

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT



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	Staking
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Contract	0X28F2E8822B6EB1BBA8F1C8AB8DF970D813B677D4
Timeline	11 TH JAN 2021 – 14 TH JAN 2021
Changelog	13 TH JAN 2021 - Initial Audit 15 TH JAN 2021 – SECONDARY Audit



Table of contents

Document	2
Table of contents	3
Introduction	4
Scope	4
Executive Summary	5
Severity Definitions	7
AS-IS overview	8
Conclusion	16
Disclaimers.....	17

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. Secondary audit January 15th, 2021.

Scope

The scope of the project is smart contract in the mainnet:

0X28F2E8822B6EB1BBA8F1C8AB8DF970D813B677D4

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	<ul style="list-style-type: none">ReentrancyOwnership TakeoverTimestamp DependenceGas Limit and LoopsDoS with (Unexpected) ThrowDoS with Block Gas LimitTransaction-Ordering DependenceStyle guide violationCostly LoopERC20 API violationUnchecked external callUnchecked mathUnsafe type inferenceImplicit visibility levelDeployment ConsistencyRepository ConsistencyData Consistency

Functional review	<ul style="list-style-type: none"> ■ 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 contract does not 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.



You are

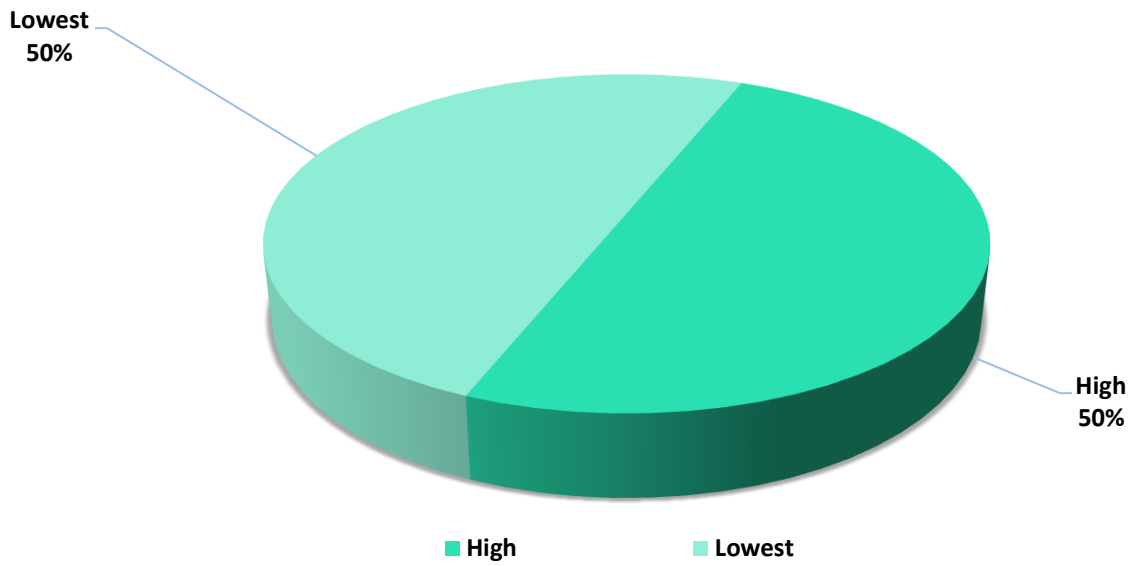


Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. A general overview is presented in AS-IS section,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

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

¹ Look for details and justification in conclusion section

Graph 1. The distribution of vulnerabilities.



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.

AS-IS overview

Description

YfDFI_staking is a contract that allows users to make deposits and to get a reward for it.

Imports

YfDFI_staking contract has 4 imports:

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

Inheritance

YfDFI_staking contract inherits *Ownable*.

Usings

YfDFI_staking contract use:

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

Fields

YfDFI_staking contract has 12 fields:

- *address public constant tokenAddress* — YfDFI token contract address;
- *uint public constant rewardRate* — a reward rate;
- *uint public constant rewardInterval* — a reward interval;
- *uint public constant stakingFeeRate* — a staking fee rate;
- *uint public constant unstakingFeeRate* — an unstaking fee rate;
- *uint public constant cliffTime* — the period of time when unstaking is not available;
- *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 stakingTime* — 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;
- *uint private constant stakingAndDaoTokens*

Functions

YfDFI_staking contract has 8 functions:

- ***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

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.



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.
- Fee 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

- *claimDivs*

Description

Used to claim rewards.

Visibility

public

Input parameters

None

Constraints

None

Events emit

None

Output

None

- ***getStakingAndDaoAmount***

Description

Calculates how many tokens the administrator can withdraw.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns the number of tokens.

- ***transferAnyERC20Tokens***

Description

Used by admin to withdraw ANY tokens.

Visibility

public

Input parameters

- *address_tokenAddr* — an address of token;
- *address_to* — an address of tokens receiver;
- *uint_amount* — an amount of tokens;

Constraints



- Only owner can call it.

Events emit

None

Output

None

Audit overview

■ ■ ■ ■ Critical

No critical issues were found.

■ ■ ■ High

1. *transferAnyERC20Tokens* allows the owner to transfer *YfDFI* tokens out from the contract. Out tokens amount limited by *getStakingAndDaoAmount* function but it does not guarantee deposit safety.

Since ownership was transferred to the contract itself (December 25th for [this](#) transaction), this issue cannot be exploited.

■ ■ 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.

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** high and **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	<ul style="list-style-type: none">Style guide violation	<ul style="list-style-type: none">According to the best practices constants should be named as UPPER_CASE_WITH_UNDERSCORES.
Functional review	<ul style="list-style-type: none">User Balances manipulation	<ul style="list-style-type: none"><i>transferAnyERC20Tokens</i> allows the owner to transfer <i>YfDFI</i> tokens out from the contract. Out tokens amount limited by <i>getStakingAndDaoAmount</i> function but it does not guarantee deposit safety.

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.