CERTIK

# Security Assessment

# Studyum

Jun 5th, 2021

# Table of Contents

# Summary

This report has been prepared for Studyum smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

# Overview

## Project Summary

| | |
|---|---|
| Project Name | Studyum |
| Platform | Ethereum |
| Language | Solidity |
| Codebase | https://github.com/STUDYUM/Studyum-Smart-Contracts |
| Commits | 8be103b82943753ef7faefcccc827e6b354ff26e |

## Audit Summary

| | |
|---|---|
| Delivery Date | Jun 05, 2021 |
| Audit Methodology | Static Analysis, Manual Review |
| Key Components | |

## Vulnerability Summary

| | |
|---|---|
| Total Issues | 13 |
| ● Critical | 0 |
| ● Major | 2 |
| ● Medium | 0 |
| ● Minor | 8 |
| ● Informational | 3 |
| ● Discussion | 0 |

# Audit Scope

| ID | file | SHA256 Checksum |
|---|---|---|
| BTS | BurnableToken.sol | ef2bb744749dfd865e4a0ec4bd50943b61086afac0e672f3b2f8583646b9d34a |
| ERC | ERC20.sol | 16e9b66b7295676f7ac0d946e0816aa64ed0283c52ea181fbf161f87db6a559f |
| ERD | ERC20Detailed.sol | 69cfc00b9dd07b5abcb87668f6741fdb70600dd1368a2aeaa40e7efa17e5fb6a |
| IER | IERC20.sol | c2a5b5503217256a446ff461a45a8b1c12d9c42bb95c916f4320e7ad780b7449 |
| OSS | Ownable.sol | 05782e95e69d7c1837b8debdc6be5122ba63b27fa8de71aa375e31d3e209520c |
| STU | STUDTeamVesting.sol | 7b0b1a360d6194c4d3b456e4129cdca30b7eb44de9852540fc61bac727aceb04 |
| STD | STUDToken.sol | 764e86c09a0ac49563918f5870d883d151753010210a72a7647729e920d2cb28 |
| SMS | SafeMath.sol | 431545171867d22f08db8feb2a468026d30fcfb14c7cec730ae65acbb464c173 |

# Decentralization Efforts

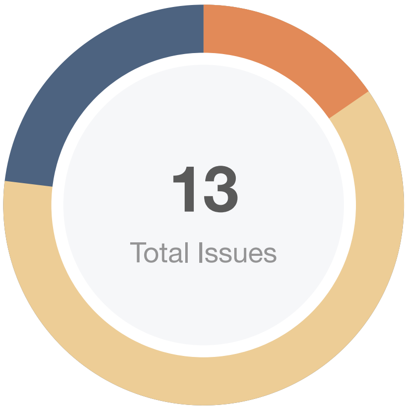
To improve the trustworthiness and transparency of the project, `Studyum` team deploys the Gnosis Safe Multi-Sig wallet to improve the decentralization of the `Studyum` project to resolve the `STD-02`, `STU-03` and `STU-04` `Centralized Risk` findings.

The `owner` role in deployment of contract `STUDToken.sol` at address 0x8f48e457b4b0708c999a1e088c005e977cfdd707 and in deployment of contract `STUDTeamVesting.sol at address 0xc18f55718260bb5a60eb5d99ec03576ae976c777 are transferred to the Multi-Sig wallet at address 0xBb672314E9BEaDBD1D83cb55272c722B0AFc7C21 through transaction 0xd87f0a268789a5d5435fd89664f1a9e22ad0d36e95dfed0d67d80168e3eb9464 and 0xff7e569ddc5fa5a7f7e5fe3a4a93df7a8a5ece2a9ca6e3c4b77ace68f2b9d1a5

Governance is distributed internally between CEO, COO and CTO of Studyum team with 2/3 consensus. The addresses of cosigners of Multi-Sig wallet are:

- 0x6bC59227b8eC8bb27b996C17D1f9277f061154F5
- 0x16b59dFBD119C7F7559613a219a7D86350FB4715
- 0x1DB392c9eEf175617AFf641155fFe88A3c8f7749

# Findings



| | |
|---|---|
| 🟥 **Critical** | **0** (0.00%) |
| 🟧 **Major** | **2** (15.38%) |
| 🟨 **Medium** | **0** (0.00%) |
| 🟨 **Minor** | **8** (61.54%) |
| 🟦 **Informational** | **3** (23.08%) |
| 🟩 **Discussion** | **0** (0.00%) |

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| **STD-01** | Unknown Implementation of `token.balanceOf` and `token.transfer` | **Centralization / Privilege** | **Minor** | **Acknowledged** |
| **STD-02** | Centralized Risk | **Centralization / Privilege** | **Major** | **Resolved** |
| STD-03 | SafeMath Not Used | Mathematical Operations | Minor | Acknowledged |
| STD-04 | Implementation Logic of `claimTokens` | Logical Issue | Minor | Acknowledged |
| STD-05 | Unhandled Return Value | Volatile Code | Informational | Acknowledged |
| **STU-01** | Unknown Implementation of `_token.transfer` | **Centralization / Privilege** | **Minor** | **Acknowledged** |
| **STU-02** | Unknown Implementation of `token.balanceOf` and `token.transfer` | **Centralization / Privilege** | **Minor** | **Acknowledged** |
| **STU-03** | Centralized Risk | **Centralization / Privilege** | **Major** | **Resolved** |
| **STU-04** | Centralized Risk | **Centralization / Privilege** | **Minor** | **Resolved** |
| STU-05 | SafeMath Not Used | Mathematical Operations | Minor | Acknowledged |
| STU-06 | Missing Emitting Events | Data Flow | Informational | Acknowledged |
| STU-07 | Implementation Logic of `claimTokens` | Logical Issue | Minor | Acknowledged |

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| STU-08 | Unhandled Return Value | Volatile Code | ● Informational | ⓘ Acknowledged |

## STD-01 | Unknown Implementation of `token.balanceOf` and `token.transfer`

| Category | Severity | Location | Status |
|---|---|---|---|
| Centralization / Privilege | ● Minor | STUDToken.sol: 68~70 | ⓘ Acknowledged |

### Description

On L68, `_tokenAddress` can be any contract address where the `IERC20` interface is implemented. As a result, the invocations of `token.balanceOf` on L69 and `token.transfer` on L70 may bring dangerous effects as the implementation of the functions `balanceOf` and `transfer` for `_token` is unknown to the user.

### Recommendation

We advise the client to check that the contract at address `_tokenAddress` on L68 is a standard smart contract that follows the `IERC20` interface with correct implementation logic.

### Alleviation

`[Studyum]`: The function `claimTokens()` was implemented to address the specific requirement to extract mistakenly sent ERC20 tokens or Ether to the contract address. The issue which was addressed here is the frequent locked tokens situation that has happened in other ICO/IEO/UTOs. The `_tokenAddress` was created to extract any kind of ERC20 tokens.

# STD-02 | Centralized Risk

| Category | Severity | Location | Status |
|---|---|---|---|
| **Centralization / Privilege** | ● **Major** | STUDToken.sol: 62~72 | ✓ **Resolved** |

## Description

In function `claimTokens`, the owner of the contract `owner` could transfer `token.balanceOf(address(this))` amount of tokens from `_tokenAddress` to itself.

## Recommendation

We advise the client to carefully manage the `owner` account's private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

## Alleviation

`[Studyum]`: As already mentioned `claimTokens()` was implemented based on the requirement presented. The requirement was that the owner should be a single person in the company responsible for the management of smart contracts. Creating a multi-sig wallet or DAO, in this case, would be an overkill, especially taking into consideration that `claimTokens()` function is a “special case” functionality.

`[Studyum]`: The client heeded the recommendation and deployed the Gnosis Safe Multi-Sig wallet at address 0xBb672314E9BEaDBD1D83cb55272c722B0AFc7C21 through transaction 0x45e61c87fbe22f1e4932eef87e3d272f12207140f0f329c144d69f14c07e2130. 3 decentralized governance addresses are added as cosigners: 0x6bC59227b8eC8bb27b996C17D1f9277f061154F5, 0x16b59dFBD119C7F7559613a219a7D86350FB4715 and 0x1DB392c9eEf175617AFf641155fFe88A3c8f7749. Governance is distributed internally between CEO, COO and CTO of Studyum with 2/3 consensus.

The owner of both STUD token and Team vesting have been both transferred to Multi-Sig wallet through transaction 0xd87f0a268789a5d5435fd89664f1a9e22ad0d36e95dfed0d67d80168e3eb9464 and 0xff7e569ddc5fa5a7f7e5fe3a4a93df7a8a5ece2a9ca6e3c4b77ace68f2b9d1a5

# STD-03 | SafeMath Not Used

| Category | Severity | Location | Status |
|---|---|---|---|
| Mathematical Operations | ● Minor | STUDToken.sol: 41 | ⓘ Acknowledged |

## Description

The SafeMath library is not used in the expression `amountSum += amounts[i]`, which makes it possible for overflows and incorrect results.

## Recommendation

We advise the client to adopt the SafeMath library for that expression.

## Alleviation

`[Studyum]`: The function `bulkTransfer()` was created to distribute the tokens more easily (and spend less Gas) in the case of Airdrop which eventually did not happen. This function is obviously not a part of ERC20 standard interface. Any external user can utilize this function by providing beneficiaries and amounts. In this case it is his responsibility to provide correct data (addresses and amounts). We consider this as a minor issue that should not be fixed in order for contract to function properly.

# STD-04 | Implementation Logic of `claimTokens`

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Minor | STUDToken.sol: 64 | ⓘ Acknowledged |

## Description

In the function `claimTokens` on L62, when `_tokenAddress` is address zero, the `balance` associated with `address(this)` is transferred to the owner of the contract. While when `_tokenAddress` is not address zero, the `token.balanceOf(address(this))` amount of tokens are transferred to the owner of the contract.

## Recommendation

We advise the client to check if the implementation logic is correct for the function `claimTokens`.

## Alleviation

`[Studyum]`: If the `_tokenAddress` is address `zero`, the `claimTokens()` extracts mistakenly sent Ether to the token address. That was stated as a special requirement in this case, and the function was implemented in that way. We consider this function implementation logic to be implemented as intended.

# STD-05 | Unhandled Return Value

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | ● Informational | STUDToken.sol: 45, 70 | ⓘ Acknowledged |

## Description

Return value of function `transfer` based on interface `IERC20` is ignored in function `bulkTransfer` and `claimTokens`.

## Recommendation

We advise the client to handle the return value of `transfer` to check if it's implementation is executed without any error.

## Alleviation

`[Studyum]`: This is informational. These are not a standard functions (ERC20). We acknowledge your statement to have the return value, but we consider that it is not needed.

# STU-01 | Unknown Implementation of `_token.transfer`

| Category | Severity | Location | Status |
|---|---|---|---|
| **Centralization / Privilege** | ● **Minor** | STUDTeamVesting.sol: 57, 85 | ⓘ **Acknowledged** |

## Description

On L57, `token` can be any contract address where the `IERC20` interface is implemented. As a result, the invocation of `_token.transfer` on L85 may bring dangerous effects as the implementation of the function `transfer` for `_token` is unknown to the user.

## Recommendation

We advise the client to check that the contract at address `token` on L57 is a standard smart contract that follows the `IERC20` interface with correct implementation logic.

## Alleviation

`[Studyum]`: We have already deployed the TeamVesting contract, where we have provided the ERC20 token address to it. We acknowledge that this can be an issue if somebody does not pay attention during the smart contract deployment, but that is obviously not the case here.

# STU-02 | Unknown Implementation of `token.balanceOf` and `token.transfer`

| Category | Severity | Location | Status |
|---|---|---|---|
| **Centralization / Privilege** | ● **Minor** | STUDTeamVesting.sol: 150~152 | ⓘ **Acknowledged** |

## Description

On L150, `_tokenAddress` can be any contract address where the `IERC20` interface is implemented. As a result, the invocations of `token.balanceOf` on L151 and `token.transfer` on L152 may bring dangerous effects as the implementation of the functions `balanceOf` and `transfer` for `_token` is unknown to the user.

## Recommendation

We advise the client to check that the contract at address `_tokenAddress` on L150 is a standard smart contract that follows the `IERC20` interface with correct implementation logic.

## Alleviation

`[Studyum]`: This function `claimTokens()` was implemented as a special requirement as already stated. This function will be called by owner who has an appropriate knowledge in this area. We acknowledge that interface was not checked, but the option for this to go wrong in any case basically does not exist.

# STU-03 | Centralized Risk

| Category | Severity | Location | Status |
|---|---|---|---|
| **Centralization / Privilege** | ● **Major** | STUDTeamVesting.sol: 144~154 | ✓ **Resolved** |

## Description

In function `claimTokens`, the owner of the contract `owner` could transfer `token.balanceOf(address(this))` amount of tokens from `_tokenAddress` to itself.

## Recommendation

We advise the client to carefully manage the `owner` account's private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

## Alleviation

`[Studyum]`: As already mentioned `claimTokens()` was implemented based on the requirement presented. The requirement was that the owner should be a single person in the company responsible for the management of smart contracts. Creating a multi-sig wallet or DAO, in this case, would be an overkill, especially taking into consideration that claimTokens() function is a “special case” functionality

`[Studyum]`: The client heeded the recommendation and deployed the Gnosis Safe Multi-Sig wallet at address 0xBb672314E9BEaDBD1D83cb55272c722B0AFc7C21 through transaction 0x45e61c87fbe22f1e4932eef87e3d272f12207140f0f329c144d69f14c07e2130. 3 decentralized governance addresses are added as cosigners: 0x6bC59227b8eC8bb27b996C17D1f9277f061154F5, 0x16b59dFBD119C7F7559613a219a7D86350FB4715 and 0x1DB392c9eEf175617AFf641155fFe88A3c8f7749. Governance is distributed internally between CEO, COO and CTO of Studyum with 2/3 consensus.

The owner of both STUD token and Team vesting have been both transferred to Multi-Sig wallet through transaction 0xd87f0a268789a5d5435fd89664f1a9e22ad0d36e95dfed0d67d80168e3eb9464 and 0xff7e569ddc5fa5a7f7e5fe3a4a93df7a8a5ece2a9ca6e3c4b77ace68f2b9d1a5

# STU-04 | Centralized Risk

| Category | Severity | Location | Status |
|---|---|---|---|
| **Centralization / Privilege** | ● **Minor** | STUDTeamVesting.sol: 127~131 | ✓ **Resolved** |

## Description

In function `changeBeneficiary`, the owner of the contract `owner` could change the beneficiary of the contract to an arbitrary payable address `_newBeneficiary`.

## Recommendation

We advise the client to carefully manage the `owner` account's private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

## Alleviation

`[Studyum]`: Even though is a centralized risk, this function `changeBeneficiary()` is implemented as intended based on specifications provided. A single person in the company should be able to change the team’s beneficiary address if he decides to do so. Creating a multi-sig wallet or DAO in this case would be an overkill, especially taking into consideration that `changeBeneficiary()` function is a “special case” functionality.

`[Studyum]`: The client heeded the recommendation and deployed the Gnosis Safe Multi-Sig wallet at address 0xBb672314E9BEaDBD1D83cb55272c722B0AFc7C21 through transaction 0x45e61c87fbe22f1e4932eef87e3d272f12207140f0f329c144d69f14c07e2130. 3 decentralized governance addresses are added as cosigners: 0x6bC59227b8eC8bb27b996C17D1f9277f061154F5, 0x16b59dFBD119C7F7559613a219a7D86350FB4715 and 0x1DB392c9eEf175617AFf641155fFe88A3c8f7749. Governance is distributed internally between CEO, COO and CTO of Studyum with 2/3 consensus.

The owner of both STUD token and Team vesting have been both transferred to Multi-Sig wallet through transaction 0xd87f0a268789a5d5435fd89664f1a9e22ad0d36e95dfed0d67d80168e3eb9464 and 0xff7e569ddc5fa5a7f7e5fe3a4a93df7a8a5ece2a9ca6e3c4b77ace68f2b9d1a5

# STU-05 | SafeMath Not Used

| Category | Severity | Location | Status |
|---|---|---|---|
| Mathematical Operations | ● Minor | STUDTeamVesting.sol: 70, 75, 83, 96 | ⓘ Acknowledged |

## Description

The SafeMath library is not used in the aforementioned lines, which makes it possible for overflows and incorrect results.

## Recommendation

We advise the client to adopt the SafeMath library for that expression.

## Alleviation

`[Studyum]`: Withdraw option is a “special case” option for the team to extract STUD tokens for the team. We acknowledge these aforementioned lines where SafeMath is not used. Since this is a minor issue and TeamVesting is only for internal use, we think that this does not bring any of potential threats.

# STU-06 | Missing Emitting Events

| Category | Severity | Location | Status |
|---|---|---|---|
| Data Flow | ● Informational | STUDTeamVesting.sol: 50~62 | ⓘ Acknowledged |

## Description

The constructor of the `STUDTeamVesting` contract does not emit an event when `_beneficiary` is set and `transferOwnership(owner)` is invoked.

## Recommendation

We advise the client to consider adding a `BeneficiaryChanged` event for `_beneficiary` and another event for transferring ownership.

## Alleviation

`[Studyum]`: We acknowledge that we should have added the event when `_beneficiary` is set. However, event for transferring ownership will be triggered as `STUDTeamVesting` contract inherits `Ownable` contract.

# STU-07 | Implementation Logic of `claimTokens`

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Minor | STUDTeamVesting.sol: 146 | ⓘ Acknowledged |

## Description

In the function `claimTokens` on L144, when `_tokenAddress` is address zero, the `balance` associated with `address(this)` is transferred to the owner of the contract. While when `_tokenAddress` is not address zero, the `token.balanceOf(address(this))` amount of tokens are transferred to the owner of the contract.

## Recommendation

We advise the client to check if the implementation logic is correct for the function `claimTokens`.

## Alleviation

`[Studyum]`: If the `_tokenAddress` is address zero, the `claimTokens()` extracts mistakenly sent Ether to the token address. That was stated as a special requirement in this case, and the function was implemented in that way. We consider this function implementation logic to be implemented as intended.

# STU-08 | Unhandled Return Value

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | ● Informational | STUDTeamVesting.sol: 85, 152 | ⓘ Acknowledged |

## Description

Return value of function `transfer` based on interface `IERC20` is ignored in function `withdraw` and `claimTokens`.

## Recommendation

We advise the client to handle the return value of `transfer` to check if it's implementation is executed without any error.

## Alleviation

`[Studyum]`: This is informational. These are not standard function (ERC20). We acknowledge your statement to have the return value, but we consider that it is not needed.

# Appendix

## Finding Categories

### Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

### Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

### Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

### Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

### Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in-storage one.

## Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

# Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

# About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

CERTIK