HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: WowSwap
Date: June 3rd, 2021

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

This document may contain confidential information about IT systems and the intellectual property of the Customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed — upon a decision of the Customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for WowSwap.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Multiple purposes contracts
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/wowswap-io/protocol/tree/release/audit
Commit	52b3cb8eb21212d9e588df74ed0d962fb8dfaeff
Timeline	17 MAY 2021 - 25 MAY 2021
Changelog	25 MAY 2021 - Initial Audit 03 JUN - 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 WowSwap (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer's smart contract and its code review conducted between May 17th, 2021 - May 25th, 2021. The second code review conducted on June 03rd, 2021.

Scope

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

```
Repository:
      https://github.com/wowswap-io/protocol/tree/release/audit
Commit:
      52b3cb8eb21212d9e588df74ed0d962fb8dfaeff
Files:
      ├── dependencies
      │   ├── contracts
      │   │   ├── Address.sol
      │   │   ├── Context.sol
      │   │   ├── ERC20.sol
      │   │   ├── IERC20.sol
      │   │   ├── IERC20Detailed.sol
      │   │   ├── Multicall.sol
      │   │   ├── Ownable.sol
      │   │   ├── SafeERC20.sol
      │   │   └── SafeMath.sol
      │   ├── pancake
      │   │   ├── core
      │   │   │   ├── PancakeERC20.sol
      │   │   │   ├── PancakeFactory.sol
      │   │   │   ├── PancakePair.sol
      │   │   │   ├── interfaces
      │   │   │   │   ├── IPancakeCallee.sol
      │   │   │   │   ├── IPancakeERC20.sol
      │   │   │   │   ├── IPancakeFactory.sol
      │   │   │   │   └── IPancakePair.sol
      │   │   │   └── libraries
      │   │   │       ├── Math.sol
      │   │   │       ├── SafeMath.sol
      │   │   │       └── UQ112x112.sol
      │   │   ├── lib
```

```
│   │   │   ├── AddressStringUtil.sol
│   │   │   ├── Babylonian.sol
│   │   │   ├── BitMath.sol
│   │   │   ├── FixedPoint.sol
│   │   │   ├── FullMath.sol
│   │   │   ├── SafeERC20Namer.sol
│   │   │   └── TransferHelper.sol
│   │   └── periphery
│   │       ├── PancakeRouter.sol
│   │       ├── PancakeRouter01.sol
│   │       ├── interfaces
│   │       │   ├── IPancakeMigrator.sol
│   │       │   ├── IPancakeRouter01.sol
│   │       │   ├── IPancakeRouter02.sol
│   │       │   ├── IWETH.sol
│   │       │   └── V1
│   │       │       ├── IUniswapV1Exchange.sol
│   │       │       └── IUniswapV1Factory.sol
│   │       └── libraries
│   │           ├── PancakeLibrary.sol
│   │           └── SafeMath.sol
│   └── upgradeability
│       ├── AdminUpgradeabilityProxy.sol
│       ├── BaseAdminUpgradeabilityProxy.sol
│       ├── BaseUpgradeabilityProxy.sol
│       ├── Initializable.sol
│       ├── InitializableAdminUpgradeabilityProxy.sol
│       ├── InitializableUpgradeabilityProxy.sol
│       ├── Proxy.sol
│       └── UpgradeabilityProxy.sol
├── interfaces
│   ├── IPair.sol
│   ├── IPairExplorer.sol
│   ├── IPairFactory.sol
│   ├── IParamProvider.sol
│   ├── IParamProviderFactory.sol
│   ├── IPriceGuard.sol
│   ├── IReserve.sol
│   ├── IReserveFactory.sol
│   ├── IRouter.sol
│   └── uniswap
│       └── IUniswapRouter.sol
```

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

```
├── mocks
│   ├── MockInterestStrategy.sol
│   ├── MockPancackeRouter.sol
│   ├── MockPriceGuard.sol
│   ├── MockPricedPancakeRouter.sol
│   ├── MockReserveLogic.sol
│   ├── MockToken.sol
│   ├── MockTokenWithFeeOnTransfer.sol
│   ├── guards
│   │   └── MockChainlinkPriceGuard.sol
│   └── v1
│       ├── PairFactoryV1.sol
│       ├── PairV1.sol
│       ├── ParamProviderFactoryV1.sol
│       ├── ParamProviderV1.sol
│       ├── ReserveFactoryV1.sol
│       ├── ReserveV1.sol
│       └── interfaces
│           ├── IPairFactoryV1.sol
│           ├── IPairV1.sol
│           ├── IParamProviderFactoryV1.sol
│           ├── IParamProviderV1.sol
│           ├── IReserveFactoryV1.sol
│           ├── IReserveV1.sol
│           └── IRouterV1.sol
└── protocol
    ├── Pair.sol
    ├── PairExplorer.sol
    ├── PairFactory.sol
    ├── ParamProvider.sol
    ├── ParamProviderFactory.sol
    ├── Reserve.sol
    ├── ReserveFactory.sol
    ├── Router.sol
    ├── guards
    │   ├── AllowAnyPriceGuard.sol
    │   ├── ChainlinkPriceFactory.sol
    │   ├── ChainlinkPriceGuard.sol
    │   └── MultiChainlinkPriceGuard.sol
    └── libraries
        ├── CoreLibrary.sol
        ├── DataTypes.sol
```

```
├── EnumerableSet.sol
├── FactoryLibrary.sol
├── helpers
│   ├── Errors.sol
│   └── TransferHelper.sol
├── logic
│   ├── DebtLibrary.sol
│   ├── InterestStrategy.sol
│   └── LiquidityMath.sol
├── math
│   ├── MathUtils.sol
│   ├── PercentageMath.sol
│   └── WadRayMath.sol
└── upgradeability
    └── Versioned.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.

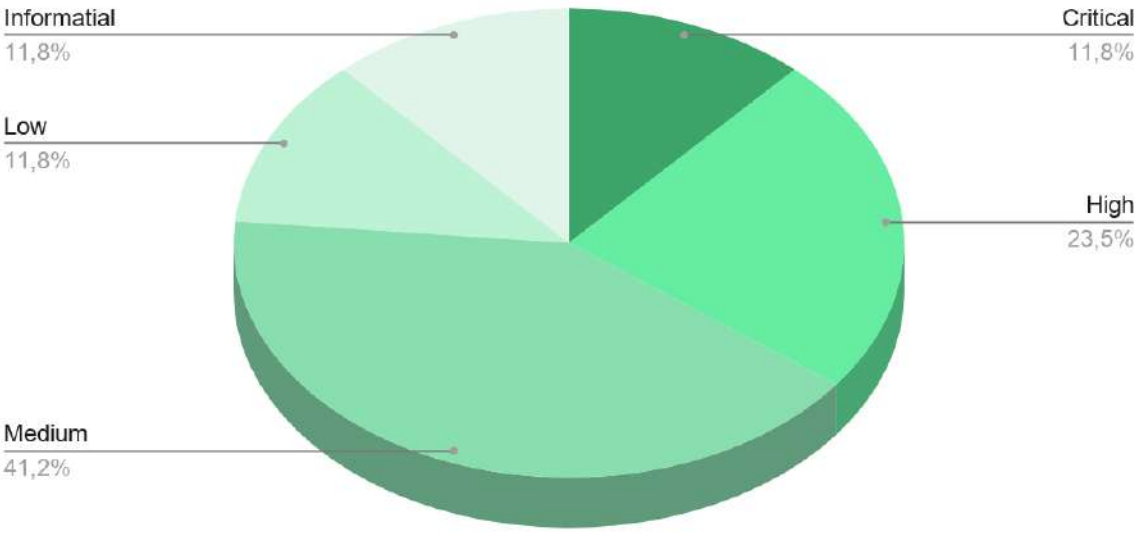
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview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

Security engineers found **2** critical, **4** high, **7** medium, **2** low, and **2** informational issues during the audit.

After the **second** review no vulnerabilities were found.

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

AS-IS overview

Contracts within Dependencies

Description

Numerous contracts exist within the dependencies directory, primarily focused on three primary elements: core functionality contracts (which should be transferred to their OpenZeppelin equivalents), PancakeSwap contracts (which is a known vulnerable codebase that has had broader security issues in the past), and upgradability (which again should be transferred to their OpenZeppelin equivalents). It is unclear why certain elements of the Pancake library are present, namely the ERC20 deployment (rather than referencing an existing deployment). In particular, the inclusion of PancakeRouter01 brings much concern, as this router had to be removed by the pancake team due to existing security issues within the contract.

Contracts within Interfaces

Description

Numerous contracts exist within the Interfaces directory. These compose the function structure of the contracts, as interfaces do not have any definition or any state variables, constructors, or any function with implementation. Interfaces only contain function declarations.

Contracts within Mocks

Description

Mock contracts are non-deployed contracts focused upon functional unit testing of various functionality. Mock contracts are easily controlled by the developer, allowing rapid iteration and development.

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

Contracts within Protocol/guards

Description

Numerous contracts

Contracts within Protocol/guards

Description

A number of singular function contracts exist within the protocol/guards directory: **AllowAnyPriceGuard, ChainlinkPriceFactory, ChainlinkPriceGuard, and MultiChainlinkPriceGuard**.

These functions (and the contracts which contain them) aim to prevent execution over a certain threshold.

Contracts within Protocol/libraries

Description

Multiple singular function contracts exist within **protocol/libraries** (as structured below).

Per the code, these libraries provide methods to calculate math, find addresses and so on. Importantly, the contract **Errors.sol** also defines the key for error messages in the WOWswap protocol:

VL = ValidationLogic

MATH = Math libraries

CT = Common errors between tokens

LP = LiquidityProvider token (pair tokens)

DT = DebtToken

P = Pair

RL = ReserveLogic

F = Factory

R = Reserve

RT = Router

CoreLibrary.sol -- Empty library definition

DataTypes.sol -- Defines a number of structs: **Debt, ReserveDebt, ReserveConfig, ReserveState** (which contains a TODO item), **ReserveData, Position, ProtocolParameters, TokenParameters and MinWOWBalanceParameters.**

EnumerableSet.sol -- Contains logic for set manipulation and creation.

FactoryLibrary.sol -- Creates an implementation of provided bytecode through **getOrCreateImplementation**.

├── helpers

│ ├── Errors.sol -- Defines error structure

│ └── TransferHelper.sol -- Helper methods for interacting

with ERC20 tokens and sending ETH that do not consistently return true/false

├── logic

│ ├── DebtLibrary.sol -- Calculation methods for user debt and

accumulated interest from **ReserveDebt** storage data.

│ ├── InterestStrategy.sol -- Calculates utilization,

interest, and borrow rates based on current debt and available liquidity.

│ └── LiquidityMath.sol -- Calculation of share and debt

interests

├── math

│ ├── MathUtils.sol -- Contains a function to calculate the

interest using a compounded interest rate formula. This calculation has multiple issues as denoted within the audit findings.

│ ├── PercentageMath.sol -- Provides functions to perform

percentage calculations

│ └── WadRayMath.sol -- Provides mul and div function for wads

(decimal numbers with 18 digits precision) and rays (decimals with 27 digits)

└── upgradeability

└── Versioned.sol -- Helper contract to implement

initializer functions.

Pair.sol

Description

Entry point to create trading position with leverage

Users can:

- Deposit liquidity
- Withdraw liquidity
- Open position
- Close position
- Liquidate unhealthy positions

Imports

Pair imports the following contracts:

- import "@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol";
- import "@openzeppelin/contracts-upgradeable/proxy/Initializable.sol";
- import "@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol";
- import "./../dependencies/pancake/core/interfaces/IPancakePair.sol";
- import "./../dependencies/contracts/SafeMath.sol";
- import "./../dependencies/contracts/IERC20.sol";
- import "./../interfaces/IPair.sol";
- import "./../interfaces/IRouter.sol";
- import "./../interfaces/IReserve.sol";
- import "./../interfaces/IPriceGuard.sol";
- import "../interfaces/IReserveFactory.sol";
- import "./libraries/upgradeability/Versioned.sol";
- import "./libraries/helpers/TransferHelper.sol";
- import "../dependencies/pancake/periphery/interfaces/IPancakeRouter02.sol";
- import "./libraries/math/PercentageMath.sol";
- import "../interfaces/IParamProvider.sol";

Inheritance

Pair inherits PairStorage and IPair.

Usages

Pair contract uses the following:

- using SafeMath for uint256

- using PercentageMath for uint256;

Structs

Pair contract has no structs (instead mainly inheriting data structure from PairStorage)

Enums

Pair contract has no custom enums.

Events

Pair contract emits no custom events.

Modifiers

Pair has no modifiers.

Fields

Pair has multiple fields:

- uint256 constant ONE = 10000 -- A constant, value of one
- uint256 public constant REVISION = 0x4 -- The present revision number

Functions

Pair has a number of public functions:

- ***initialize***

Description
Initializes a pool instance
Visibility
external
Modifiers
The initializer modifier is required.
Input parameters
- address reserve
- address paramProvider
- address treasurer
- address wow

- address[] calldata path
- string calldata name
- string calldata symbol

Constraints

No constraints exist.

Events emit

No event is emit.

Output

Nothing is returned.

- ***openPosition***

Description

Opens a position for trading

Visibility

external

Modifiers

None.

Input parameters

- address trader,
- uint256 leverageFactor
- uint256 amountOutMin

Constraints

Requires leverage to be greater than or equal to one, but less than the maximum amount of leverage.
Requires traders balance to meet minimum threshold for trading.

Events emit

No event is emit.

Output

Returns uint256 amountOut

- ***closePosition***

Description

Closes a position taken by a trader

Visibility

external

Modifiers

None

Input parameters

- address trader

- uint256 amountOutMin

Constraints

Amount out must be at least the minimum amount out

Events emit

No event is emit.

Output

Returns uint256 amountOut

- ***liquidatePosition***

Description

Allows a position to be liquidated

Visibility

external

Modifiers

None.

Input parameters

- address trader
- address liquidator

Constraints

No constraints exist.

Events emit

No event is emit.

Output

Boolean (returns true by default)

- ***positionCosts***

Description

Calculate costs associated with a position

Visibility

external

Modifiers

None

Input parameters

- address trader

Constraints

No constraints exist.

Events emit

No event is emit

Output

Returns the balance and debt of the trader

- ***getRateMultiplier, getBorrowLimit, getLiquidationCost, calcProfitFee, getAmountOut, getDeposit, getTotalDeposit, getLoan, getTotalLoan***

Single use getter and view functions with minimal calculations associated.

PairExplorer.sol

Description

PairExplorer contract manages data of the pair

Imports

PairExplorer has the following imports:

```
1. import "./../dependencies/contracts/IERC20.sol";
2. import "../interfaces/IPair.sol";
3. import {Position} from "./libraries/DataTypes.sol";
4. import "../interfaces/IPairExplorer.sol";
5. import "../interfaces/IReserve.sol";
6. import "./../dependencies/contracts/SafeMath.sol";
7. import "./libraries/math/PercentageMath.sol";
```

Inheritance

PairExplorer contract inherits IPairExplorer.

Usages

PairExplorer contract has two usages, using SafeMath for uint256 and using PercentageMath for uint256;

Structs

PairExplorer contract has no structs.

Enums

PairExplorer contract has no custom enums.

Events

PairExplorer contract has no events.

Modifiers

PairExplorer has no modifiers.

Fields

PairExplorer has no fields.

Functions

PairExplorer has the following public functions:

getPair, getRoutablePair, getReserve, getPosition, getProxyPosition, calculateBalance, calculateProxyBalance, calculateOpenPosition, calculateOpenProxyPosition

Description
Getter functions with minimal computation.

PairFactory.sol

Description

PairFactory is responsible for Pair creation

Imports

PairFactory contract has multiple inputs:

- import "@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol";
- import "@openzeppelin/contracts-upgradeable/proxy/Initializable.sol";

- import "@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol";
- import "../dependencies/upgradeability/InitializableAdminUpgradeabilityProxy.sol";
- import "./libraries/helpers/Errors.sol";
- import "./libraries/EnumerableSet.sol";
- import "./libraries/upgradeability/Versioned.sol";
- import "./Pair.sol";
- import "../dependencies/contracts/IERC20Detailed.sol";
- import "../dependencies/pancake/core/interfaces/IPancakeFactory.sol";
- import "../interfaces/IReserveFactory.sol";
- import "../interfaces/IPairFactory.sol";
- import "../interfaces/IParamProviderFactory.sol";
- import "./libraries/FactoryLibrary.sol";

Inheritance

PairFactory contract inherits PairFactoryStorage and IPairFactory.

Usages

PairFactory contract has one usage, using EnumerableSet for EnumerableSet.AddressSet.

Structs

PairFactory contract has no structs.

Enums

PairFactory contract has no enums

Events

PairFactory emits no events.

Modifiers

PairFactory has two modifiers:

- isTradable
- isProxyLendable

Fields

PairFactory has one custom field:

- uint256 public constant REVISION = 0x4;

Functions

registerTradable, registerTradables, registerProxyLendable, registerProxyLendables, upgrade,

Owner gated or single use functions with minimal computation. Primarily used for maintenance and setup

getPair, getOrCreatePair, getRoutablePair, getOrCreateRoutablePair, getAllTradeables, getAllProxyLendables
Description

Getter functions with minimal computation.

ParamProvider.sol

Description

ParamProvider defines contract storage to reuse in future implementations without copy and paste.

Imports

ParamProvider contract has multiple imports:

- import "@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol";
- import "./libraries/upgradeability/Versioned.sol";
- import "../interfaces/IParamProvider.sol";
- import {ProtocolParameters, TokenParameters} from "./libraries/DataTypes.sol";

Inheritance

ParamProvider contract inherits OwnableUpgradeable and Versioned.

Usages

ParamProvider contract has no usages.

Structs

ParamProvider contract has no structs.

Enums

ParamProvider contract has no enums.

Events

ParamProvider has no event emittance.

Modifiers

ParamProvider has no custom modifiers.

Fields

ParamProvider has one custom fields

- uint256 public constant REVISION = 0x4;

Functions

- ***initialize***

Description

Initialize pool instance

Visibility

external

Input parameters

Multiple input parameters exist:

- address owner -- The owner's wallet address
- address swapRouter -- The SwapRouter address
- ProtocolParameters calldata defaultParameters -- Configuration parameters for the protocol
- TokenParameters calldata defaultTokenParameters -- Configuration parameters for the token
- MinWOWBalanceParameters[] calldata minWOWBalances -- Minimum balance required for a position

Constraints

No constraints exist.

Events emit

No event is emit.

Output

Nothing is returned.

The remaining functions of ParamProvider are gated controls limited to onlyOwner.

ParamProviderFactory.sol

Description

ParamProviderFactory defines methods for ParamProvider creation. Beyond the **initialize** function, which validates that the proper creation of the ParamProvider is within set bounds (as determined by the **defaultParameters**)

Reserve.sol

Description

Holds investors funds to provide loans for trading positions

Usages

Reserve contract has multiple usages:

- using SafeERC20 for IERC20;
- using SafeMath for uint256;
- using WadRayMath for uint256;
- using PercentageMath for uint256;
- using LiquidityMath for ReserveData;
- using DebtLibrary for ReserveDebt;
- using InterestStrategy for ReserveConfig;

Structs

Reserve contract has no structs.

Enums

Reserve contract has no enums

Events

Reserve emits no events.

Modifiers

Reserve has one modifiers:

- onlyPair -- Only pair may call a function with this modifier

Fields

Reserve has one custom field:

- uint256 public constant REVISION = 0x4;

Functions

Reserve has the following public functions:

- ***initialize***

Description

Initialize reserve instance

Visibility

external

Input parameters

Multiple input parameters exist:

- address pairFactory_ -- Address of the associated pair factory
- address param_provider -- Address of the *ParamProvider*
- string calldata name_ -- Name associated with the reserve
- string calldata symbol_ -- Symbol associated with the reserve
- address liquidityToken_ -- Liquidity token address associated with the reserve.

Constraints

No constraints exist.

Events emit

No event is emit.

Output

Nothing is returned.

- ***fill***

Description

Fills the reserve

Visibility

external

Input parameters

No input parameters exist.

Constraints

No constraints exist.

Events emit

The *Fill* event is emit.

Output

Nothing is returned.

- ***deposit***

Description

Deposits into a reserve instance

Visibility

external

Input parameters

One input parameter exists:

- address investor -- Address of the investor

Constraints

No constraints exist.

Events emit

The Deposit event is emitted.

Output

Nothing is returned.

- ***withdraw***

Description

Withdraws from the reserve instance, and sets the interest rate to 0.

Visibility

external

Input parameters

Multiple input parameters exist:

- address from, address to -- Self explanatory

Constraints

Available balance must be greater than zero.

Events emit

No event is emitted.

Output

Nothing is returned.

- ***repay***

Description

Repay a borrow

Visibility

external

Input parameters

Multiple input parameters exist:

- address pair, address trader -- Self explanatory

Constraints

No constraints exist.

Events emit

The Repay event is emit.

Output

debtLeft is returned.

getState, getConfig, getReserveDebt, getDebtState, getDebt, getLiquidity, getLiquidityIncrease, getHolder, getLiquidityFeeAccrued, shareOf, liquidityOf, getTotalLiquidity, getBorrowRate, getLiquidityRate, getAvailableLiquidity, getTotalDebt, getUtilizationRate, transferDebt, calculateDeposit, calculateWithdraw, calculateBorrow

Description

Getter functions with minimal computation.

ReserveFactory.sol

Description

ReserveFactory defines methods for methods for Reserve creation. Beyond the **initialize** function, which validates that the proper creation of the Reserve is within set bounds (as determined by the **defaultParameters** of the **ParamProvider**)

Router.sol

Description

Router defines methods for overall position creation and routing.

Inheritance

Router inherits RouterStorage, IRouter, and PairExplorer.

Usages

Router contract has multiple usages:

- using SafeERC20 for IERC20;
- using SafeMath for uint256;

Structs

Router contract has no structs.

Enums

Router contract has no enums

Events

Router emits no events.

Modifiers

Router has one modifiers:

- ensure -- Ensure the block timestamp based deadline has not passed

Fields

Reserve has one custom field:

- uint256 public constant REVISION = 0x4;

Functions

Router has the following public functions:

- ***openPosition, openProxyPosition***

Description

Opens a position

Visibility

external

Input parameters

Multiple input parameters exist:

- uint256 amountIn -- Amount inbound
- uint256 leverageFactor -- Amount of leverage used
- uint256 amountOutMin -- Minimum amount out
- address lendable -- The lendable address
- (only on **openProxyPosition**) address proxyLendable -- The lendable proxy address
- address tradable -- The tradeable address
- address trader -- The trader address
- uint256 deadline - The deadline (in epoch seconds)

Constraints

No constraints exist.

Events emit

No event is emit.

Output

Return IPair(pair)

- *openPositionETH*

Description

Opens a position

Visibility

external

Input parameters

Multiple input parameters exist:

- uint256 leverageFactor -- Amount of leverage used
- uint256 amountOutMin -- Minimum amount out
- address tradable -- The tradeable address
- address trader -- The trader address
- uint256 deadline - The deadline (in epoch seconds)

Constraints

No constraints exist.

Events emit

No event is emit.

Output

Return IPair(pair)

- *closePosition, closeProxyPosition, closePositionETH*

Description

Closes a position

Visibility

external

Input parameters

Multiple input parameters exist:

- uint256 amountIn -- Amount inbound
- uint256 amountOutMin -- Minimum amount out
- (only on **closePosition** and **closeProxyPosition**) address lendable -- The lendable address
- (only on **closeProxyPosition**) address proxyLendable -- The lendable proxy address
- address tradable -- The tradeable address
- address trader -- The trader address
- uint256 deadline - The deadline (in epoch seconds)

Constraints

No constraints exist.

Events emit

No event is emit.

Output

Return IPair(pair), or nothing, in the case of **closePositionEth**.

- ***liquidatePosition, liquidateProxyPosition***

Description

Liquidates a position

Visibility

external

Input parameters

Multiple input parameters exist:

- address lendable -- Lendable address
- address proxyLendable (only with *liquidateProxyPosition*) -- *Proxy address*
- address tradable -- Tradable address
- address trader -- Trader address

Constraints

The pair must not be address 0.

Events emit

No event is emit.

Output

Nothing is returned.

- ***deposit, depositETH***

Description

Deposit into the reserve

Visibility

external

Input parameters

Multiple input parameters exist:

- address lendable -- Lendable address
- uint256 amount -- Amount
- address to -- To address

Constraints

The pair must not be address 0.

Events emit

No event is emitted.

Output

Nothing is returned.

- ***withdraw, withdrawETH***

Description

Withdraw from the reserve

Visibility

external

Input parameters

Multiple input parameters exist:

- address lendable -- Lendable address, not present on **withdrawEth**
- uint256 amount -- Amount withdrawn
- address to -- Address received

Constraints

Reserve for withdrawal must exist

Events emit

No event is emitted.

Output

Nothing is returned.

WETH, swapFactory, swapRouter, reserveFactory, pairFactory

Description

Getter functions with minimal computation.

sweepFee, getReserve

Description

Proxy functions with minimal computation.

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

Audit overview

■ ■ ■ ■ Critical

1. Described functionality not present in CoreLibrary despite code comments suggesting functionality should exist. If the constructor should be empty, a comment should be placed alluding to such.

Fixed before the second audit

2. High interest rates and long compounding times can lead to high inaccuracies as a result of choices made by the team to save gas. The error of this approximation can become quite substantial, especially for per-second compounding. For example, a 25 % APR could have an error of as much as 5 % using the three-term Taylor series rather than a more complete approximation. Be cautioned that most standards calculators may also hide this error since they typically use approximations of their own, albeit more robust ones.

Partially fixed before the second audit and it's an acceptable risk by the customer

■ ■ ■ High

1. Contract code size of multiple contracts exceeds 24576 bytes (a limit introduced in Spurious Dragon). This contract may not be deployable on mainnet. Consider enabling the optimizer (with a low "runs" value!), turning off revert strings, or using libraries. **protocol/PairFactory.sol, protocol/ParamProviderFactory.sol, protocol/Reserve.sol, protocol/ReserveFactory.sol**

Fixed before the second audit

2. Numerous reentrant functions throughout the protocol could benefit from the usage of **ReentrancyGuard**. While the bulk of contracts that do possess reentrant behaviors are trusted (e.g.

native to the protocol), there are a number of those which are not.

Fixed before the second audit

3. Throughout the protocol, numerous functions aim to save gas at the cost of a loss of precision for end-users (such as within **Babylonian.sol**, **MathUtils.sol**, and many others). These optimizations should be made apparent to end-users.

Customer accepts this risk

4. FIXME item remains in code (**Pair.sol**, **PairExplorer.sol**)

Fixed before the second audit

■ ■ Medium

1. Unused local variable located at protocol/Reserve.sol:204 (**uint256 fee**).

Fixed before the second audit

2. Unused function parameter (**amount**). Remove or comment out the variable name to silence this warning (protocol/Reserve.sol:448) function calculateDeposit(uint256 **amount**, address investor)

Fixed before the second audit

3. Loss of precision in ParamProvider.minWOWBalance(uint256) (protocol/ParamProvider.sol#160-169) as it performs a multiplication on the result of a division: leverageFactorRoundedUp = (leverageFactor / 10000) * 10000

Omitted as intentional behaviour

4. Legacy (read: non-utilized) code remains within **PairStorage** and **PairFactoryStorage**. These elements should be removed if no longer in use.

Customer accepts this risk

5. **Burn** within **PancakePair.sol** should have safety checks native to the function (as it is externally facing) rather than rely upon the external contract (as individuals are not restricted from interfacing with it).

Customer accepts this risk

6. TODO item remains in code (within **DataTypes.sol**)

Fixed before the second audit

7. **registerLendables** within **ReserveFactory.sol** can result in resource exhaustion when too many lendables are registered at one time.

Customer accepts this risk

- **Low**

1. No zero / null checks exist on setters (seen most prominently within **ParamProvider.sol**).

Omitted as intentional behaviour

2. Error message in PancakePair could be more verbose **require(balance0Adjusted.mul(balance1Adjusted) >= uint(_reserve0).mul(_reserve1).mul(1000**2), 'Pancake: K');**

Customer accepts this risk

- **Lowest / Code style / Best Practice**

1. Solidity style guide is not followed for variable or function naming.

Customer accepts this risk

2. Extensive typographical errors throughout the contract, primarily within code comments.

Fixed before the second audit

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.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security engineers found **2** critical, **4** high, **7** medium, **2** low, and **2** informational issues during the audit.

After the **second** review no vulnerabilities were found.

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 security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree 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 security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.