Smart Contract Audit Report

By Globalized ® I May 2018

Executive Summary

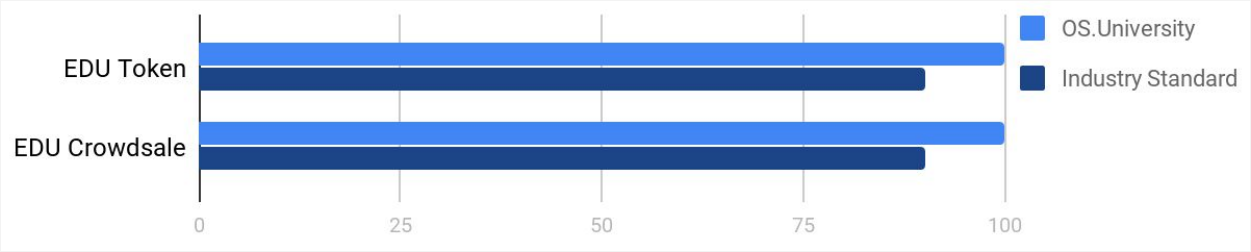
This document outlines the overall security of Cool Cousin's smart contract as evaluated by Globalized's Smart Contract auditing team.

The scope of this audit was to analyze and document Cool Cousin's token contract codebase for quality, security, and correctness.

Contract Status

All issues have been remediated.
(See Complete Analysis)

Test Coverage



Testable code is higher than industry standard.
(See Coverage Report)

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, merely an assessment of its logic and implementation.

In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Globalized recommend that the OS.University team put in place a bug bounty program to encourage further and active analysis of the smart contract.

Table of Contents

1. Auditing Strategy and Techniques Applied

The Globalized Team has performed a thorough review of the smart contract code, the latest version as written and updated on May 21st, 2018.

All main contract files were reviewed using the following tools and processes.

Throughout the review process, care was taken to ensure that the token contract:

A. Implements and adheres to existing ERC-20 Token standard appropriately and effectively:

- Documentation and code comments match logic and behavior
- Distributes tokens in a manner that matches calculations
- Follows best practices in efficient use of gas, without unnecessary waste
- Uses methods safe from reentrance attacks
- Is not affected by the latest vulnerabilities

The Globalized Team has followed best practices and industry-standard techniques to verify the implementation of OS.University token contract.

To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as they were discovered. Part of this work included writing a unit test suite using the Truffle testing framework.

In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1. Due diligence in assessing the overall code quality of the codebase.
2. Cross-comparison with other, similar smart contracts by industry leaders.
3. Testing contract logic against common and uncommon attack vectors.
4. Thorough, manual review of the codebase, line-by-line.
5. Deploying the smart contract to testnet and production networks using multiple client implementations to run live tests.

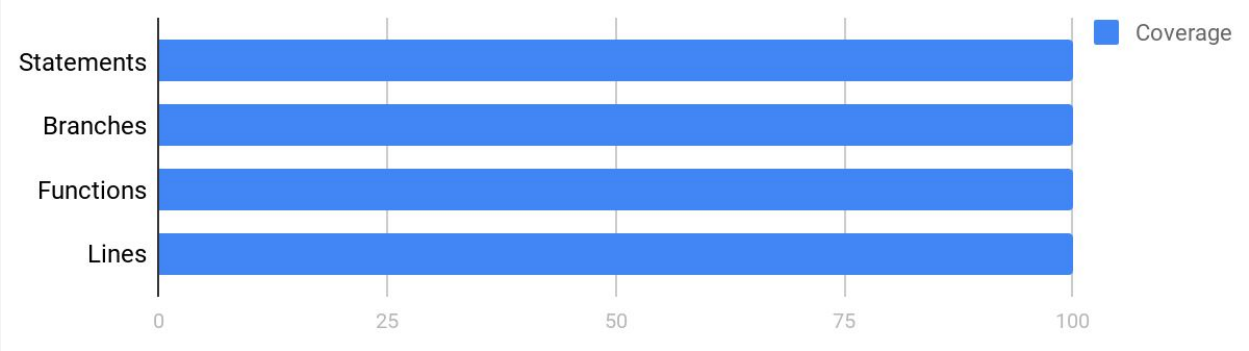
2. Structure Analysis and Test Results

2.1. Summary

The OS.University contracts are an ERC-20 contracts. The crowdsale utilizes a system with bonuses for both increasing funds and early purchases.

2.2. Coverage Report

As part of our work assisting OS.University in verifying the correctness of their contract code, our team was responsible for testing using the Truffle testing framework.



2.3. Failing Tests

No Failing Tests.

See Test Suite Results for all tests.

3. Complete Analysis

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

- **Informational** - The issue has no impact on the contract’s ability to operate.
- **Low** - The issue has minimal impact on the contract’s ability to operate.
- **Medium** - The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.
- **High** - The issue affects the ability of the contract to compile or operate in a significant way.
- **Critical** - The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

All comments discovered during the audit of the smart contract can be found in the GitHub repository of the contracts audit.

3.1. Resolved, Critical: bypassing KYC verification intransferFrom

Explanation:
The isKnownCustomer modifier used within the transferFrom function of EDU Token smart contract should pass over the _from parameter instead of the _to one. This behavior could be used in order to bypass KYC verification.

Resolution:
The OS.University team confirmed the issue and adjusted the argument passed to the modifier, so that vector of bypassing could not be exploited.

3.2. Resolved, Informational - redundant usage of rate

Explanation:
The rate argument does not need to be hardcoded during the construction of EDU Crowdsale contract. Instead it could be dynamically retrieved by the getCurrentRate() function.

Resolution:
The OS.University team recognized the proposed improvement as legit, and instead of using a hardcoded value updated the code to dynamically retrieve the current rate during construction.

4. Closing Statement

We are grateful to have been given the opportunity to work with the OS.University Team. The OS.University contracts are an ERC-20 EDU token contract and a crowdsale contract. The crowdsale system utilizes a system with bonuses for increasing funds and early purchases. These contracts are built around the OpenZeppelin libraries and are functional.

The team of experts at Globalized, having backgrounds in all aspects of blockchain, cryptography, and cybersecurity, we can say with confidence that the OS.University contract is free of any critical issues. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

We at Globalized recommend that the OS.University Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

5. Test Suite Results

Contract: EDUCrowdsale
- ✓ wei raised updated correctly (384ms)
- ✓ could not contribute over cap (280ms)
- ✓ delayed transfer blocking transactions (387ms)

crowdsale new investor
- ✓ should be able to contribute (209ms)
- ✓ should not be able to transfer without kyc (237ms)
- ✓ should be able to transfer after KYC (287ms)

investor passed kyc before crowdsale
- ✓ should be able to contribute (417ms)

opening/closing time
- ✓ should not be able to contribute before opening time
- ✓ should not be able to contribute after closing time (214ms)

wallet change
- ✓ token wallet change (468ms)
- ✓ eth wallet change (601ms)
- ✓ test volume bonus over 50 (199ms)
- ✓ test volume bonus over 150 (162ms)
- ✓ test volume bonus over 250 (171ms)

Contract: EDUToken
token properties
✓ has a name
✓ has a symbol
✓ has 18 decimals
token deployment
✓ total supply equals 48000000
✓ assigns the initial total supply to the creator
✓ new account has zero balance
token transfer
✓ transfer of 1000 tokens from creator to empty account (60ms)
✓ revert on transfer from zero balance account
✓ revert on transfer to zero address
approve functionality
✓ approved account is able to transfer (65ms)
✓ approved account is not able to transfer over limit (97ms)
✓ increase approval (87ms)
✓ decrease approval (60ms)
✓ allowance view returns correct value
token burning
✓ account balance and initial supply changed after burn (63ms)
✓ revert on burning insufficient amount (40ms)

Contract: EventsTest
certifier events
✓ confirmed event emited
✓ revoked event emited (69ms)
token events
✓ burn event emited
✓ transfer event emited
✓ approval event emited
✓ add manager event emited
✓ remove manager event emited
✓ certifier changed event emited
✓ ownership transfered event emited
crowdsale events
✓ token purchase event emited (175ms)
✓ token wallet changed event emited
✓ wallet changed event emited

42 passing (16s)

6. Contracts Tested

File	Fingerprint (SHA256)
EDUCrowdsale.sol	8e273686a16ce59e3e97e89d5bd936ffdb1723df81d567187db9e7715f3bbc4e
EDUToken.sol	bac196df1f908b711d01b0907035e5ac4c004ffa05f98a578f52f1cd5213e3d3
Migrations.sol	fba1b52fd068b61782c862ab3a9bcec53a598b23d66f29784ff609221c43345a
Certifiable.sol	7101c711596460fa83b914dab6d5445ec5304e884a843230897a71e3e0b71bca
Certifier.sol	91021f2ee0ae2ac04b2e75d0dfac09859a60e2fd122c89bdfc59b8c39766bdd2
KYCToken.sol	ea37c065cd8291a4938bac67c1eb9a83b99d62a3f8f2a70aa7e4a54defe28c0b