HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Warp

Date: December 16th, 2020

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 and 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 fixed - upon a decision of the Customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for Warp (66 pages)
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Collateralized Loans
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Git	HTTPS://GITHUB.COM/WARPFINANCE/WARP-CONTRACTS/
Commit	3E25B22C73290F64023A59850E17293A1F41094E
Timeline	18TH NOV 2020 – 10TH DEC 2020
Changelog	24TH NOV 2020 - Initial Audit 25TH NOV 2020 – Remediation check 11TH DEC 2020 – Remediation check 15TH DEC 2020 – Remediation check 16TH DEC 2020 – Remediation check

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 Warp (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 18th, 2020 – December 11th, 2020.

Scope

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

Intial audit
(1)Archive name: Warp-master.zip
SHA256 checksum: 0AF09EEFF431848635FBF3BD4B5CEAFC2D31CA0E50D5F8595A7776375D0D587E
or
(2)HTTPS://GITHUB.COM/WARPFINANCE/WARP/TREE/MASTER/SMART-CONTRACTS
3AA37FF41AB561CAA09EA54AF606E4258B261E67

Files in scope of review
Warp-master/Smart-Contracts/contracts/WarpVaultSC.sol
Warp-master/Smart-Contracts/contracts/WarpControl.sol
Warp-master/Smart-Contracts/contracts/UniswapLPOracleFactory.sol
Warp-master/Smart-Contracts/contracts/UniswapLPOracleInstance.sol
Warp-master/Smart-Contracts/contracts/WarpVaultLP.sol
Warp-master/Smart-Contracts/contracts/WarpVaultSCFactory.sol
Warp-master/Smart-Contracts/contracts/WarpVaultLPFactory.sol
Warp-master/Smart-Contracts/contracts/WarpWrapperToken.sol
Warp-master/Smart-Contracts/contracts/interfaces/UniswapLPOracleFactoryI.sol
Warp-master/Smart-Contracts/contracts/interfaces/WarpControlI.sol
Warp-master/Smart-Contracts/contracts/interfaces/WarpVaultSCI.sol
Warp-master/Smart-Contracts/contracts/interfaces/WarpVaultLPI.sol
Warp-master/Smart-Contracts/contracts/interfaces/WarpVaultSCFactoryI.sol
Warp-master/Smart-Contracts/contracts/interfaces/WarpVaultLPFactoryI.sol

Remediation check GIT and ETHERSCAN
(1)HTTPS://GITHUB.COM/WARPFINANCE/WARP-CONTRACTS/
F0523B4BCC26B819E5206517ADA77A93D2D01D0D
(2)Adresses on mainnet:

Contract	Address
Warp Control	0xcc8d17feeb20969523f096797c3d5c4a490ed9a8
DAI Vault	0x31Cd9B3525946B521B01FaB324B0aa1807A078a2
USDC Vault	0x2BE5e4E7711ccC1c665b718AB2D22aA11307638e
ETH-DAI Vault	0x500D083a118A23b805332BF4F8Be57257d9C70Be
ETH-USDT Vault	0x84be8517AA1Ac3027b89A83e64B8C039C71B9176
ETH-USDC Vault	0xaB3442Be99d4F291234437769EDE690f39a24851
ETH-WBTC Vault	0x22A9fb8704bc2B89CdAf0E0Bac0B8fd5Ae7cD98d

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

Remediation check GIT
HTTPS://GITHUB.COM/WARPFINANCE/WARP-CONTRACTS/
3E25B22C73290F64023A59850E17293A1F41094E

Final version of contracts:

Contract	Address
Warp Oracle	0x4A224CD0517f08B26608a2f73bF390b01a6618c8
Warp Control	0xBa539B9a5C2d412Cb10e5770435f362094f9541c
WBTC-WETH LP Vault	0x3c37f97F7d8f705cc230f97a0668f77a0e05D0aA
WETH-DAI LP Vault	0x13db1CB418573f4c3A2ea36486F0E421bC0D2427
USDT-WETH LP Vault	0xCDb97F4C32F065b8e93cF16BB1E5d198bcF8cA0d
USDC-WETH LP Vault	0xb64dfae5122D70Fa932f563c53921FE33967B3E0
DAI Vault	0x6046c3Ab74e6cE761d218B9117d5c63200f4b406
USDT Vault	0xDadd9bA311192d360Df13395E137f1E673C91deB
USDC Vault	0xae465FD39B519602eE28F062037F7B9c41FDc8cF

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 do not have high-level vulnerabilities and can be considered secure. But some fixes are recommended.

During the second audit, we established that all found issues were fixed by the Customer.

We described issues in the conclusion of these documents. Please read the whole document to estimate the risks well.

Insecure	Poor secured	Secured	Well-secured

You are here[1]

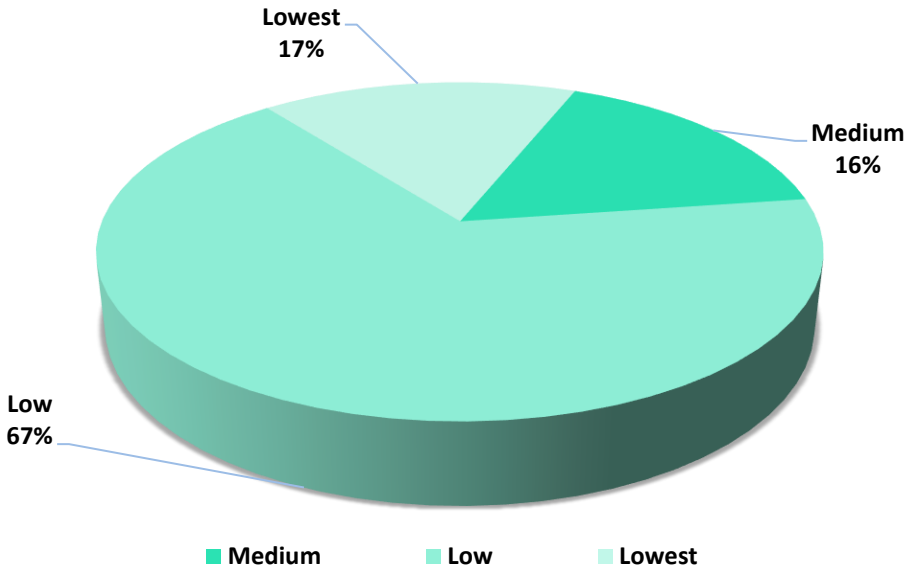
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. A general overview is presented in AS-IS section,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

Security engineers found **1** medium and **4** low, and **1** lowest severity issues during the audit.

[1] Look for details and justification in conclusion section

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

Graph 1. The distribution of vulnerabilities.



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 essential 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
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored.

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

AS-IS overview

WarpVaultSC.sol

Description

WarpVaultSC is a contract used to manage Warp Vaults.

Imports

WarpVaultSC contract has 10 imports:

- *Ownable* — from OpenZeppelin;
- *SafeMath* — from OpenZeppelin;
- *IERC20* — from OpenZeppelin;
- *Exponential* — from project files;
- *InterestRateModel* — from project files;
- *UniswapLPOracleFactoryI* — from project files;
- *WarpWrapperToken* — from project files;
- *WarpControlI* — from project files;

Inheritance

WarpVaultSC contract inherits *Ownable* and *Exponential*.

Usings

WarpVaultSC contract use:

- *SafeMath* for *uint256*;

Structs

WarpVaultSC contract has 5 data structs:

- *BorrowSnapshot* — the borrow balance information, has 2 fields:
  - *uint256 principal* — total balance;
  - *uint256 interestIndex* — global borrowing index;
- *MintLocalVars*
  - *MathError mathErr* — a math error;
  - *uint256 exchangeRateMantissa* — an exchange rate mantissa;

    - *uint256 mintTokens* — the number of minted tokens;
- *RedeemLocalVars*
    - *MathError mathErr* — a math error;
    - *uint256 exchangeRateMantissa* — an exchange rate mantissa;
    - *uint256 burnTokens* — the number of burned tokens;
    - *uint256 currentWarpBalance* — current Warp balance;
    - *uint256 currentCoinBalance* — current stablecoin balance;
    - *uint256 principalRedeemed* — principal redeemed;
- *BorrowLocalVars*
    - *MathError mathErr* — a math error;
    - *uint256 accountBorrows* — the last calculated account's borrow balance using the prior borrowIndex;
    - *uint256 accountBorrowsNew* — the *accountBorrows* after new borrow;
    - *uint256 totalBorrowsNew* — the *totalBorrows* after new borrow;
- *RepayBorrowLocalVars*
    - *MathError mathErr* — a math error;
    - *uint256 repayAmount* — a repay amount;
    - *uint256 borrowerIndex* — borrowing index of the account;
    - *uint256 accountBorrows* — the last calculated account's borrow balance;
    - *uint256 accountBorrowsNew* — the *accountBorrows* after borrow repay;
    - *uint256 totalBorrowsNew* — the *totalBorrows* after borrow repay;
    - *uint256 totalOwed* — the summ of account's *principal* and *interestIndex*;
    - *uint256 borrowPrinciple* — account's *principal*;
    - *uint256 interestPayed* — interest paid;

Modifiers

WarpVaultSC contract has 2 modifiers:

- *onlyWC* — checks if caller is warp control account;
- *angryWizard* — checks if timeWizard delay has passed;

Fields

WarpVaultSC contract has 21 fields:

- *uint256 internal initialExchangeRateMantissa* — initial exchange rate mantissa;
- *uint256 public reserveFactorMantissa* — reserve factor mantissa;
- *uint256 public accrualBlockNumber* — accrual block number;
- *uint256 public borrowIndex* — the borrow index;
- *uint256 public totalBorrows* — total borrows;
- *uint256 public totalReserves* — total reserves;
- *uint256 internal constant borrowRateMaxMantissa* — maximum borrow rate mantissa;
- *uint256 public percentA* — fee percent;
- *uint256 public percentB* — fee percent for liquidation;
- *uint256 public divisor* — the divisor;
- *uint256 public timeWizard* — the number of seconds;
- *address public warpTeam* — the address of the Warp Team used for fees;
- *ERC20 public stablecoin* — the stablecoin;
- *WarpWrapperToken public wStableCoin* — a token Wrapper to represent ownership of stablecoin;
- *WarpControlI public WC* — the *WarpControlI*;
- *InterestRateModel public InterestRate* — the Interest Rate Model;
- *mapping(address => BorrowSnapshot) public accountBorrows* — mapping addresses to *BorrowSnapshot*;
- *mapping(address => uint256) public principalBalance* — mapping addresses to principal balance;
- *mapping(address => uint256) public historicalReward* — mapping addresses to rewards paid;
- *mapping(address => address) public collateralAddressTracker* — mapping addresses to address;
- *mapping(address => bool) public collateralLocked* — mapping addresses to bool;

Functions

WarpVaultSC has 20 functions:

- ***constructor***

  Description

  Initializes contract.

Visibility

public

Input parameters

- *address _InterestRate* — the address of the Interest Rate Model;
- *address _StableCoin* — the address of the stablecoin;
- *address _warpControl* — the address of the Warp Control contract;
- *address _warpTeam* — the address of the Warp Team;
- *uint256 _initialExchangeRate* — the initial exchange rate mantissa;
- *uint256 _timelock* — the number of seconds;

Constraints

None

Events emit

None

Output

None

- ***getCashPrior***

Description

Used to get the balance of the stablecoin token of this contract.

Visibility

internal view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns the balance.

- ***calculateFee***

  Description

  Used to calculate the fee earned by the Warp Platform.

  Visibility

  public view

  Input parameters

  - *uint256 _payedAmount* — the full amount of stablecoin earned as interest;
  - *bool _isLiquidation* — a bool representing whether of not a fee is being calculated for a liquidation;

  Constraints

  None

  Events emit

  None

  Output

  Returns fee.

- ***accrueInterest***

  Description

Applies accrued interest to total borrows and reserves.

Visibility

public

Input parameters

None

Constraints

- Only one call is allowed in one block.
- Calculated borrow rate mantissa should be less or equal maximum borrow rate mantissa.

Events emit

- *InterestAccrued(accrualBlockNumber, borrowIndex, totalBorrows, totalReserves);*

Output

None

withdrawReserves

Description

Allows the warp team to withdraw an *_amount* from fee reserves.

Visibility

public

Input parameters

o uint _amount

Constraints

o onlyWarpT modifier.

o _amount should not exceed reserves.

Events emit

Emits the ReserveWithdraw event.

Output

None

- ***borrowBalancePrior***

  Description

  Used to get last calculated account's borrow balance using the prior *borrowIndex*.

  Visibility

  public view

  Input parameters

  - *address account* — the address of the account;

  Constraints

  None

  Events emit

  None

  Output

  Returns calculated balance.

- ***borrowBalanceCurrent***

  Description

Accrues interest to updated *borrowIndex* and then calculates account's borrow balance using the updated *borrowIndex*

Visibility

public

Input parameters

- *address account* — the address of the account;

Constraints

None

Events emit

None

Output

Returns calculated balance.

- ***getBlockNumber***

Description

Used to get block number.

Visibility

internal view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns block number.

- ***borrowRatePerBlock***

  Description

  Used to get current per-block borrow interest rate.

  Visibility

  public view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns current per-block borrow interest rate.

- ***supplyRatePerBlock***

  Description

  Used to get current per-block supply interest rate.

  Visibility

  public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns current per-block supply interest rate.

- ***totalBorrowsCurrent***

Description

Used to get current total borrows.

Visibility

external

Input parameters

None

Constraints

None

Events emit

None

Output

Returns current total borrows.

- ***exchangeRatePrior***

  Description

  Used to get not up-to-date exchange rate mantissa.

  Visibility

  public view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns not up-to-date exchange rate mantissa.

- ***exchangeRateCurrent***

  Description

  Used to get up-to-date exchange rate mantissa.

  Visibility

  public

  Input parameters

  None

  Constraints

None

Events emit

None

Output

Returns up-to-date exchange rate mantissa.

- ***getCash***

  Description

  Used to get the balance of the stablecoin token of this contract.

  Visibility

  external view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns the balance.

- ***lendToWarpVault***

  Description

  Used to lend stablecoin assets to a WaprVault.

Visibility

public

Input parameters

- *uint256 _amount* — the amount of the asset being lent;

Constraints

None

Events emit

- *StableCoinLent(msg.sender, _amount, vars.mintTokens);*

Output

None

- ***redeem***

Description

Used to redeem Warp Wrapper Token for the appropriate amount of underlying stablecoin asset.

Visibility

public

Input parameters

- *uint256 _amount* — the amount of stablecoin the user wishes to exchange;

Constraints

- *timeWizard* delay must have passed.
- The vault must have enough stablecoin.
- Account's *principalRedeemed* should be less or equal *principalBalance*.

Events emit

- *StableCoinLent(msg.sender, feeAdjusted, vars.burnTokens);*

Output

None

- ***viewAccountBalance***

  Description

  Used to view the balance of the account.

  Visibility

  public view

  Input parameters

  - *address _account* — the address of the account;

  Constraints

  None

  Events emit

  None

  Output

  Returns the balance of the account.

- ***viewHistoricalReward***

  Description

  Used to view historical reward.

  Visibility

  public view

Input parameters

- *address _account* — the address of the account;

Constraints

None

Events emit

None

Output

Returns total gains.

- ***_borrow***

Description

Borrows stablecoin assets.

Visibility

public

Input parameters

- *uint256 _borrowAmount* — the borrowed amount;
- *address _borrower* — the address of the borrower;

Constraints

- Only *WarpControl* contract can call it.
- *timeWizard* delay must have passed.
- The vault must have enough stablecoin.

Events emit

None

Output

None

- ***repayBorrow***

  Description

  Used to repay borrow.

  Visibility

  public

  Input parameters

  - *uint256 _repayAmount* — the repaid amount;

  Constraints

  - *timeWizard* delay must have passed.
  - The payback amount should be less or equal to the owed amount.
  - The caller must have enough stablecoin.

  Events emit

  - *LoanRepayed(msg.sender, feeAdjusted, accountBorrows[msg.sender].principal, accountBorrows[msg.sender].interestIndex);*

  Output

  None

- ***_repayLiquidatedLoan***

  Description

  Used to repay a loan on behalf of a liquidator.

  Visibility

  public

Input parameters

- *address _borrower* — the address of the borrower;
- *address _liquidator* — the address of the liquidator;
- *uint256 _amount* — the repaid amount;

Constraints

- Only *WarpControl* contract can call it.
- *timeWizard* delay must have passed.

Events emit

None

Output

None

WarpVaultLP.sol

Description

WarpVaultLP contract is responsible for distributing WarpWrapper tokens in exchange for stablecoin assets, holding and accounting of stablecoins and LP tokens and all associates lending/borrowing calculations for a specific Warp LP asset class.

Imports

WarpVaultLP contract has 7 imports:

- *Ownable* — from OpenZeppelin;
- *SafeMath* — from OpenZeppelin;
- *IERC20* — from OpenZeppelin;
- *WarpWrapperToken* — from project files;
- *WarpControlI* — from project files;

Inheritance

WarpVaultLP contract inherits *Ownable*.

Usings

WarpVaultLP contract use:

- *SafeMath* for *uint256*;

Modifiers

WarpVaultLP contract has 2 modifiers:

- *onlyWC* — checks if caller is warp control account;
- *angryWizard* — checks if timeWizard delay has passed;

Fields

WarpVaultLP contract has 6 fields:

- *uint256 public timeWizard* — the number of seconds;
- *string public lpName* — the name of the lp token;
- *IERC20 public LPtoken* — the lp token;
- *WarpWrapperToken public WLP* — the warp wrapper token;
- *WarpControlI public WC* — the *WarpControlI*;
- *mapping(address => uint256) public collateralizedLP* — a mapping of addresses to collateral lp tokens amount;

Functions

WarpVaultLP has 7 functions:

- ***constructor***

  Description

  Initializes contract. Sets *lpName*, *LPtoken*, *WC* and *timeWizard* fields.

  Visibility

  public

  Input parameters

  - *uint256 _timelock* — the number of seconds;

- *address _lp* — the address of the lp token;
- *address _WarpControl* — the address of *WarpControlI*;
- *string memory _lpName* — the name of the lp token;

Constraints

None

Events emit

None

Output

None

- ***provideCollateral***

Description

Used to collateralize the number of LP tokens.

Visibility

public

Input parameters

- *uint256 _amount* — the number of tokens;

Constraints

- *WarpVaultLP* contract must have enough allowance.
- The caller must have enough tokens.

Events emit

- *CollateralProvided(msg.sender, _amount);*

Output

None

- ***withdrawCollateral***

  Description

  Used to withdraw collateral LP tokens.

  Visibility

  public

  Input parameters

  - *uint256 _amount* — the number of tokens;

  Constraints

  - *timeWizard* delay must have passed.
  - The caller must have enough withdrawal allowance.
  - The caller must have enough locked collateral tokens.

  Events emit

  - *CollateralWithdraw(msg.sender, _amount);*

  Output

  None

- ***getAssetAdd***

  Description

  Used to get the address of *LPtoken*.

  Visibility

  public view

  Input parameters

  None

  Constraints

None

Events emit

None

Output

Returns the address of *LPtoken*.

- ***collateralOfAccount***

  Description

  Used to get the number of collateral tokens by account.

  Visibility

  public view

  Input parameters

  - *address _account* — the address of the account;

  Constraints

  None

  Events emit

  None

  Output

  Returns the number of collateral tokens.

- ***_liquidateAccount***

  Description

  Used to liquidate the LP tokens of the input account.

Visibility

public

Input parameters

- *address _account* — the address of account;
- *address _liquidator* — the address of liquidator;

Constraints

- Only the Warp Control contract can call it.
- *timeWizard* delay must have passed.

Events emit

None

Output

None

- ***valueOfAccountCollateral***

Description

Used to get value of all collateral LP tokens by account.

Visibility

external view

Input parameters

- *address _account* — the address of the account;

Constraints

None

Events emit

None

Output

Returns value of all collateral LP tokens by account.

WarpControl.sol

Description

WarpControl is a contract used to manage Warp Vaults.

Imports

WarpControl contract has 10 imports:

- *Ownable* — from OpenZeppelin;
- *SafeMath* — from OpenZeppelin;
- *IERC20* — from OpenZeppelin;
- *WarpVaultSCI* — from project files;
- *WarpVaultLPI* — from project files;
- *WarpVaultLPFactoryI* — from project files;
- *WarpVaultSCFactoryI* — from project files;
- *UniswapLPOracleFactoryI* — from project files;
- *JumpRateModelV2* — from project files;
- *Exponential* — from project files;

Inheritance

WarpControl contract inherits *Ownable* and *Exponential*.

Usings

WarpControl contract use:

- *SafeMath* for *uint256*;

Modifiers

WarpControl contract has 1 modifier:

- *onlyVault* — checks if caller is warp vault;

Fields

WarpControl contract has 11 fields:

- *UniswapLPOracleFactoryI public Oracle* — the *UniswapLPOracleFactoryI* interface;
- *WarpVaultLPFactoryI public WVLPF* — the *WarpVaultLPFactoryI* interface;
- *WarpVaultSCFactoryI public WVSCF* — the *WarpVaultSCFactoryI* interface;
- *address public warpTeam* — the address of the Warp Team used for fees;
- *address[] public lpVaults* — a list of LP vaults;
- *address[] public scVaults* — a list of SC vaults;
- *mapping(address => address) public instanceLPTracker* — mapping the LP token address to this Warp Vault address;
- *mapping(address => address) public instanceSCTracker* — mapping the SC token address to this Warp Vault address;
- *mapping(address => uint256) public lockedLPValue* — mapping the address to the number of locked LP tokens;
- *mapping(address => bool) public isVault* — mapping the address to 'is vault' indicator;
- *mapping(address => uint256) nonCompliant* — mapping the address to the time of noncompliance;

Functions

WarpControl has 20 functions:

- ***constructor***

  Description

  Initializes contract. Sets *Oracle*, *WVLPF*, *WVSCF* and *warpTeam* fields.

  Visibility

  public

  Input parameters

  - *address _oracle* — the *UniswapLPOracleFactory* address;
  - *address _WVLPF* — the *WarpVaultLPFactory* address;

- *address _WVSCF* — the *WarpVaultSCFactory* address;
- *address _warpTeam* — the address of the Warp Team;

Constraints

None

Events emit

None

Output

None

- ***viewNumLPVaults***

Description

Used to get the number of LP vaults.

Visibility

external view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns the number of LP vaults.

- ***viewNumSCVaults***

Description

Used to get the number of SC vaults.

Visibility

external view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns the number of SC vaults.

- ***createNewLPVault***

  Description

  Used to create a new WarpVaultLP contract for a specific LP token.

  Visibility

  public

  Input parameters

  - *uint256 _timelock* — the number of seconds;
  - *address _lp* — the address of LP token;
  - *address _lpAsset1* — the address for the first asset in a pair that the LP token represents;
  - *address _lpAsset2* — the address for the second asset in a pair that the LP token represents;

- *string memory _lpName* — the name of the LP token;

Constraints

- Only owner can call it.

Events emit

- *NewLPVault(_WarpVault);*

Output

None

- ***importLPVault***

Description

Used to import LP vault.

Visibility

public

Input parameters

- *address _lpVault* — the address of LP vault;

Constraints

- Only owner can call it.

Events emit

- *ImportedLPVault(_lpVault);*

Output

None

- ***manuallyCreateOracles***

Description

Creates a new oracle.

Visibility

public

Input parameters

- *address _token0* — the address of token;
- *address _token1* — the address of token;
- *address _lpToken* — the address of LP token;

Constraints

- Only owner can call it.

Events emit

None

Output

None

- ***createNewSCVault***

Description

Used to create a new WarpVaultSC contract for a specific SC token.

Visibility

public

Input parameters

- *uint256 _timelock* — the number of seconds;
- *uint256 _baseRatePerYear* — the base rate per year;
- *uint256 _multiplierPerYear* — the multiplier per year;
- *uint256 _jumpMultiplierPerYear* — the jump multiplier per year;
- *uint256 _optimal* — the utilization point or "kink" at which the jump multiplier is applied;

- *uint256 _initialExchangeRate* — the initial exchange rate
- *address _StableCoin* — the address of the StableCoin;

Constraints

- Only owner can call it.

Events emit

- *NewSCVault(_WarpVault, IR);*

Output

None

- ***importSCVault***

Description

Used to import SC vault.

Visibility

public

Input parameters

- *address _scVault* — the address of SC vault;

Constraints

- Only owner can call it.

Events emit

- *ImportedSCVault(_scVault);*

Output

None

- ***getMaxWithdrawAllowed***

Description

Used to get the number of LP tokens that an account is allowed to withdraw.

Visibility

public

Input parameters

- *address account* — the address of account;
- *address lpToken* — the address of LP token;

Constraints

None

Events emit

None

Output

Returns the number of LP tokens.

- ***viewMaxWithdrawAllowed***

Description

Used to view the number of LP tokens that an account is allowed to withdraw.

Visibility

public view

Input parameters

- *address account* — the address of account;
- *address lpToken* — the address of LP token;

Constraints

None

Events emit

None

Output

Returns the number of LP tokens.

- ***getTotalAvailableCollateralValue***

  Description

  Used to get total available collateral value.

  Visibility

  public

  Input parameters

  - *address account* — the address of account;

  Constraints

  None

  Events emit

  None

  Output

  Returns total available collateral value.

- ***viewTotalAvailableCollateralValue***

  Description

Used to view total available collateral value.

Visibility

public view

Input parameters

- *address account* — the address of account;

Constraints

None

Events emit

None

Output

Returns total available collateral value.

- ***viewPriceOfCollateral***

Description

Used to fetch the price of collateral.

Visibility

public view

Input parameters

- *address lpToken* — the address of LP token;

Constraints

None

Events emit

None

Output

Returns the price of collateral.

- ***viewPriceOfToken***

  Description

  Used to fetch the price of token.

  Visibility

  public view

  Input parameters

  - *address token* — the address of token;

  Constraints

  None

  Events emit

  None

  Output

  Returns the price of token.

- ***viewTotalBorrowedValue***

  Description

  Used to view the total borrowed value.

  Visibility

  public view

Input parameters

- *address account* — the address of account;

Constraints

None

Events emit

None

Output

Returns the total borrowed value.

- ***getTotalBorrowedValue***

Description

Used to get the total borrowed value.

Visibility

public

Input parameters

- *address account* — the address of account;

Constraints

None

Events emit

None

Output

Returns the total borrowed value.

- ***calcBorrowLimit***

  Description

  Used to calc the borrow limit for collateral value.

  Visibility

  public pure

  Input parameters

  - *uint256 _collateralValue* — a value of collateral;

  Constraints

  None

  Events emit

  None

  Output

  Returns the borrow limit.

- ***calcCollateralRequired***

  Description

  Used to calc the required collateral.

  Visibility

  public view

  Input parameters

  - *uint256 _borrowAmount* — an amount of the borrow;

  Constraints

  None

Events emit

None

Output

Returns the required collateral.

- ***getBorrowLimit***

  Description

  Used to get the borrow limit for the account.

  Visibility

  public

  Input parameters

  - *address account* — the address of account;

  Constraints

  None

  Events emit

  None

  Output

  Returns the borrow limit.

- ***viewBorrowLimit***

  Description

  Used to view the borrow limit for the account.

  Visibility

public view

Input parameters

- *address account* — the address of account;

Constraints

None

Events emit

None

Output

Returns the borrow limit.

- ***borrowSC***

Description

Used to borrow StableCoin.

Visibility

public

Input parameters

- *address _StableCoin* — the address of StableCoin;
- *uint256 _amount* — the borrow amount;

Constraints

- The caller must have enough borrow allowed amount.

Events emit

- *NewBorrow(msg.sender, _StableCoin, _amount);*

Output

None

- ***markAccountNonCompliant***

  Description

  Used by a potential liquidator to mark an account as noncompliant.

  Visibility

  public

  Input parameters

  - *address _borrower* the address of the borrower;

  Constraints

  - The borrower is not yet noncompliant.

  Events emit

  - *NotCompliant(_borrower, now);*

  Output

  None

- ***liquidateAccount***

  Description

  Used to liquidate a noncompliant loan.

  Visibility

  public

  Input parameters

  - *address _borrower* the address of the borrower;

  Constraints

- The caller cannot be *_borrower*.
- Half an hour should pass after the borrower is marked as noncompliant.

Events emit

- *complianceReset(_borrower, now);*

Output

None

UniswapLPOracleFactory.sol

Description

UniswapLPOracleFactory contract is designed to produce individual UniswapLPOracleInstance contracts

Imports

UniswapLPOracleFactory contract has 5 imports:

- *Ownable* — from OpenZeppelin;
- *UniswapV2Router02* — from Uniswap;
- *UniswapV2Library* — from Uniswap;
- *UniswapLPOracleInstance* — from project files;
- *ExtendedIERC20* — from project files;

Inheritance

UniswapLPOracleFactory contract inherits *Ownable*.

Usings

UniswapLPOracleFactory contract use:

- *SafeMath* for *uint256*;

Fields

UniswapLPOracleFactory contract has 6 fields:

- *address public usdc_add* — the address of the ERC20 USDC;
- *address public factory* — the address of the Uniswap Factory contract;
- *IUniswapV2Router02 public uniswapRouter* — the Uniswap Router;
- *mapping(address => address[]) LPAssetTracker* — mapping of LP token address to its oracles addresses ;
- *mapping(address => address) instanceTracker* — mapping of oracle address to token address;
- *mapping(address => address) public tokenToUSDC* — mapping of token address to oracle address;

Functions

UniswapLPOracleFactory has 5 functions:

- ***constructor***

  Description

  Initializes contract.

  Visibility

  public

  Input parameters

  - *address usdcAdd* — the address of the ERC20 USDC;
  - *address _uniFactoryAdd* — the address of the Uniswap Factory contract;
  - *address _uniRouterAddress* — the address of the Uniswap Router contract;

  Constraints

  None

  Events emit

  None

  Output

None

- ***USDC***

  Description

  Used to get the address of the ERC20 USDC.

  Visibility

  public view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns the address of the ERC20 USDC.

- ***OneUSDC***

  Description

  Used to get USDC token multiplier.

  Visibility

  public

  Input parameters

  None

Constraints

None

Events emit

None

Output

Returns USDC token multiplier.

- ***createNewOracles***

  Description

  Creates new oracle.

  Visibility

  public

  Input parameters

  - *address _tokenA* — the address of the first token in a liquidity pair;
  - *address _tokenB* — the address of the second token in a liquidity pair;
  - *address _lpToken* — the address of the token that this oracle will provide a price feed for;

  Constraints

  - Only owner can call it.

  Events emit

  None

  Output

  None

- ***getUnderlyingPrice***

  Description

  Used to calculate price of one LP token.

  Visibility

  public

  Input parameters

  - *address _lpToken* — the address of the LP token;

  Constraints

  None

  Events emit

  None

  Output

  Returns price of one LP token.

- ***viewUnderlyingPrice***

  Description

  Used to retrieve LP token price.

  Visibility

  public view

  Input parameters

  - *address _lpToken* — the address of the LP token;

  Constraints

  None

Events emit

None

Output

Returns price of one LP token.

- ***viewPriceOfToken***

  Description

  Used to retrieve token price.

  Visibility

  public view

  Input parameters

  - *address _token* — the address of the token;

  Constraints

  - The token should be registered with a USDC pairing.

  Events emit

  None

  Output

  Returns price token.

UniswapLPOracleInstance.sol

Description

UniswapLPOracleInstance contract used to get prices of LP tokens.

Imports

UniswapLPOracleInstance contract has 6 imports:

- *Ownable* — from OpenZeppelin;
- *IUniswapV2Factory* — from Uniswap;
- *IUniswapV2Pair* — from Uniswap;
- *FixedPoint* — from Uniswap;
- *UniswapV2OracleLibrary* — from Uniswap;
- *UniswapV2Library* — from Uniswap;

Inheritance

UniswapLPOracleInstance contract inherits *Ownable*.

Usings

UniswapLPOracleInstance contract use:

- *SafeMath* for *uint256*;
- *FixedPoint* for *;

Fields

UniswapLPOracleInstance contract has 9 fields:

- *uint256 public constant PERIOD* — 1 hour;
- *IUniswapV2Pair public pair* — Uniswap pair;
- *address public token0* — the address of the first token in the pair;
- *address public token1* — the address of the second token in the pair;
- *uint256 public price0CumulativeLast* — the current accumulated price value;
- *uint256 public price1CumulativeLast* — the current accumulated price value;
- *uint32 public blockTimestampLast* — the last update timestamp;
- *FixedPoint.uq112x112 public price0Average* — an avarage price;
- *FixedPoint.uq112x112 public price1Average* — an avarage price;

Functions

UniswapLPOracleInstance has 4 functions:

- ***constructor***

Description

Initializes contract.

Visibility

public

Input parameters

- *address _factory* — the address of the Uniswap factory contract;
- *address _tokenA* — the address of the asset being looked up;
- *address _tokenB* — the address of the USDC token;

Constraints

- The pair must have reserves.

Events emit

None

Output

None

- ***update***

Description

Updates the prices.

Visibility

public

Input parameters

None

Constraints

None

Events emit

None

Output

None

- ***consult***

  Description

  Updates the prices and gets the price of a token.

  Visibility

  external

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns the price of a token.

- ***viewPrice***

  Description

  Retrieves the current price of a token.

Visibility

external view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns the price of a token.

WarpVaultSCFactory.sol

Description

WarpVaultSCFactory contract is designed to produce individual WarpVaultSC contracts.

Imports

WarpVaultSCFactory contract has 2 imports:

- *Ownable* — from OpenZeppelin;
- *WarpVaultSC* — from project files;

Inheritance

WarpVaultSCFactory contract inherits *Ownable*.

Functions

WarpVaultSCFactory has 1 function:

- ***createNewWarpVaultSC***

  Description

  Used to create new WarpVaultSC contract instances.

  Visibility

  public

  Input parameters

  - *address _InterestRate* — the address of the Interest Rate Model;
  - *address _StableCoin* — the address of the stablecoin;
  - *address _warpControl* — the address of the Warp Control contract;
  - *uint256 _initialExchangeRate* — the initial exchange rate mantissa;
  - *uint256 _timelock* — the number of seconds;

  Constraints

  - Only owner can call it.

  Events emit

  None

  Output

  Returns the address of created WarpVaultSC contract instance.

WarpVaultLPFactory.sol

Description

WarpVaultLPFactory contract is designed to produce individual WarpVaultLP contracts.

Imports

WarpVaultLPFactory contract has 2 imports:

- *Ownable* — from OpenZeppelin;
- *WarpVaultLP* — from project files;

Inheritance

WarpVaultLPFactory contract inherits *Ownable*.

Functions

WarpVaultLPFactory has 1 function:

- ***createWarpVaultLP***

  Description

  Used to create a new WarpVaultLP contract for a specific LP token.

  Visibility

  public

  Input parameters

  - *uint256 _timelock* — the number of seconds;
  - *address _lp* — the address of the lp token;
  - *string memory _lpName* — the name of the lp token;

  Constraints

  - Only owner can call it.

  Events emit

  None

  Output

  Returns the address of created WarpVaultLP contract instance.

WarpWrapperToken.sol

Description

WarpWrapperToken contract is designed as a token Wrapper to represent ownership of stablecoins added to a specific WarpVault.

Imports

WarpWrapperToken contract has 2 imports:

- *Ownable* — from OpenZeppelin;
- *ERC20* — from OpenZeppelin;

Inheritance

WarpWrapperToken contract inherits *Ownable* and *ERC20*.

Fields

WarpWrapperToken has 1 field:

- *address public stablecoin* — the address of stablecoin;

Functions

WarpWrapperToken has 3 functions:

- ***constructor***

  Description

  Initializes contract.

  Visibility

  public

  Input parameters

  - *address _SC* — the address of stablecoin;
  - *string memory _tokenName* — the name of the token;
  - *string memory _tokenSymbol* — the symbol of the token;

  Constraints

None

Events emit

None

Output

None

- ***mint***

  Description

  Used to mint tokens.

  Visibility

  public

  Input parameters

  - *address _to* — the address that will receive the new tokens;
  - *uint256 _amount* — the amount of tokens;

  Constraints

  - Only owner can call it.

  Events emit

  None

  Output

  None

- ***burn***

  Description

  Used to burn tokens.

Visibility

public

Input parameters

- *address _from* — the address where the tokens will be burnt
- *uint256 _amount* — the amount of tokens;

Constraints

- Only owner can call it.

Events emit

None

Output

None

UniswapLPOracleFactoryI.sol

Description

UniswapLPOracleFactoryI abstract contract used to interface with the *UniswapLPOracleFactory* to retrieve token prices.

Functions

UniswapLPOracleFactoryI defines 5 functions:

- *createNewOracles(address _tokenA, address _tokenB, address _lpToken) public virtual;*
- *OneUSDC() public virtual view returns (uint256);*
- *getUnderlyingPrice(address _MMI) public virtual returns (uint256);*
- *viewUnderlyingPrice(address _MMI) public view virtual returns (uint256);*
- *viewPriceOfToken(address token) public view virtual returns (uint256);*

WarpControlI.sol

Description

WarpControlI is an abstract contract used to call functions of the WarpControl contract.

Functions

WarpControlI defines 2 functions:

- *getMaxWithdrawAllowed(address account, address lpToken) public virtual returns (uint256);*
- *viewPriceOfCollateral(address lpToken) public virtual view returns (uint256);*

WarpVaultSCI.sol

Description

WarpVaultSCI is an abstract contract used to interface with a *WarpVaultSC* contract.

Functions

WarpVaultSCI defines 5 functions:

- *borrowBalanceCurrent(address account) public virtual returns (uint256);*
- *borrowBalancePrior(address account) public view virtual returns (uint256);*
- *exchangeRateCurrent() public virtual returns (uint256);*
- *_borrow(uint256 _borrowAmount, address _borrower) external virtual;*
- *_repayLiquidatedLoan(address _borrower, address _liquidator, uint256 _amount) public virtual;*

WarpVaultLPI.sol

Description

WarpVaultLPI is an abstract contract used to interface with a *WarpVaultLP* contract.

Functions

WarpVaultLPI defines 3 functions:

- *getAssetAdd() public view virtual returns (address);*
- *collateralOfAccount(address _account) public view virtual returns (uint256);*
- *_liquidateAccount(address _account, address _liquidator) public virtual;*

WarpVaultSCFactoryI.sol

Description

WarpVaultSCFactoryI is an abstract contract used to interface with a *WarpVaultSCFactory* contract.

Functions

WarpVaultSCFactoryI defines 1 function:

- *createNewWarpVaultSC(address _InterestRate, address _StableCoin, address _warpTeam, uint256 _initialExchangeRate, uint256 _timelock) public virtual returns (address);*

WarpVaultLPFactoryI.sol

Description

WarpVaultLPFactoryI is an abstract contract used to interface with a *WarpVaultLPFactory* contract.

Functions

WarpVaultLPFactoryI defines 1 function:

- *createWarpVaultLP(uint256 _timelock, address _lp, string memory _lpName) public virtual returns (address);*

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. *provideCollateral* and *withdrawCollateral* functions of the *WarpVaultLP* contract have conditions in lines 77 and 94 that seems should be '>=' not '>'.

   Fixed during the second audit in *provideCollateral* function. In the *withdrawCollateral* function, it was not an issue.

Low

1. The *WLP* field of the *WarpVaultLP* contract is not used.
2. The *collateralAddressTracker* field of the *WarpVaultSC* contract is not used.
3. The *collateralLocked* field of the *WarpVaultSC* contract is not used.
4. *viewAccountBalance* function of the *WarpVaultSC* contract has unused code.

   All low-level issues were fixed during the second audit.

Lowest / Code style / Best Practice

1. *withdrawReserves* function of the *WarpVaultSC* contract has inappropriate name and can confuse. The function controls the reserves from interest accrued from loans at 5%. No user funds are at risk.

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. For the contract, high-level description of functionality was presented in As-is overview section of the report.

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

Security engineers found **1** medium and **4** low, and **1** lowest severity issues during the audit.

Violations in the following categories were found and addressed to the Customer:

Category	Check Item	Comments
Code review	▪ Functionality Checks	▪ The project has unused code. ▪ *provideCollateral* and *withdrawCollateral* functions of the *WarpVaultLP* contract have conditions in lines 77 and 94 that seems should be '>=' not '>'.
	▪ Best practices	▪ withdrawReserves function of the WarpVaultSC contract has inappropriate name and can confuse.

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, 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 the 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 own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.