MOONISWAP V2 SMART CONTRACT SECURITY AUDIT

January 22, 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 Mooniswap (name of Client). 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

Mooniswap is the next generation of an automated market maker with virtual balances — enabling liquidity providers to capture profits otherwise captured by arbitrageurs

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 audited scope includes a smart contracts set which is a part of Mooniswap v2 project's swap mechanism. The project uses the UniERC20 wrapper to provide unified interface to ETH and ERC20 tokens. LiquidVoting, MooniswapGovernance, VirtualBalance, Mooniswap provide a swap mechanism. Converter implements methods to make swaps.

1.5 PROJECT DASHBOARD

Client	Mooniswap
Audit name	v2
Initial version	355ee41415aa4ab97cccb18b77004caa03c07ac3
Final version	b9c06335fb1d68b19054442547fc677f42795b44
SLOC	1145
Date	2020-12-03 - 2021-01-22
Auditors engaged	3 auditors

FILES LISTING

MooniswapDeployer.sol	MooniswapDeployer.sol
MooniswapConstants.sol	MooniswapConstants.sol
ReferralFeeReceiver.sol	ReferralFeeReceiver.sol
Mooniswap.sol	Mooniswap.sol
MooniswapFactory.sol	MooniswapFactory.sol
GovernanceMothership.sol	GovernanceMothership.sol
Vote.sol	Vote.sol
LiquidVoting.sol	LiquidVoting.sol
VirtualBalance.sol	VirtualBalance.sol
Sqrt.sol	Sqrt.sol
UniERC20.sol	UniERC20.sol
IRewardDistributionRecipient.sol	IRewardDistributionRecipient.sol
Rewards.sol	Rewards.sol
BaseGovernanceModule.sol	BaseGovernanceModule.sol
MooniswapGovernance.sol	MooniswapGovernance.sol
GovernanceFeeReceiver.sol	GovernanceFeeReceiver.sol
MooniswapFactoryGovernance.sol	MooniswapFactoryGovernance.sol
IMooniswapFactoryGovernance.sol	IMooniswapFactoryGovernance.sol
IReferralFeeReceiver.sol	IReferralFeeReceiver.sol
IGovernanceModule.sol	IGovernanceModule.sol
IMooniswapDeployer.sol	IMooniswapDeployer.sol
Converter.sol	Converter.sol
BalanceAccounting.sol	BalanceAccounting.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	3
Warning	10
Comment	5

CONCLUSION

Smart contracts have been audited and several suspicious places have been spotted. During the audit no critical issues were found, three 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, contracts are assumed as secure to use according to our security criteria.

2.FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

MJR-1	Denial of Service bug during ETH transfer
File	UniERC20.sol
Severity	Major
Status	Acknowledged

DESCRIPTION

At lines UniERC20.sol#L29 and UniERC20.sol#L42 not all recipients are addresses or contracts with minimal fallback functions. Since the transfer and send function forwards exactly 2300 gas to the recipient, a contract with a fallback function will throw an out-of-gas exception. The intention of this upper limit is to prevent reentrancy vulnerabilities, but it requires gas usage to be constant.

RECOMMENDATION

It is recommended to use call instead of transfer/send to avoid possible DoS.
For example like this:

```
(bool success, ) = to.call{value: amount}("");
require(success, "Transfer failed");
```

MJR-2	Unlimited access to change token balances for any users
File	GovernanceMothership.sol
Severity	Major
Status	Fixed at 2ce549dc

DESCRIPTION

At line GovernanceMothership.sol#L48 the `notifyFor()` function is external and has no access modifiers.

Calling these function affects the token balances of any other users.
An attacker can block the project by calling these functions and continuously changing the token balances of other users.

RECOMMENDATION

It is necessary to make the following at line 49:

```
_notifyFor(account, balanceOf(account));
```

MJR-3	It is possible to carry out attacks to manipulate pools within one transaction using a flash loan
File	GovernanceMothership.sol
Severity	Major
Status	Acknowledged

DESCRIPTION

In contract GovernanceMothership.sol, any user can exchange tokens with a contract. An attacker can take a flash loan and perform multiple liquidity manipulations within a single transaction. These manipulations can lead to the loss of funds for other users.

RECOMMENDATION

It is recommended to add protection against token manipulation with flash loans. Here's some sample code:

```
mapping(address => uint256) private _lastSwapBlock;

function some() external {
   _preventSameTxOrigin();
   ....
   some logic
   ...
}

function _preventSameTxOrigin() private {
   require(block.number > _lastSwapBlock[tx.origin], "SAME_TX_ORIGIN");
   _lastSwapBlock[tx.origin] = block.number;
}
```

CLIENT'S COMMENTARY

> The current project modules have a logic "smeared" in time. So the problem is not relevant. In future if we add a module with logic that can be affected by flash attacks we'll implement protection inside this module.

2.3 WARNING

WRN-1	No validation of address parameter value in contract constructor
File	MooniswapFactory.sol MooniswapGovernance.sol BaseGovernanceModule.sol GovernanceFeeReceiver.sol Converter.sol GovernanceMothership.sol Rewards.sol Mooniswap.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

The variable is assigned the value of the constructor input parameter. But this parameter is not checked before this. If the value turns out to be zero, then it will be necessary to redeploy the contract, since there is no other functionality to set this variable:

- At line MooniswapFactory.sol#L27 the `poolOwner` variable is set to the value of the `_poolOwner` input parameter. The `mooniswapDeployer` variable is set to the value of the `_mooniswapDeployer` input parameter.
- At line MooniswapGovernance.sol#L27 the `mooniswapFactoryGovernance` variable is set to the value of the `_mooniswapFactoryGovernance` input parameter.
- At line BaseGovernanceModule.sol#L18 the `mothership` variable is set to the value of the `_mothership` input parameter.
- At line GovernanceFeeReceiver.sol#L18 the `rewards` variable is set to the value of the `_rewards` input parameter.
- At line Converter.sol#L25 the `targetToken` variable is set to the value of the `_targetToken` input parameter.
- At line GovernanceMothership.sol#L25 the `inchToken` variable is set to the value of the `_inchToken` input parameter.
- At line Rewards.sol#L43 the `gift` variable is set to the value of the `_gift` input parameter.
- At line Mooniswap.sol#L94 the `token0` variable is set to the value of the `_token0` input parameter and the `token1` variable is set to the value of the `_token1` input parameter.

RECOMMENDATION

It is necessary to add a check that the address does not contain a zero value.

WRN-2	Gas overflow during iteration — DoS
File	GovernanceMothership.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

Each iteration of the cycle requires a gas flow. A moment may come when more gas is required than it is allocated for recording one block.
In this case, all iterations of the loop will fail:

- At line GovernanceMothership.sol#L58
- At line GovernanceMothership.sol#L75

RECOMMENDATION

We recommend adding a check for the maximum possible number of elements of the `maxAmounts` array.

WRN-3	Successful token transfer operation is not checked
File	Converter.sol GovernanceMothership.sol ReferralFeeReceiver.sol UniERC20.sol Rewards.sol
Severity	Warning
Status	**No issue**

DESCRIPTION

According to the ERC-20 specification, the token transfer functions return a boolean value. But in the source code, this value is not checked:

- At line Converter.sol#L105
- At line GovernanceMothership.sol#L31
- At line GovernanceMothership.sol#L39
- At line ReferralFeeReceiver.sol#L115
- At line ReferralFeeReceiver.sol#L127
- At line ReferralFeeReceiver.sol#L155
- At line ReferralFeeReceiver.sol#L163
- At line ReferralFeeReceiver.sol#L171
- At line UniERC20.sol#L31
- At line UniERC20.sol#L45
- At line Rewards.sol#L85

RECOMMENDATION

It is necessary to add processing in case the token transfer fails.

WRN-4	There is no event record about minting and burning of tokens
File	BalanceAccounting.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At line BalanceAccounting.sol#L22 the `_mint()` function is for minting new tokens.
But there is no record of the token transfer event:

```
emit Transfer(address(0), account, amount);
```

At line BalanceAccounting.sol#L27 the `_burn()` function is designed to burn existing tokens.
But there is no record of the token transfer event:

```
emit Transfer(account, address(0), amount);
```

RECOMMENDATION

It is recommended that you add a record of these events.

WRN-5	Possible loss of tokens
File	BalanceAccounting.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At line BalanceAccounting.sol#L24 when minting new tokens, the wallet address to which tokens will be transferred is not checked. It may not be an existing address equal to zero.

RECOMMENDATION

It is recommended to add verification of the wallet address:

```
require(account != address(0), "mint to the zero address
```

WRN-6	It is necessary to check the correctness of the variable value
File	MooniswapDeployer.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At line MooniswapDeployer.sol#L23 the `deploy()` function sets the `poolOwner` owner for the pool. But if the address is zero, then it will not be possible to work with the created pool.

RECOMMENDATION

It is recommended adding a check of the current value of the `poolOwner` variable for zero:

```
require(poolOwner != address(0), "burn from the zero address");
```

WRN-7	There is no check of the amount of ether before calling the paid function
File	UniERC20.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

- At line UniERC20.sol#L29 in the `uniTransfer()` function, ether is transferred from the balance of the contract to the `to` address. But on the balance of the contract there may not be enough ether to perform these functions and the `uniTransfer()` function will not be executed.

- At line UniERC20.sol#L42 in the `uniTransferFrom()` function, ether is transferred from the balance of the contract to the `from` address. But on the balance of the contract there may not be enough ether to perform these functions and the `uniTransferFrom()` function will not be executed.

RECOMMENDATION

It is recommended adding such a check.

WRN-8	Function does not always return a value
File	Mooniswap.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

- At line Mooniswap.sol#L314 function `_getReturn ()` must return a value of type `uint256`. But it returns a value only if the condition on line 318 is met.

RECOMMENDATION

It is recommended that you modify this function so that it always returns a value.

WRN-9	Initialization potential overflow
File	Mooniswap.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At lines Mooniswap.sol#L239 and Mooniswap.sol#L240 in the `swapFor()` function, the values are added to the `volumes[src].confirmed` and `volumes[src].result` variables.
Normal addition is used for the `uint128` data type.
In this case, an overflow may occur and an unexpected value of the variables will be obtained.

RECOMMENDATION

It is recommended to use functions from the SafeMath Library.

WRN-10	Require the argument to not break the `Vote.Data`
File	Vote.sol#23L
Severity	Warning
Status	Acknowledged

DESCRIPTION

If you call the `init` function with `vote = type(uint256).max`, `value = vote + 1` will overflow the `uint256` and will be set to `0` and will be handled as a default: Vote.sol#23L

RECOMMENDATION

In this project this is not a big problem because in every usage of `Vote.init` the code require `vote < SOME_MAX`, but anyway we suggest adding require `vote < type(uint256).max`.

2.4 COMMENTS

CMT-1	Using "magic" numbers
File	UniERC20.sol Mooniswap.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

The use in the source code of some unknown where taken values impair its understanding:

- At line UniERC20.sol#L55 and
  at line UniERC20.sol#L59
  the value is `20000`.
- At line UniERC20.sol#L66 and
  at line UniERC20.sol#L73
  the value is `0x20`.
- At line UniERC20.sol#L73
  the value is `0x7E`.
- At line UniERC20.sol#L101 and
  at line UniERC20.sol#L102
  the values are `48` and `39`.
- At line Mooniswap.sol#L139
  the value is `99`.

RECOMMENDATION

It is recommended that you create constants with meaningful names for using numeric values.

CMT-2	Add protection against attacks like Reentrancy
File	UniERC20.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At lines UniERC20.sol#L29 and UniERC20.sol#L42 this is where the ETH is transferred in the library functions `uniTransfer()` and `uniTransferFrom()`. Now, other contracts that call these functions in external functions have added the `nonReentrant` access modifier. However, there is a possibility that in other versions of the code the programmer, when working with these functions, will forget to add this access modifier. In this case, his contract would be vulnerable.

RECOMMENDATION

It is recommended to add the `nonReentrant` access modifier for the library functions `uniTransfer()` and `uniTransferFrom()`.

CMT-3	Avoid complex one-liners
File	Mooniswap.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

A one-liner can make simple logic quite difficult to read and verify. It's better to expand them to ifs: Mooniswap.sol#L134

RECOMMENDATION

We recommend to expand one-liner to several ifs.

CMT-4	Mixed solidity versions
File	LiquidVoting.sol MooniswapFactory.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

Somewhere it's `^0.6.12`, somewhere it's `^0.6.0`:

- LiquidVoting.sol#L3 (0.6.12)
- MooniswapFactory.sol#L3 (0.6.0)

It's better to freeze the version to the selected one.

RECOMMENDATION

We suggest using the same solidity version

CMT-5	Change the visibility of a variable
File	MooniswapFactory.sol
Severity	Comment
Status	**No issue**

DESCRIPTION

Line MooniswapFactory.sol#L23 has a public variable `allPools` declared. To view it, there is a function `getAllPools()` at line MooniswapFactory.sol#L32.

Public variable will be more expensive than a private one in terms of gas. If you change the visibility of a variable to private, then the cost will be lower.

RECOMMENDATION

We recommend changing the visibility of the `allPools` variable to private.

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