

# Smart Contract Audit

# **Super Token**

---

By,

Shebin John

21st November 2020

---

## Index

1. [Introduction](#)
2. [High Threats](#)
3. [Medium Threats](#)
4. [Low Threats](#)
5. [Optimization & Readability](#)
6. [Typo's & Comments](#)
7. [Suggestions](#)

**Note:** Some threat levels or headings might be empty if there is no vulnerability/updates/suggestions found.



## Introduction

The contract audited here is currently not deployed in any net (main/test) and was shared by Yvan Nasr, and for checking purpose, is added into a private GitHub Gist:

<https://gist.github.com/remedcu/c72125eeffb583d14f48ef7927dbac38>

The contract is a custom version of the ERC20 smart contract with some additional functions built into the system.

Based on the audit, we were able to find 4 High threats, 1 Medium threat, and 1 Low threat with some other changes in the optimization, readability, typos, and comments section.

## High Threats

1. [L27](#) can cause underflow, resulting in transferring tokens that the sender doesn't have. Say User A is transferring tokens to User B. User A and B has 0 as balance. User A is transferring 1 Token (the smallest unit of Token). After the transfer, the balance of User A will be  $2^{256} - 1$ , the balance of User B will be 1.

**Recommended:** Use of SafeMath along with a balance check in a `require`` statement recommended.

*`require(_balances[msg.sender] >= amount, "Sender does not have enough balance.");`*

2. Same problem as Point 1 of High Threats can be seen in [L32](#) `modifier easyTransfer``.

**Recommended:** Use of SafeMath. Also, the check in [L35](#) should be to check if the amount is less than the balances of the sender, the current check cannot work in all cases (where the balance is higher than twice the amount, then the balances won't be less than or equal to the amount, thus resulting in reverting the transaction.).

*`require(_balances[from] >= amount, "Sender does not have enough balance.");`*

3. A subtle `^^` changes the meaning of the allowance in [L43](#), making it check that the allowance is not one, instead of zero. Similarly, the amount should be less than or equal to the allowances allowed, rather than greater than or equal to. Also the switch between `msg.sender`` and `from` in the `_allowances`.

**Recommended:** Use of `require` statement to check the allowance limit aptly.

*`require(_allowance[from][msg.sender] >= amount, "Initiator does not have enough allowance.");`*


4. L69, instead of `<=`, it should have been `>`.

**Recommended:**

*`require(stk.timestamp + stk.duration > block.timestamp, "Locked");`*

## Medium Threats

1. [L73](#), the deletion of the element in the array like that can result in an element not found, and creating a gap between the array. Say, for 10 elements in an array, calling the `redeem` function with parameter 5 will create two divisions, like from 0 to 4 elements and 6 to 9 elements. After that, if we call the `redeem` function without any parameter, the `for` loop will run from 0 to 8 elements (less than 9). But as there is no element in the 6th position (or index 5), it will result in an error.



**Recommended:** Replacing the last element in the array with that particular entry, and then deleting the last entry.

## Low Threats

1. [L77](#), can result in out of gas error if a sufficiently large amount of stakes are provided by a single address and calls the redeem function.

**Recommended:** Do it as batches with small numbers.

## Optimization & Readability

1. [L76](#), calling the redeem function with multiple stakes, and one of them having the stakes locked, will result in reverting the whole transaction. Say a user has made 10 stakes, and 9 of them have matured based on the duration provided. And only one of them is not. Calling the redeem function without any parameter will result in a failed transaction, even if he has 9 matured stakes.

**Recommended:** An if statement might be helpful in this scenario based on the business needs.

## Typo's & Comments

1. No comments provided for any line of code.

## Suggestions

1. No documentation has been done.
2. [L43](#), making a require, instead of an if, will fail much more elegantly, than reverting in the opposite case without an error message.

