HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Para
Date: December 22nd, 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 Para.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 token; ERC721 token
Platform	Polygon / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/para-nft/tokens
Commit	b561c86f0c69f8dedb3039bda7ed67d5c8a424d9
Technical Documentation	YES
JS tests	NO
Website	para-nft.com
Timeline	16 DECEMBER 2021 - 22 DECEMBER 2021
Changelog	17 DECEMBER 2021 - Initial Audit 22 DECEMBER 2021 - Second Review

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Para (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 16th, 2021 - December 17th, 2021.

Second review conducted on December 22nd, 2021.

Scope

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

Repository:
https://github.com/para-nft/tokens
Commit:
b561c86f0c69f8dedb3039bda7ed67d5c8a424d9
Technical Documentation: Yes (WP:
https://drive.google.com/file/d/18OD8BJbAfojJKW74QPJyz3Dt5OP0hrCs/view)
JS tests: No
Contracts:
ERC20.sol
ERC721.sol

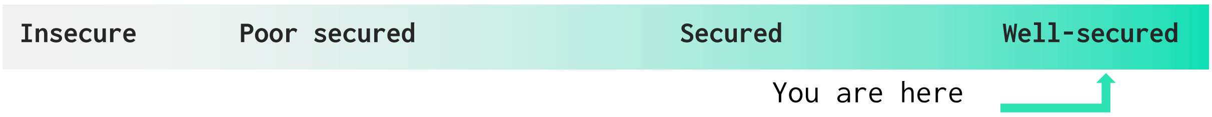
We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

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 **4** low severity issues.

After the second review security engineers found **1** low severity issue.

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. Unclear number constants.

It's always better to write "365 days" in solidity instead of "31536000" as seconds. Solidity gives us time units suffixes that make our code more readable.

Contract: ERC20.sol

Constant: _vestingPeriod

Recommendation: Please use time unit suffixes.

Status: Fixed

2. Conformance to Solidity naming conventions

Solidity defines a naming convention that should be followed.

Contract: ERC20.sol

Constants: _vestingPeriod, _supply, _vestingRate

Recommendation: Follow the Solidity naming convention.

Status: Fixed

3. Duplicated code

It is always a good idea to put the code into some function and call this function from other instead of duplicating the code. The main reason here is that you don't need to change the code in multiple places and you'll be sure that the code works the same.

Contract: ERC20.sol

Functions: checkClaim, claimOutstanding

Recommendation: Please consider declaring checkClaim as **public** and call it from the claimOutstanding.

Status: Fixed

4. Not emitting events

It is recommended to emit events on the contract state change to the community can keep on track the state off-chain.

Contract: ERC721.sol

Functions: flipSale

Recommendation: Please emit events on contract state changes.

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 **4** low severity issues.

After the second review security engineers found **1** low severity issue.

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.