SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: FEAR
Date: October 29th, 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 FEAR.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 token; Staking
Platform	Polygon / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Zip archive	FearContracts.zip
Commit	8b88156e1cfe5a3fd9879af6a39ea601518513d6
Technical Documentation	YES
JS tests	YES
Website	fear.io
Timeline	25 OCTOBER 2021 – 29 OCTOBER 2021
Changelog	29 OCTOBER 2021 – INITIAL AUDIT

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by FEAR (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 October 25th, 2021 - October 29th, 2021.

Scope

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

Zip archive:
FearContracts.zip
Commit:
8b88156e1cfe5a3fd9879af6a39ea601518513d6
Technical Documentation: Yes, provided in text
JS tests: Yes
Contracts:
FearPlayToEarn.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

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 are well-secured.

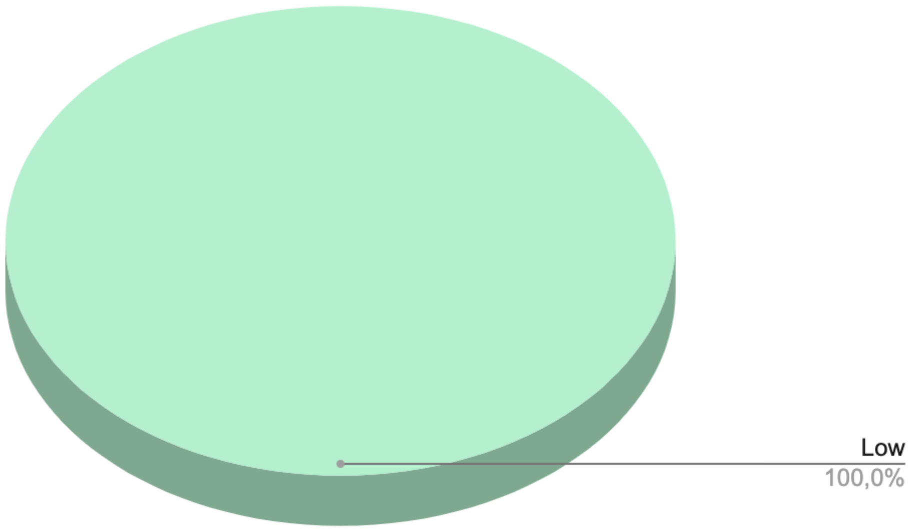
Insecure	Poor secured	Secured	Well-secured

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.

As a result of the audit, security engineers found **4** low severity issues.

Graph 1. The distribution of vulnerabilities after the audit.



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

Audit overview

🟥 🟥 🟥 🟥 Critical

No critical issues were found.

🟨 🟨 🟨 High

No high severity issues were found.

🟩 🟩 Medium

No medium severity issues were found.

🟦 Low

1. Unavailable tokens after changing milestones (if only one milestone is active at the time) or inconsistency in view methods (if more than one milestones are active at the time).

   Contracts: FearPlayToEarn.sol

   Every milestone has a single flag **enabled** that controls users' ability to both stake and unstake. If only one milestone is enabled at the time, the user is not able to unstake tokens from past milestones. If more than one milestone is enabled, information in variables fearStakers and stakingTime could become inconsistent.

   Recommendation: add the ability to unstake from disabled milestones, store set of stakers and staking time separately for each milestone.

2. The owner has too broad power over the contract.

   The owner has the ability to directly change the number of tokens staked for any address, withdraw any amount of any ERC20 token from the staking contract, including the FEAR token itself, all that without emitting any event.

   Contracts: FearPlayToEarn.sol

   Functions: setPlayerTokens, emergencyAssetWithdrawal

   Recommendation: Remove the ability for the owner to directly manipulate with player tokens, add a check to prevent withdrawal FEAR tokens. If described functionality is necessary, please emit events on these methods for better tracking off-chain.

3. Missing event for changing fearToken, nftToken, stakeLockPeriod

   Contracts: FearPlayToEarn.sol

   Functions: setPlayToEarn, setFearTokenContract, setStakeLockPeriod

   Changing critical values should be followed by the event emitting for better tracking off-chain.

   Recommendation: Please emit events on the critical values changing.

4. A public function that could be declared external

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

   Contracts: FearPlayToEarn.sol

   Function: getRemainingStakeAllowed

   Recommendation: Use the **external** attribute for functions never called from the contract.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

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

As a result of the audit, security engineers found **4** low severity issues.

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, bug-free 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 a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.