SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Champion Games SL
Date: December 20th, 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 Champion Games SL.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC721 Token
Platform	Polygon / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/metasoccer/vouchers/blob/main/contracts/Vouchers.sol
Commit	92100c41b49caae2eaf541e6154e0728965c8772
Technical Documentation	YES
JS tests	YES
Website	Metasoccer.com
Timeline	16 NOVEMBER 2021 - 20 DECEMBER 2021
Changelog	18 NOVEMBER 2021 - INITIAL AUDIT 23 NOVEMBER 2021 - SECOND REVIEW 20 DECEMBER 2021 - THIRD REVIEW

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Champion Games SL (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 November 16th, 2021 - November 18th, 2021.

Second audit conducted on November 23rd, 2021.

Third audit conducted on December 20th, 2021.

Scope

The scope of the project is smart contracts in the repository:
Repository:
https://github.com/metasoccer/vouchers
Commit:
92100c41b49caae2eaf541e6154e0728965c8772
Technical Documentation: Yes (provided in text)
JS tests: Yes
Contracts:
Vouchers.sol
libraries/Base64.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.

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** high and **2** low severity issues.

After the second review security engineers found that some issues were fixed but also some new ones were discovered. Therefore there are **1** high, **1** medium, and **1** low severity issue.

After the third 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

1. Possible reentrancy issue

While Vouchers contract implements the ERC721 contract, it has a callback to the token receiver while safeMinting/safeTransferring.

Malicious users could call “buyVouchers” function and reenter it in the middle of the process, before ‘soldNum’ are updated.

Contract: Vouchers.sol

Function: buyVouchers

Recommendation: Please either move all minting at the end of the code or use reentrancy-guard to prevent a reetrancy possibility.

Status: Fixed

2. MetaSoccerToken contract is missing

The previously audited contract “MetaSoccerToken” is missing in the latest scope.

Recommendation: Please make sure this contract is not required anymore.

Status: Replaced by the TestMintableToken contract

🟩 🟩 Medium

Test is failing

The test is failing with the message: “Artifact for contract "MetaSoccerToken" not found.”

Recommendation: Make sure tests are executed successfully.

Status: Fixed

🟦 Low

1. Test coverage is not enough

Currently, only about 60% of the “Vouchers.sol” code branches are covered by tests.

Contract: Vouchers.sol

Recommendation: Make sure code coverage is at least 95-100% for branches.

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.

Contract: Vouchers.sol

Functions: getAllVouchers, getAllOwned, getDropsById

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

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** high and **2** low severity issues.

After the second review security engineers found that some issues were fixed but also some new ones were discovered. Therefore there are **1** high, **1** medium, and **1** low severity issue.

After the third 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.