HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: VoirStudio
Date: November 11th, 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 VoirStudio.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Liquidity Manager
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/VoirStudio/unipilot-protocol-contract-v2
Commit	e5ac07dea4dc11d5b163467fddc39945fe781c5d
Technical Documentation	YES
JS tests	YES
Website	voirstudio.io
Timeline	18 OCTOBER 2021 - 11 NOVEMBER 2021
Changelog	29 OCTOBER 2021 - INITIAL AUDIT 11 NOVEMBER 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 VoirStudio (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 October 18th, 2021 - October 29th, 2021

Second code review conducted on November 10th, 2021.

Scope

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

Repository:
https://github.com/VoirStudio/unipilot-protocol-contract-v2
Commit:
e5ac07dea4dc11d5b163467fddc39945fe781c5d
Technical Documentation: Yes; Business logic, no technical specs
- https://docs.google.com/document/d/1heX04nZ_f7cP7JVgSAzedpXMbuS9erN-YnbJI6NFcR0/edit
- https://docs.google.com/document/d/1-RorxePAvG6yooTtmpX27OwuBljd-qtvOOpviqUtJzk/edit?pli=1

JS tests: Yes; Included (“/test/”)
Contracts:
libraries/TransferHelper.sol
interfaces/IERC721Permit.sol
oracle/libraries/OracleLibrary.sol
interfaces/external/IERC20PermitAllowed.sol
oracle/interfaces/IOracle.sol
interfaces/ILiquidityMigrator.sol
libraries/PositionKey.sol
libraries/SafeCast.sol
interfaces/external/IWETH9.sol
libraries/Sqrt.sol
base/ERC721Permit.sol
base/BlockTimestamp.sol
libraries/LowGasSafeMath.sol
interfaces/IUniStrategy.sol
interfaces/IUnipilot.sol
V3Oracle.sol
interfaces/uniswap/INonfungiblePositionManager.sol
test/ERC20.sol
interfaces/uniswap/IULMState.sol
oracle/libraries/SafeUint128.sol
base/PeripheryPayments.sol
libraries/LiquidityReserves.sol
libraries/LiquidityAmounts.sol
interfaces/external/IERC1271.sol
interfaces/uniswap/IULMEvents.sol
libraries/FixedPoint128.sol
interfaces/IExchangeManager.sol
interfaces/external/IERC20.sol
base/ULMState.sol

```
interfaces/uniswap/IUniswapLiquidityManager.sol
Unipilot.sol
base/UniswapLiquidityManager.sol
UniStrategy.sol
libraries/ChainId.sol
LiquidityMigrator.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
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 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 **2** medium and **5** low severity issues.

After the second review security engineers found some changes in the code related to interfaces and some code reorganization. Unfortunately, inconsistencies between the code and documentation weren't fixed therefore there are still **2** medium and **2** low issues.

Notice:

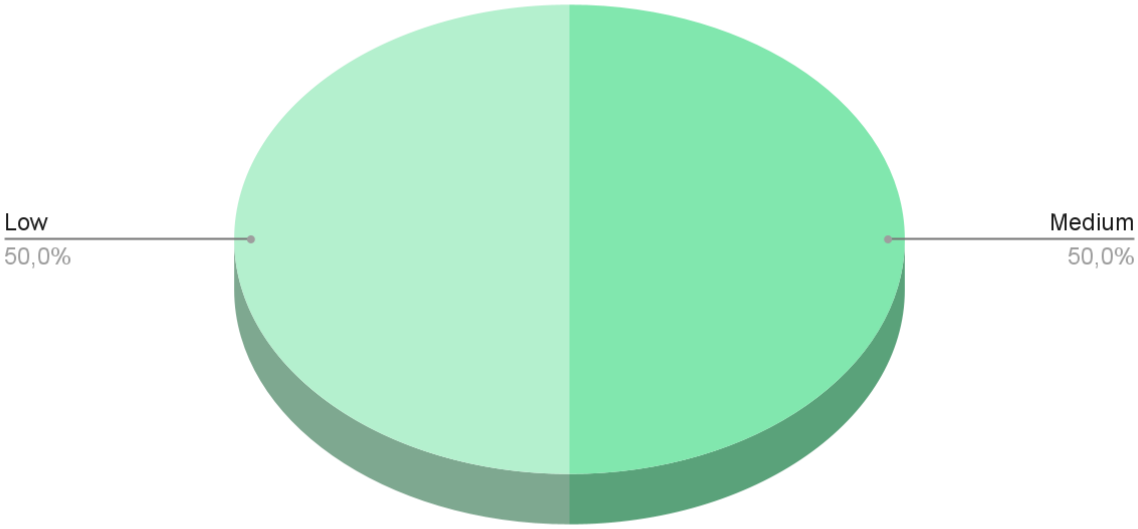
Contracts are written in a very SDKish manner which makes it difficult to understand all inputs and outputs. There are some inconsistencies with the provided business logic documentation as well as no technical documentation.

Notice 2:

We'd recommend rewriting the UniswapLiquidityManager contract to be more straightforward and linear logic. Right now there are too many logic branches that could bring to misunderstanding or just confuse anyone who tries to unravel the logic.

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. Tests could not be run.

Following the instruction. Run:

```
$ yarn install
$ yarn compile
$ yarn test
```

but receiving an error:

```
An unexpected error occurred:

test/stubs.ts:2:30 - error TS2307: Cannot find module '../artifacts/contracts/Unipilot.sol/Unipilot.json' or its corresponding type declarations.

2 import UniPilotArtifact from "../artifacts/contracts/Unipilot.sol/Unipilot.json";
                               ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
test/stubs.ts:3:27 - error TS2307: Cannot find module '../artifacts/contracts/test/ERC20.sol/ERC20.json' or its corresponding type declarations.

3 import Erc20Artifact from "../artifacts/contracts/test/ERC20.sol/ERC20.json";
                            ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
test/stubs.ts:4:25 - error TS2307: Cannot find module '../artifacts/contracts/base/UniswapLiquidityManager.sol/UniswapLiquidityManager.json' or its corresponding type declarations.

4 import ULMArtifact from "../artifacts/contracts/base/UniswapLiquidityManager.sol/UniswapLiquidityManager.json";
                          ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
test/stubs.ts:6:30 - error TS2307: Cannot find module '../artifacts/contracts/base/ULMState.sol/ULMState.json' or its corresponding type declarations.

6 import UlmStateArtifact from "../artifacts/contracts/base/ULMState.sol/ULMState.json";
                               ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
test/stubs.ts:7:33 - error TS2307: Cannot find module '../artifacts/contracts/UniStrategy.sol/UniStrategy.json' or its corresponding type declarations.

7 import UniStrategyArtifact from "../artifacts/contracts/UniStrategy.sol/UniStrategy.json";
                                  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
test/stubs.ts:11:28 - error TS2307: Cannot find module '../artifacts/contracts/V3Oracle.sol/V3Oracle.json' or its corresponding type declarations.

11 import OracleArtifact from "../artifacts/contracts/V3Oracle.sol/V3Oracle.json";
                              ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Recommendation: Please make sure tests could be run and cover at least 95% of code branches.

2. Inconsistency with provided docs.

While it said in the docs:

> 4. The position could be rebased if the current price/tick is outside the base range or +-15% from the upper and lower tick of the base range.
> 5. The smart contracts should not allow rebasing for pairs whose twap is diverged 10% from its current price.
> 6. The PILOT token would be rewarded for the rebasing (gas fees + (150000 gwei * gas price)) if the pair's liquidity exceeds $100,000.

in the code, we see that “$100,000” is the constant (**LIQUIDITY_VALIDATION_AMOUNT**) while it should be changeable by the governance, and also, instead of taking “150000 gwei” which also should be configurable by governance, we couldn’t find it at all. As well as items 4-5.

Contracts: UniswapLiquidityManager.sol

Constants: readjustLiquidity

Recommendation: Please make sure contracts are aligned with the docs.

Low

1. Unused constants.

Contracts: PeripheryPayments.sol

Constants: DAI, USDC, USDT

Recommendation: Remove unused constants.

Status: Fixed

2. No events on values changed.

While contract changes critical values it is recommended to emit events so the community may track such changes off-chain.

Contracts: UniStrategy.sol

Functions: setRangeMultiplier, setBaseMutiplier, setMaxTwapDeviation, setTwapDuration

Recommendation: Remove unused constants.

Status: Fixed

3. Boolean equality.

Boolean constants can be used directly and do not need to be compared to **true** or **false**.

Contracts: UniswapLiquidityManager.sol

Functions: readjustLiquidity

Recommendation: Remove the equality to the boolean constant.

Status: Fixed

4. Too many digits.

Literals with many digits are difficult to read and review.

Contracts: UniswapLiquidityManager.sol, V3Oracle.sol

Functions: readjustLiquidity

Constants: LIQUIDITY_VALIDATION_AMOUNT

Recommendation: Please use either scientific notation or ether units suffix (ie: *0.2e18* or *0.2 ether*; *100e6 ether* instead of *100000000000000000000000*).

Status: Partly 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.

Contracts: V3Oracle.sol, ULMState.sol

Functions: V3Oracle.checkPoolValidation, ULMState.getPoolAddress, V3Oracle.getPilotAmountWethPair, V3Oracle.getPilotAmountForTokens

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

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 **5** low severity issues.

After the second review security engineers found some changes in the code related to interfaces and some code reorganization. Unfortunately, inconsistencies between the code and documentation weren't fixed therefore there are still **2** medium and **2** low issues.

Notice:

Contracts are written in a very SDKish manner which makes it difficult to understand all inputs and outputs. There are some inconsistencies with the provided business logic documentation as well as no technical documentation.

Notice 2:

We'd recommend rewriting the UniswapLiquidityManager contract to be more straightforward and linear logic. Right now there are too many logic branches that could bring to misunderstanding or just confuse anyone who tries to unravel the logic.

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.