Exeedme Audit Report

Version 1.0.0

Serial No. 2021102000022012

Presented by Fairyproof

October 20, 2021

FAIRYPROOF

01. Introduction

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

Audit Start Time:

October 18, 2021

Audit End Time:

October 19, 2021

Audited Source Files' Ethereum Onchain Addresses:

Token: 0xee573a945b01b788b9287ce062a0cfc15be9fd86

DistibutionContract1: 0xd582a0740f39abda6aa3d28bf1ae343253ff3352

DistibutionContract2:0xa58ddecbc9ad958caf2aa8a2236d9abae74652a5

DistibutionContract3:0x74194e77d1cb4b07e0d49faddf413930cdf40973

DistibutionContract4:0xc55de2477b67080478de57e9133080f194119592

DistibutionContract5:0x698def85217fdf0797e3f6179ea3e31c8bcd54f9

Audited Source Code's URLs:

https://etherscan.io/address/0xee573a945b01b788b9287ce062a0cfc15be9fd86#code

https://etherscan.io/address/0xd582a0740f39abda6aa3d28bf1ae343253ff3352#code

https://etherscan.io/address/0xa58ddecbc9ad958caf2aa8a2236d9abae74652a5#code

https://etherscan.io/address/0x74194e77d1cb4b07e0d49faddf413930cdf40973#code

https://etherscan.io/address/0xc55de2477b67080478de57e9133080f194119592#code

https://etherscan.io/address/0x698def85217fdf0797e3f6179ea3e31c8bcd54f9#code

The goal of this audit is to review Exeedme's solidity implementation for its token issurance and vesting 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 Exeedme team for specified versions. Whenever the code, software, materials, settings, enviroment 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 Exceedme application should work:

https://www.exeedme.com/

project docs

These were considered the specification.

— Comments from Auditor

No vulnerabilities with critical, high, medium or low-severity were found in the above source code.

Additional notice: 0.

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 Exeedme

Exeedme is a blockchain-powered tournament platform that allows gamers of all skill levels to monetize their skills. The platform's vision is to build a fair and trusted play-to-earn platform where gamers can play their favorite games, challenge opponents and profit from betting on their own victory. Gamers can earn money from their bets, earn XED for engagement, and exclusive NFT prizes for winning tournaments and events.

04. Major functions of audited code

The audited code mainly implements the following functions:

- Issurance of ERC-20 Token
  - Name: XEDToken
  - Symbol: XED
  - Precisions: 18
  - Max Supply: 100,000,000
  - Flexible Max Supply: No
  - Transaction Charge: No
- Token Vesting
  - There are five vesting contracts: DistibutionContract1, DistibutionContract2, DistibutionContract3, DistibutionContract4, DistibutionContract5
  - The XED tokens will be sent to these five contracts right after the token's contract is deployed
  - These five contracts define the token's distribution addresses, vesting periods, schedules and amounts for distribution in the vesting periods
  - After the admin sets a time to activate a vesting schedule whenever the current time is equal to or greater than "activation time + vesting period" a specific amount of the tokens will be distributed to the distribution addresses.

Note:

- The admin can change the vesting's activation time. If a newly updated activation time is earlier than the previous activation time, the tokens that are locked may be unlocked in advance.
- The admin can pause transfers of the XED token

05. Admin rights

In this application the admin has the following previlleges:

- pausing token transfer
- resetting a vesting schedule's activation time to unlock tokens in advance
- resetting a contract's address that distrbutes tokens to the vesting contracts
- migrating tokens that are held in the vesting contracts

06. Key points in audit

During the audit Fairyproof mainly worked on the following items:

- 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.

- 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.

07. 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

08. 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.

09. List of issues by severity

A. Critical

- N/A

B. High

- N/A

C. Medium

- N/A

D. Low

- N/A

10. List of issues by source file

- N/A

11. Issue descriptions

- N/A

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.

- Using Memory Instead of Storage to Reduce Gas Consumption

In the `DistibutionContract1` file, starting from line 415 to line 425 , the `triggerTokenSend()` function has the following code section:

```
    function triggerTokenSend() external whenNotPaused  {
      //...
        for(uint i = 0; i < tokenOwners.length; i++) {
            /* Get Address Distribution */
            DistributionStep[] memory d = distributions[tokenOwners[i]];
            /* Go thru all distributions array */
            for(uint j = 0; j < d.length; j++){
                if( (block.timestamp.sub(TGEDate) > d[j].unlockDay) /* Verify if
unlockDay has passed */
                    && (d[j].currentAllocated > 0) /* Verify if currentAllocated > 0,
so that address has tokens to be sent still */
                ){
                    uint256 sendingAmount;
                    sendingAmount = d[j].currentAllocated;
                    distributions[tokenOwners[i]][j].currentAllocated =
distributions[tokenOwners[i]][j].currentAllocated.sub(sendingAmount);
```

```
                    distributions[tokenOwners[i]][j].amountSent =
distributions[tokenOwners[i]][j].amountSent.add(sendingAmount);
                    require(erc20.transfer(tokenOwners[i], sendingAmount));
                }
            }
        }
      //...
    }
```

`distributions` is a `storage` type, the value of `distributions[tokenOwners[i]]` is assigned to a variable `d` whose data type is `memory`. In Solidity, a memory type's assignment operation is done with `mload` and the gas consumption with this operation is lower than that of an `storage` type's assignment operation.

Consider using using `d` instead of `distributions[tokenOwners[i]]` to perform the subsequent arithmetic operations to reduce gas consumption.

- Removing Subtraction Operation

In line 423 of the `DistibutionContract1` contract file, the `triggerTokenSend()` function has the following code section:

```
  uint256 sendingAmount;
  sendingAmount = d[j].currentAllocated;
  distributions[tokenOwners[i]][j].currentAllocated = distributions[tokenOwners[i]]
[j].currentAllocated.sub(sendingAmount);
```

In the above code section the value of `d[j].currentAllocated` is the same as the value of `sendingAmount`, therefore in the subsequent directive, the value of `distributions[tokenOwners[i]][j].currentAllocated` is 0 and there is no need to perform a subtraction operation.

Consider using `distributions[tokenOwners[i]][j].currentAllocated = 0` to replace `distributions[tokenOwners[i]][j].currentAllocated = distributions[tokenOwners[i]][j].currentAllocated.sub(sendingAmount)`