HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: CryptoToday
Date: March 8th, 2022

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 CryptoToday.
Approved by	Andrew Matiukhin | CTO Hacken OU Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type	ERC20 token; ERC1155 token; Voting;
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Automated Review, Manual Review
Repository	https://github.com/cryptotodaycom/contracts
Commit	548c1ef24d996a3adc0557638601d099a5ef745d
Deployed contract	
Technical Documentation	No
JS tests	Yes
Website	https://cryptotoday.com
Timeline	27 FEB 2022 - 8 MAR 2022
Changelog	1 MAR 2022 - Initial Audit 8 MAR 2022 - Second Review

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by CryptoToday (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.

Scope

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

Repository:
https://github.com/cryptotodaycom/contracts
Commit:
548c1ef24d996a3adc0557638601d099a5ef745d
Technical Documentation: Partial
JS tests: Yes
Contracts:
- LIST.sol
- VotingEngine.sol
- TrooperNFT.sol
- Signature.sol
- LISTFuture.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

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

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

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

Score measurements details can be found in the corresponding section of the methodology.

Documentation quality

The customer provided a whitepaper where tokenonmics is described. The code has clear comments. Total Documentation Quality score is **8** out of **10**.

Code quality

Total CodeQuality score is **10** out of **10**. Some code duplications. Some logic is covered with unit tests.

Architecture quality

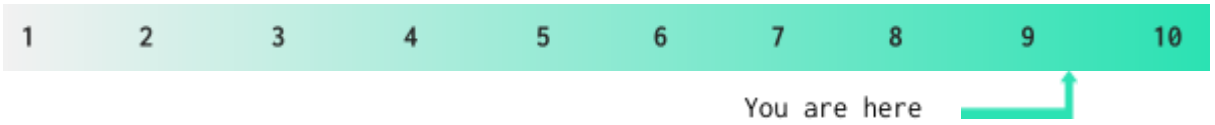
Architecture quality score is **5** out of **10**. The voting engine does not actually designed for votes.

Security score

As a result of the audit, security engineers found no issues. Security score is **10** out of **10**. All found issues are displayed in the “Issues overview” section of the report.

Summary

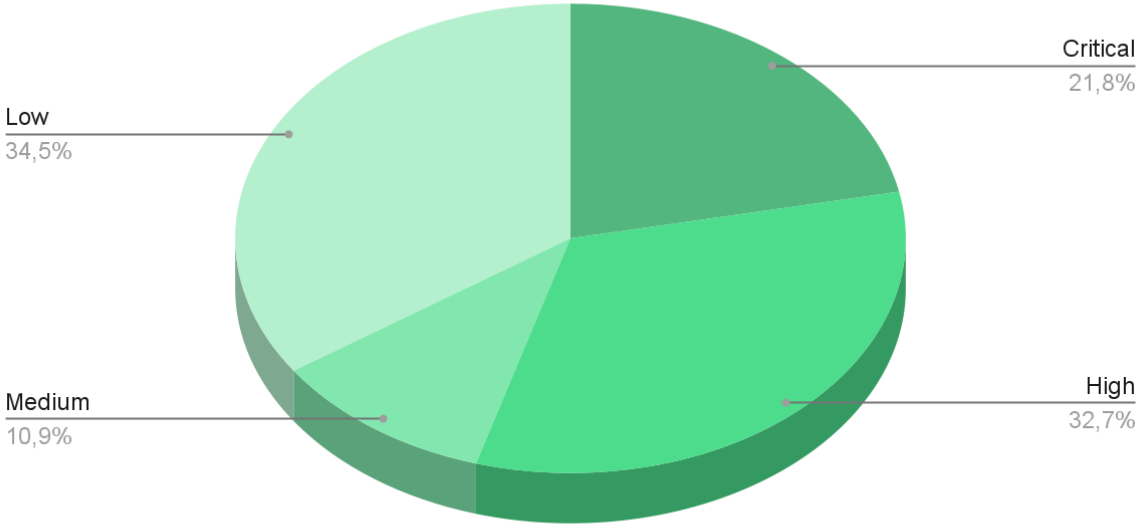
According to the assessment, the Customer's smart contracts have the following score: **9.3**



Notice

1. 87% of the token supply is moved to a VotingEngine contract that is fully controlled by Owners. The off-chain part of the system responsible for funds transfers is **OUT** of the audit scope and can not be verified by Hacken.

Graph 1. The distribution of vulnerabilities after the first audit.



Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

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

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Audit overview

Critical

No critical severity issues were found.

High

1. Misleading contract name and purpose.

   The VotingEngine contract is not designed for any fair voting and is used as funds storage for the project owners. Funds withdrawal can be done at any time and with any amount.

   Contracts: VotingEngine.sol

   Recommendation: re-design voting functionality and move funds storage functionality to separate contract.

   Status: Mitigated. All voting logic is off-chain.

2. Transfer fail.

   The function can fail on ethers transfer if a msg.sender is a contract with fallback function (e.g. multi-sig wallet with advanced fallback mechanisms).

   Contracts: VotingEngine.sol

   Function: endSale

   Recommendation: use call to transfer ethers or ensure that the owner is not a contract.

   Status: Mitigated. The Customer approved that address is not a contract.

Medium

1. Constructor overwhelmed.

   The code can fail if a list of token receivers is big enough.

   Contracts: LISTFuture.sol

   Recommendation: ensure that list of receivers is not big enough for the function to fail on gaslimit. Or use a separate function for tokens distribution.

   Status: Mitigated. The Customer approved that the gas limit would not be reached.

Low

1. Redundant import.

“@openzeppelin/contracts/token/ERC20/ERC20.sol" import is redundant.

Contracts: LIST.sol

Recommendation: remove unused imports.

Status: Fixed.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

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.