PUBLIC REPORT

Mooniswap Security Audit

peppersec

Foreword

Status of a vulnerability or security issue

Clarity is a rare commodity. That is why for the convenience of both the client and the reader, we have introduced a system of marking vulnerabilities and security issues we discover during our security audits.

No issue

Let's start with an ideal case. If an identified security imperfection bears no impact on the security of our client, we mark it with the label.

✓ Fixed

The fixed security issues get the label that informs those reading our public report that the flaws in question should no longer be worried about.

Addressed

In case a client addresses an issue in another way (e.g., by updating the information in the technical papers and specification) we put a nice tag right in front of it.

Acknowledged

If an issue is planned to be addressed in the future, it gets the tag, and a client clearly sees what is yet to be done.

Although the issues marked "Fixed" and "Acknowledged" are no threat, we still list them to provide the most detailed and up-to-date information for the client and the reader.

Severity levels

We also rank the magnitude of the risk a vulnerability or security issue pose. For this purpose, we use 4 "severity levels" namely:

1. Minor
2. Medium
3. Major
4. Critical

More details about the ranking system as well as the description of the severity levels can be found in Appendix 1. Terminology.

TABLE OF CONTENTS

01. Introduction

SCOPE OF WORK

The scope of the security audit comprised the Mooniswap smart contract located at the commit #3a6fbf00faed063fa1221f614b4d567ddf3ee620

SECURITY ASSESSMENT METHODOLOGY

The smart contract's code is scanned both manually and automatically for known vulnerabilities and logic errors that can lead to potential security threats. The conformity of requirements (e.g., White Paper) and practical implementations are reviewed as well on a consistent basis. More about the methodology can be found here.

AUDITORS

1. PepperSec

02. Discovered issues

CRITICAL ISSUES

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

MAJOR ISSUES

During the audit, PepperSec's experts found **no Major issues** in the code of the smart contract.

MEDIUM ISSUES

1 Deposits to pools with tokens with fee can withdraw additional token amount

Severity: **MEDIUM**

The Mooniswap does its best to avoid issues with unusual ERC20 tokens (that have a transfer fee). However, there is still a possible case when a depositor loses some token amount.

For demonstration purposes, here is an example.

1. Token A - takes fee on each transfer (user sends 10 tokens, recipient recieves 9)
2. Token B - Native ETH

The depositor provides tokens in the [ETH, A] order. In this example above, the user sends 10 ETH and 10 A tokens (assuming 1/1 rate). According to the code, the pool gets 10ETH, 9A and mint 9 Mooniswap tokens (shares). As a result, the user loses 1 ETH.

Recommendation:

1. Consider effectively handling the case or notify users about it.

Status:

Acknowledged

2 Proxy factory

Severity: **MEDIUM**

Under the current implementation of Factory.sol, the whole Mooniswap contract is deployed for each pool, which requires a lot of Gas.

Recommendation:

1. Consider implementing a Factory that deploys a minimal viable proxy contract, which proxies all calls to single Mooniswap contract implementation.

Status:

Acknowledged

MINOR ISSUES

1 Native ETH support expands attack surface

Severity: **MINOR**

The Mooniswap contract supports both native ETH and ERC20 tokens. Although it saves some Gas, consider using WETH with a special proxy (router) contract that handles the Ether wrap/unwrap. On the UI side, there would be similar behavior. On the smart-contract side, it will decrease the code complexity and reduce attack surface.

Recommendation:

1. Consider using WETH instead of ETH.

Status:

Acknowledged

2 Using immutable keyword

Severity: **MINOR**

The Factory contract is stored in Storage and requires SLOAD upon each interaction. Using the immutable keyword will reduce the cost of Gas.

Recommendation:

1. Consider using immutable for the Factory address.

Status:

Acknowledged

Conclusion

In comparison to other AMM protocols, the Mooniswap protocol has a significant advantage. It utilizes arbitrage to provide actual market prices and enables liquidity providers to earn more interest than other AMMs. In the worst-case scenario, Mooniswap pools serve as Uniswap pools in terms of swaps and collecting fees.

In addition to this:

1. The product of token balances does not decrease, supposing these balances are not externally decreased. However, users' funds are still subject to impermanent losses.
2. Deposits and withdrawals to the imbalanced pool may lead to value losses (not reducing the product).
3. The pool should capture more value performing trades based on virtual balances comparing to trades based on real balances.

During the audit, the experts proposed some style and architecture impovements. There were **no Critical** or **Major** issues found.

Appendix 1. Terminology

1 Severity

Assessment of the magnitude of an issue.

		Severity		
Impact	Major	Medium	Major	Critical
	Medium	Minor	Medium	Major
	Minor	None	Minor	Medium
		Minor	Medium	Major
		Likelihood of exploitation		

MINOR

Minor issues are generally subjective in nature or potentially associated with topics like “best practices” or “readability”. As a rule, minor issues do not indicate an actual problem or bug in the code. The maintainers should use their own judgment as to whether addressing these issues will improve the codebase.

MEDIUM

Medium issues are generally objective in nature but do not represent any actual bugs or security problems. These issues should be addressed unless there is a clear reason not to.

MAJOR

Major issues are things like bugs or vulnerabilities. These issues may be unexploitable directly or may require a certain condition to arise to be exploited. If unaddressed, these issues are likely to cause problems with the operation of the contract or lead to situations which make the system exploitable.

CRITICAL

Critical issues are directly exploitable bugs or security vulnerabilities. If unaddressed, these issues are likely or guaranteed to cause major problems or, ultimately, a full failure in the operations of the contract.

About Us

Worried about the security of your project? You're on the right way! The second step is to find a team of seasoned cybersecurity experts who will make it impenetrable. And you've just come to the right place.

PepperSec is a group of whitehat hackers seasoned by many-year experience and have a deep understanding of the modern Internet technologies. We're ready to battle for the security of your project.

LET'S KEEP IN TOUCH


peppersec.com


hello@peppersec.com


Github