# Smart contract security audit

# Pokechain

v.1.2

# Table of Contents


**1.0 Introduction** **3**

1.1 Project engagement 3

1.2 Disclaimer 3

**2.0 Coverage** **4**

2.1 Target Code and Revision 4

2.2 Attacks made to the contract 5

**3.0 Security Issues** **7**

3.1 High severity issues [0] 7

3.2 Medium severity issues [2] 7

3.3 Low severity issues [1] 8

**4.0 Summary of the audit** **9**

# 1.0 Introduction

## 1.1 Project engagement

During June of 2021, Pokechain engaged CTDSec to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. Pokechain provided CTDSec with access to their code repository and whitepaper.

## 1.2 Disclaimer

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the network's fast-paced and rapidly changing environment, we at CTDSec recommend that Pokechain team put in place a bug bounty program to encourage further and active analysis of the smart contract.

# 2.0 Coverage

## 2.1 Target Code and Revision

For this audit, we performed research, investigation, and review of the Pokechain contract followed by issue reporting, along with mitigation and remediation instructions outlined in this report. The following code files are considered in-scope for the review:

Source:

https://github.com/Pokechain/Pokechain/blob/master/Pokemon.sol

## 2.2 Attacks made to the contract

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices.

| № | Issue description. | Checking status |
|---|---|---|
| 1 | Compiler warnings. | PASSED |
| 2 | Race conditions and Reentrancy. Cross-function race conditions. | PASSED |
| 3 | Possible delays in data delivery. | PASSED |
| 4 | Oracle calls. | PASSED |
| 5 | Front running. | PASSED |
| 6 | Timestamp dependence. | PASSED |
| 7 | Integer Overflow and Underflow. | PASSED |
| 8 | DoS with Revert. | PASSED |
| 9 | DoS with block gas limit. | PASSED |
| 10 | Methods execution permissions. | PASSED |
| 11 | Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc. | PASSED |
| 12 | The impact of the exchange rate on the logic. | PASSED |
| 13 | Private user data leaks. | PASSED |
| 14 | Malicious Event log. | PASSED |
| 15 | Scoping and Declarations. | PASSED |
| 16 | Uninitialized storage pointers. | PASSED |
| 17 | Arithmetic accuracy. | PASSED |

| | | |
|---|---|---|
| 18 | Design Logic. | SOLVED BY DEV TEAM |
| 19 | Cross-function race conditions. | PASSED |
| 20 | Safe Zeppelin module. | PASSED |
| 21 | Fallback function security. | PASSED |
| 22 | Overpowered functions / Owner privileges | SOLVED BY DEV TEAM |

# 3.0 Security Issues

## 3.1 High severity issues [0]

No high severity issues found.

## 3.2 Medium severity issues [2]

**1. Fees aren't limited**

```
function setMaxTxPercent(uint256 maxTxPercent) public onlyOwner() {
        _maxTxAmount = _tTotal.mul(maxTxPercent).div(10000);
    }
function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
        _taxFee = taxFee;
    }
    function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner() {
        _liquidityFee = liquidityFee;
    }
```

The indicated functions of the owner of the contract do not have any limits.

Recommendation:

Apply limits to these functions.

**2. Random numbers are generated onchain**

The contract creates a lottery system which generates random numbers within the blockchain. Historically it has been shown that generating randoms onchains is hackable.

We recommend obtaining other types of solutions such as chainlink VRF.

## 3.3 Low severity issues [1]

1. Comments

bool public swapAndLiquifyEnabled = false; // should be true

By default swapandliquifyenabled is false it should be called during the activation.

# 4.0 Summary of the audit

All smart contract issues were solved in the next commit:
c35bc982c50d08a280ccbfadfae47c4235b161ec
https://github.com/Pokechain/Pokechain/commit/c35bc982c50d08a280ccbfadfae47c4235b161ec#diff-0a36d093bd6774d464fb1a9b3ba04bea2ca35c664eeb2f678676c1443cb41c89

Lottery functions were deleted and owner privileges are limited now.