HACKEN

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer**: GiveUs
**Date**: 07 July, 2023

This report 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 can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

## Document

| | |
|---|---|
| **Name** | Smart Contract Code Review and Security Analysis Report for GiveUs |
| **Approved By** | Paul Fomichov \| Lead Solidity SC Auditor at Hacken OU |
| **Type** | DAO, Voting |
| **Platform** | EVM |
| **Language** | Solidity |
| **Methodology** | Link |
| **Website** | https://giveus.io/ |
| **Changelog** | 12.05.2023 - Initial Review<br>13.06.2023 - Second Review<br>07.07.2023 - Third Review |

## Table of contents

## Introduction

Hacken OÜ (Consultant) was contracted by GiveUs (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

## System Overview

The scope of this audit consists of a crowdfunding contract that aims to collect donations for project owners in the form of ERC20 tokens. The projects' specifications are determined by an admin role called *UPDATER_ROLE*, which acts as the coordinator of each project funding in the system.

Every project has several threshold donation values. When the donations reach thresholds, the sessions are voted by the donors for passage to the next session. If the votes are successful, project owners can withdraw the donations. If not, the session is reset and gets into a cooldown period.

The files in the scope:

- **Crowdfunding.sol** - The main contract of the system that is responsible for storing project specifications, the collection of funds, and withdrawal of funds.
- **ICrowdfunding.sol** - Interface for Crowdfunding.sol.

### Privileged roles

*Crowdfunding* contract use Access Control to restrict access to important functions. In the contract, there are 3 key privileged roles:

- *PAUSER_ROLE* - address with this privilege is permitted to pause and unpause contract in case of emergency.
- *UPDATER_ROLE* - address with this privilege has access to most functions. This role can create new crowdfunding projects, start a vote session for a threshold for a given project, add new supported token, update donation fee for given project, update project status (set if project is active or not), update project vote cooldown and withdraw funds to other projects.
- *WITHDRAWER_ROLE* - this role is defined in the system but never used. Address with this role does not have more privileges than other addresses. This role should be deleted.

## Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

### Documentation quality

The total Documentation Quality score is **10** out of **10**.

- Technical description is provided.
- Description of the development environment is provided.
- NatSpec is sufficient.

### Code quality

The total Code Quality score is **10** out of **10**.

- Best practices are followed.
- Development environment is configured.

### Test coverage

Code coverage of the project is **100%** (branch coverage).

- Deployment and basic user interactions are covered with tests.
- Negative cases coverage is present.
- Interactions by several users are not tested thoroughly.

### Security score

As a result of the audit, the code contains **no** issues. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

### Summary

According to the assessment, the Customer's smart contract has the following score: **10**. The system users should acknowledge all the risks summed up in the risks section of the report.

1
2
3
4
5
6
7
8
9
10
The final score

*Table. The distribution of issues during the audit*

| Review date | Low | Medium | High | Critical |
|---|---|---|---|---|
| 12 May 2023 | 4 | 4 | 2 | 2 |
| 13 June 2023 | 0 | 0 | 1 | 0 |
| 07 July 2023 | 0 | 0 | 0 | 0 |

## Risks

- No risks have been identified for this project.

## Checked Items

We have audited the Customers' smart contracts for commonly known and specific vulnerabilities. Here are some items considered:

| Item | Description | Status | Related Issues |
|---|---|---|---|
| **Default Visibility** | Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously. | Passed | |
| **Integer Overflow and Underflow** | If unchecked math is used, all math operations should be safe from overflows and underflows. | Passed | |
| **Outdated Compiler Version** | It is recommended to use a recent version of the Solidity compiler. | Passed | |
| **Floating Pragma** | Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly. | Passed | |
| **Unchecked Call Return Value** | The return value of a message call should be checked. | Passed | |
| **Access Control & Authorization** | Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users. | Passed | |
| **SELFDESTRUCT Instruction** | The contract should not be self-destructible while it has funds belonging to users. | Not Relevant | |
| **Check-Effect-Interaction** | Check-Effect-Interaction pattern should be followed if the code performs ANY external call. | Passed | |
| **Assert Violation** | Properly functioning code should never reach a failing assert statement. | Passed | |
| **Deprecated Solidity Functions** | Deprecated built-in functions should never be used. | Passed | |
| **Delegatecall to Untrusted Callee** | Delegatecalls should only be allowed to trusted addresses. | Passed | |
| **DoS (Denial of Service)** | Execution of the code should never be blocked by a specific contract state unless required. | Passed | |

| | | | |
|---|---|---|---|
| **Race Conditions** | Race Conditions and Transactions Order Dependency should not be possible. | Passed | |
| **Authorization through tx.origin** | tx.origin should not be used for authorization. | Not Relevant | |
| **Block values as a proxy for time** | Block numbers should not be used for time calculations. | Passed | |
| **Signature Unique Id** | Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. All parameters from the signature should be used in signer recovery. EIP-712 should be followed during a signer verification. | Not Relevant | |
| **Shadowing State Variable** | State variables should not be shadowed. | Passed | |
| **Weak Sources of Randomness** | Random values should never be generated from Chain Attributes or be predictable. | Not Relevant | |
| **Incorrect Inheritance Order** | When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. | Passed | |
| **Calls Only to Trusted Addresses** | All external calls should be performed only to trusted addresses. | Passed | |
| **Presence of Unused Variables** | The code should not contain unused variables if this is not justified by design. | Passed | |
| **EIP Standards Violation** | EIP standards should not be violated. | Not Relevant | |
| **Assets Integrity** | Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract. | Passed | |
| **User Balances Manipulation** | Contract owners or any other third party should not be able to access funds belonging to users. | Passed | |
| **Data Consistency** | Smart contract data should be consistent all over the data flow. | Passed | |

| | | | |
|---|---|---|---|
| **Flashloan Attack** | When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used. | Not Relevant | |
| **Token Supply Manipulation** | Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the Customer. | Not Relevant | |
| **Gas Limit and Loops** | Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit. | Passed | |
| **Style Guide Violation** | Style guides and best practices should be followed. | Failed | I04 |
| **Requirements Compliance** | The code should be compliant with the requirements provided by the Customer. | Passed | |
| **Environment Consistency** | The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code. | Passed | |
| **Secure Oracles Usage** | The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles. | Not Relevant | |
| **Tests Coverage** | The code should be covered with unit tests. Test coverage should be sufficient, with both negative and positive cases covered. Usage of contracts by multiple users should be tested. | Passed | |
| **Stable Imports** | The code should not reference draft contracts, which may be changed in the future. | Passed | |

## Findings

### ■■■■ Critical

#### C01. Funds Lock / Undocumented Behavior

| | |
|---|---|
| **Impact** | High |
| **Likelihood** | High |

Several functions accept native coins (with *payable* modifier) to Smart Contract, but there is no withdrawn mechanism for these funds. If, for some reason, a user sends native coins to contract, these coins will be locked there.

The contract accepts token deposits but lacks a withdrawal mechanism, which can result in funds being locked in the contract and is not documented.

**Path:**

./contracts/Crowdfunding.sol : createProject(), pause(), unpause(), endTresholdVoting(), addNewSupportedToken(), setDonationFee(), updateProjectStatus(), updateProjectVoteCooldown(), withdrawFundsToOtherProject()

**Recommendation**: Remove payable modifier from mentioned functions.

**Found in:** af8ce9eb0405098f53ed3d6584ec4bd3271c1941

**Status**: Fixed (Revised commit: 84d6bd1)

#### C02. Funds Lock

| | |
|---|---|
| **Impact** | High |
| **Likelihood** | High |

If a given project has set *donationFee > 0*, when the user is donating tokens to a given project, transactionFee value is calculated but not saved to the variable. *Amount*(function parameter) is transferred to contract, but only *donationAmount(amount - transactionFee)* is tracked. These funds will be locked inside the contract without being able to withdraw them.

**Path:**

./contracts/Crowdfunding.sol : donateToProject(),

**Recommendation**: Add variable to track total value of transactions fee and create mechanism to withdraw them using dedicated or existing role.

**Found in:** af8ce9eb0405098f53ed3d6584ec4bd3271c1941

**Status**: Fixed (Revised commit: 84d6bd1)

## ■■■ High

### H01. Unchecked Transfer

| | |
|---|---|
| **Impact** | High |
| **Likelihood** | Medium |

It is considered following best practices to avoid unclear situations and prevent common attack vectors.

The functions do not use SafeERC20 library for checking result of ERC20 token transfers. Tokens may not follow ERC20 standard and return false in case of transfer failure or not returning any value at all.

This can lead to denial of service vulnerabilities during interactions with non-standard tokens.

**Path:**

./contracts/Crowdfunding.sol : donateToProject(), withdrawFunds(),

**Recommendation**: Follow common best practices, use SafeERC20Upgradeable library to interact with tokens safely.

**Found in:** af8ce9eb0405098f53ed3d6584ec4bd3271c1941

**Status**: Fixed (Revised commit: 84d6bd1)

### H02. Undocumented Mathematical Operation

| | |
|---|---|
| **Impact** | High |
| **Likelihood** | Medium |

In the *deliberateVote()* function, there is a statement:

*project.availableToWithdraw += currentTreshold.budget++*

The usage and reason for increasing *currentTreshold.budget* by using the *++* statement is not clear.

This may lead to unexpected behavior.

**Path:**

./contracts/Crowdfunding.sol : deliberateVote()

**Recommendation**: Correct incorrect mathematical operation or document its behavior in detail.

**Found in:** af8ce9eb0405098f53ed3d6584ec4bd3271c1941

**Status**: Fixed (Revised commit: 84d6bd1)

### H03. Highly Permissive Role Access / Token Balance Manipulation

| | |
|---|---|
| **Impact** | High |
| **Likelihood** | Medium |

The UPDATER_ROLE role has access to user project owner funds and can move the funds between projects without any restrictions.

If a key leak were to occur, the potential consequences could be significant, potentially leading to security breaches and undermining the overall integrity of the system.

**Path:**

./contracts/Crowdfunding.sol : withdrawFundsToOtherProject()

**Recommendation**: It is recommended to restrict the scope of permissions for those roles.

To ensure transparency and accountability, it is advised to provide a comprehensive explanation of highly-permissive access in the system's public documentation. This would help to ensure that users are fully informed of the implications of such access and can make informed decisions accordingly.

Implement a multi-sig access management system with a Timelock controller (like OpenZeppelin Defender) and provide clear explanations to the users in the public documentation.

**Found in:** af8ce9eb0405098f53ed3d6584ec4bd3271c1941

**Status**: Fixed (Revised commit: 74e5746)

## ■■ Medium

### M01. Non-Finalized Code

| | |
|---|---|
| **Impact** | Medium |
| **Likelihood** | Medium |

The code should not contain TODO comments. Otherwise, it means that the code is not finalized and additional changes will be introduced in the future.

**Path:**

./contracts/Crowdfunding.sol : createProject(),

**Recommendation**: Add changes described in TODO comment.

**Found in:** af8ce9eb0405098f53ed3d6584ec4bd3271c1941

**Status**: Fixed (Revised commit: 84d6bd1)

### M02. Contradiction - Missing Validation

| | |
|---|---|
| **Impact** | Medium |
| **Likelihood** | Medium |

It is considered that the project should be consistent and contain no self-contradictions.

According to comment in *ICrowdfunding* the value of *Project.requiredVotePercentage* should be in boundaries 0-10000. However, in the functions the validation is missed.

According to the implementation of *setDonationFee()* the *donationFee* should be lower than 10000. However, in the function *createProject()* the validation is missed. Without this check and value greater than 10000, newly created plans will calculate that fee is bigger than the actual *amount* deposited.

According to good practice the values *projectData.owner*, *projectData.exchangeTokenAddress* should be different that 0x0 address. However, in the functions the validation is missed.

This may lead to unexpected value processed by the contract.

**Path:**

./contracts/Crowdfunding.sol : createProject(),

**Recommendation**: Provide documentation, comments and identifiers in code consciously, implement the validations.

**Found in:** af8ce9eb0405098f53ed3d6584ec4bd3271c1941

**Status**: Fixed (Revised commit: 84d6bd1)

### M03. Uninitialized Inheritance

| | |
|---|---|
| **Impact** | Medium |
| **Likelihood** | Medium |

The Crowdfunding.sol contract inherits AccessControlUpgradeable and PausableUpgradeable contracts; however, it does not initialize them.

This may lead to unexpected behavior and it is best practice to initialize all inherited contracts in the initialization.

**Path:**

./contracts/Crowdfunding.sol : initialize(),

**Recommendation**: Call *__Pausable_init()* and *__AccessControl_init()* in the initializer.

**Found in:** af8ce9eb0405098f53ed3d6584ec4bd3271c1941

**Status**: Fixed (Revised commit: 84d6bd1)

### M04. Contradiction

| | |
|---|---|
| **Impact** | Medium |
| **Likelihood** | Medium |

The NatSpec of the function *endTresholdVoting()* suggests that the function is used to start a voting session; however, the implementation ends a voting session.

Misleading NatSpec can lead to unexpected behavior.

**Path:**

./contracts/Crowdfunding.sol : endTresholdVoting(),

**Recommendation**: Correct NatSpec.

**Found in:** af8ce9eb0405098f53ed3d6584ec4bd3271c1941

**Status**: Fixed (Revised commit: 84d6bd1)

## Low

### L01. Missing Zero Address Validation

| | |
|---|---|
| **Impact** | Medium |
| **Likelihood** | Low |

Address parameters are being used without checking against the possibility of 0x0.

This can lead to unwanted external calls to 0x0.

**Path:** ./contracts/Crowdfunding.sol : addNewSupportedToken()

**Recommendation**: Implement zero address checks.

**Found in:** af8ce9eb0405098f53ed3d6584ec4bd3271c1941

**Status**: Fixed (Revised commit: 84d6bd1)

### L02. Functions that Can Be Declared External

| | |
|---|---|
| **Impact** | Low |
| **Likelihood** | Low |

In order to save Gas, public functions that are never called in the contract should be declared as external.

**Path:**

./contracts/Crowdfunding.sol : initialize(),

**Recommendation**: Use the external attribute for functions never called from the contract.

**Found in:** af8ce9eb0405098f53ed3d6584ec4bd3271c1941

**Status**: Fixed (Revised commit: 84d6bd1)

### L03. Variables that Can Be Boolean

| | |
|---|---|
| **Impact** | Low |
| **Likelihood** | Low |

Some of the variables in Crowdfunding.sol are only used as 0s or 1s like flags. These could be declared as bool in order to increase readability and optimize storage.

**Path:**

./contracts/Crowdfunding.sol : *,

**Recommendation**: Use *bool* data type instead of *uint256* where it is appropriate.

**Found in:** af8ce9eb0405098f53ed3d6584ec4bd3271c1941

**Status**: Fixed (Revised commit: 84d6bd1)

### L04. Redundant Modifier Usage

| | |
|---|---|
| **Impact** | Low |
| **Likelihood** | Medium |

In some of the private functions, the same modifiers are used twice, both with the private functions and the functions that call them.

Unnecessary checks can decrease readability and decrease Gas efficiency.

**Path:**

./contracts/Crowdfunding.sol : startTresholdVoting(), deliberateVote(), resetVoteSession()

**Recommendation**: Remove redundant modifier calls.

**Found in:** af8ce9eb0405098f53ed3d6584ec4bd3271c1941

**Status**: Fixed (Revised commit: 84d6bd1)

## Informational

### I01. Typos in Code

There are typos in several places in *Crowdfunding* and *ICrowdfunding*. It says *tresholds*; however, it should be *threshold*.

There are typos in several places in *ICrowdfunding*. It says *emited*; however, it should be *emitted*.

There are typos in several places in *Crowdfunding*. It says *lenght*; however, it should be *length*.

**Paths:**

./contracts/Crowdfunding.sol : *,

./contracts/ICrowdfunding.sol : *,

**Recommendation**: Fix typos.

**Found in:** af8ce9eb0405098f53ed3d6584ec4bd3271c1941

**Status**: Reported (In the NatSpec of withdrawFundsToOtherProject(), there is the type *beetween* which should be *between*)

### I02. Style Guide Violation - Order of Layout

The project should follow the official code style guidelines.
Inside each contract, library, or interface, use the following order:

- Type declarations
- State variables
- Events
- Modifiers
- Functions

Functions should be grouped according to their visibility and ordered:

- constructor
- receive function (if exists)
- fallback function (if exists)
- external
- public
- internal
- private

Within a grouping, place the view and pure functions at the end.

**Paths:**

./contracts/Crowdfunding.sol : *,

./contracts/ICrowdfunding.sol : *,

**Recommendation**: The official Solidity style guidelines should be followed.

**Found in:** af8ce9eb0405098f53ed3d6584ec4bd3271c1941

**Status**: Fixed (Revised commit: 84d6bd1)

### I03. Unused Variable

Unused variables should be removed from the contracts. Unused variables are allowed in Solidity and do not pose a direct security issue. It is best practice to avoid them as they can cause an increase in computations (and unnecessary Gas consumption) and decrease readability.

The variable *WITHDRAWER_ROLE* is never used.

**Path:**

./contracts/Crowdfunding.sol : WITHDRAWER_ROLE,

**Recommendation**: Remove unused variable.

**Found in:** af8ce9eb0405098f53ed3d6584ec4bd3271c1941

**Status**: Fixed (Revised commit: 84d6bd1)

### I04. Style Guide Violation - Naming Conventions

The project should follow the official code style guidelines.

Each function should use mixedCase style for name declaration.

**Path:**

./contracts/Crowdfunding.sol : CheckAndStartThresholdVoting(),

**Recommendation**: The official Solidity style guidelines should be followed. Change name *CheckAndStartThresholdVoting()* to *checkAndStartThresholdVoting()* to fit naming convention for functions.

**Found in:** 84d6bd1534b0616214d1a493722bc7767ac9e40b

**Status**: New

# Disclaimers

## Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with 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 report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

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 the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

## Technical Disclaimer

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

## Appendix 1. Severity Definitions

When auditing smart contracts Hacken is using a risk-based approach that considers the potential impact of any vulnerabilities and the likelihood of them being exploited. The matrix of impact and likelihood is a commonly used tool in risk management to help assess and prioritize risks.

The impact of a vulnerability refers to the potential harm that could result if it were to be exploited. For smart contracts, this could include the loss of funds or assets, unauthorized access or control, or reputational damage.

The likelihood of a vulnerability being exploited is determined by considering the likelihood of an attack occurring, the level of skill or resources required to exploit the vulnerability, and the presence of any mitigating controls that could reduce the likelihood of exploitation.

| Risk Level | High Impact | Medium Impact | Low Impact |
|---|---|---|---|
| **High Likelihood** | Critical | High | Medium |
| **Medium Likelihood** | High | Medium | Low |
| **Low Likelihood** | Medium | Low | Low |

### Risk Levels

**Critical**: Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.

**High**: High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.

**Medium**: Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.

**Low**: Major deviations from best practices or major Gas inefficiency. These issues won't have a significant impact on code execution, don’t affect security score but can affect code quality score.

## Impact Levels

**High Impact**: Risks that have a high impact are associated with financial losses, reputational damage, or major alterations to contract state. High impact issues typically involve invalid calculations, denial of service, token supply manipulation, and data consistency, but are not limited to those categories.

**Medium Impact**: Risks that have a medium impact could result in financial losses, reputational damage, or minor contract state manipulation. These risks can also be associated with undocumented behavior or violations of requirements.

**Low Impact**: Risks that have a low impact cannot lead to financial losses or state manipulation. These risks are typically related to unscalable functionality, contradictions, inconsistent data, or major violations of best practices.

## Likelihood Levels

**High Likelihood**: Risks that have a high likelihood are those that are expected to occur frequently or are very likely to occur. These risks could be the result of known vulnerabilities or weaknesses in the contract, or could be the result of external factors such as attacks or exploits targeting similar contracts.

**Medium Likelihood**: Risks that have a medium likelihood are those that are possible but not as likely to occur as those in the high likelihood category. These risks could be the result of less severe vulnerabilities or weaknesses in the contract, or could be the result of less targeted attacks or exploits.

**Low Likelihood**: Risks that have a low likelihood are those that are unlikely to occur, but still possible. These risks could be the result of very specific or complex vulnerabilities or weaknesses in the contract, or could be the result of highly targeted attacks or exploits.

## Informational

Informational issues are mostly connected to violations of best practices, typos in code, violations of code style, and dead or redundant code.

Informational issues are not affecting the score, but addressing them will be beneficial for the project.

## Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

### Initial review scope

| | |
|---|---|
| **Repository** | https://github.com/Krayt78/GiveUsContracts |
| **Commit** | af8ce9eb0405098f53ed3d6584ec4bd3271c1941 |
| **Whitepaper** | https://giveus.gitbook.io/white-paper/i.-introduction/what-is-giveus |
| **Requirements** | https://github.com/Krayt78/GiveUsContracts#readme |
| **Technical Requirements** | https://giveus.gitbook.io/white-paper/i.-introduction/what-is-giveus |
| **Contracts** | File: contracts/Crowdfunding.sol<br>SHA3: ab8fac6dfb88b4f5d17ed0699f0dbaef437ffa7cf1253853c314930d726dd154<br>File: contracts/ICrowdfunding.sol<br>SHA3: b03691859738fd5a0621ccd3a32ea776ba9d742cdd3a08fa2352d522f1122983 |

### Second review scope

| | |
|---|---|
| **Repository** | https://github.com/Krayt78/GiveUsContracts |
| **Commit** | 84d6bd1534b0616214d1a493722bc7767ac9e40b |
| **Whitepaper** | https://giveus.gitbook.io/white-paper/i.-introduction/what-is-giveus |
| **Requirements** | https://github.com/Krayt78/GiveUsContracts/tree/Audit-Changes#readme |
| **Technical Requirements** | https://giveus.gitbook.io/white-paper/i.-introduction/what-is-giveus |
| **Contracts** | File: contracts/Crowdfunding.sol<br>SHA3: dd50a54518ced7dfff934e0fe29215c6e3f282e49da9f321ad0b074a95fe6aa9<br>File: contracts/ICrowdfunding.sol<br>SHA3: 2d6f628385a458228de9a68f90fd692ab69ee735cd3672f2d6ab23525611cd4a |

### Third review scope

| | |
|---|---|
| **Repository** | https://github.com/Krayt78/GiveUsContracts/tree/Audit-Changes |
| **Commit** | 74e5746634be8f74042b38b7a0c982a2b82f2a36 |
| **Whitepaper** | https://giveus.gitbook.io/white-paper/i.-introduction/what-is-giveus |
| **Requirements** | https://github.com/Krayt78/GiveUsContracts/tree/Audit-Changes#readme |

| | |
|---|---|
| **Technical Requirements** | https://giveus.gitbook.io/white-paper/i.-introduction/what-is-giveus |
| **Contracts** | File: contracts/Crowdfunding.sol<br>SHA3: dd50a54518ced7dfff934e0fe29215c6e3f282e49da9f321ad0b074a95fe6aa9<br><br>File: contracts/ICrowdfunding.sol<br>SHA3: 2d6f628385a458228de9a68f90fd692ab69ee735cd3672f2d6ab23525611cd4a |