FAIRYPROOF

TMG35S NFT Auction

AUDIT REPORT

Version 1.0.0

Serial No. 2022052400022021

Presented by Fairyproof

May 24, 2022

01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the TMG35S' NFT issuance and auction project.

Audit Start Time:

May 23, 2022

Audit End Time:

May 24, 2022

Project Token's Name:

TMG35S

Audited Source Files:

The calculated SHA-256 values for the audited files when the audit was done are as follows:

```
Auction.sol:  0x89e99e4f40364f29a6e3ce5630573f08e7a38da018b988ac984f78faa579f987
TMG35S.sol :  0x1173493039467530aaa704cd169dbd448ea217bdd2855614a58cc630d7b79942
```

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

```
contracts/
├── Auction.sol
└── TMG35S.sol

0 directories, 2 files
```

The goal of this audit is to review TMG35S' solidity implementation for its NFT issuance and auction functions, 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 TMG35S 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

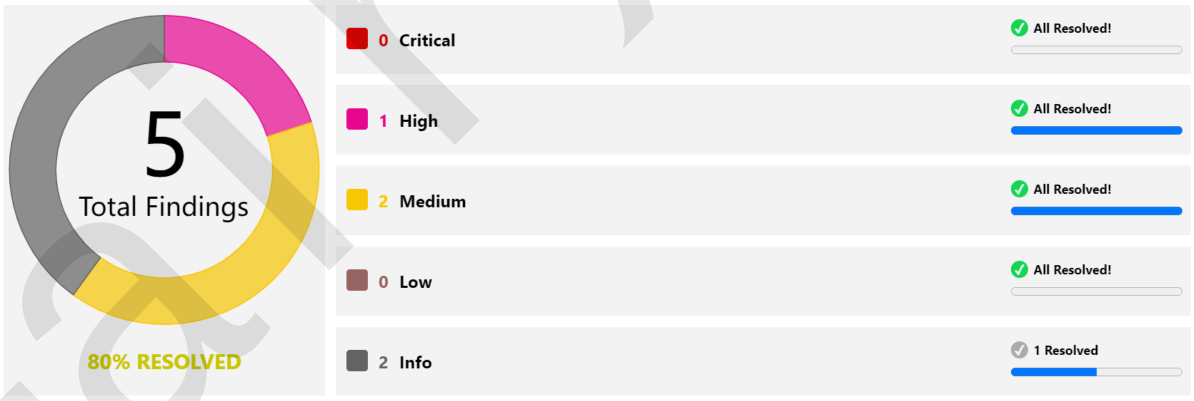
For this audit, we used the following sources of truth about how the NFT issuance and auction functions should work:

https://tmg35s.d1verse.io/#/home

These were considered the specification, and when discrepancies arose with the actual code behavior, we consulted with the TMG35S team or reported an issue.

— Comments from Auditor

Serial Number	Auditor	Audit Time	Result
2022052400022021	Fairyproof Security Team	2022.05.23 - 2022.05.24	Low



Summary:

The Fairyproof security team used its auto analysis tools and manual work to audit the project. During the audit, 1 issue of high-severity, 2 issues of medium-severity and 2 issues of informational-severity were discovered. The issue of high-severity has been partially fixed, 2 issues of medium-severity and 1 issue of informational-severity have been fixed, and 1 issue of informational-severity has been ignored.

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. Major functions of audited code

The audited code mainly implements the following functions:

Issuance of NFTs:

NFT Name: TMG35S

NFT Symbol: TMG35S

Issuer: The TMG35S Team

NFT Auction:

Users can list/delist their NFTs for auction or participate in NFT auctions. All the auctions are timed auctions. If a bidder bids a price in the last ten minutes in an auction, the auction will be automatically extended by an additional 10 minutes.

When an NFT auction ends, the last effective bidder wins and pays for the NFT and some fees will be charged by the application from the winner.

An NFT auction's parameters are set by the seller and NFTs can be priced in either ETH or an ERC-20 token.

Note: NFTs cannot be priced in an token which burns on transactions.

04. 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
- Parameter Check
- Design Vulnerability
- Token Issuance
- Admin Rights
- Inappropriate Proxy Design
- Inappropriate Use of Slots
- Asset Security
- Contract Upgrade/Migration
- Code Improvement
- Misc

05. 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.

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

06. Major areas that need attention

Based on the provided source code 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 had arithmetic operations and might introduce integer overflow or underflow if no safe libraries were used. All of them used safe libraries.

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

- Access Control

We checked each of the functions that could modify a state, especially those functions that could only be accessed by "owner".

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

- Token Issuance and Transactions

We checked whether or not the contract files that minted tokens or transferred tokens worked normally.

We found one issue. For more details please refer to FP-3 in "08. Issue description".

- State Update

We checked some key state variables which should only be set at initialization.

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

- Asset Security

We checked whether or not all the functions that transfer assets were safely handled.

We found one issue. For more details please refer to FP-2 in "08. Issue description".

- Miscellaneous

We found three issues. For more details please refer to FP-1, FP-4 and FP-5 in "08. Issue description".

07. List of issues by severity

Index	Title	Issue/Risk	Severity	Status
FP-1	Unsafe Withdrawal Method	DDoS Attack	High	Partially Fixed
FP-2	Redundant `receive` function	Asset Security	Medium	✓ Fixed
FP-3	Unsafe Token Transfer	Design Vulnerability	Medium	✓ Fixed
FP-4	Missing Check for Zero Address	Parameter Check	Informational	✓ Fixed
FP-5	Buyer Could Be Seller	Design Vulnerability	Informational	Ignored

08. Issue descriptions

[FP-1] Unsafe Withdrawal Method High Partially Fixed

Issue/Risk: DDoS Attack

Description:

In `Auction.sol` when an NFT was priced in ETH in an auction, a malicious bidder could block other bidders' participation by purposely rejecting returned staked assets and eventually won the NFT in a low price.

Here was the code section:

```
if (collectible.currency == address(0)) {
      require(msg.value >= price, "Insufficient payment");
      _executeFundsTransfer(collectible.currency, lastBidder, paymentAmount);
} else {
    IERC20(collectible.currency).safeTransferFrom(_msgSender(), address(this),
price);
    IERC20(collectible.currency).safeTransferFrom(_msgSender(), lastBidder,
paymentAmount);
}
```

Recommendation:

1 Consider changing the code such that a bidder should get back his/her staked assets manually

2 Consider disallowing a contract to participate in an auction

Update:

The TMG35S changed the code as follows to disallow a contract to participate in an auction:

```
require(!Address.isContract(_msgSender()), "Invalid caller");
require(tx.origin == _msgSender(),"only eoa");
```

Status:

It has been partially fixed. In some extreme cases such as one in which a bidder cannot get back his/her staked assets, this implementation will still face a DDoS attack but the probability becomes much lower so we marked this as a low-severity.

[FP-2] Redundant `receive` Function Medium ✓ Fixed

Issue/Risk: Asset Security

Description:

In `Auction.sol`, the following function was redundant since the implementation didn't have such a scenario. If a user sent ETHs to this contract the ETHs would never be gotten back.

The code section was as follows:

```
receive() payable external {
}
```

Recommendation:

Consider removing this function.

Update/Status:

It has been fixed by the TMG35S team.

[FP-3] Unsafe Token Transfer Medium ✓ Fixed

Issue/Risk: Design Vulnerability

Description:

In `Auction.sol` the `_executeFundsTransfer` function didn't use a safe method to transfer ERC-20 tokens. Therefore when a token was not a standard ERC-20 token, its transfer might encounter issues.

The code section was as follows:

```
function _executeFundsTransfer(address currency, address to, uint256 amount)
private {
    if (currency == address(0)) {
        Address.sendValue(payable(to), amount);
    } else {
        IERC20(currency).transfer(to, amount);
    }
}
```

Recommendation:

Consider using `safeTransfer` instead of `transfer`.

Update/Status:

It has been fixed by the TMG35S team.

[FP-4] Missing Check for Zero Address Informational

✓ Fixed

Issue/Risk: Parameter Check

Description:

In `Auction.sol` the `setPayee` function didn't check whether or not `newPayee` was a zero-address. When it was a zero-address, ETHs sent to it would be lost, ERC-20 tokens would not be sent out from it and it wouldn't be able to get its won NFT.

Recommendation:

Consider adding `require(newPayee != address(0))`.

Update/Status:

It has been fixed by the TMG35S team.

[FP-5] Buyer Could Be Seller Informational Ignored

Issue/Risk: Design Vulnerability

Description:

In `Auction.sol` the buyer's validity was not verified for an NFT auction and a buyer could be the NFT's seller.

Recommendation:

Consider adding a validity check.

Update:

The TMG35S team didn't think this was an issue.

Status:

It has been ignored by the TMG35S team.

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.

1 Consider adding a function to read an NFT auction's bidder number.

2 In the `withdraw` function, consider adding a directive to require the number of bidders to be greater than 0.

3 For both the `Bided` event and the `Withdrawn` event, consider adding a parameter to record the auctioned NFT for tracing.

4 When auctioning an ERC-1155 NFT, if `collectible.amount` is greater than 1, a bidder's price is the number of auctioned items * the bidding price. This is different from auctioning an ERC-721 NFT. Consider announcing this difference to all bidders.

5 There are two kinds of admin rights. Both need to be securely handled. if the admin rights are compromised, winning bidders may not be able to get their NFTs.

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