HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Splash
Date: March 11th, 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 Splash.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC721 and ERC1155 Marketplace; Staking; Tournaments
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/NFTCoach/splash-core
Commit	b38add0749f450a528197909005f6666cf609960
Technical Documentation	NO
JS tests	YES
Website	-
Timeline	16 FEBRUARY 2022 - 11 MARCH 2022
Changelog	28 FEBRUARY 2022 - Initial Audit 11 MARCH 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 Splash (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 February 16th, 2022 - February 28th, 2022.

The second review was conducted on March 11th, 2022

Scope

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

Repository:
https://github.com/NFTCoach/splash-core
Commit:
b38add0749f450a528197909005f6666cf609960
Technical Documentation: No
JS tests: Yes
Contracts:
marketplace/Marketplace.sol
gameplay/PaidTournaments.sol
gameplay/MatchMaker.sol
gameplay/WeeklyTournaments.sol
utils/ABDKMath64x64.sol
staking/Staking.sol
utils/Errors.sol

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

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

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

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 2 low issues which were acknowledged by the customer and accepted.

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 medium severity issues were found.

Low

1. Some remappings are missed.

Foundry could not compile contracts because some remappings are missed in the “foundry.toml”: oz-contracts, chainlink

Scope: compilation, tests

Recommendation: Please make sure all remappings are specified correctly.

Status: Added remappings.txt

2. No events on fees set.

There is a recommendation to emit events in the case the contract’s crucial state is changed to let track changes off-chain. Functions listed below change valuable state values.

Contracts: WeeklyTournaments.sol, Marketplace.sol, Staking.sol

Functions: Staking.enterStake, WeeklyTournaments.setFirstWeight, WeeklyTournaments.setSecondWeight, Marketplace.setCommission

Recommendation: Please emit events on changing contract state to be able to track changes off-chain.

Status: Fixed

3. Calls inside a loop.

Functions listed below are calling external contracts and are called in the loop. Calling external contracts in the loop could cost a lot of excess gas.

Contracts: WeeklyTournaments.sol, MatchMaker.sol

Functions: MatchMaker.matchMaker, PaidTournaments.playTournamentRound

Recommendation: Please revise the logic to keep retrieved external values in the memory variables.

Status: Acknowledged.

Comment from customer: we assume that loops are not going to be too large to cause a problem so we're not fixing them. Also calls inside loops is bounded to inside contracts so we don't see any problems.

4. Costly operations inside a loop.

createTournaments function increments *_tournamentNonce* state variable in the loop which is very costly.

Contracts: WeeklyTournaments.sol

Functions: createTournaments

Recommendation: Please store the _tournamentNonce in the local memory variable, increment it in the loop, and save it to the state afterward.

Status: Acknowledged.

Comment from customer: we assume that loops are not going to be too large to cause a problem so we're not fixing them. Also calls inside loops is bounded to inside contracts so we don't see any problems.

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 **4** low severity issues.

After the second review security engineers found **2** low issues which were acknowledged by the customer and accepted.

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.