Audit Whale

Smart contract code review and security analysis report

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 CFC.
Approved by	Jameson | COO AuditWhale
Type	ERC20 tokens; ERC721 token; Staking
Platform	Binance / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/Crypto-Fight-Club/smart-contracts/tree/main/contracts
Commit	371ada481411f0000f665c64c983c484df7382a3
Technical Documentation	YES
JS tests	YES
Website	cryptofightclub.io
Timeline	01 DECEMBER 2021 – 13 DECEMBER 2021
Changelog	10 DECEMBER 2021 – INITIAL AUDIT 13 DECEMBER 2021 – SECOND REVIEW

Table of contents

Introduction

AuditWhale (Consultant) was contracted by CFC (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 December 1st, 2021 - December 10th, 2021.

Second review conducted on December 13th, 2021.

Scope

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

Repository:
https://github.com/Crypto-Fight-Club/smart-contracts/tree/main/contracts
Commit:
371ada481411f0000f665c64c983c484df7382a3
Technical Documentation: Yes, https://docs.google.com/document/d/1k3l-uUbys8Kg0WlNSlR-n89RPzfNludo5BnNLkAfSBM
JS tests: Yes, https://github.com/Crypto-Fight-Club/smart-contracts/tree/main/test
Contracts:
CutOffMultiRecipients.sol
FightToken.sol FighterNFT.sol
StakingFight.sol StakingLP.sol
wrappedFight.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.

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 6 low severity issues. After the second review, security engineers found 2 low severity issues.

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. Some tests are failing

   Testcase *"StakeandUnstakeBasicWithoutReward"* triedtostakewith zero duration causing a fail to itself and all sequential cases.

   Recommendation: Fix the test.

   Status: case fixed, but introduced one failing test in vesting.js. As of staking.js — we recommend extending test coverage including all possible scenarios, eg: unstaking with late penalty etc. Consider using solidity-coverage plugin to check methods and branches that are not yet covered.

2. Outdated versions of Solidity

   Both versions of Solidity used in the project(0.6.2 and 0.8.0) are outdated and not recommended.

   Recommendation: Consider switching to 0.6.12 and 0.8.9+ Status:

   fixed

3. Misleading revert message

   Revert message included double negative and is not consistent with revert condition

   Contracts: FighterNFT.sol

   Modifier: noContract()

   Recommendation: Change revert message.

   Status: fixed

4. Missing event for changing maxUserMint, fightRequiredForMint

   Contracts: FighterNFT.sol

   Functions: pushNewBatch

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

   Recommendation: Please emit events on the critical values changing.

   Status: fixed

5. State variables that could be declared constant

   Constant state variables should be declared constant to save gas.

   Contracts: CutOffMultiRecipients.sol

   Variables: divisionFactor

   Recommendation: Add the constant attributes to state variables that never change.

   Status: not fixed. The immutable variable is not a constant

6. 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: FighterNFT.sol, StakingLP.sol, StakingFight.sol

   Functions: setTokenURI, setBaseURI, unstake

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

   Status: fixed

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 6 low severity issues. After the second review, security engineers found 2 low severity issues.

Disclaimers

AuditWhale 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.