

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: CryptoDragons

Date: November 5th, 2021



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 CryptoDragons.		
Approved by	Andrew Matiukhin CTO Hacken OU		
Туре	ERC20 token; ERC721 token		
Platform	Ethereum / Solidity		
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review		
File name	<pre>smartcontracts-rc_1.0-20211104.zip (md5: 96ad53eecca4ac4e4fcca231f09233df)</pre>		
Technical	NO		
Documentation			
JS tests	YES		
Timeline	07 OCTOBER 2021 - 13 OCTOBER 2021		
Changelog	13 OCTOBER 2021 - Initial Audit		
	25 OCTOBER 2021 - Second Review 05 NOVEMBER 2021 - Third Review		

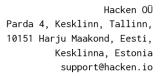




Table of contents

Introduction	
Scope	4
Executive Summary	5
Severity Definitions	10
Audit overview	8
Conclusion	10
Disclaimers	11



Introduction

Hacken OÜ (Consultant) was contracted by CryptoDragons (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 $07^{\rm th}$, 2021 - October $13^{\rm th}$, 2021.

Second review conducted on October 25th, 2021.

Third review conducted on November 5th, 2021.

Scope

```
The scope of the project is smart contracts in the repository:
File name:
      smartcontracts-rc_1.0-20211104.zip
md5:
      96ad53eecca4ac4e4fcca231f09233df
Technical Documentation: No
JS tests: Yes, included
Contracts:
      AccessControlManager.sol
      BaseAuctionReceiver.sol
      DragonArena.sol
      DragonCoin.sol
      DragonCreator.sol
      DragonCrossbreed.sol
      DragonMarket.sol
      DragonToken.sol
      EggMarket.sol
      EggToken.sol
      LockableReceiver.sol
      MockCoinSpender.sol
      access/BaseAccessControl.sol
      interfaces/IChangeableVariables.sol
      structs/FightInfo.sol
      structs/EggInfo.sol
      structs/DragonInfo.sol
      utils/GenesLib.sol
      utils/Random.sol
      utils/BytesLib.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
Tunecional Teview	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		ere

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 1 medium and 2 low severity issues.

After the second review security engineers found that some contracts were slightly changed. Therefore found 1 medium and 2 low severity issues.

After the third review security engineers found that all issues were fixed.

Notice: Some functionality (creation and battles of dragons) based on PRNG could be manipulated by malicious miners.



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

1. "Random" value is predictable

While using block.difficulty, block.timestamp and the previously set seed, it is very easy to predict the next random number for a malicious miner. Any miner having powers to produce the next block could generate a transaction and include it in the block with the needed "random" number they want to use.

Contracts: Random.sol

Functions: rand

Recommendation: Please take a look at the Chainlink VRF or anything like that if you need a random number.

Status: non-issue because affects only non-essential functionality.

Low

1. Missing event for changing <u>tokenAddress</u> and <u>accessControl</u>

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

Contracts: BaseAccessControl.sol, EggMarket.sol

Functions: setAccessControlAddress, setTokenAddress

Recommendation: Please emit events on the critical values changing.

Status: fixed.

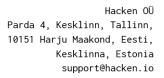
2. A public function that could be declared external.

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

Contracts: EggToken.sol, DragonArena.sol, DragonToken.sol

Functions: price, updatePrice, tokenURI, fightInfo, canHatch, isHatched

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





Status: fixed

3. Tautology or contradiction

Contracts: GenesLib.sol

Functions: randomInheritGenesInRange (line #58)

Recommendation: remove tautologic comparison $r \ge 0$

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 1 medium and 2 low severity issues.

After the second review security engineers found that some contracts were slightly changed. Therefore found 1 medium and 2 low severity issues.

After the third review security engineers found that all issues were fixed.

Notice: Some functionality (creation and battles of dragons) based on PRNG could be manipulated by malicious miners.



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.