HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Hedgey
Date: December 10th, 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 Hedgey.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Trading tools
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/hedgey-finance/hedgey_audit
Commit	7eb9dc8de364a1ab3b53ba0ad3194f380c03e94b
Technical Documentation	YES
JS tests	NO
Website	Hedgey.finance
Timeline	24 NOVEMBER 2021 - 10 DECEMBER 2021
Changelog	06 DECEMBER 2021 - INITIAL AUDIT 10 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 Hedgey (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 24th, 2021 - December 6th, 2021.

Second review conducted on December 10th, 2021.

Scope

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

Repository:
https://github.com/hedgey-finance/hedgey_audit
Commit:
7eb9dc8de364a1ab3b53ba0ad3194f380c03e94b
Technical Documentation: Yes
(https://docs.hedgey.finance/hedgey/developers/smart-contracts-overview)
JS tests: No
Contracts:
HedgeyAnySwap.sol
HedgeyCalls.sol
HedgeyCallsFactory.sol
HedgeyCeloAnySwap.sol
HedgeyCeloCalls.sol
HedgeyCeloPuts.sol
HedgeyPuts.sol
HegdeyPutsFactory.sol
libraries.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

Executive Summary

According to the assessment, the Customer's smart contracts are secured but not compilable.

Insecure	Poor secured	Secured	Well-secured

You are here ↑

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

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, **4** medium, and **7** low severity issues.

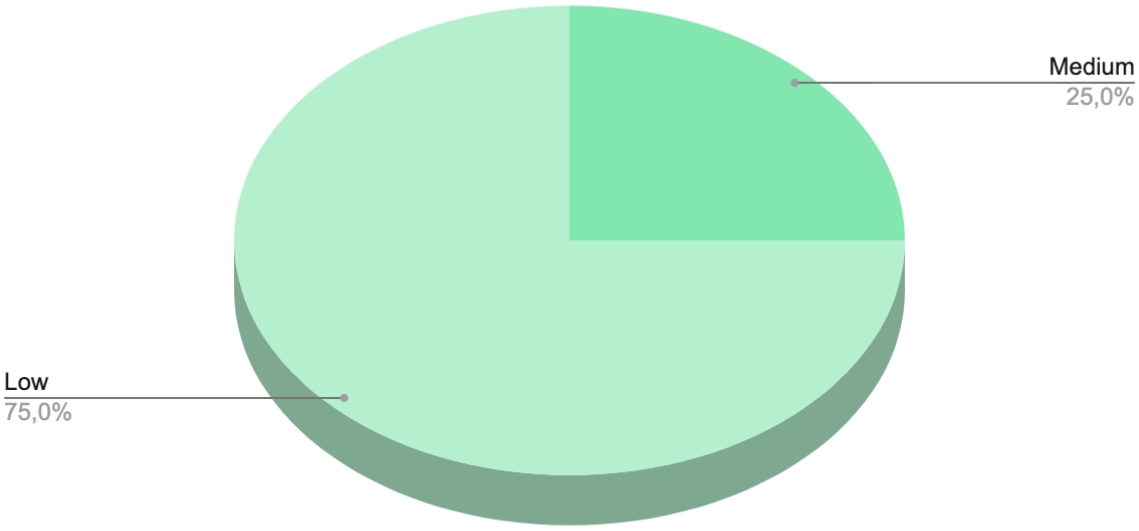
After the second review security engineers found **1** medium and **3** low severity issues.

Notice:

Do not place WETH for the HedgeyAnySwap contract because anyone calling hedgeyCallSwap or hedgeyPutSwap function will get the entire WETH balance.

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 issues were found.

Medium

1. Contracts are not compilable.

All provided contracts could not be compiled because the importing file should be presented as a string literal, while it's given without quotes.

Contracts: HedgeyAnySwap.sol, HedgeyCalls.sol, HedgeyCallsFactory.sol, HedgeyCeloAnySwap.sol, HedgeyCeloCalls.sol, HedgeyCeloPuts.sol, HedgeyPuts.sol, HegdeyPutsFactory.sol

Recommendation: Please fix import statements.

Status: Partially fixed. Contracts still could not be compiled, since there is no ';' after the import statement.

2. Syntax error

HedgeyCallsFactory contract has an excess "/" symbol before the "import" statement.

Contracts: HedgeyCallsFactory.sol

Recommendation: Please fix syntax error.

Status: Fixed

3. Contracts that lock Ether

Contract with a payable function, but without a withdrawal capacity.

Contract: HedgeyCeloCalls.sol

Function: rollExpired

Recommendation: Remove the payable attribute or add a withdraw function

Status: Fixed

4. Contracts that lock Ether

Contract with a payable function, but without a withdrawal capacity.

Contract: HedgeyCeloPuts.sol

Function: rollExpired

Recommendation: Remove the payable attribute or add a withdraw function

Status: Fixed

Low

1. Possible ETH leakage.

In the “hedgeyCallSwap” function, which is external and unrestricted, the contract sends the entire WETH balance to the “originalOwner” address, provided by the function caller.

This balance could contain any WETH amount that is not related to the caller. This way, a malicious contract could leak all WETH balance.

Contract: HedgeyAnySwap.sol

Functions: hedgeyCallSwap, hedgeyPutSwap

Recommendation: Please fix the leakage. One of the possible solutions is to store WETH balance before doing swap, and then get the delta and send only delta to the “originalOwner” instead of sending entire balance.

Status: After communicating with the customer we decided to move it to the “Low severity” issue since there shouldn’t be a case when someone is putting WETH on the contract’s balance other than in the transaction, which will get it back.

2. SPDX license identifier not provided in source file

Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.

Contracts: HedgeyAnySwap.sol, HedgeyCalls.sol, HedgeyCallsFactory.sol, HedgeyCeloAnySwap.sol, HedgeyCeloCalls.sol, HedgeyCeloPuts.sol, HedgeyPuts.sol, HegdeyPutsFactory.sol

Recommendation: Please provide SPDX-License-Identifier to each contract.

3. Function state mutability can be restricted to pure

Functions that never access state variables and do not make any external calls should be declared as pure to save gas.

Contracts: HedgeyAnySwap.sol

Function: sortTokens

Recommendation: Please use keyword **pure** instead of **view**.

4. State variables that could be declared constant

Constant state variables should be declared constant to save gas.

Contract: HedgeyCeloPuts.sol

Variable: uniFactory

Recommendation: Add the **constant** attributes to state variables that never change.

Status: Fixed

5. 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: HedgeyCalls.sol

Functions: updateAMM, newBid, cancelNewBid, sellOpenOptionToNewBid, sellNewOption, changeNewOption, newAsk, cancelNewAsk, buyNewOption, buyOptionFromAsk, setPrice, buyOpenOption, exercise, cashClose, returnExpired, rollExpired, changeFee, getPair, createContract

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

Status: Fixed

6. 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: HedgeyCeloCalls.sol

Functions: changeFee, updateAMM, newBid, cancelNewBid, sellOpenOptionToNewBid, sellNewOption, changeNewOption, newAsk, cancelNewAsk, buyNewOption, buyOptionFromAsk, setPrice, buyOpenOption, exercise, cashClose, returnExpired, rollExpired

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

Status: Fixed

7. 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: HedgeyCeloPuts.sol

Functions: updateAMM, newBid, cancelNewBid, sellOpenOptionToNewBid, sellNewOption, changeNewOption, newAsk, cancelNewAsk, buyNewOption, buyOptionFromAsk, setPrice, buyOpenOption, exercise, cashClose, returnExpired, rollExpired

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

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

Status: Fixed

8. 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: HedgeyPuts.sol, HegdeyPutsFactory.sol

Functions: updateAMM, newBid, cancelNewBid, sellOpenOptionToNewBid, sellNewOption, changeNewOption, newAsk, cancelNewAsk, buyNewOption, buyOptionFromAsk, setPrice, buyOpenOption, exercise, cashClose, returnExpired, rollExpired, changeFee, getPair, createContract

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 **1** high, **4** medium, and **7** low severity issues.

After the second review security engineers found **1** medium and **3** low severity issues.

Notice:

Do not place WETH for the HedgeyAnySwap contract because anyone calling hedgeyCallSwap or hedgeyPutSwap function will get the entire WETH balance.

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.