AntiMatter Audit Report

Version 1.0.1

Serial No. 2021042000022015

Presented by Fairyproof

April 20, 2021

FAIRYPROOF

01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the AntiMatter project, at the request of the AntiMatter team.

Audited Code's Github Repository:

https://github.com/antimatter-finance/contracts

Audited Code's Github Commit Number:

50f3bd8e0feb86d92c1146d9ab4bb8c762e5992f

Audited Contract Files' Onchain Address:

N/A

Audited Contract Files:

The contract file audited is only one of the files with the extension "sol" under the "contracts" directory as follows:

```
contracts/
├── Mining.sol
└── PerpetualOption.sol
```

Attention: this audit only covered the Mining.sol and PerpetualOption.sol files and all the other files such as MATTER.sol were not covered by this audit.

The goal of this audit is to review AntiMatter's solidity implementation for its option trading, algorithm for calculating returns and liquidity mining, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

We make observations on specific areas of the code that present concrete problems, as well as general observations that traverse the entire codebase horizontally, which could improve its quality as a whole.

— Disclaimer

Note that as of the date of publishing, the contents of this report reflect the current understanding of known security patterns and state of the art regarding smart contract security. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk.

The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. Risks or issues introduced by using data feeds from offchain sources are not extended by this review either.

Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit is recommended after the issues covered are fixed.

To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

— Methodology

The above files' code was studied in detail in order to acquire a clear impression of how the its specifications were implemented. The codebase was then subject to deep analysis and scrutiny, resulting in a series of observations. The problems and their potential solutions are discussed in this document and, whenever possible, we identify common sources for such problems and comment on them as well.

The Fairyproof 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 Fairyproof to make sure we understand the size, scope, and functionality of the project's smart contracts.
   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 Fairyproof 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 the test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

— Structure of the document

This report contains a list of issues and comments on all the above contract files. Each issue is assigned a severity level based on the potential impact of the issue and recommendations to fix it, if applicable. For ease of navigation, an index by topic and another by severity are both provided at the beginning of the report.

— Documentation

For this audit, we used the following sources of truth about how the AntiMatter system should work:

https://antimatter.finance/

https://antimatter-finance.github.io/

These were considered the specification, and when discrepancies arose with the actual code behavior, we consulted with the AntiMatter team or reported an issue.

— Comments from Auditor

No vulnerabilities with critical, high or low-severity were found in the above contract files.

One vulnerability with medium-severity was found in the above contract files.

02. About Fairyproof

Fairyproof is a leading technology firm in the blockchain industry, providing consulting and security audits for organizations. Fairyproof has developed industry security standards for designing and deploying smart contract systems.

03. Introduction to AntiMatter

AntiMatter is an onchain DeFi derivative protocol that provides trading of tokenized perpetual call and put options. The team plans to deploy it on Ethereum first and then to Binance Smart Chain and Polkadot.

04. Major functions of audited code

The audited code implements its option trading, algorithm for calculating returns and liquidity mining functions.

Note: some contract files can be migrated or upgraded. Whenever there is a contract migration or upgrade, an audit should be conducted to mitigate vulnerabilities. The liquidity mining function has a blacklist. if a user is added to the blacklist he/she will not be able to withdraw his/her rewards but can only withdraw his/her staked assets

05. Key points in audit

During the audit, we worked closely with the AntiMatter team, checked the possible vulnerabilities associated with asset security, access control, token issurance etc.

06. Coverage of issues

The issues that the Fairyproof team covered when conducting the audit include but are not limited to the following ones:

- Re-entrancy Attack
- DDos Attack
- Integer Overflow
- Function Visibility
- Logic Vulnerability
- Uninitialized Storage Pointer
- Arithmetic Precision
- Tx.origin
- Shadow Variable
- Design Vulnerability

- Token Issurance
- Asset Security
- Access Control

07. Severity level reference

Every issue in this report was assigned a severity level from the following:

Critical severity issues need to be fixed as soon as possible.

High severity issues will probably bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

08. Major areas that need attention

Based on the provided contract files the Fairyproof team focused on the possible issues and risks related to the following functions or areas.

- Integer Overflow/Underflow

We checked all the code sections, which have arithmetic operations and might introduce integer overflow or underflow if no safe libraries are used. All of them use safe libraries.
We didn't find issues or risks in this function or area at the time of writing.

- Setting of Transaction Fees

We checked whether or not the setting of transaction fees is correct and appropriate.
We didn't find issues or risks in this function or area at the time of writing.

- Algorithm for Option Trading

We checked whether or not the algorithm for option trading matches what is described in the project's white paper and whether or not there might be mistakes.
We didn't find issues or risks in this function or area at the time of writing.

- Access Control

We checked each of the functions that can modify a state, especially those functions that can only be accessed by "owner".
We didn't find issues or risks in this function or area at the time of writing.

- Token Issurance

We checked whether or not the contract files can mint tokens at will.
We didn't find issues or risks in this function or area at the time of writing.

- State Update

We checked some key state variables which should only be set at initialization.
We didn't find issues or risks in this function or area at the time of writing.

- Asset Security

We checked whether or not all the functions that transfer assets are safely hanlded.
We didn't find issues or risks in this function or area at the time of writing.

- Contract Migration/Upgrade

We checked whethere the contract files could be migrated or upgraded.

We found an issue. For more detailes please refer to section 11.

- Miscellaneous

We didn't find issues or risks in other functions or areas at the time of writing.

09. List of issues by severity

A. Critical

- N/A

B. High

- N/A

C. Medium

- PerpetualOption.sol

Contract Migration/Upgrade

D. Low

- N/A

10. List of issues by contract file

- PerpetualOption.sol

Contract Migration/Upgrade: Medium

11. Issue descriptions and recommendations by contract file

- PerpetualOption.sol

Contract Migration/Upgrade: Medium

Source and Description：

In line 1661, the function `upgradeCallPut` can be used to upgrade a put or call contract. When mistakes are made in contract upgrade the locked assets may suffer losses.

Recommendation:

Consider migrating or upgrading contract files with extreme caution and conducting a new audit prior to migration, or completely disabling this feature,

Update：Acknowledged by the AntiMatter team. The team will upgrade contracts with extreme caution and conduct a new audit prior to contract upgrade.

12. Recommendations to enhance the overall security

We list some recommendations in this section. They are not mandatory but will enhance the overall security of the system if they are adopted.

- N/A