HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Axion
Date: September 23rd, 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 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 fixed - upon a decision of the customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for Axion
Type	Token, auction, tokenswap, staking
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository URL	https://github.com/Rock-n-Block/axion-contracts
Last Commit	51957A13A54F187B6B1C5B89A1C2A8D77FE168C0
Timeline	18TH SEP 2020 - 23RD SEP 2020
Changelog	23RD SEP 2020 - Initial Audit

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 Axion (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 September 18, 2020 - September 23, 2020.

Scope

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

Repository URL – https://github.com/Rock-n-Block/axion-contracts
Last Commit - 51957A13A54F187B6B1C5B89A1C2A8D77FE168C0

contracts\interfaces\IAuction.sol	In Scope of Review
contracts\interfaces\IBPD.sol	In Scope of Review
contracts\interfaces\IForeignSwap.sol	In Scope of Review
contracts\interfaces\IStaking.sol	In Scope of Review
contracts\interfaces\ISubBalances.sol	In Scope of Review
contracts\interfaces\IToken.sol	In Scope of Review
contracts\interfaces\IUniswapV2Router01.sol	In Scope of Review
contracts\interfaces\IUniswapV2Router02.sol	In Scope of Review
contracts\test\AuctionMock.sol	**Out of Scope of Review**
contracts\test\BPDMock.sol	**Out of Scope of Review**
contracts\test\HEXMock.sol	**Out of Scope of Review**
contracts\test\StakingMock.sol	**Out of Scope of Review**
contracts\test\SubBalancesMock.sol	**Out of Scope of Review**
contracts\test\TERC20.sol	**Out of Scope of Review**
contracts\test\UniswapV2Router02Mock.sol	**Out of Scope of Review**
contracts\Auction.sol	In Scope of Review
contracts\BPD.sol	In Scope of Review
contracts\ForeignSwap.sol	In Scope of Review
contracts\Migrations.sol	In Scope of Review
contracts\NativeSwap.sol	In Scope of Review
contracts\Staking.sol	In Scope of Review
contracts\SubBalances.sol	In Scope of Review
contracts\Token.sol	In Scope of Review

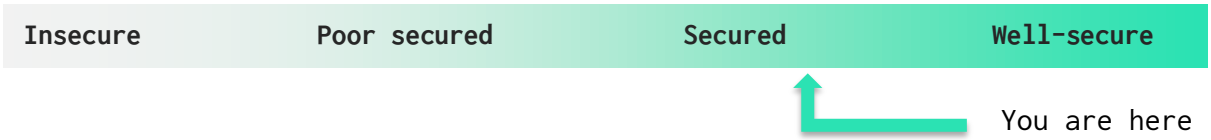
We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

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 ▪ 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 vulnerability and can be considered secure. Some fixes are required though.

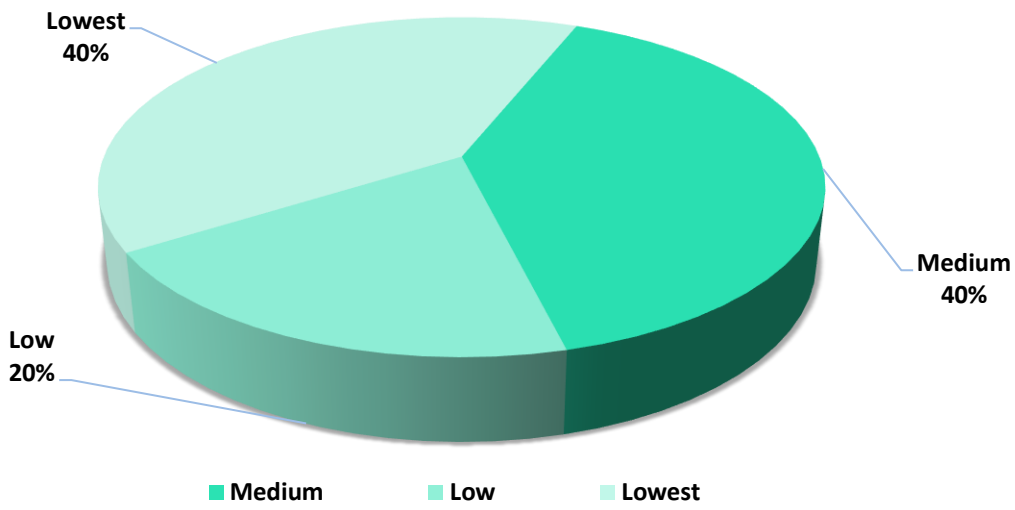


Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed and important vulnerabilities are presented in the Audit overview section. A general overview is presented in AS-IS section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

Security engineers found 4 medium, 2 low and 4 lowest severity issues during audit.

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 lose or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have 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 lose or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have 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

Interfaces

Description

Folder *./contracts/interfaces* contains all Axion interfaces:

- *IAuction.sol* defines 2 functions:
  - *callIncomeDailyTokensTrigger(uint256 amount) external;*
  - *callIncomeWeeklyTokensTrigger(uint256 amount) external;*
- *IBPD.sol* defines 3 functions:
  - *callIncomeTokensTrigger(uint256 incomeAmountToken) external;*
  - *transferYearlyPool(uint256 poolNumber) external returns (uint256);*
  - *getPoolYearAmounts() external view returns (uint256[5] memory poolAmounts);*
- *IForeignSwap.sol* defines 4 functions:
  - *getCurrentClaimedAmount() external view returns (uint256);*
  - *getTotalSnapshotAmount() external view returns (uint256);*
  - *getCurrentClaimedAddresses() external view returns (uint256);*
  - *getTotalSnapshotAddresses() external view returns (uint256);*
- *IStaking.sol*
  - *externalStake(uint256 amount, uint256 stakingDays, address staker) external;*
- *ISubBalances.sol* defines 2 functions:
  - *callIncomeStakerTrigger(address staker, uint256 sessionId, uint256 start, uint256 end, uint256 shares) external;*
  - *callOutcomeStakerTrigger(address staker, uint256 sessionId, uint256 start, uint256 end, uint256 shares) external;*
- *IToken.sol* defines 2 functions:
  - *mint(address to, uint256 amount) external;*
  - *burn(address from, uint256 amount) external;*
- *IUniswapV2Router01.sol* defines 18 functions:
  - *factory() external pure returns (address);*
  - *WETH() external pure returns (address);*
  - *addLiquidity(address tokenA, address tokenB, uint amountADesired, uint amountBDesired, uint amountAMin,*

uint amountBMin, address to, uint deadline) external returns (uint amountA, uint amountB, uint liquidity);

- *addLiquidityETH(address token, uint amountTokenDesired, uint amountTokenMin, uint amountETHMin, address to, uint deadline) external payable returns (uint amountToken, uint amountETH, uint liquidity);*
- *removeLiquidity(address tokenA, address tokenB, uint liquidity, uint amountAMin, uint amountBMin, address to, uint deadline) external returns (uint amountA, uint amountB);*
- *removeLiquidityETH(address token, uint liquidity, uint amountTokenMin, uint amountETHMin, address to, uint deadline) external returns (uint amountToken, uint amountETH);*
- *removeLiquidityWithPermit(address tokenA, address tokenB, uint liquidity, uint amountAMin, uint amountBMin, address to, uint deadline, bool approveMax, uint8 v, bytes32 r, bytes32 s) external returns (uint amountA, uint amountB);*
- *removeLiquidityETHWithPermit(address token, uint liquidity, uint amountTokenMin, uint amountETHMin, address to, uint deadline, bool approveMax, uint8 v, bytes32 r, bytes32 s) external returns (uint amountToken, uint amountETH);*
- *swapExactTokensForTokens(uint amountIn, uint amountOutMin, address[] calldata path, address to, uint deadline) external returns (uint[] memory amounts);*
- *swapTokensForExactTokens(uint amountOut, uint amountInMax, address[] calldata path, address to, uint deadline) external returns (uint[] memory amounts);*
- *swapExactETHForTokens(uint amountOutMin, address[] calldata path, address to, uint deadline) external payable returns (uint[] memory amounts);*
- *swapTokensForExactETH(uint amountOut, uint amountInMax, address[] calldata path, address to, uint deadline) external returns (uint[] memory amounts);*
- *swapExactTokensForETH(uint amountIn, uint amountOutMin, address[] calldata path, address to, uint deadline) external returns (uint[] memory amounts); function swapETHForExactTokens(uint amountOut, address[] calldata path, address to, uint deadline) external payable returns (uint[] memory amounts);*
- *quote(uint amountA, uint reserveA, uint reserveB) external pure returns (uint amountB);*

    - *getAmountOut(uint amountIn, uint reserveIn, uint reserveOut) external pure returns (uint amountOut);*
    - *getAmountIn(uint amountOut, uint reserveIn, uint reserveOut) external pure returns (uint amountIn);*
    - *getAmountsOut(uint amountIn, address[] calldata path) external view returns (uint[] memory amounts);*
    - *getAmountsIn(uint amountOut, address[] calldata path) external view returns (uint[] memory amounts);*
- *IUniswapV2Router02.sol* imports and inherits *IUniswapV2Router01*. It defines 5 functions:
    - *removeLiquidityETHSupportingFeeOnTransferTokens(address token, uint liquidity, uint amountTokenMin, uint amountETHMin, address to, uint deadline) external returns (uint amountETH);*
    - *removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address token, uint liquidity, uint amountTokenMin, uint amountETHMin, address to, uint deadline, bool approveMax, uint8 v, bytes32 r, bytes32 s) external returns (uint amountETH);*
    - *swapExactTokensForTokensSupportingFeeOnTransferTokens(uint amountIn, uint amountOutMin, address[] calldata path, address to, uint deadline) external;*
    - *swapExactETHForTokensSupportingFeeOnTransferTokens(uint amountOutMin, address[] calldata path, address to, uint deadline) external payable;*
    - *swapExactTokensForETHSupportingFeeOnTransferTokens(uint amountIn, uint amountOutMin, address[] calldata path, address to, uint deadline) external;*

Auction.sol

Description

Auction contract used to exchange ETH for tokens

Imports

Auction contract has 6 imports:

- *IERC20* - from OpenZeppelin;
- *AccessControl* - from OpenZeppelin;
- *SafeMath* - from OpenZeppelin;
- *IUniswapV2Router02* - from project files;
- *IToken* - from project files;
- *IAuction* - from project files;

Inheritance

Auction contract inherits *IAuction*, *AccessControl*.

Usings

Auction contract use:

- *SafeMath* for *uint256*;

Structs

Auction contract has 2 data structs:

- *AuctionReserves* defines 2 fields:
  - *eth* – an amount of ETH;
  - *token* – an amount of tokens;
- *UserBet* defines 2 fields:
  - *eth* – an amount of ETH;
  - *ref* – an address of ref;

Fields

Auction contract has 14 fields:

- *bytes32 public constant MANAGER_ROLE* – an indicator of Manager role;
- *bytes32 public constant CALLER_ROLE* – an indicator of Caller role;
- *mapping(uint256 => AuctionReserves) public reservesOf* – a list of *AuctionReserves*;
- *mapping(address => uint256[]) public auctionsOf* – a list of user`s auctions;
- *mapping(uint256 => mapping(address => UserBet)) public auctionBetOf* – a list of bets of users by auction id;
- *mapping(uint256 => mapping(address => bool)) public existAuctionsOf* – a list of checks for the user's bid on the auction;
- *uint256 public start* – start time of the auction;
- *uint256 public stepTimestamp* – auctions duration;
- *uint256 public uniswapPercent* – an amount of UniSwap percent;
- *address public mainToken* – an address of main token;
- *address public staking* – an address of staking;
- *address payable public uniswap* – an address of UniSwap;
- *address payable public recipient* – an address of recepient;
- *bool public init_* – the contract initialization indicator;

Modifiers

Auction contract has 2 modifiers:

- *onlyCaller* – checks if caller has Caller role;
- *onlyManager* – checks if caller has Manager role;

Functions

Auction contract has 15 functions:

- ***constructor***

  Description

  Used to set *init_* field to *false*.

  Visibility

  public

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  None

- ***init***

  Description

  Initialized contract. Sets roles and initial fields values.

  Visibility

  external

  Input parameters

  - *uint256 _stepTimestamp* – auctions duration;

- *address _manager* – an address of Manager;
- *address _mainToken* – an address of main token;
- *address _staking* – an address of staking;
- *address payable _uniswap* – an address of UniSwap;
- *address payable _recipient* – an address of recepient;
- *address _nativeSwap* – an address of native swap;
- *address _foreignSwap* – an address of foreign swap;

Constraints

- *init* function has not been called yet.

Events emit

None

Output

None

- ***auctionsOf_***

Description

Returns auctions ids of the user.

Visibility

public view

Input parameters

- *address account* – an address of account;

Constraints

None

Events emit

None

Output

Returns auctions ids of the user.

- ***setUniswapPercent***

Description

Sets UniSwap percent.

Visibility

external

Input parameters

- *uint256 percent* – a percent value;

Constraints

- Only Manager can call it.

Events emit

None

Output

None

- ***bet***

Description

Used to make a bet.

Visibility

external payable

Input parameters

- *uint256 deadline* – the deadline;
- *address ref* – an address of ref;

Constraints

- A ref cannot be caller.

Events emit

None

Output

None

- ***withdraw***

  Description

  Used to withdraw tokens from auction.

  Visibility

  external

  Input parameters

  - *uint256 auctionId* – an auction ID;

  Constraints

  - The auction must end.
  - The caller's balance must be greater than 0.

  Events emit

  None

  Output

  None

- ***callIncomeDailyTokensTrigger***

  Description

  Sets reserve amount of the next auction.

  Visibility

  external

  Input parameters

  - *uint256 amount* – an amount of reserve;

  Constraints

  - Only Caller can call it.

  Events emit

None

Output

None

- ***callIncomeWeeklyTokensTrigger***

  Description

  Sets reserve amount of weekly auction.

  Visibility

  external

  Input parameters

  - *uint256 amount* – an amount of reserve;

  Constraints

  - Only Caller can call it.

  Events emit

  None

  Output

  None

- ***calculateNearestWeeklyAuction***

  Description

  Calculates nearest weekly auction.

  Visibility

  public view

  Input parameters

  None

  Constraints

None

Events emit

None

Output

Returns nearest weekly auction.

- ***calculateStepsFromStart***

  Description

  Used to calculate a current auction.

  Visibility

  public view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns a current auction.

- ***_calculatePayoutWithUniswap***

  Description

  Calculates payout with UniSwap.

  Visibility

  internal view

  Input parameters

- *uint256 amount* – user`s ETH balance;
- *uint256 payout* – a payout amount;

Constraints

None

Events emit

None

Output

Returns calculated payout.

- ***_calculatePayout***

Description

Calculates payout.

Visibility

internal view

Input parameters

- *uint256 auctionId* – an auction ID;
- *uint256 amount* – user`s ETH balance;

Constraints

None

Events emit

None

Output

Returns calculated payout.

- ***_calculateRecipientAndUniswapAmountsToSend***

Description

Calculates recipient and UniSwap amounts to send.

Visibility

private

Input parameters

None

Constraints

None

Events emit

None

Output

Returns recipient and UniSwap amounts to send.

- ***_calculateRefAndUserAmountsToMint***

  Description

  Calculates ref and user amounts to mint.

  Visibility

  private pure

  Input parameters

  - *uint256 amount* – a payout amount;

  Constraints

  None

  Events emit

  None

  Output

  Returns ref and user amounts to mint.

- ***_swapEth***

Description

Used to swap ETH to tokens via UniSwap.

Visibility

private

Input parameters

- *uint256 amount* – an ETH amount;
- *uint256 deadline* – the deadline;

Constraints

None

Events emit

None

Output

None

BPD.sol

Description

BPD contract used to work with tokens pool.

Imports

BPD contract has 4 imports:

- *IERC20* – from OpenZeppelin;
- *SafeMath* – from OpenZeppelin;
- *AccessControl* – from OpenZeppelin;
- *IBPD* – from project files;

Inheritance

BPD contract inherits *IBPD*, *AccessControl*.

Usings

BPD contract use:

- *SafeMath* for *uint256*;

Structs

BPD contract does not have data structs.

Fields

BPD contract has 8 fields:

- *bytes32 public constant SETTER_ROLE* – an indicator of Setter role;
- *bytes32 public constant SWAP_ROLE* – an indicator of Swap role;
- *bytes32 public constant SUBBALANCE_ROLE* – an indicator of Subbalance role;
- *uint256[5] public poolYearAmounts* – a list of amounts;
- *bool[5] public poolTransferred* – an indicator of pool amount transfer;
- *uint256[5] public poolYearPercentages* – a list of pool percentages;
- *address public mainToken* – an address of main token;
- *uint256 public constant PERCENT_DENOMINATOR* – the percent denominator;

Modifiers

BPD contract has 1 modifiers:

- *onlySetter* – checks if caller has Setter role;

Functions

BPD contract has 6 functions:

- ***constructor***

  Description

  Sets Setter role.

  Visibility

  public

  Input parameters

- *address _setter* – an address of Setter;

Constraints

None

Events emit

None

Output

None

- ***init***

Description

Initialized contract. Sets roles and main token.

Visibility

public

Input parameters

- *address _mainToken* – an address of main token;
- *address _foreignSwap* – an address of Swap;
- *address _subBalancePool* – an address of Subbalance;

Constraints

- Only Setter can call it.

Events emit

None

Output

None

- ***getPoolYearAmounts***

Description

Used to get pool amounts.

Visibility

external view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns pool amounts.

- ***getClosestPoolAmount***

  Description

  Used to get first not transfered pool amount.

  Visibility

  public view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns first not transfered pool amount.

- ***callIncomeTokensTrigger***

Description

Used to add amount to pool.

Visibility

external

Input parameters

- *uint256 incomeAmountToken* – an income amount;

Constraints

- Only Swap can call it.

Events emit

None

Output

None

- ***transferYearlyPool***

Description

Used to transfer part of a pool amount to Subbalance.

Visibility

external

Input parameters

- *uint256 poolNumber* – a pool number;

Constraints

- Only Subbalance can call it.

Events emit

None

Output

Returns transfered amount.

ForeignSwap.sol

Description

ForeignSwap contract used to swap tokens.

Imports

ForeignSwap contract has 9 imports:

- *IERC20* – from OpenZeppelin;
- *AccessControl* – from OpenZeppelin;
- *SafeMath* – from OpenZeppelin;
- *ECDSA* – from OpenZeppelin;
- *IToken* – from project files;
- *IAuction* – from project files;
- *IStaking* – from project files;
- *IBPD* – from project files;
- *IForeignSwap* – from project files;

Inheritance

ForeignSwap contract inherits *IForeignSwap*, *AccessControl*.

Usings

ForeignSwap contract use:

- *SafeMath* for *uint256*;

Structs

ForeignSwap contract does not have data structs.

Fields

ForeignSwap contract has 15 fields:

- *bytes32 public constant SETTER_ROLE* – an indicator of Setter role;
- *uint256 public start* – start time of the auction;
- *uint256 public stepTimestamp* – auctions duration;
- *uint256 public maxClaimAmount* – a maximum amount to claim;
- *uint256 public constant PERIOD* – the period constant;
- *address public mainToken* – an address of main token;

- *address public staking* – an address of staking;
- *address public auction* – an address of auction;
- *address public bigPayDayPool* – an address of BPD;
- *address public signerAddress* – an address of signer;
- *mapping(address => uint256) public claimedBalanceOf* – a list of claimed balances;
- *uint256 internal claimedAmount* – an amount of claimed tokens;
- *uint256 internal totalSnapshotAmount* – a total snapshot amount;
- *uint256 internal claimedAddresses* – a claimed addresses count;
- *uint256 internal totalSnapshotAddresses* – a total snapshot addresses;

Modifiers

ForeignSwap contract has 1 modifier:

- *onlyCaller* – checks if caller has Setter role;

Functions

ForeignSwap contract has 12 functions:

- ***constructor***

  Description

  Sets Setter role.

  Visibility

  public

  Input parameters

  - *address _setter* – an address of Setter;

  Constraints

  None

  Events emit

  None

Output

None

- ***init***

  Description

  Initialized contract. Sets roles and initial fields values.

  Visibility

  external

  Input parameters

  - *address _signer* – an address of signer;
  - *uint256 _stepTimestamp* – auctions duration;
  - *uint256 _maxClaimAmount* – a maximum amount to claim;
  - *address _mainToken* – an address of main token;
  - *address _auction* – an address of auction;
  - *address _staking* – an address of staking;
  - *address _bigPayDayPool* – an address of BPD;
  - *uint256 _totalSnapshotAmount* – a total snapshot amount;
  - *uint256 _totalSnapshotAddresses* – a total snapshot addresses;

  Constraints

  - Only Setter can call it.

  Events emit

  None

  Output

  None

- ***getCurrentClaimedAmount***

  Description

  Used to get current claimed amount.

  Visibility

external view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns claimed amount.

- ***getTotalSnapshotAmount***

  Description

  Used to get total snapshot amount.

  Visibility

  external view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns total snapshot amount.

- ***getCurrentClaimedAddresses***

  Description

Used to get current claimed addresses count.

Visibility

external view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns current claimed addresses count.

- ***getTotalSnapshotAddresses***

  Description

  Used to get total snapshot addresses.

  Visibility

  external view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns total snapshot addresses.

- *getMessageHash*

  Description

  Calculates message hash.

  Visibility

  public pure

  Input parameters

  - *uint256 amount* – an amount of tokens;
  - *address account* – an address of account;

  Constraints

  None

  Events emit

  None

  Output

  Returns message hash.

- *check*

  Description

  Checks that the message was sent by a Signer.

  Visibility

  public view

  Input parameters

  - *uint256 amount* – an amount of tokens;
  - *bytes memory signature* – a signature;

  Constraints

  None

  Events emit

None

Output

Returns true if the message was sent by a Signer.

- ***getUserClaimableAmountFor***

  Description

  Calculates the penalized amount and the actual number of tokens a user can receive by a claim.

  Visibility

  public view

  Input parameters

  - *uint256 amount* – an amount of tokens;

  Constraints

  None

  Events emit

  None

  Output

  Returns the actual amount and the penalized amount.

- ***claimFromForeign***

  Description

  Used to claim from foreign.

  Visibility

  public

  Input parameters

  - *uint256 amount* – an amount of tokens;
  - *bytes memory signature* – a signature;

Constraints

- A claimed amount must be greater than 0.
- Signature must be correct.
- User claimed balance must be greater than 0.

Events emit

None

Output

Returns true if claimed.

- ***calculateStepsFromStart***

  Description

  Used to calculate a current auction.

  Visibility

  public view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns a current auction.

- ***getClaimableAmount***

  Description

  Calculates the claimable amount.

  Visibility

internal view

Input parameters

- *uint256 amount* – an amount of tokens;

Constraints

None

Events emit

None

Output

Returns the amount to out and deltas.

Migrations.sol

Description

Migrations contract used to track migrations.

Imports

Migrations contract does not have imports.

Inheritance

Migrations contract inherits nothing.

Usings

Migrations contract does not have usings.

Structs

Migrations contract does not have data structs.

Fields

Migrations contract has 2 fields:

- *address public owner* – an address of Owner;
- *uint public last_completed_migration* – the last completed migration;

Modifiers

Migrations contract has 1 modifier:

- *restricted* – checks if caller is Owner;

Functions

Migrations contract has 2 functions:

- ***constructor***

  Description

  Sets Owner.

  Visibility

  public

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  None

- ***setCompleted***

  Description

  Sets *last_completed_migration* field.

  Visibility

  public

  Input parameters

- *uint completed* – the last completed migration;

Constraints

- Only Owner can call it.

Events emit

None

Output

None

NativeSwap.sol

Description

NativeSwap contract used to swap tokens.

Imports

NativeSwap contract has 4 imports:

- *IERC20* – from OpenZeppelin;
- *SafeMath* – from OpenZeppelin;
- *IToken* – from project files;
- *IAuction* – from project files;

Inheritance

NativeSwap contract inherits nothing.

Usings

NativeSwap contract use:

- *SafeMath* for *uint256*;

Structs

NativeSwap contract does not have data structs.

Fields

NativeSwap contract has 7 fields:

- *uint256 public start* – start time of the auction;

- *uint256 public stepTimestamp* – auctions duration;
- *IERC20 public swapToken* – a swap token;
- *IToken public mainToken* – a main token;
- *IAuction public auction* – an auction;
- *bool public init_* – the contract initialization indicator;
- *mapping(address => uint256) public swapTokenBalanceOf* – a list of swap token balances;

Modifiers

NativeSwap contract does not have modifiers.

Functions

NativeSwap contract has 6 functions:

- ***constructor***

  Description

  Used to set *init_* field to *false*.

  Visibility

  public

  Input parameters

  - *address _setter* – an address of Setter;

  Constraints

  None

  Events emit

  None

  Output

  None

- ***init***

  Description

  Initialized contract. Sets roles and initial fields values.

Visibility

external

Input parameters

- *uint256 _stepTimestamp* – auctions duration;
- *address _swapToken* – a swap token address;
- *address _mainToken* – a main token address;
- *address _auction* – an auction address;

Constraints

- *init* function has not been called yet.

Events emit

None

Output

None

- ***deposit***

Description

Used to deposit.

Visibility

external

Input parameters

- *uint256 _amount* – an amount of tokens;

Constraints

- Tokens transferred successfully.

Events emit

None

Output

None

- ***withdraw***

  Description

  Used to withdraw.

  Visibility

  external

  Input parameters

  - *uint256 _amount* – an amount of tokens;

  Constraints

  - The user balance must be greater or equal the amount.

  Events emit

  None

  Output

  None

- ***swapNativeToken***

  Description

  Used to swap all user`s tokens.

  Visibility

  external

  Input parameters

  None

  Constraints

  - The user balance must be greater than 0.

  Events emit

  None

Output

None

- ***calculateDeltaPenalty***

  Description

  Calculates the delta penalty amount.

  Visibility

  public view

  Input parameters

  - *uint256 amount* – an amount of tokens;

  Constraints

  None

  Events emit

  None

  Output

  Returns the delta penalty amount.

- ***calculateStepsFromStart***

  Description

  Used to calculate a current auction.

  Visibility

  public view

  Input parameters

  None

  Constraints

  None

Events emit

None

Output

Returns a current auction.

Staking.sol

Description

Staking contract used to allow users to stake and to unstake tokens.

Imports

Staking contract has 7 imports:

- *IERC20* – from OpenZeppelin;
- *AccessControl* – from OpenZeppelin;
- *SafeMath* – from OpenZeppelin;
- *IToken* – from project files;
- *IAuction* – from project files;
- *IStaking* – from project files;
- *ISubBalances* – from project files;

Inheritance

Staking contract inherits *IStaking*, *AccessControl*.

Usings

Staking contract use:

- *SafeMath* for *uint256*;

Structs

Staking contract has 2 data structs.

- *Payout* that has 2 fields:
  - *uint256 payout* – a payout;
  - *uint256 sharesTotalSupply* – a total supply to share;
- *Session* that has 5 fields:
  - *uint256 amount* – an amount;
  - *uint256 start* – a start timestamp;

- *uint256 end* – an end timestamp;
- *uint256 shares* – shares amount;
- *uint256 nextPayout* – the next payout index;

Fields

Staking contract has 16 fields:

- *uint256 private _sessionsIds* – sessions length;
- *bytes32 public constant EXTERNAL_STAKER_ROLE* – an indicator of Externa staker role;
- *address public mainToken* – an address of main token;
- *address public auction* – an address of auction;
- *address public subBalances* – an address of subbalances;
- *uint256 public shareRate* – a share rate;
- *uint256 public sharesTotalSupply* – a total supply to share;
- *uint256 public nextPayoutCall* – the next payout timestamp;
- *uint256 public stepTimestamp* – auctions duration;
- *uint256 public startContract* – the timestamp when the contract was deployed;
- *uint256 public globalPayout* – a global payout amount;
- *uint256 public globalPayin* – a global payin amount;
- *bool public init_* – the contract initialization indicator;
- *mapping(address => mapping(uint256 => Session)) public sessionDataOf* – a list of sessions;
- *mapping(address => uint256[]) public sessionsOf* – a list of user`s sessions IDs;
- *Payout[] public payouts* – a list of payouts;

Modifiers

Staking contract has 1 modifier:

- *onlyExternalStaker* – checks if caller is External staker;

Functions

Staking contract has 16 functions:

- ***constructor***

  Description

  Used to set *init_* field to *false*.

  Visibility

public

Input parameters

None

Constraints

None

Events emit

None

Output

None

- *init*

  Description

  Initialized contract. Sets roles and initial fields values.

  Visibility

  external

  Input parameters

  - *address _mainToken* – an address of main token;
  - *address _auction* – an address of auction;
  - *address _subBalances* – an address of subbalances;
  - *address _foreignSwap* – an address of foreign swap;
  - *uint256 _stepTimestamp* – auctions duration;

  Constraints

  - *init* function has not been called yet.

  Events emit

  None

  Output

  None

- ***sessionsOf_***

  Description

  Used to get sessions IDs for the account;

  Visibility

  external view

  Input parameters

  - *address account* – an account address;

  Constraints

  None

  Events emit

  None

  Output

  Returns sessions IDs.

- ***stake***

  Description

  Used to stake some amount for some period of time.

  Visibility

  external

  Input parameters

  - *uint256 amount* – an amount of tokens;
  - *uint256 stakingDays* – a staking period;

  Constraints

  - The staking period must be greater than 0.

  Events emit

  None

Output

None

- ***externalStake***

  Description

  Used to make external stake.

  Visibility

  external

  Input parameters

  - *uint256 amount* – an amount of tokens;
  - *uint256 stakingDays* – a staking period;
  - *address staker* a staker address;

  Constraints

  - Only External staker can call it.
  - The staking period must be greater than 0.

  Events emit

  None

  Output

  None

- ***_initPayout***

  Description

  Used to mint tokens for a global payout;

  Visibility

  internal

  Input parameters

  - *address to* – an address to which tokens will be minted;
  - *uint256 amount* – an amount of tokens;

Constraints

None

Events emit

None

Output

None

- ***calculateStakingInterest***

  Description

  Calculates staking interest.

  Visibility

  public view

  Input parameters

  - *uint256 sessionId* – an ID of the session;
  - *address account* – an address of the account;
  - *uint256 shares* – shares amount;

  Constraints

  None

  Events emit

  None

  Output

  Returns staking interest.

- ***_updateShareRate***

  Description

  Used to update share rate.

  Visibility

internal

Input parameters

- *address account* – an address of the account;
- *uint256 shares* – shares amount;
- *uint256 stakingInterest* – a staking interest;
- *uint256 sessionId* – an ID of the session;

Constraints

None

Events emit

None

Output

None

- ***unstake***

Description

Used to unstake.

Visibility

external

Input parameters

- *uint256 sessionId* – an ID of the session;

Constraints

- The user`s shares balance must be greater than 0.
- The session must have payouts.

Events emit

None

Output

None

- *getAmountOutAndPenalty*

  Description

  Used to get the amount that can be out and the penalty amount.

  Visibility

  public view

  Input parameters

  - *uint256 sessionId* – an ID of the session;
  - *uint256 stakingInterest* – a staking interest;

  Constraints

  None

  Events emit

  None

  Output

  Returns the amount that can be out and the penalty amount.

- *makePayout*

  Description

  Used to make payout.

  Visibility

  external

  Input parameters

  None

  Constraints

  - It is required that the timestamp of the next payout call has been passed.

  Events emit

None

Output

None

- ***readPayout***

  Description

  Used to get the current supply of tokens including inflation.

  Visibility

  external view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns the current supply of tokens including inflation.

- ***_getPayout***

  Description

  Used to make payout and to get the current supply of tokens including inflation.

  Visibility

  internal

  Input parameters

  None

Constraints

None

Events emit

None

Output

Returns the current supply of tokens including inflation.

- ***_getStakersSharesAmount***

  Description

  Used to calculate the amount of stakers shares.

  Visibility

  internal view

  Input parameters

  - *uint256 amount* – an amount of tokens;
  - *uint256 start* – a start timestamp;
  - *uint256 end* – an end timestamp;

  Constraints

  None

  Events emit

  None

  Output

  Returns the amount of stakers shares.

- ***_getShareRate***

  Description

  Used to calculate share rate.

  Visibility

internal view

Input parameters

- *uint256 amount* – an amount of tokens;
- *uint256 shares* – an amount of shares;
- *uint256 start* – a start timestamp;
- *uint256 end* – an end timestamp;
- *uint256 stakingInterest* – a staking interest;

Constraints

None

Events emit

None

Output

Returns share rate.

- ***getNow0x***

  Description

  Helper function used to get now timestamp.

  Visibility

  external view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns now timestamp.

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

SubBalances.sol

Description

SubBalances contract used to manage subbalances.

Imports

SubBalances contract has 8 imports:

- *IERC20* – from OpenZeppelin;
- *SafeMath* – from OpenZeppelin;
- *AccessControl* – from OpenZeppelin;
- *IToken* – from project files;
- *IAuction* – from project files;
- *IForeignSwap* – from project files;
- *IBPD* – from project files;
- *ISubBalances* – from project files;

Inheritance

SubBalances contract inherits *ISubBalances*, *AccessControl*.

Usings

SubBalances contract use:

- *SafeMath* for *uint256*;

Structs

SubBalances contract has 2 data structs.

- *StakeSession* that has 7 fields:
  - *address staker* – an address of staker;
  - *uint256 shares* – shares amount;
  - *uint256 start* – a start timestamp;
  - *uint256 end* – an end timestamp;
  - *uint256 finishTime* – a finish timestamp;
  - *bool[5] payDayEligible* – a list of stake days eligibility;
  - *bool withdrawn* – an indicator of withdrawing;
- *SubBalance* that has 5 fields:
  - *uint256 totalShares* – a total shares;
  - *uint256 totalWithdrawAmount* – a total withdrawing amount;
  - *uint256 payDayTime* – a pay day tamstamp;

- *uint256 requiredStakePeriod* – the required stake period;
- *bool minted* – an indicator of minting;

Fields

SubBalances contract has 14 fields:

- *bytes32 public constant SETTER_ROLE* – an indicator of Setter role;
- *bytes32 public constant STAKING_ROLE* – an indicator of Staker role;
- *SubBalance[5] public subBalanceList* – a list of subbulances;
- *address public mainToken* – an address of main token;
- *address public foreignSwap* – an address of foreign swap;
- *address public bigPayDayPool* – an address of BPD pool;
- *address public auction* – an address of auction;
- *uint256 public startTimestamp* – a start timestamp;
- *uint256 public stepTimestamp* – auctions duration;
- *uint256 public basePeriod* – a base period;
- *uint256[5] public PERIODS* – a list of periods;
- *uint256 public currentSharesTotalSupply* – a total supply of current shares;
- *mapping (address => uint256[]) userStakings* – a list of users stakes;
- *mapping (uint256 => StakeSession) stakeSessions* – a list of stake sessions;

Modifiers

SubBalances contract has 1 modifier:

- *onlySetter* – checks if caller is Setter;

Functions

SubBalances contract has 15 functions:

- ***constructor***

  Description

  Sets Setter role.

  Visibility

public

Input parameters

- *address _setter* – an address of Setter;

Constraints

None

Events emit

None

Output

None

- ***init***

Description

Initialized contract. Sets roles and initial fields values.

Visibility

public

Input parameters

- *address _mainToken* – an address of main token;
- *address _foreignSwap* – an address of foreign swap;
- *address _bigPayDayPool* – an address of BPD;
- *address _auction* – an address of auction;
- *address _staking* – an address of staking;
- *uint256 _stepTimestamp* – auctions duration;;
- *uint256 _basePeriod* – a base period;

Constraints

- Only Setter can call it.

Events emit

None

Output

None

- ***getStartTimes***

  Description

  Used to get a list of pay days.

  Visibility

  public view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns a list of pay days.

- ***getPoolsMinted***

  Description

  Used to get a list of minted pool indicators.

  Visibility

  public view

  Input parameters

  None

  Constraints

  None

  Events emit

None

Output

Returns a list of minted pool indicators.

- ***getPoolsMintedAmounts***

  Description

  Used to get a list of minted pool amounts.

  Visibility

  public view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns a list of minted pool amounts.

- ***getClosestYearShares***

  Description

  Used to get closest not minted pool total shares amount.

  Visibility

  public view

  Input parameters

  None

  Constraints

None

Events emit

None

Output

Returns closest not minted pool total shares amount.

- ***getSessionStats***

  Description

  Used to get the session info by the session ID.

  Visibility

  public view

  Input parameters

  - *uint256 sessionId* – the session ID;

  Constraints

  None

  Events emit

  None

  Output

  Returns session info.

- ***getSessionEligibility***

  Description

  Used to get a list of session's eligible pay days.

  Visibility

  public view

  Input parameters

- *uint256 sessionId* – the session ID;

Constraints

None

Events emit

None

Output

Returns a list of session's eligible pay days.

- ***calculateSessionPayout***

Description

Calculates a payout amount and a penalty amount of the session.

Visibility

public view

Input parameters

- *uint256 sessionId* – the session ID;

Constraints

None

Events emit

None

Output

Returns a payout amount and a penalty amount of the session.

- ***withdrawPayout***

Description

Used to withdraw payout of session.

Visibility

public

Input parameters

- *uint256 sessionId* – the session ID;

Constraints

- The session must not be finished.
- The session must not be withdrawn.
- The caller must be the session staker.

Events emit

None

Output

None

- ***callIncomeStakerTrigger***

Description

Used to add shares to total shares supply. Creates a new session if user makes stake for period that greater than base period.

Visibility

external

Input parameters

- *address staker* – an address of staker;
- *uint256 sessionId* – the session ID;
- *uint256 start* – a start timestamp;
- *uint256 end* – an end timestamp;
- *uint256 shares* – a shares amount;

Constraints

- Only Staking role can call it.
- Stake end must be after stake start.

Events emit

None

Output

None

- ***callOutcomeStakerTrigger***

  Description

  Used to subtract total shares supply and to finish session.

  Visibility

  external

  Input parameters

  - *address staker* – an address of staker;
  - *uint256 sessionId* – the session ID;
  - *uint256 start* – a start timestamp;
  - *uint256 end* – an end timestamp;
  - *uint256 shares* – a shares amount;

  Constraints

  - Only Staking role can call it.
  - Stake end must be after stake start.

  Events emit

  None

  Output

  None

- ***generatePool***

  Description

  Generates tokens pool.

  Visibility

  external

  Input parameters

None

Constraints

None

Events emit

None

Output

Returns true if generated.

- ***getPoolFromBPD***

  Description

  Used to get a pool amount from BPD.

  Visibility

  internal

  Input parameters

  - *uint256 poolNumber* – a pool number;

  Constraints

  None

  Events emit

  None

  Output

  Returns a pool amount.

- ***_bpdAmountFromRaw***

  Description

  Calculates a total amount of tokens and an amount that need to be minted for subbalance.

  Visibility

internal view

Input parameters

- *uint256 yearTokenAmount* – an amount of tokens;

Constraints

None

Events emit

None

Output

Returns a total amount of tokens and an amount that need to be minted

Token.sol

Description

Token an ERC20 token contract.

Imports

Token contract has 4 imports:

- *IERC20* – from OpenZeppelin;
- *AccessControl* – from OpenZeppelin;
- *SafeMath* – from OpenZeppelin;
- *IToken* – from project files;

Inheritance

Token contract inherits *IToken*, *ERC20*, *AccessControl*.

Usings

Token contract use:

- *SafeMath* for *uint256*;

Structs

Token contract does not have data structs.

Fields

Token contract has 5 fields:

- *bytes32 private constant MINTER_ROLE* – an indicator of Minter role;
- *bytes32 public constant SETTER_ROLE* – an indicator of Setter role;
- *IERC20 private swapToken* – a swap token;
- *bool private swapIsOver* – an indicator of swap is over or not;
- *uint256 private swapTokenBalance* – a swap token balance;

Modifiers

Token contract has 2 modifiers:

- *onlyMinter* – checks if caller is Minter;
- *onlySetter* – checks if caller is Setter;

Functions

Token contract has 12 functions:

- ***constructor***

  Description

  Sets setter role and swap token.

  Visibility

  public

  Input parameters

  - *string memory _name* – a token name;
  - *string memory _symbol* – a token symbol;
  - *address _swapToken* – an address of swap token;
  - *address _setter* – an address of Setter;

  Constraints

  None

  Events emit

None

Output

None

- ***init***

  Description

  Initialized contract. Sets Minters.

  Visibility

  external

  Input parameters

  - *address[] calldata instances* – a list of Minters.

  Constraints

  - Only Setter role can call it.
  - Must be 5 Minters.

  Events emit

  None

  Output

  None

- ***getMinterRole***

  Description

  Used to get Minter role indicator.

  Visibility

  external pure

  Input parameters

  None

  Constraints

None

Events emit

None

Output

Returns Minter role indicator.

- ***getSetterRole***

  Description

  Used to get Setter role indicator.

  Visibility

  external pure

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns Setter role indicator.

- ***getSwapTOken***

  Description

  Used to get swap token.

  Visibility

  external view

  Input parameters

None

Constraints

None

Events emit

None

Output

Returns swap token.

- ***getSwapTokenBalance***

  Description

  Used to get swap token balance.

  Visibility

  external view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns swap token balance.

- ***initDeposit***

  Description

  Used to add tokens to swap token balance when initialization not finished.

  Visibility

external

Input parameters

- *uint256 _amount* – an amount of tokens;

Constraints

- Only Setter role can call it.
- Tokens must be transfered.

Events emit

None

Output

None

- ***initWithdraw***

  Description

  Used to subtract tokens from swap token balance when initialization not finished.

  Visibility

  external

  Input parameters

  - *uint256 _amount* – an amount of tokens;

  Constraints

  - Only Setter role can call it.
  - Swap token balance must be greater than subtracted amount.

  Events emit

  None

  Output

  None

- ***initSwap***

  Description

  Used to mint all swap token balance to Setter.

  Visibility

  external

  Input parameters

  None

  Constraints

  - Only Setter role can call it.
  - Swap is not over.
  - Swap token balance must be greater than 0.

  Events emit

  None

  Output

  None

- ***mint***

  Description

  Used to mint tokens.

  Visibility

  external

  Input parameters

  - *address to* – an address to;
  - *uint256 amount* – an amount of tokens;

  Constraints

  - Only Minter role can call it.

  Events emit

None

Output

None

- ***burn***

  Description

  Used to burn tokens.

  Visibility

  external

  Input parameters

  - *address from* – an address from;
  - *uint256 amount* – an amount of tokens;

  Constraints

  - Only Minter role can call it.

  Events emit

  None

  Output

  None

- ***getNow***

  Description

  Helper function used to get now timestamp.

  Visibility

  external view

  Input parameters

  None

  Constraints

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

None

Events emit

None

Output

Returns now timestamp.

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. It's recommended to move a value of the stepTimestamp to constant because it's implied that its value should always be equal to 1 day.
2. *_getShareRate* and *_getStakersSharesAmount* function of the *Staking* contract has "magic numbers" that should be moved to a named constants.
3. The *_getShareRate* function of the *Staking* contract uses hardcoded value of the token decimals. This value should be fetched from the token contract.
4. *calculateSessionPayout* function of *SubBalances* contract has "magic numbers" that should be moved to a named constants.

Low

1. *stake* and *externalStake* functions of the *Staking* contract have common code that can be moved to separate function to decrease code duplication.
2. *callOutcomeStakerTrigger* function of *SubBalances* contract has unused parameter *staker*.

Lowest / Code style / Best Practice

1. *onlyManager* modifier of the *Auction* contract has wrong message.
2. *swapNativeToken* function of the *NativeSwap* contract has require that checks balance. It can be placed in the beginning of the function.
3. *getSwapTokenBalance* function of *Token* contract has redundant type in parameters.
4. A lot of other code-style issues can be found by running any static code analyzer specified in the Executive Summary section.

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

Conclusion

Smart contracts within the scope was 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.

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

Violations in following categories were found and addressed to Customer:

Category	Check Item	Comments
Code review	■ Style guide violation	It is recommended to fix the found severity issues in order to make the code more readable and to avoid troubles when developing these contracts in the future.

Security engineers found 4 medium, 2 low and 4 lowest severity issues during audit. It is highly recommended to fix them all.

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