

# Audit Whale

Smart contract code  
review and security  
analysis report

This document may contain confidential information about IT systems and the intellectual property of the Customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities fixed - upon a decision of the Customer.

## Document

<b>Name</b>	Smart Contract Code Review and Security Analysis Report for StudentCoin.
<b>Approved by</b>	Jameson   COO Audit Whale
<b>Type</b>	Token
<b>Platform</b>	Ethereum / Solidity
<b>Methods</b>	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
<b>Repository</b>	<a href="https://github.com/StudentCoinTeam/stc-erc20-v2">https://github.com/StudentCoinTeam/stc-erc20-v2</a>
<b>Commit</b>	412905FFA414C34B75FA4C52E5993318B2A5F067
<b>Deployed contract</b>	0x15B543e986b8c34074DFc9901136d9355a537e7E
<b>Timeline</b>	6 APR 2021 – 8 APR 2021
<b>Changelog</b>	8 APR 2021 – INITIAL AUDIT

## Table of contents

Document.....	2
Table of contents.....	3
Introduction .....	4
Scope .....	4
Executive Summary .....	5
Severity Definitions.....	6
AS-IS overview.....	7
Conclusion .....	11
Disclaimers.....	12

## Introduction

Audit Whale (Consultant) was contracted by StudentCoin (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer's smart contract and its code review conducted between April 6<sup>th</sup>, 2021 – April 8<sup>th</sup>, 2021.

## Scope

The scope of the project is smart contracts in the repository:

Contract deployment address: 0x15B543e986b8c34074DFc9901136d9355a537e7E

Repository: <https://github.com/StudentCoinTeam/stc-erc20-v2>

File:

STCERC20.sol

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	<ul style="list-style-type: none"> <li>▪ Reentrancy</li> <li>▪ Ownership Takeover</li> <li>▪ Timestamp Dependence</li> <li>▪ Gas Limit and Loops</li> <li>▪ DoS with (Unexpected) Throw</li> <li>▪ DoS with Block Gas Limit</li> <li>▪ Transaction-Ordering Dependence</li> <li>▪ Style guide violation</li> <li>▪ Costly Loop</li> <li>▪ ERC20 API violation</li> <li>▪ Unchecked external call</li> <li>▪ Unchecked math</li> <li>▪ Unsafe type inference</li> <li>▪ Implicit visibility level</li> <li>▪ Deployment Consistency</li> <li>▪ Repository Consistency</li> <li>▪ Data Consistency</li> </ul>

Functional review	<ul style="list-style-type: none"><li>▪ Business Logics Review</li><li>▪ Functionality Checks</li><li>▪ Access Control &amp; Authorization</li><li>▪ Escrow manipulation</li><li>▪ Token Supply manipulation</li><li>▪ Assets integrity</li><li>▪ User Balances manipulation</li><li>▪ Kill-Switch Mechanism</li><li>▪ Operation Trails &amp; Event Generation</li></ul>
-------------------	--

## Executive Summary

According to the assessment, the Customer's smart contracts are secured.

Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. A general overview is presented in AS-IS section, and all found issues can be found in the Audit overview section.

Security engineers found **1** medium issue during the audit.

## Severity Definitions

Risk Level	Description
<b>Critical</b>	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
<b>High</b>	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
<b>Medium</b>	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
<b>Low</b>	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution
<b>Lowest / Code Style / Best Practice</b>	Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored.

## **AS-IS**

### **overview**

## **STCERC20.s**

### **ol**

#### **Description**

STCERC20 is token contract.

#### **Imports**

STCERC20 contract has following imports:

- ERC20 – from the OpenZeppelin.

#### **Inheritance**

STCERC20 is ERC20.

#### **Structs**

STCERC20 contract has no data structures.

#### **Enums**

STCERC20 contract has no enums.

#### **Events**

STCERC20 contract has no custom events.

#### **Modifiers**

STCERC20 contract has no modifiers.

#### **Fields**

STCERC20 contract has no fields and constants.

#### **Functions**

STCERC20 contract has following functions:

- ***constructor***

## **Description**

Initializes the contract. Mints 10 billion tokens to the deployer address.

## **Visibility**

public

## **Input parameters**

None

## **Constrain**

ts None

## **Events**

emit None

## **Output**

None

- ***batchTransfer***

## **Description**

Used to send tokens to multiple addresses.

## **Visibility**

public

## **Input parameters**

- address[] calldata destinations
- uint256[] calldata amounts

## **Constraints**

- The length of the amounts should be the same as destinations length.

## **Events**

**emit** None

## **Output**

None

## Audit overview

### ■ ■ ■ ■ Critical

No critical issues were found.

### ■ ■ ■ High

No high issues were found.

### ■ ■ Medium

1. batchTransfer function has a costly loop. For big input arrays, the gas limit can be reached.

### ■ Low

No low severity issues were found.

### ■ Lowest / Code style / Best Practice

No lowest severity issues were found.

## Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, high-level description of functionality was presented in As-Is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security engineers found **1** medium issue during the audit.

Violations in the following categories were found and addressed to Customer:

Category	Check Item	Comments
Code review	<ul style="list-style-type: none"><li>▪ Costly loop</li></ul>	<ul style="list-style-type: none"><li>▪ batchTransfer function has a costly loop.</li></ul>

## **Disclaimers**

### **Disclaimer**

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

### **Technical Disclaimer**

Smart contracts are deployed and executed on blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.