HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: CropToken
Date: February 3rd, 2021

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

Document

Name	Smart Contract Code Review and Security Analysis Report for CropToken.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Token with vesting.
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	
Commit	
Deployed contract	https://ropsten.etherscan.io/address/0x7E5B7d03af060c88A129aF45378023E6360e7657#code
Timeline	27 JAN 2020 - 10 FEB 2021
Changelog	29 JAN 2021 - INITIAL AUDIT 03 FEB 2021 - SECOND REVIEW 10 FEB 2021 - FINAL REPORT

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

Table of contents

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

Introduction

Hacken OÜ (Consultant) was contracted by CropToken (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 January 27th, 2021 - January 29th, 2021.

Second review conducted on February 3rd, 2021.

Scope

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

```
Contract deployment address:
Repository
File:
      CropToken.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	■ Reentrancy ■ Ownership Takeover ■ Timestamp Dependence ■ Gas Limit and Loops ■ DoS with (Unexpected) Throw ■ DoS with Block Gas Limit ■ Transaction-Ordering Dependence ■ Style guide violation ■ Costly Loop ■ ERC20 API violation ■ Unchecked external call ■ Unchecked math ■ Unsafe type inference ■ Implicit visibility level ■ Deployment Consistency ■ Repository Consistency ■ Data Consistency

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

Functional review	■ Business Logics Review ■ Functionality Checks ■ Access Control & Authorization ■ Escrow manipulation ■ Token Supply manipulation ■ Assets integrity ■ User Balances manipulation ■ Kill-Switch Mechanism ■ Operation Trails & Event Generation

Executive Summary

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

Insecure	Poor secured	Secured	Well-secured

You are 

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.

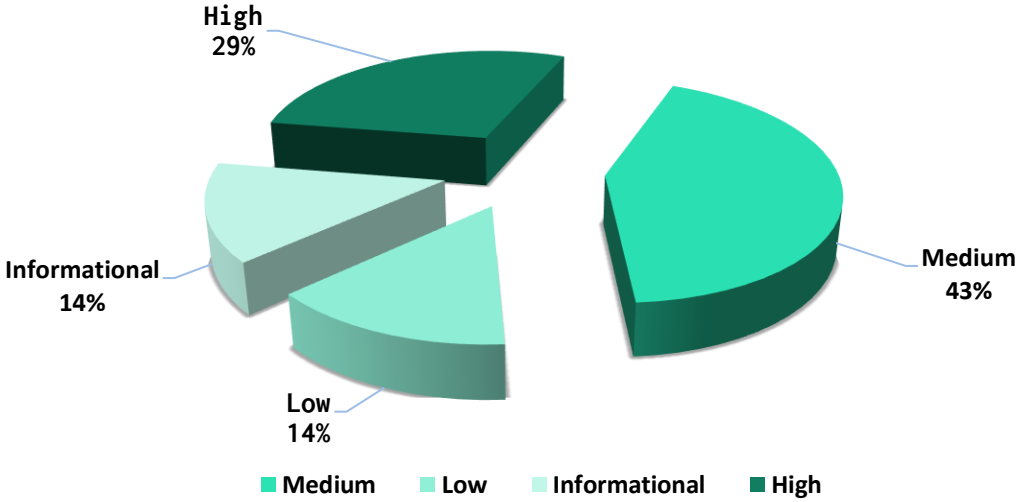
Security engineers found **2** high, **3** medium, **1** low and **1** informational issues during the audit.

After the second review Customers` smart contracts contains: **1** medium, **1** low and **1** informational issues during the audit.

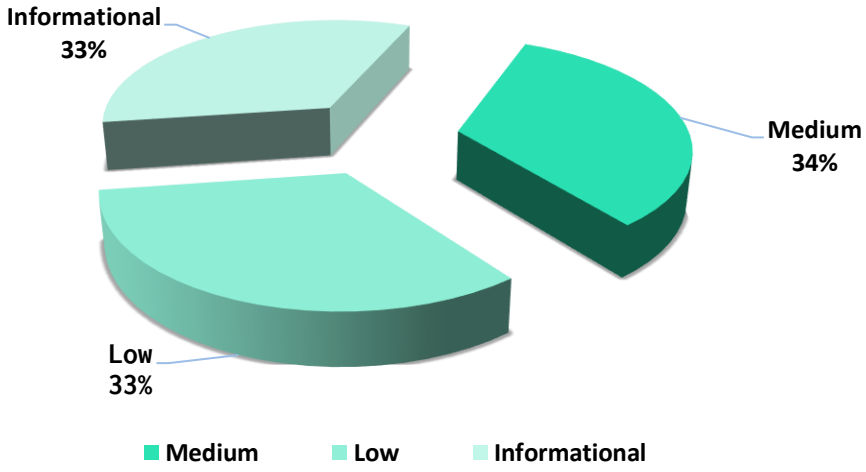
Notice: the contract is Pausable. That means that owners may stop all transfers.

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

Graph 1. The distribution of vulnerabilities after the first review.



Graph 2. The distribution of vulnerabilities after the second review.



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

Severity Definitions

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

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

AS-IS overview

CropToken.sol

Description

CropToken is an ERC-20 token with vesting functionality. The contract is designed in a way so that it can be used from proxy.

Initial tokens allocations:

1000000 - for 0x19a953816F5dCb7cCDb8b37bB684320304E2fD18 address.

1050000 - for 0x0dAE97dE55Af3243B6cD95F1Ffc3443f1da77B72 address.

Imports

CropToken contract has the following imports:

- @openzeppelin/contracts-upgradeable/token/ERC20/ERC20PausableUpgradeable.sol
- @openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol
- @openzeppelin/contracts-upgradeable/proxy/Initializable.sol

Inheritance

CropToken contract is Initializable, OwnableUpgradeable, ERC20PausableUpgradeable.

Usages

CropToken contract has no custom usages.

Structs

CropToken contract has the following data structures:

- FrozenWallet
- VestingType

Enums

CropToken contract has no custom enums.

Events

CropToken contract has no custom events.

Modifiers

CropToken has no custom modifiers.

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

Fields

CropToken contract has following constants:

- mapping(address => FrozenWallet) public frozenWallets;
- VestingType[] public vestingTypes;

Functions

CrowdSale has following public functions:

- ***initialize***
  Description
  Initializes the contract. Sets Token name and symbol. Mints initial supply of tokens. Initializes vesting types.
  Visibility
  public
  Input parameters
  None
  Constraints
  - `initializer` modifier.

  Events emit
  None
  Output
  None
- ***getReleaseTime, getMaxTotalSupply, getTimestamp, getMonths, isStarted***

  Description
  Simple getter functions.
- ***mulDiv - removed before the second review***
  Description
  Multiplies `x` to `y` and divides the result by `z`. May lead to overflow.
  Visibility
  public pure
  Input parameters
  - uint256 x
  - uint256 y
  - uint256 z

  Constraints
  None
  Events emit
  None
  Output
  uint256
- ***addAllocations***
  Description

Mints tokens with a `vestingTypeIndex` vesting type.
Visibility
external payable
Input parameters
- address[] memory addresses,
- uint256[] memory totalAmounts,
- uint256 vestingTypeIndex

Constraints
- onlyOwner modifier.

Events emit
None
Output
None

- ***getTransferableAmount***
  Description
  Returns an amount of tokens that are available to be transferred by `sender`.
  Visibility
  public view
  Input parameters
  - address sender

  Constraints
  None
  Events emit
  None
  Output
  uint256
- ***getRestAmount***
  Description
  Returns an amount of tokens that are still locked.
  Visibility
  public view
  Input parameters
  - address sender

  Constraints
  None
  Events emit
  None
  Output
  uint256
- ***canTransfer***
  Description
  Check whether a `sender` is allowed to transfer an `amount`.
  Visibility
  public view
  Input parameters

    - address sender
    - uint256 amount

  Constraints
  None
  Events emit
  None
  Output
  bool

- ***withdraw - removed before the second review***
  Description
  Transfers an `amount` of ETH to the message sender.
  Visibility
  public
  Input parameters
    - uint256 amount

  Constraints
    - onlyOwner modifier.

  Events emit
  None
  Output
  None

- ***pause***
  Description
  Changes the contract pause status to `status`.
  Visibility
  public
  Input parameters
    - bool status

  Constraints
    - onlyOwner modifier.

  Events emit
  None
  Output
  None

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

Audit overview

Critical

No critical issues were found.

High

1. The token description says:

    a. Team tokens: a total of 8.950.000 CROP tokens locked for 180 days then 5% released each month for 20 months thereafter.

    b. Tokens sold on the seed sale: a total of 6.000.000 CROP tokens 10% released immediately after TGE, 10% released every month for 9 months thereafter.

    c. 65.000.000 tokens allocated as staking/liquidity mining/yield farming rewards will only be minted underways with a vesting schedule when the different platforms has been deployed

    d. Private sale tokens: a total of 6.000.000 CROP tokens 10% released immidiatly after TGE, 15% released monthly thereafter for 6 months thereafter

    e. Public ico tokens: a total of 1.000.000 CROP tokens

    f. Development tokens: a total of 12.000.000 CROP tokens 5% released immidiatly after TGE and 5% thereafter each month for 19 months

    g. Uniswap innitial liquidity: a total of 1.000.000 CROP tokens (liquidity locked)

    h. Hotbit innitial liquidity: a total of 50.0000 CROP tokens

    None of the described locks are implemented in the code.

    Partial Fixed. All locks are assigned in the migration scripts.

2. The `mulDiv` function performs unsafe math operations.

    Fixed before the second review.

Medium

1. The `addAllocations` has no reasons to be payable.

We recommend to forbid ETH transfers to the contract and to remove the `withdraw` function.

Fixed before the second review.

2. The `scheduled` value of the `FrozenWallet` is never assigned as `true`.

   This variable may be removed.

   Customer feedback: It is assigned true in FrozenWallets creation.

   Hacken feedback: It is not. The `scheduled` variable is used 3 times in the contract:

   1. Declaration in the FrozenWallet structure (Default value for bool is `true`).
   2. Line 149 - `if` condition.
   3. Line 238 - `if` condition.

3. The `addAllocations` function has no input arrays size validation.

   We recommend to add validation that will check whether input arrays are of the same size.

   Fixed before the second review.

Low

1. The contract has a lot of hardcoded values that can be moved to named constants and be assigned in the `initialize` function.

Lowest / Code style / Best Practice

1. We recommend using exponential form of numbers instead of using standard form.

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

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 **2** high, **3** medium, **1** low and **1** informational issues during the audit.

After the second review Customers` smart contracts contains: **1** high, **1** medium, **1** low and **1** informational issues during the audit.

Notice: the contract is Pausable. That means that owners may stop all transfers.

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

Category	Check Item	Comments
Code review	■ Business Logics Review	■ No tokens are locked after the contract initialization as it is stated in the description.

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

Disclaimers

Hacken 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.