HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Polkamarkets
Date: October 6th, 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 Polkamarkets.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Market prediction game
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/bepronetwork/bepro-js/tree/feature/prediction-markets-hacken-changes
Commit	3CF69D00A0261E986FC312F8307E4BA468769397
Technical Documentation	NO
JS tests	YES
Timeline	17 SEPTEMBER 2021 - 06 OCTOBER 2021
Changelog	28 SEPTEMBER 2021 - INITIAL AUDIT 06 OCTIBER 2021 - 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 Polkamarkets (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 September 17th, 2021 - September 28th, 2021.

The second code review conducted on October 6th, 2021.

Scope

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

Repository:
https://github.com/bepronetwork/bepro-js/tree/feature/prediction-markets-hacken-changes
Commit:
3cf69d00a0261e986fc312f8307e4ba468769397
Technical Documentation: No
JS tests: Yes
Contracts:
PredictionMarket.sol
RealitioERC20.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 well-secured.

Insecure	Poor secured	Secured	Well-secured

You are here

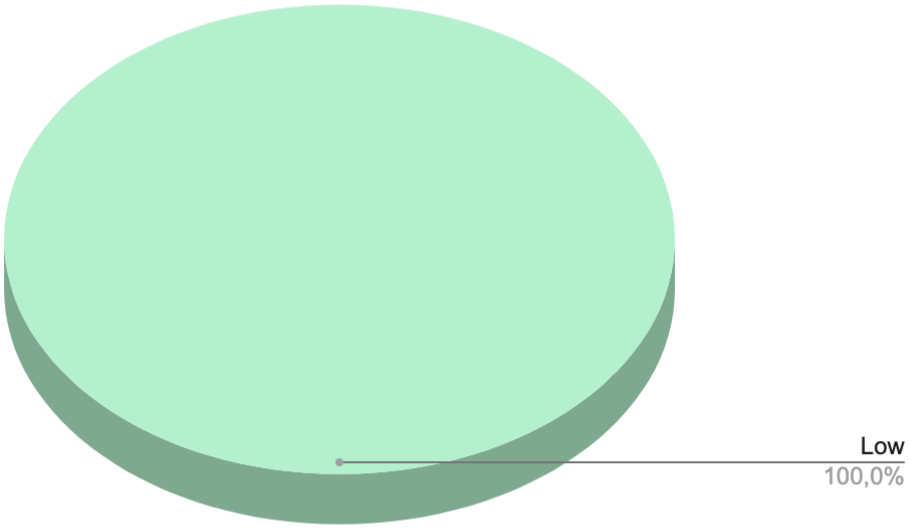
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview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

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

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

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

Tests could not be run

Recommendation: Please make sure tests are able to be executed and test coverage is at least 95% for branches.

Fixed before the second review

Low

A public function that could be declared external

public functions that are never called by the contract should be declared **external** to save gas.

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

Lines: RealitioERC20.sol#280

```
function setToken(IERC20 _token)
public
```

Lines: RealitioERC20.sol#333

```
function createTemplateAndAskQuestion(
   string memory content,
   string memory question, address arbitrator, uint32 timeout, uint32
opening_ts, uint256 nonce
)
   // stateNotCreated is enforced by the internal _askQuestion
public returns (bytes32) {
```

Lines: RealitioERC20.sol#379

```
function askQuestionERC20(uint256 template_id, string memory question,
address arbitrator, uint32 timeout, uint32 opening_ts, uint256 nonce,
uint256 tokens)
   // stateNotCreated is enforced by the internal _askQuestion
public returns (bytes32) {
```

Lines: RealitioERC20.sol#818

```
function claimMultipleAndWithdrawBalance(
   bytes32[] memory question_ids, uint256[] memory lengths,
   bytes32[] memory hist_hashes, address[] memory addrs, uint256[] memory
bonds, bytes32[] memory answers
)
   stateAny() // The finalization checks are done in the claimWinnings f
public {
```

Lines: RealitioERC20.sol#849

```
function getContentHash(bytes32 question_id)
public view returns(bytes32) {
```

Lines: RealitioERC20.sol#856

```
function getArbitrator(bytes32 question_id)
public view returns(address) {
```

Lines: RealitioERC20.sol#863

```
function getOpeningTS(bytes32 question_id)
public view returns(uint32) {
```

Lines: RealitioERC20.sol#870

```
function getTimeout(bytes32 question_id)
public view returns(uint32) {
```

Lines: RealitioERC20.sol#877

```
function getFinalizeTS(bytes32 question_id)
public view returns(uint32) {
```

Lines: RealitioERC20.sol#884

```
function isPendingArbitration(bytes32 question_id)
public view returns(bool) {
```

Lines: RealitioERC20.sol#892

```
function getBounty(bytes32 question_id)
public view returns(uint256) {
```

Lines: RealitioERC20.sol#899

```
function getBestAnswer(bytes32 question_id)
public view returns(bytes32) {
```

Lines: RealitioERC20.sol#907

```
function getHistoryHash(bytes32 question_id)
public view returns(bytes32) {
```

Lines: RealitioERC20.sol#914

```
function getBond(bytes32 question_id)
public view returns(uint256) {
```

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 **1** low severity issue.

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

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.