Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Hacken.IO Sample Token Vesting

## Sample Audit Techspec

## Project Overview

Sample Solidity project is an ERC20 based vesting project. It allows owners to create multiple payment plans and deposit tokens to allow users to receive those funds in a way described by the plan.

## 1. Functional Requirements

### 1.1. Roles

Hacken.IO samples project has two roles:

- **Vesting Admin:** All control regarding the vesting contract belongs to a vesting admin. A vesting admin can create payment plans, and in the case of necessity, can revoke a payment plan. Allows to lock funds for users.
- **User:** can withdraw funds according to a provided payment plan.

### 1.2. Features

Hacken.IO Sample Token Vesting has the following features:

- Create new payment plans. (Admin)
- Lock funds (tokens) into payment plans. Each address can have only 1 active vesting plan.
- Revoke payment plans. (Admin)
- Release vested tokens. (User)
- Calculate amount that should be unlocked. (Everyone)
- Get detailed information about a user participation in the payment plans. (Everyone)
- Get the number of payment plans. (Everyone)

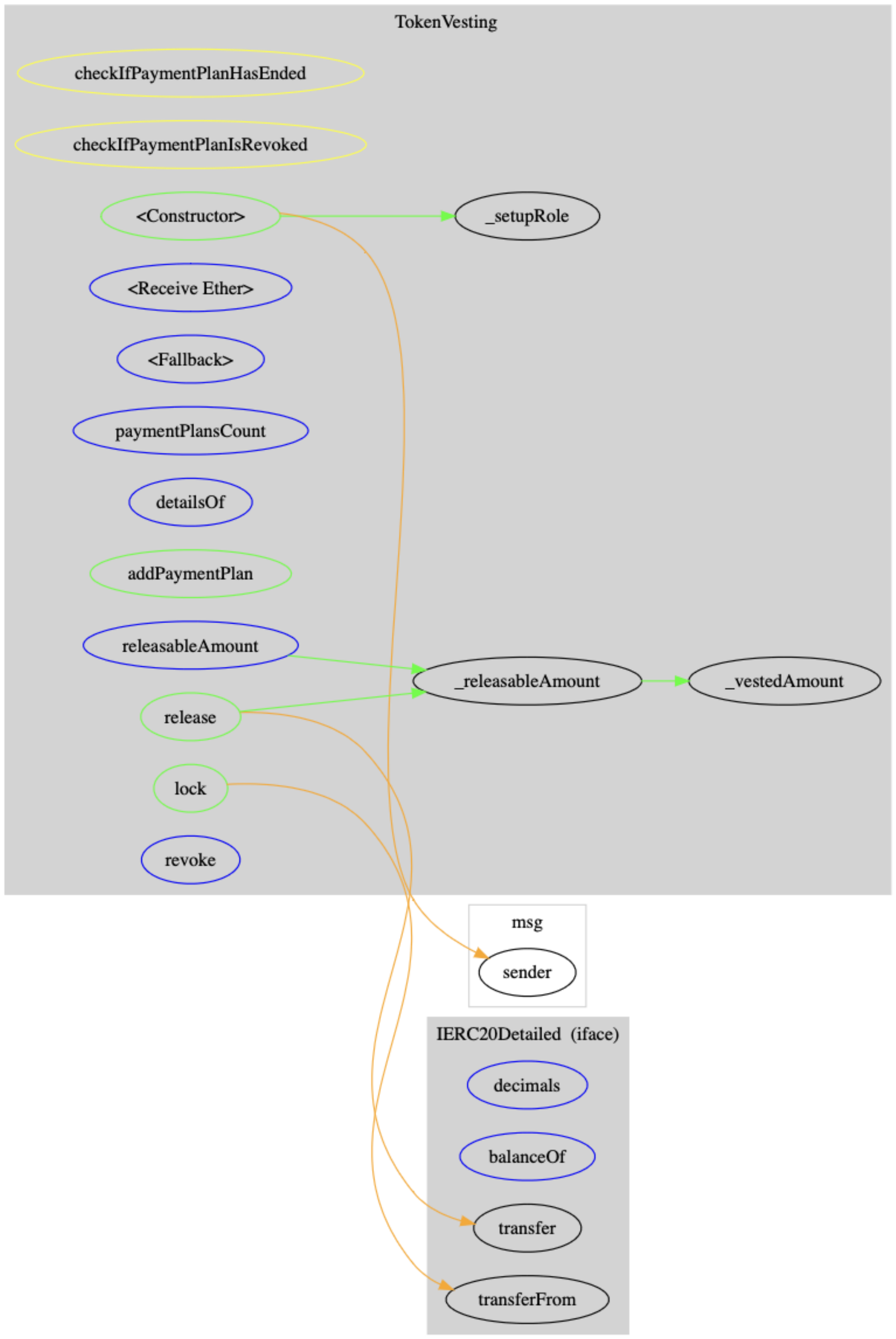
### 1.3 Use Cases

1. The vesting admin creates different vesting plans. A period plan should contain following information: duration of 1 vesting period, total number of periods, cliff period.
2. And admin can lock funds using one of the vesting plans. Tokens for each active user vesting should be guaranteed by the contract.
3. Users can withdraw their funds according to a provided payment plan. Tokens can be unlocked after a start of each period in equal parts. If

cliff period is provided, the user should not be able to withdraw funds befor it passes.

# 2. Technical Requirements

## 2.1. Architecture Overview



## 2.2. Contract Information

This section contains detailed information (their purpose, assets, functions, and events) about the contracts used in the project.

### 2.2.1. TokenVesting.sol

A token holder contract that can release its token balance gradually like a typical vesting scheme, with a cliff and vesting period. Optionally revocable by the admin.

#### 2.2.1.1. Assets

Hacken.IO Sample Token Vesting contains two Structs:

- **PaymentPlan:** This object contains information about a payment plan.
  - uint256 periodLength - length of 1 period in seconds.
  - uint256 periods - total vesting periods.
  - uint256 cliffPeriods - number of periods that will be skipped.
  - bool revoked - True if plan is revoked, false otherwise.
- **Lock:** This structure holds information about a user participation in a payment plan.
  - address beneficiary - address of the beneficiary to whom vested tokens are transferred.
  - uint256 start - start the time (Unix timestamp), at which point vesting starts.
  - uint256 paymentPlan - payment plan to apply.
  - uint256 totalAmount - total amount to be unlocked.
  - uint256 released - the released amount of the lock (so far).

Besides the mentioned structs, the following entities are present in the project:

- **paymentPlans**: A public array of type Struct PaymentPlan holds the created payment plans of the Hacken.IO Sample Token Vesting.
- **lock:** An address <-> Lock object mapping. Aims to store a lock assigned for a specific address.
- **token:** ERC20 token address.
- **PERCENT_100:** Constant denominator for %100.

### 2.2.1.2. Modifiers

Hacken.IO Sample Token Vesting has the following modifiers:

- **checkIfPaymentPlanIsRevoked(PaymentPlan plan):** Checks if the given payment plan has been revoked.

### 2.2.1.3. Functions

Hacken.IO Sample Token Vesting has the following functions:

- **constructor(IERC20Detailed _token):** Handles configurations and sets related addresses.
- **paymentPlansCount():** Returns number of payment plans created so far.
- **addPaymentPlan(uint256 periodLength, uint256 periods, uint256 cliffPeriods):** Allows a vesting admin to add a new payment plan.
- **lock(address beneficiary, uint256 amount, uint256 start, uint256 paymentPlan):** Allows an admin to lock funds into a payment plan. This payment plan must not be revoked.
- **releasableAmount(address beneficiary):** Returns an amount that can be unlocked at this moment.
- **release(address beneficiary):** Allows a user to release his/her lock.
- **revoke(uint256 paymentPlanId):** External function allows a vesting admin to revoke a payment plan.