HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Harvester DAO
Date: June 24th, 2021

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

This document 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 containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed - upon a decision of the Customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for Harvester DAO
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Staking
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Git repository	https://github.com/harvester-dao/contracts/blob/8bafff49e74b7bd137e3741988106e9583be46c5/GreatHarvester.sol
Timeline	21 JUNE 2021 - 24 JUNE 2021
Changelog	24 JUNE 2021 - INITIAL AUDIT 29 JUNE 2021 - SECOND REVIEW

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 Harvester DAO (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 June 21st, 2021 - June 24th, 2021. The second review conducted on June 29th, 2021.

Scope

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

Repository:
https://github.com/harvester-dao/contracts/blob/8bafff49e74b7bd137e3741988106e9583be46c5/GreatHarvester.sol

Out of scope:

- Crop.sol
- GHTimelock.sol
- GreatHarvester.sol
- Harvester.sol
- IUniswapV2Factory.sol
- IUniswapV2Pair.sol
- Create IUniswapV2Pair.sol
- IUniswapV2Router01.sol
- IUniswapV2Router02.sol

We have scanned these smart contracts 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
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Asset’s integrity ▪ User Balances manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

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

Executive Summary

According to the assessment, the Customer's smart contract is well-secured.

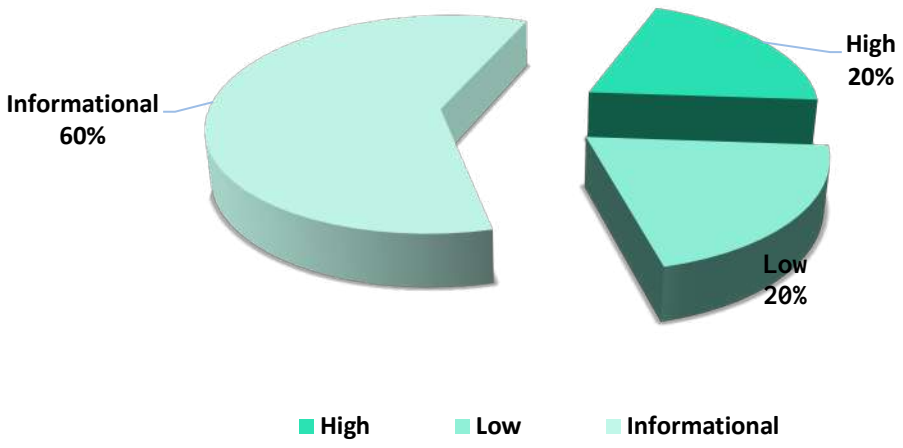
Insecure	Poor secured	Secured	Well-secured
			You are here

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. All found issues can be found in the Audit overview section.

Security engineers found **1** high, **1** low and **3** informational issues during the first review.

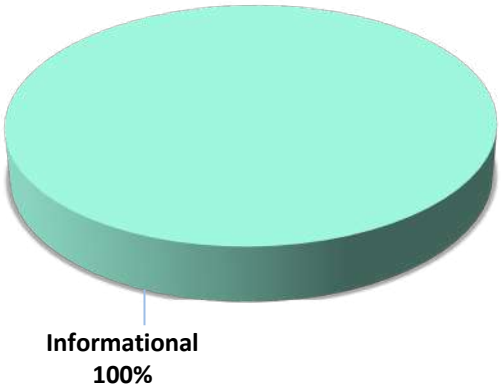
Security engineers found **1** informational issue during the second review.

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



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

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

Audit overview

Critical

No Critical severity issues were found.

High

Vulnerability: Reward allocation manipulation

While it's allowed to set allocPoints of the farm (farms[_fid]) to zero (0), which will disallow all rewards and allowing to call **setFarm** function with argument _sync set to false, which will NOT recalculate already received rewards, we see here possible reward allocation manipulation by the contract owner.

Recommendation: Please consider adding implicit call to the **farms[_fid]** if _sync is false.

Fixed before the second review.

Medium

No Medium severity issues were found.

Low

Vulnerability: Missing zero address validation

Accidentally setting the treasury address to 0x0 could lead to unexpected transaction reversions.

Recommendation: Please consider adding zero-address validation in constructor and changeTreasury function.

Fixed before the second review.

Lowest / Code style / Best Practice

1. **Vulnerability:** No purpose events

Event is emitted even if the value is not changed.

Lines: #341-344

```
function changeTreasury(address _treasury) external onlyGovernance {
  emit TreasuryChanged(treasury, _treasury);
  treasury = _treasury;
}
```

Lines: #347-350

```
function toggleFeeRotation() external onlyGovernance {
  hasFeeRotation = !hasFeeRotation;
  emit FeeRotationToggled(!hasFeeRotation, hasFeeRotation);
}
```

Lines: #353-357

```
function alterCropGrowth(uint256 _cropPerBlock) external onlyGovernance
{
  massSyncFarms();
  emit CropGrowthAltered(cropPerBlock, _cropPerBlock);
  cropPerBlock = _cropPerBlock;
}
```

Lines: #360-364

```
function delayStartBlock(uint256 _startBlock) external onlyGovernance {
  require(block.number < startBlock, "GreatHarvester::delayStartBlock:
too late to delay farming");
  emit RewardsStartDelayed(startBlock, _startBlock);
  startBlock = _startBlock;
}
```

2. **Vulnerability**: Boolean equality

   Boolean constants can be used directly and do not need to be compared to **true** or **false**.

   Recommendation: Please consider removing the equality to the boolean constant.

   Fixed before the second review.

3. **Vulnerability:** Public function that could be declared external

   public functions that are never called by the contract should be declared **external** to save gas.

   Fixed before the second review.

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.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security engineers found **1** high, **1** low and **3** informational issues during the first review.

Security engineers found 1 informational issue during the second review.

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 the 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.