HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Premia
Date: September 9th, 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 Client.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Options Platrform
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/PremiaFinance/premia-contracts-private
Commit	dfc07ca0c6d6c1f76025a646a8616bcf26dc5d0b
Technical Documentation	YES
JS tests	YES
Timeline	02 AUGUST 2021 - 18 AUGUST 2021
Changelog	11 AUGUST 2021 - INITIAL AUDIT 09 SEPTEMBER 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 Premia (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 August 2nd, 2021 - August 11th, 2021.

Additional review was done in period September 6th, 2021 - September 9th, 2021

Scope

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

Repository:
https://github.com/PremiaFinance/premia-contracts-private
Commit:
dfc07ca0c6d6c1f76025a646a8616bcf26dc5d0b
Technical Documentation: Yes
JS tests: Yes
Contracts:
core/IProxyManager.sol
core/Premia.sol
core/ProxyManager.sol
core/ProxyManagerStorage.sol
interface/IFlashLoanReceiver.sol
interface/IKeeperCompatible.sol
interface/INewPremiaFeeDiscount.sol
interface/IPremiaFeeDiscount.sol
interface/IPremiaMaker.sol
interface/IPremiaOption.sol
interface/IPriceOracleGetter.sol
keeper/AutoExerciseKeeper.sol
keeper/PremiaMakerKeeper.sol
keeper/ProcessExpiredKeeper.sol
libraries/ABDKMath64x64Token.sol
libraries/OptionMath.sol
mining/IPremiaMining.sol
mining/PremiaMining.sol
mining/PremiaMiningProxy.sol
mining/PremiaMiningStorage.sol
oracle/IVolatilitySurfaceOracle.sol
oracle/VolatilitySurfaceOracle.sol
oracle/VolatilitySurfaceOracleStorage.sol
pool/IPool.sol
pool/IPoolBase.sol
pool/IPoolEvents.sol
pool/IPoolExercise.sol
pool/IPoolIO.sol
pool/IPoolView.sol
pool/IPoolWrite.sol
pool/PoolBase.sol
pool/PoolExercise.sol
pool/PoolIO.sol

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

```
pool/PoolProxy.sol
pool/PoolStorage.sol
pool/PoolSwap.sol
pool/PoolView.sol
pool/PoolWrite.sol
vesting/PremiaMultiVesting.sol
vesting/PremiaVesting.sol
vesting/PremiaVestingCancellable.sol
PremiaDevFund.sol
PremiaErc20.sol
PremiaFeeDiscount.sol
PremiaMaker.sol
PremiaMakerStorage.sol
PremiaStaking.sol
PremiaVoteProxy.sol
ProxyUpgradeableOwnable.sol
ProxyUpgradeableOwnableStorage.sol
WETH.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

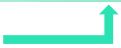
Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

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 secured, but it’s possible to use some assets in the Flash Loan arbitrage attack. Also, there are some tests failing.

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 **2** medium and **2** low severity issues.

After the second review security engineers found **1** medium and **2** low severity issues.

Notice:

The PremiaStaking contract contains token assets which could be used in the FlashLoan attack which could lead to money loss.

Notice 2:

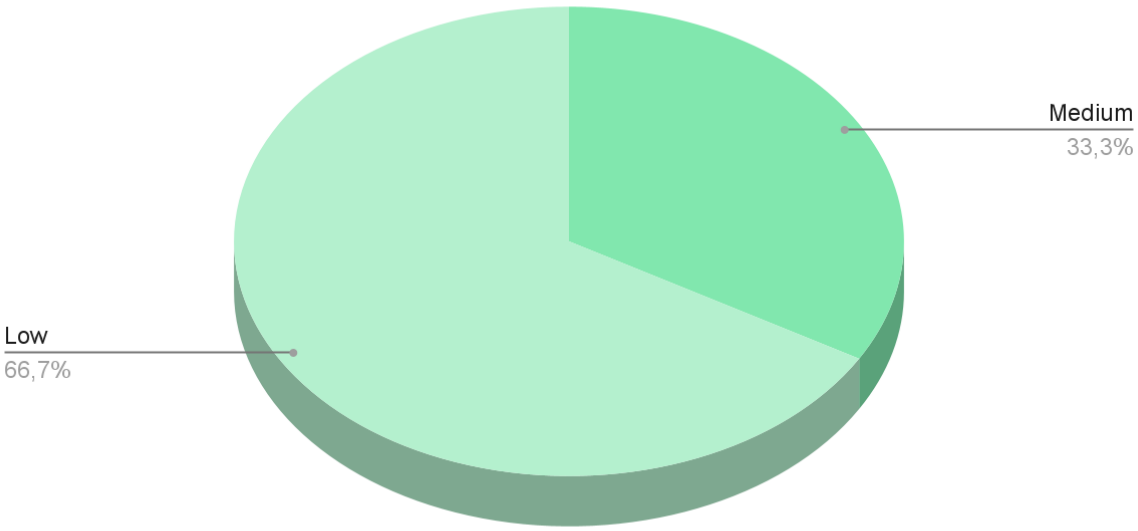
Failing tests could show real issues with business logic. Please check them.

Notice 3:

A lot of TODOs could mean that contracts are not done yet.

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

1. Possible Flash Loan arbitrage attack

   It is possible that xPremia token (PremiaStaking) could be used in the Flash Loan arbitrage attack. As soon as it traded on any DEX, it could be “borrowed”, then “leave” method called, to receive some premia which could be used on other DEX where traded.

   This is not direct contract vulnerability, it’s rather a possibility to use produced and cross-changed tokens to receive those for free in the same transaction trade those, receive rewards and return tokens back.

   Recommendation: Please consider either not trading xPremia on DEXes or create a timelock for at least one block between taking premia and minting xPremia, and also between taking xPremia and sending premia back.

2. Not all tests succeed

   There are bunch of tests provided with contracts. In total 288 tests. But, unfortunately, **1** test is failing and 9 of them aren’t finished and marked as **todo**.

   After the second review: We’ve found that now **9** of test are failing while 9 more are still in pending state.

   Recommendation: We’d recommend checking the failing test and also complete the pending ones. One more, as far as your code coverage is about 85%, which is not bad, but we recommend to add more test cases to cover at least 95% of the code.

```
    ✓ returns number of holders of given token (263ms)
  #accountsByToken
    ✓ returns list of addresses holding given token (258ms)
  #tokensByAccount
    1) returns list of tokens held by given address
__internal
  #_formatTokenId
```

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

```
PremiaDevFund
  2) "before each" hook for "should correctly deprecate bonding curve"

PremiaDevFund
  3) "before each" hook for "should not allow withdrawal if timelock has not passed"

PremiaFeeDiscount
  4) "before each" hook for "should correctly overwrite existing stake levels"

PremiaMaker
  5) "before each" hook for "should make premia successfully"

PremiaStaking
  6) "before each" hook for "should successfully enter with permit"

PremiaVesting
  7) "before each" hook for "should withdraw 200 premia, then 50 premia if withdrawing after 100 days and then after 25 days"

PremiaVestingCancellable
  8) "before each" hook for "should withdraw 100 premia, then 25 premia if withdrawing after 100 days and then after 25 days"

PremiaMultiVesting
  9) "before each" hook for "should correctly handle vesting for multiple deposits"


261 passing (6m)
9 pending
9 failing
```

Low

1. TODOs in the code

   Having a bunch of TODOs in the code could mean there is some business or other logic unimplemented. Right now, we've discovered 10 TODOs in the code.

   Recommendation: Please consider implementing unimplemented logic, re-check places with TODO marks and remove those.

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.

Lines: keeper/AutoExerciseKeeper.sol#45

```
function addPriceOrder(AutoExerciseOrder memory order) public {
```

Lines: keeper/AutoExerciseKeeper.sol#55

```
function addExpirationOrder(AutoExerciseOrder memory order) public {
```

Lines: keeper/AutoExerciseKeeper.sol#67

```
function removeOrder(AutoExerciseOrder memory order) public {
```

Lines: keeper/AutoExerciseKeeper.sol#78

```
function cleanupExpiredOrders(uint256 expiration) public {
```

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 **2** medium and **2** low severity issues.

After the second review security engineers found **1** medium and **2** low severity issues.

Notice:

The PremiaStaking contract contains token assets which could be used in the FlashLoan attack which could lead to money loss.

Notice 2:

Failing tests could show real issues with business logic. Please check them.

Notice 3:

A lot of TODOs could mean that contracts are not done yet.

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.