Openmeta

AUDIT REPORT

Version 1.0.0

Serial No. 20220308000220221

Presented by Fairyproof

March 8, 2022

01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the Openmeta project.

Audit Start Time:

March 4, 2022

Audit End Time:

March 7, 2022

Audited Code's Github Repository:

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

Audited Code's Github Commit Number When Audit Started:

d8749545140dc1559222ae55bb80190e18ab17d1

Audited Code's Github Commit Number When Audit Ended:

498da778c2562623f18175b574de785267d127fe

Audited Source Files:

The source files audited include all the files with the extension "sol" as follows:

```
contracts/
├── OpenmetaController.sol
├── OpenmetaNFT.sol
├── OpenmetaTrade.sol
├── interface
|   ├── IERCToken.sol
|   ├── IMnftController.sol
|   ├── IOpenmetaController.sol
|   └── IOpenmetaTrade.sol
├── libraries
|   └── TransferHelper.sol
└── utils
    ├── BlockTimestamp.sol
    └── Validation.sol

3 directories, 10 files
```

The goal of this audit is to review Openmeta's solidity implementation for its NFT auction application, 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.

This audit only applies to the specified code, software or any materials supplied by the Openmeta team for specified versions. Whenever the code, software, materials, settings, environment etc is changed, the comments of this audit will no longer apply.

— 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 system 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. If the audited source files are smart contract files, 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 source code.
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 source code 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 source 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

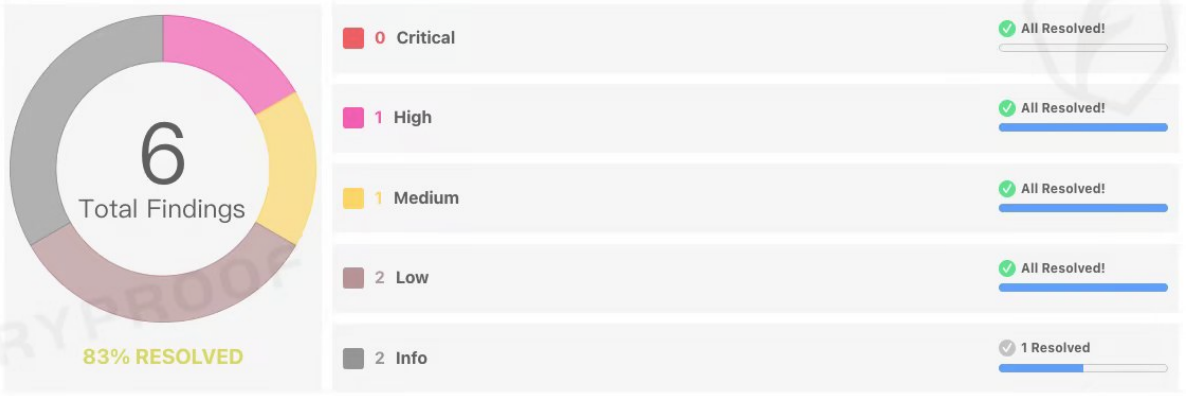
For this audit, we used the following sources of truth about how the NFT auction application should work:

https://nft.openmeta.finance/

This was considered the specification, and when discrepancies arose with the actual code behavior, we consulted with the Openmeta team or reported an issue.

— Comments from Auditor

Serial Number	Auditor	Audit Time	Result
2022030800022021	Fairyproof Security Team	Mar 4, 2022 - Mar 7, 2022	Informational



Summary:

The Fairyproof security team used its auto analysis tools and manual work to audit the project. During the audit, 1 risk of high-severity, 1 risk of medium-severity, 2 risks of low-severity and 2 risks of informational-severity were found. The risk of high-severity, the risk of medium-severity, 2 risks of low-severity and 1 risk of informational-severity have been fixed and 1 risk of informational-severity has been confirmed.

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 blockchain applications.

03. Introduction to Openmeta

Openmeta is an NFT auction application.

04. Major functions of audited code

The audited code mainly implements an NFT auction application which has the following functions:

The `OpenmetaNFT.sol` contract implements an ERC-1155 NFT contract.

The `OpenmetaController.sol` contract implements a controller which controls NFT mint, access control and etc.

The `OpenmetaTrade.sol` contract implements an auction application which supports both ERC-721 tokens and ERC-1155 tokens. A seller can sell his/her NFTs and select a type of token from a list of supported tokens as his/her preferred payment method. A buyer can buy NFTs by paying the seller's preferred token.

Note: the application has off-chain functions which were not covered by this audit.

05. 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
- Replay Attack
- Reordering Attack
- Miner's Advantage
- Rollback Attack
- DDos Attack
- Transaction Ordering Attack
- Race Condition
- Access Control
- Integer Overflow/Underflow
- Timestamp Attack
- Gas Consumption
- Inappropriate Callback Function
- Function Visibility
- Implementation Vulnerability
- Uninitialized Storage Pointer
- Arithmetic Precision
- Tx.origin
- Fake Deposit
- Shadow Variable
- Incorrect Parameter Setting
- Design Vulnerability
- Token Issurance
- Admin Rights
- Inappropriate Proxy Design
- Inappropriate Use of Slots
- Asset Security
- Contract Upgrade/Migration

- Code Improvement
- Misc

06. 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.

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

Informational is not an issue or risk but a suggestion for code improvement.

07. List of issues by severity

Index	Title	Issue/Risk	Severity	Status
FP-1	Incorrect Payment Function	Design Vulnerability	High	✓Fixed
FP-2	Incorrect Auction Function	Design Vulnerability	Medium	✓Fixed
FP-3	Missing Constrains for Variable	Incorrect Parameter Setting	Low	✓Fixed
FP-4	Missing Validation for Payment	Design Vulnerability	Low	✓Fixed
FP-5	Redundant Code	Code Improvement	Informational	✓Fixed
FP-6	Unused Imported Interfaces	Code Improvement	Informational	Confirmed

08. Issue descriptions

[FP-1] Incorrect Payment Function High ✓Fixed

Issue/Risk: Design Vulnerability

Description:

In the `OpenmetaTrade.sol` file, the `performOrder` function allowed a buyer to pay ETHs. However when an auction was going on, only the one that signed the `OpenmetaController` contract was able to call the `performOrder` function(more details can be checked in `modifier checkOrderCaller`). In this situation when a buyer won an item and paid for it, it was the signer i.e. `msg.sender` that paid for it. That is to say the buyer didn't need to pay for the won item and as long as the `msg.sender` had sufficient asset, the assets held by `msg.sender` were used to pay for it.

Recommendation:

Consider using `WETH` instead of `ETH` to pay for a won item in an auction.

Status:

It has been fixed by the Openmeta team. A conditional check has been added: `require(!isOriginToken, "auctions do not support chain-based coins");`. This enforces a buyer to pay ERC-20 tokens.

[FP-2] Incorrect Auction Function Medium ✓Fixed

Issue/Risk: Design Vulnerability

Description:

In the `OpenmetaTrade.sol` file, the `performOrder` function called the `getOrderUserBalance` function to verify the winner's balance and the seller's NFTs in an auction. When the seller didn't have sufficient ERC-1155 NFTs, an auction wouldn't be started. However the implementation allowed a seller to sell an NFT which wasn't minted yet. The correct implementation should only allow a seller to auction an already minted NFT.

Recommendation:

Consider adding a directive to check whether or not all the NFTs for an auction are already minted before checking whether or not the seller has sufficient NFTs.

Status:

It has been fixed by the Openmeta team. A conditional check has been added: `if ((_dealOrder.minted && nftBalance < _makerOrder.quantity) || amountBalance < _dealOrder.dealAmount)`

[FP-3] Missing Constrains for Variable Low ✓Fixed

Issue/Risk: Incorrect Parameter Setting

Description:

In the `OpenmetaController.sol` file, the `setMaxFeeLimit` function didn't have constraints for the setting of `_maxFeeLimit`. If `_maxFeeLimit` were inappropriately set, buyers wouldn't be able to participate in an auction.

Recommendation:

Consider adding constraints for the setting of `_maxFeeLimit`.

Status:

It has been fixed by the Openmeta team. The following condition has been added:

```
require(
    _maxFeeLimit >= feeRate && _maxFeeLimit < BASE_ROUND,
    "verification fee limit failed"
);
```

[FP-4] Missing Validation for Payment Low ✓Fixed

Issue/Risk: Design Vulnerability

Description:

In the `OpenmetaTrade.sol` file, the `performOrder` function didn't check whether or not the winner's payment was equal to or greater than the winning bid's price. This validation was done off-chain.

Recommendation:

Consider adding validation for this in smart contract implementation as well.

Status:

It has been fixed by the Openmeta team and the following condition has been added:

```
uint256 dealAmount = _makerOrder.price * _makerOrder.quantity;
require(_dealOrder.dealAmount >= dealAmount, "order deal amount verification
failed");
```

[FP-5] Redundant Code Informational ✓Fixed

Issue/Risk: Code Improvement

Description:

In the `OpenmetaController.sol` file, both the `getMaximumFee` function and the `checkFeeAmount` function had the following redundant code:

```
maximumFee = _amount * (maxFeeLimit * BASE_ROUND) / 10000 / BASE_ROUND;
authorFee = _amount * (_authorProtocolFee * BASE_ROUND) / 10000 / BASE_ROUND;
txFee = _amount * (feeRate * BASE_ROUND) / 10000 / BASE_ROUND;
```

In the above code section, the `* BASE_ROUND` operation and the `/BASE_ROUND` operation didn't improve a precision.

Recommendation:

Consider simplifying the code.

Status:

It has been fixed by the Openmeta team and here is the updated code:

```
uint256 public constant BASE_ROUND = 10000;
maximumFee = _amount * maxFeeLimit / BASE_ROUND;
```

[FP-6] Unused Imported Interfaces Informational Confirmed

Issue/Risk: Code Improvement

Description:

Under the `interface` directory, the `IMnftController.sol` file defined multiple interfaces. However these interfaces were not used and the interfaces were not implemented either.

Recommendation:

Consider reorganizing interfaces' definitions and importing them in the files that use them and implementing them in the files that inherit them. If there are interfaces which are not used by other, define them internally in smart contracts.

Status:

It has been confirmed by the Openmeta team. The team will improve code in its future upgrades.

09. 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

Appendix

Audited Files' SHA-256 Values:

```
contracts/OpenmetaController.sol:
0x4f53964474b08c964a28953550d5d3fef7485f6649b0f21ce311f4105af75bd7

contracts/OpenmetaNFT.sol:
0x9658ebddf0b2f1ed6036c2a4c19e115524df6d125b0160907dd6b9603e128e91

contracts/OpenmetaTrade.sol:
0x733a3bb69ef289328f2e8af5835fda278cb50763288fd11d1f250e4ce8e72579

contracts/interface/IERCToken.sol:
0xc7b798f9b0a0660cf79fd6f33a6f4172b79142ad8d2269744cae52ab70c3516e

contracts/interface/IMnftController.sol:
0xeda179b24e31a22e6d7d6948ac0b24d20f10472a94d2e7b85d71758cd23df59b

contracts/interface/IOpenmetaController.sol:
0xe03f90b69532e7b48f0f30652c09855dcd64fb8b2205240d2a383df8e9972e40

contracts/interface/IOpenmetaTrade.sol:
0xf41f9f58d97478dd3d7545ec972b4f5b64cfc7f5633584efee928da8927fff13

contracts/libraries/TransferHelper.sol:
0x10e90a45583a37c724469636c3f72891b05df8758bcced41e7428e0a16d739c8
```

```
contracts/utils/BlockTimestamp.sol:
0x49ed703773e14bdff4b91e2e3c6725b113a9f98a274a85fc4b662a4e8387279b

contracts/utils/Validation.sol:
0x21adc5caeba350fa5504f8cdfe192043eea2d8e814762dd5dc1e0ecb924c71f7
```

Unit Test Result:

```
44 passing (6s)

----------------------------|----------|----------|----------|----------|----------
------|
File                        |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered
Lines |
----------------------------|----------|----------|----------|----------|----------
------|
 contracts/                 |    92.86 |    66.67 |      100 |    92.99 |
       |
  Mock20.sol                |      100 |      100 |      100 |      100 |
       |
  OpenmetaController.sol    |      100 |    85.71 |      100 |      100 |
       |
  OpenmetaNFT.sol           |      100 |      100 |      100 |      100 |
       |
  OpenmetaTrade.sol         |    86.25 |    58.82 |      100 |    86.42 |...
303,304,325 |
 contracts/interface/       |      100 |      100 |      100 |      100 |
       |
  IERCToken.sol             |      100 |      100 |      100 |      100 |
       |
  IMnftController.sol       |      100 |      100 |      100 |      100 |
       |
  IOpenmetaController.sol   |      100 |      100 |      100 |      100 |
       |
  IOpenmetaTrade.sol        |      100 |      100 |      100 |      100 |
       |
 contracts/libraries/       |       50 |       25 |       50 |       50 |
       |
  TransferHelper.sol        |       50 |       25 |       50 |       50 |
 11,12,37,38 |
 contracts/utils/           |      100 |      100 |      100 |      100 |
       |
  BlockTimestamp.sol        |      100 |      100 |      100 |      100 |
       |
  Validation.sol            |      100 |      100 |      100 |      100 |
       |
----------------------------|----------|----------|----------|----------|----------
------|
All files                   |    90.85 |    64.15 |    95.56 |    91.07 |
       |
----------------------------|----------|----------|----------|----------|----------
------|
```

/

90.85% Statements `149/164` **64.15%** Branches `68/106` **95.56%** Functions `43/45` **91.07%** Lines `153/168`

File		Statements		Branches		Functions		Lines	
contracts/		92.86%	143/154	66.67%	64/96	100%	39/39	92.99%	146/157
contracts/interface/		100%	0/0	100%	0/0	100%	0/0	100%	0/0
contracts/libraries/		50%	4/8	25%	2/8	50%	2/4	50%	4/8
contracts/utils/		100%	2/2	100%	2/2	100%	2/2	100%	3/3

FAIRYPROOF

https://medium.com/@FairyproofT

https://twitter.com/FairyproofT

https://www.linkedin.com/company/fairyproof-tech

https://t.me/Fairyproof_tech

Reddit: https://www.reddit.com/user/FairyproofTech