HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: xPocket
Date: December 21st, 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 xPocket.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 token; DEX
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/xpocket/PocketSwap
Commit	c3c87da365bfb775f80e3caa15fb76cae294ac07
Technical Documentation	YES
JS tests	YES
Website	xpocket.finance
Timeline	08 DECEMBER 2021 - 21 DECEMBER 2021
Changelog	16 DECEMBER 2021 - INITIAL AUDIT 21 DECEMBER 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 xPocket (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 8th, 2021 - December 16th, 2021

Second review conducted on December 21st, 2021.

Scope

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

Repository:
https://github.com/xpocket/PocketSwap
Commit:
c3c87da365bfb775f80e3caa15fb76cae294ac07
Technical Documentation: Yes, provided in text
JS tests: Yes
Contracts:
PocketSwap.sol
pocketswap/abstract/PeripheryImmutableState.sol
pocketswap/abstract/PeripheryPayments.sol
pocketswap/abstract/Multicall.sol
pocketswap/abstract/BlockTimestamp.sol
pocketswap/abstract/PeripheryValidation.sol
pocketswap/libraries/AddressStringUtil.sol
pocketswap/libraries/Path.sol
pocketswap/libraries/UQ112x112.sol
pocketswap/libraries/PairAddress.sol
pocketswap/libraries/Math.sol
pocketswap/libraries/CallbackValidation.sol
pocketswap/libraries/TransferHelper.sol
pocketswap/libraries/PocketSwapLibrary.sol
pocketswap/libraries/PlainMath.sol
pocketswap/libraries/BytesLib.sol
pocketswap/PocketSwapFactory.sol
pocketswap/PocketSwapRouter.sol
pocketswap/PocketSwapERC20.sol
pocketswap/PocketSwapPair.sol
pocketswap/interfaces/IPocketSwapFactory.sol
pocketswap/interfaces/IPocket.sol
pocketswap/interfaces/IPocketSwapRouter.sol
pocketswap/interfaces/IMulticall.sol
pocketswap/interfaces/IPeripheryImmutableState.sol
pocketswap/interfaces/IPocketSwapERC20.sol
pocketswap/interfaces/IPocketSwapLiquidityRouter.sol
pocketswap/interfaces/callback/IPocketSwapCallback.sol
pocketswap/interfaces/external/IWETH9.sol
pocketswap/interfaces/IPeripheryPayments.sol
pocketswap/interfaces/IPocketSwapPair.sol
pocketswap/pair/StorageData.sol
pocketswap/router/swap/SwapProcessing.sol
pocketswap/router/SwapRouter.sol
pocketswap/router/LiquidityRouter.sol
pocketswap/router/liquidity/LiquidityProcessing.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.

Insecure	Poor secured	Secured	Well-secured

You are here 

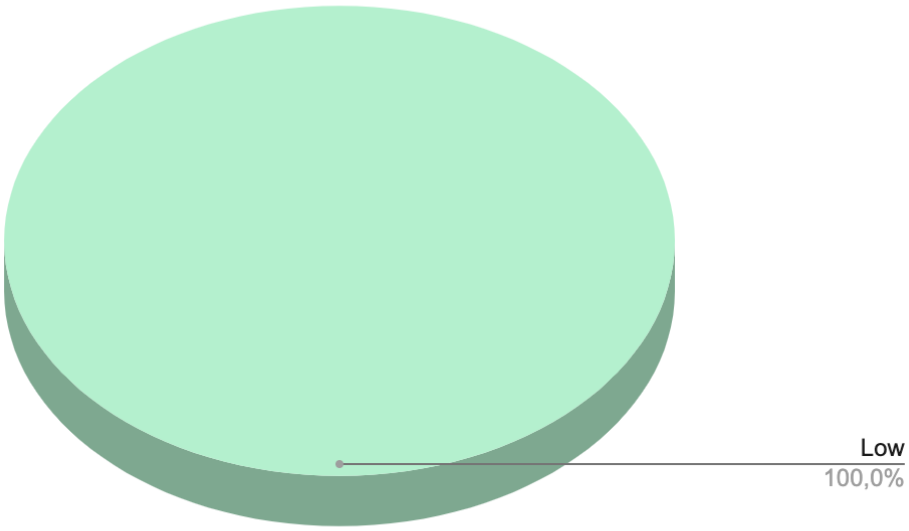
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview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

As a result of the audit, security engineers found **4** low severity issues.

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

No medium severity issues were found.

Low

1. Possible inconsistency of pair addresses obtained by creation and computation

   Contracts: PairAddress.sol

   Constant *PAIR_INIT_CODE_HASH* that is used in the calculation is hardcoded therefore it could differ from the actual hash of the pair creation code.

   Recommendation: change *PAIR_INIT_CODE_HASH* to the actual hash of pair creation code

   Status: Fixed

2. Low test coverage

   Overall coverage for the project is only **48%** for statements and **72%** for branches

   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.

   Status: Acknowledged.

   Customer's comments: *We've covered all main cases. Other cases are standard and works as in Uni and PSC. They have tests for that.*

3. Missing event for changing *fee*, *feeSetter*

   Contracts: PocketSwapFactory.sol

   Functions: setFee, setFeeSetter

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

   Recommendation: Please emit events on the critical values changing.

   Status: Fixed

4. 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: LiquidityRouter.sol, SwapRouter.sol

Functions: removeLiquidity, quote, getAmountOut, getAmountIn

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

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 **4** low severity issues.

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.