap functionality. However, there is no corresponding mechanism to shutdown the contract. Consider introducing a mechanism for the owner to pause and unpause the contract. This will be important in case of emergency or fraudulent activities.	NA	We suggest to add a shutdown() method to the Mooniswap factory

5.1.2 Potential violation of Checks-Effects-Interaction pattern
Severity: MEDIUM
Status: FIXED (https://github.com/1inch-exchange/mooniswap-v2/commit/961b4116f35a954c30435273da0cf6dba34cf053)
File(s) affected: ReferralFeeReceiver.sol
Author: ZumZoom

Attack / Description	Code Snippet	Result/Recommendation
Potential violation of Checks-Effects-Interaction pattern. Could potentially lead to re-entrancy vulnerability.	Line: 10 contract ReferralFeeReceiver is IReferralFeeReceiver, Converter Line: 52 function freezeEpoch(Mooniswap mooniswap) external validPool(mooniswap) validSpread(mooniswap) { Line: 66 function trade(Mooniswap mooniswap, IERC20[] memory path) external validPool(mooniswap) validPath(path) Line: 129 function claimCurrentEpoch(Mooniswap mooniswap) external validPool(mooniswap) Line 141: function claimFrozenEpoch(Mooniswap mooniswap) external validPool(mooniswap)	OpenZeppelin has it's own mutex implementation you can use called ReentrancyGuard. This library provides a modifier you can apply to any function called nonReentrant that guards the function with a mutex. View the source code for the OpenZeppelin ReentrancyGuard library here: https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/utils/ReentrancyGuard.sol Keep in mind that a nonReentrant function should be external. If another function calls the nonReentrant function it is no longer protected.

LOW ISSUES

5.1.3 Potential front running attack or losing of allowance
Severity: LOW
Status: Acknowledged
File(s) affected: ERC20Permit.sol (1INCH Token)

Attack / Description	Code Snippet	Result/Recommendation
The _approve method replaces the allowance, so there are two potential problems here: 1. If a signer wants to increase the allowance from A to B , a receiver may withdraw A+B using the front-running attack. 2. If a signer wants to send A and B , but a receiver forgot to withdraw A , the receiver will lose ability to withdraw A .	Line: 53 _approve(owner, spender, amount);	We suggest to add permitIncrease, permitDecrease methods and use it instead of the permit .

5.1.4 Mismatch of argument name in _PERMIT_TYPEHASH and permit function
Severity: LOW
Status: FIXED (https://github.com/1inch-exchange/1inch-token/commit/5332caa9b91403a022e74b49c9d0fc9c6d5419f4)
File(s) affected: ERC20Permit.sol (1INCH Token)
Author: ZumZoom

Attack / Description	Code Snippet	Result/Recommendation

At the line ERC20Permit.sol#L27 the value argument is used but at the line ERC20Permit.sol#L32 the argument name is amount .	Line: 27
	bytes32 private immutable _PERMIT_TYPEHASH = keccak256("Permit(address owner,address spender,uint256 value,uint256 nonce,uint256 deadline)");
	
	Line: 32
	function permit(address owner, address spender, uint256 amount, uint256 deadline, uint8 v, bytes32 r, bytes32 s) public virtual override {
		We suggest to rename amount to value in the permit function.

5.1.5 Use SafeCast library for uintXX/intXX wrapper
Severity: LOW
Status: FIXED (https://github.com/1inch-exchange/mooniswap-v2/commit/6276231d02ded3c44863b23fc04a77356671108a)
File(s) affected: MooniswapFactoryGovernance.sol, MooniswapGovernance.sol, ExplicitLiquidVoting.sol, LiquidVoting.sol, VirtualBalance.sol
Author: ZumZoom

Attack / Description	Code Snippet	Result/Recommendation
Wrappers over Solidity's uintXX/intXX casting operators with added overflow checks. Downcasting from uint256/int256 in Solidity does not revert on overflow. This can easily result in undesired exploitation or bugs, since developers usually assume that overflows raise errors. 'SafeCast' restores this intuition by reverting the transaction when such an operation overflows. Using this library instead of the unchecked operations eliminates an entire class of bugs, so it's recommended to use it always. Can be combined with {SafeMath} and {SignedSafeMath} to extend it to smaller types, by performing	Line: NA `using SafeCast for uint256;`	Consider to use SafeCast

all math on 'uint256' and 'int256' and then downcasting.		

INFORMATIONAL ISSUES

5.1.6 Missing natspec documentation
Severity: INFORMATIONAL
Status: Acknowledge
File(s) affected: all

Attack / Description	Code Snippet	Result/Recommendation
Solidity contracts can use a special form of comments to provide rich documentation for functions, return variables and more. This special form is named the Ethereum Natural Language Specification Format (NatSpec).	NA	It is recommended to include natspec documentation and follow the doxygen style including @author, @title, @notice, @dev, @param, @return and make it easier to review and understand your smart contract. The team addressed the issue while auditing and added more documentation parts. https://github.com/1inch-exchange/mooniswap-v2/pull/13/commits

5.1.7 Fix Spelling and Grammatical Errors

Severity: INFORMATIONAL
Status: FIXED
File(s) affected: ReferralFeeReceiver.sol

Attack / Description	Code Snippet	Result/Recommendation

Language mistakes were identified in the messages in the codebase. Fixing these mistakes can help improve the end-user experience by providing clear information on errors encountered, and improve the maintainability and auditability of the codebase.	Line 53:
	`require(info.lastUnprocessedEpoch == currentEpoch, "Previous epoch is not finlazed");`
	
	Line 138:
	`require(_userInfo.lastUnprocessedEpoch[mooniswap] == lastUnprocessedEpoch, "Epoch funcds already claimed");`
		`require(info.lastUnprocessedEpoch == currentEpoch, "Previous epoch is not finalized");`
		
		`require(_userInfo.lastUnprocessedEpoch[mooniswap] == lastUnprocessedEpoch, "Epoch funds already claimed");`

5.1.8 A floating pragma is set
Severity: INFORMATIONAL
Code: SWC-103
Status: Acknowledge
File(s) affected: all

Attack / Description	Code Snippet	Result/Recommendation
The current pragma Solidity directive is `^0.6.12;` It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.	Line: 1 `pragma solidity ^0.6.12;`	It is recommended to follow the example (0.6.12), as future compiler versions may handle certain language constructions in a way the developer did not foresee. Not effecting the overall contract functionality.

5.1.9 "dist:factory": "truffle-flattener Mooni* (rename)
Severity: INFORMATIONAL
Code: NA
Status: FIXED
File(s) affected: packages.json

Attack / Description	Code Snippet	Result/Recommendation
Failed at the @1inch/mooniswap@0.0.1 dist:factory script.	Line: 43 `"dist:factory": "truffle-flattener ./contracts/MooniFactory.sol | awk '/SPDX-License-Identifier/&&c++>0 {next} 1' > ./MooniFactory.full.sol && solcjs --bin --abi --optimize ./MooniFactory.full.sol && mv ./__MooniFactory_full_sol_MooniFactory.bin ./MooniFactory.full.bin && mv ./__MooniFactory_full_sol_MooniFactory.abi ./MooniFactory.full.abi && rm ./*_sol_*"` }}	Rename Mooni* to Mooniswap*

6. Executive Summary

The overall code quality of the project is very good, not overloaded with unnecessary functions and it is accompanied by unit tests, these is greatly benefiting the security of the contract. It correctly implemented widely-used and reviewed contracts from OpenZeppelin and for safe mathematical operations.

The main goal of the audit was to verify the claims regarding the security of the smart contract. During the audit, no critical or minor issues were found after the manual and automated security testing. The previous audits of mooniswap greatly benefiting the outcome, as the development team adopted the findings in the previous reports. It is recommended to include natspec documentation and follow the doxygen style including @author, @title, @notice, @dev, @param, @return and make it easier to review and understand your smart contract.

Previous Audits of Mooniswap in 2020
https://mooniswap.exchange/docs/mooniswap-audit-report-2.pdf
https://mooniswap.exchange/docs/mooniswap-audit-report-3.pdf

7. Deployed Smart Contract

VERIFIED

1INCH Token
0x111111111117dc0aa78b770fa6a738034120c302

VERIFIED

MOONISWAP V2
0xbAF9A5d4b0052359326A6CDAb54BABAa3a3A9643