HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Plant Exodus
Date: December 17th, 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 Plant Exodus.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 tokens; ERC721 token; Market
Platform	Binance / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/PlantExodus/smart-contracts
Commit	b623298a64b05e8a359a8f4863cda630c08267f3
Technical Documentation	NO
JS tests	NO
Website	Plantexodus.com
Timeline	06 DECEMBER 2021 - 17 DECEMBER 2021
Changelog	08 DECEMBER 2021 - INITIAL AUDIT 17 DECEMBER 2021 - SECOND REVIEW

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

Table of contents

Introduction 4

Scope 4

Executive Summary 5

Severity Definitions 7

Audit overview 8

Conclusion 10

Disclaimers 11

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

Introduction

Hacken OÜ (Consultant) was contracted by Plant Exodus (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 6th, 2021 - December 8th, 2021.

Second review conducted on December 17th, 2021.

Scope

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

Repository:
https://github.com/PlantExodus/smart-contracts
Commit:
b623298a64b05e8a359a8f4863cda630c08267f3
Technical Documentation: No
JS tests: No
Contracts:
ERC20_HOA.sol
ERC20_PEXO.sol
Market.sol
Market_Seed_Leafies.sol
Market_Seed_Rubblers.sol
Market_Seed_Shroomies.sol
NFT.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

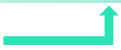
Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

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

Executive Summary

According to the assessment, the Customer's smart contracts are secured.

Insecure	Poor secured	Secured	Well-secured

You are here 

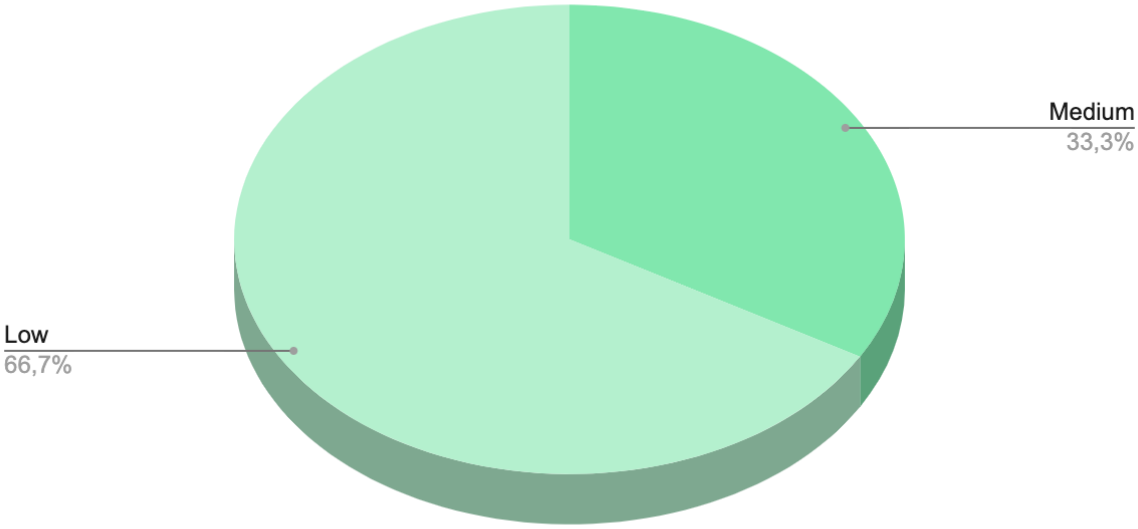
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found **1** medium and **6** low severity issues.

After the second review security engineers found **1** medium and **2** low severity issues.

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

Graph 1. The distribution of vulnerabilities after the 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 issues were found.

High

No high severity issues were found.

Medium

No tests were provided.

It’s recommended to cover all non-trivial contracts with tests.

The recommended coverage is minimum 95% for branches, while it should be definitely 100% for the main logic contracts.

Low

1. View function iterates over or returns an array of unpredictable size

   Contracts: NFT.sol, Market.sol

   Functions: tokenIdsOfAccount, nftDetailsOfAccount, fetchMarketItems, fetchItemsCreated

   Gas consumption grows with array size and starting from a certain size function could become inoperable.

   Recommendation: add(optional) *limit* and *offset* parameters

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: ERC20_HOA.sol, ERC20_PEXO.sol, Market.sol, Market_Seed_Leafies.sol, Market_Seed_Rubblers.sol, Market_Seed_Shroomies.sol, NFT.sol

   Functions: approveAndCall, batchMint, burn, cancelMarketItem, createMarketItem, createMarketSale, fetchItemsCreated, fetchMarketItems, getListingPrice, getRemainingHoldersShroomies, getRemaining, getRemainingHoldersOne, getRemainingHoldersTwo, getRemainingPriSale, getRemainingPubSale, lock, merge, mint, safeTransferAndLock, tokenOfOwnerByIndex, transferAnyBEP20Token

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

   Status: fixed

3. ETH could be locked or some functions could be unavailable.

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

In case *listingPrice* will be changed during active sales, functions *cancelMarketItem* and *createMarketSale* could leave ETH on contract account(that could not be withdrawn) or fail due to lack of ETH on contract account.

Contracts: Market.sol

Functions: setListingPrice

Status: fixed

4. Missing event for changing *_whitelist*, *baseIndex*, *SALE_TYPE*, *_holdersShroomies*, *_holdersOne*, *_holdersTwo*

   Contracts: Market_Seed_Shroomies.sol, Market_Seed_Rubblers.sol, Market_Seed_Leafies.sol

   Functions: setWhitelist, setBaseIndex, setSaleType, setHoldersShroomies, approvalHoldersShroomies, setHoldersOne, approvalHoldersOne, setHoldersTwo, approvalHoldersTwo

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

5. State variables that could be declared constant

   Constant state variables should be declared constant to save gas.

   Contracts: Market.sol

   Variables: STATUS_CANCELLED, STATUS_NEW, STATUS_SOLD

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

   Status: fixed

6. Using SafeMath in Solidity >= 0.8.0

   Starting solidity version 0.8.0 arithmetic operations revert on underflow and overflow. There’s no more need to assert the result of operations.

   Contracts: ERC20_HOA.sol, ERC20_PEXO.sol, Market_Seed_Leafies.sol, Market_Seed_Rubblers.sol, Market_Seed_Shroomies.sol

   Recommendation: Please avoid using assert for arithmetic operations.

   Status: fixed

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

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 **6** low severity issues.

After the second review security engineers found **1** medium and **2** low severity issues.

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.