HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: YFD

Date: January 14th, 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 and 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 fixed - upon a decision of the Customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for YFD (15 pages)
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Vaults
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Github	HTTPS://GITHUB.COM/YFDFI-FINANCE/YFD-FARMING-VAULT
Commit	A3218A3E62570B5CFE4EC227CE917E54BCAC9B55
Timeline	11TH JAN 2021 – 14TH JAN 2021
Changelog	14TH JAN 2021 - Initial Audit

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 YFD (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 contract and its code review conducted between January 11th, 2021 – January 14th, 2021.

Scope

The scope of the project is github repository:

Github HTTPS://GITHUB.COM/YFDFI-FINANCE/YFD-FARMING-VAULT
CommitA3218A3E62570B5CFE4EC227CE917E54BCAC9B55

Files in scope of review
./ Vaultlocking1month.sol
./ Vaultlocking2months.sol
./ Vaultlocking3months.sol
./ Vaultlocking72h.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 ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts don’t have critical vulnerabilities and can be considered secure.

We described issues in the conclusion of these documents. Please read the whole document to estimate the risks well.

Insecure	Poor secured	Secured	Well-secured

You are 

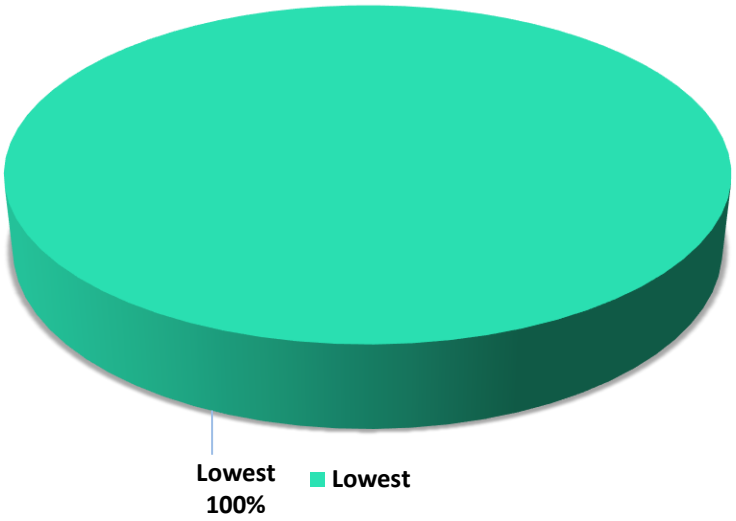
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. A general overview is presented in AS-IS section, and all found issues can be found in the Audit overview section.

Security engineers found **1** lowest severity issues during the audit.

[1] Look for details and justification in conclusion section

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

Graph 1. The distribution of vulnerabilities.



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 essential 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

Description

VaultProRewardMonth, *VaultProReward*, *VaultProReward3Months* and *VaultProReward2Months* are the same contract with different values of constants. So, it can be audited as one contract named *VaultProReward****.

Imports

*VaultProReward**** contract has 4 imports:

- *SafeMath* — from *OpenZeppelin*
- *EnumerableSet* — from *OpenZeppelin*
- *Ownable* — from *OpenZeppelin*
- *Token* — is an ERC20 token interface

Inheritance

*VaultProReward**** contract inherits *Ownable*.

Usings

*VaultProReward**** contract use:

- *SafeMath* for *uint*;
- *EnumerableSet* for *EnumerableSet.AddressSet*;

Fields

*VaultProReward**** contract has 21 fields:

- *address public trustedDepositTokenAddress* — deposit token address;
- *address public trustedRewardTokenAddress* — reward token address;
- *uint public constant withdrawFeePercentX100* — withdrawal fee;
- *uint public constant disburseAmount* — disbursement amount;
- *uint public constant disburseDuration* — disbursement duration;
- *uint public constant cliffTime* — the period of time when unstaking is not available;
- *uint public constant disbursePercentX100* — disbursement percent rate;
- *uint public contractDeployTime* — a timestamp when the contract was deployed;

- *uint public lastDisburseTime* — a timestamp of the last disbursement;
- *uint public totalClaimedRewards* — the total number of claimed rewards;
- *EnumerableSet.AddressSet private holders* — a set of the address of holders;
- *mapping (address => uint) public depositedTokens* — deposited tokens amount by address;
- *mapping (address => uint) public depositTime* — a timestamp of the last deposit by address;
- *mapping (address => uint) public lastClaimedTime* — a timestamp of the last rewards claimed by address;
- *mapping (address => uint) public totalEarnedTokens* — the total number of earned tokens by address;
- *mapping (address => uint) public lastDivPoints* — snapshot of the total div points;
- *uint public totalTokensDisbursed* — the total number of disbursed tokens;
- *uint public contractBalance* — the number of tokens on the balance of the contract;
- *uint public totalDivPoints* — the total div points;
- *uint public totalTokens* — the total number of deposited tokens;
- *uint internal pointMultiplier* — the point multiplier;

Functions

*VaultProReward**** contract has 13 functions:

- ***constructor***

  Description

  Initializes the contract. Sets *contractDeployTime* and *lastDisburseTime*.

  Visibility

  public

  Input parameters

  None

  Constraints

None

Events emit

None

Output

None

- ***addContractBalance***

  Description

  Used by admin to add tokens to the contract balance.

  Visibility

  public

  Input parameters

  None

  Constraints

  - Only owner can call it.
  - Tokens should be transferred.

  Events emit

  None

  Output

  None

- ***updateAccount***

  Description

  Used to pay rewards to account.

  Visibility

private

Input parameters

- *address account* — an address of the account;

Constraints

- Tokens should be transferred.

Events emit

- *RewardsTransferred(account, pendingDivs);*

Output

None

- ***getPendingDivs***

Description

Used to calculate the holder reward.

Visibility

public view

Input parameters

- *address _holder* — an address of the holder;

Constraints

None

Events emit

None

Output

Returns reward amount.

- ***getNumberOfHolders***

  Description

  Used to get the number of holders.

  Visibility

  public view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns the number of holders.

- ***deposit***

  Description

  Used to deposit tokens.

  Visibility

  public

  Input parameters

  - *uint amountToStake* — an amount of tokens to deposit;

  Constraints

  - The deposit amount should be greater than 0.
  - Tokens should be transferred.

Events emit

None

Output

None

- ***withdraw***

  Description

  Used to withdraw.

  Visibility

  public

  Input parameters

  - *uint amountToWithdraw* — an amount of tokens to withdraw;

  Constraints

  - The withdraw amount should be less than or equal to deposited amount.
  - The *cliffTime* period must be passed.
  - Fee should be transferred.
  - Tokens should be transferred.

  Events emit

  None

  Output

  None

- ***emergencyWithdraw***

  Description

  Used to withdraw without rewards.

Visibility

public

Input parameters

- *uint amountToWithdraw* — an amount of tokens to withdraw;

Constraints

- The withdraw amount should be less than or equal to deposited amount.
- The *cliffTime* period must be passed.
- Fee should be transferred.
- Tokens should be transferred.

Events emit

None

Output

None

- ***claim***

Description

Used to claim rewards.

Visibility

public

Input parameters

None

Constraints

None

Events emit

None

Output

None

- ***distributeDivs***

  Description

  Increases total div points.

  Visibility

  private

  Input parameters

  - *uint amount* — an amount of tokens;

  Constraints

  None

  Events emit

  - *emit RewardsDisbursed(amount);*

  Output

  None

- ***disburseTokens***

  Description

  Used to disburse tokens.

  Visibility

  public

  Input parameters

None

Constraints

- Only owner can call it.

Events emit

None

Output

None

- ***getPendingDisbursement***

  Description

  Calculates disbursement amount.

  Visibility

  public view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns disbursement amount.

- ***getDepositorsList***

  Description

Used to get depositors list.

Visibility

public view

Input parameters

- *uint startIndex* — index from;
- *uint endIndex* — index to;

Constraints

- *startIndex* should be less than *endIndex*.

Events emit

None

Output

Returns depositors info.

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

Audit overview

Critical

No critical issues were found.

High

No high issues were found.

Medium

No medium issues were found.

Low

No low severity issues were found.

Lowest / Code style / Best Practice

1. According to the best practices constants should be named as UPPER_CASE_WITH_UNDERSCORES.

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.

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

Security engineers found **1** lowest severity issues during the audit.

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

Category	Check Item	Comments
Code review	▪ Style guide violation	▪ According to the best practices constants should be named as UPPER_CASE_WITH_UNDERSCORES.

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 the 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 the 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 own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.