HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: DeXe Network
Date: September 24th, 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 DeXe Network
Type	Token, token-sale, staking.
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Archive Name	dexe-8ca55a54680edb118108384318f1867caf65565b.zip
Checksum	c3b6df51f4b88bc7f518c425786a518548d89052153a65310058f7f74057ff18
Timeline	17 SEP 2020 - 20 SEP 2020
Changelog	20 SEP 2020 - Initial Audit 24 SEP 2020 - Secondary review

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

Table of contents

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

Introduction

Hacken OÜ (Consultant) was contracted by DeXe Network (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 17, 2020 - September 20, 2020.

Scope

The scope of the project is smart contracts in the repository:
`Audit Archive File - dexe-8ca55a54680edb118108384318f1867caf65565b.zip`
`SHA256 checksum - c3b6df51f4b88bc7f518c425786a518548d89052153a65310058f7f74057ff18`
We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

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

Executive Summary

According to the assessment, the Customer's smart contracts are following code style guides and best practices. All functions are covered with tested, and the code works as described in the whitepaper.

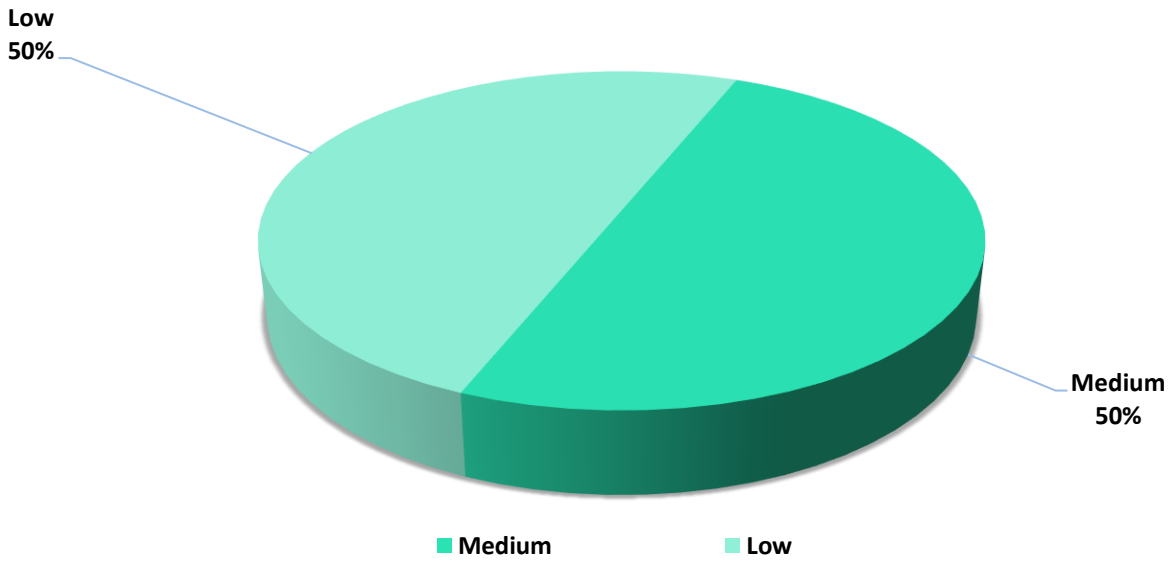
Insecure	Poor secured	Secured	Well-secured

You are here

Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed and important vulnerabilities are presented in the Audit overview section. 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 1 low severity issues during the initial audit. All the issues have been fixed before secondary audit.

Graph 1. The distribution of vulnerabilities during the initial audit.



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

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets 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

PriceFeed.sol

Description

PriceFeed contract serves as a wrapper to receive token prices from the *Uniswap*.

Imports

PriceFeed contract has following imports:

- *IUniswapV2Pair* - from the Uniswap.
- *FixedPoint* - from the Uniswap
- *UniswapV2OracleLibrary* - from the Uniswap.
- *IPriceFeed* - from the project files.

Inheritance

PriceFeed contract implements *IPriceFeed.*

Structs

PriceFeed contract has no data structures.

Usages

PriceFeed contract uses:

- *FixedPoint* for *;

Fields

PriceFeed contract has 5 fields and constants:

- address constant USDCAddress = 0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48; - the USDC contract address from the mainnet.
- IUniswapV2Pair public immutable pair; - the Uniswap pair address.
- uint public immutable multiplier; - multiplier used to receive indivisible units as a result.
- uint public priceCumulativeLast; - last price.
- uint32 public blockTimestampLast; - timestamp of a last update.

Modifiers

PriceFeed contract has no custom modifiers.

Functions

PriceFeed has 4 functions:

- ***constructor***
  Description
  Initializes contract. Sets a Uniswap pair to provide exchange rated for.
  Visibility
  public
  Input parameters
  - *IUniswapV2Pair _pair* - uniswap pair.

  Constraints
  - *token0* of the pair should always be USDC.
  - The pair should have liquidity on the Unsiwap.

  Events emit
  None
  Output
  None
- ***update***
  Description
  Updates the pair cumulative price and timestamp.
  Visibility
  public
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  - _priceCumulative - cumulative price.
  - _blockTimestamp - timestamp.
- ***consult***
  Description
  Provides exchange rate.
  Visibility
  external view
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  - exchange rate.

- *updateAndConsult*
  Description
  Updates the pair cumulative price and provides exchange rate as a result.
  Visibility
  external view
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
    - exchange rate.

Dexe.sol

Description

Dexe is a contract that implements ERC-20 interface and allows to run token-sale and to provide staking functionality.

Imports

Dexe contract has following imports:

- *Ownable* - from the OpenZeppelin.
- *SafeMath* - from the OpenZeppelin.
- *ERC20Burnable* - from the OpenZeppelin.
- *IPriceFeed* - from the project files.
- *IDexe* - from the project files.

Inheritance

Dexe contract implements IDexe and is Ownable and ERC20Burnable.

Structs

Dexe contract has 6 structures:

- *LockConfig*
- *Lock*
- *HolderRound*
- *UserInfo*
- *BalanceInfo*
- *Round*

Usages

Dexe contract uses:

- *ExtraMath for *;*
- *SafeMath for *;*

Fields

Dexe contract has 28 fields and constants:

- *uint private constant DEXE = 10**18;* - DEXE token decimals.
- *uint private constant USDC = 10**6;* - USDC token decimals.
- *uint private constant USDT = 10**6;* - USDT token decimals.
- *uint private constant DISTRIBUTOR_LIMIT = 10**10 * USDC;* - a distributor first round contribution limit.
- *uint private constant MONTH = 30 days;* - days in a month.
- *uint public constant ROUND_SIZE_BASE = 190_476;* - sale round limit in divisible units.
- *uint public constant ROUND_SIZE = ROUND_SIZE_BASE * DEXE;* - sale round limit in indivisible units.
- *uint public constant FIRST_ROUND_SIZE_BASE = 1_000_000;* - divider for the first round price calculation.
- *IERC20 public usdcToken;* - usdc token contract.
- *IERC20 public usdtToken;* - usdt token contract.
- *IPriceFeed public usdtPriceFeed;* - USDT to USDC price provider;
- *IPriceFeed public dexePriceFeed;* - DEXE to USDC price provider;
- *IPriceFeed public ethPriceFeed;* - ETH to USDC price provider;
- *address payable public treasury;* - an address where all deposits are transferred.
- *IPriceFeed public priceFeed;* - never used.
- *uint public averagePrice;* - 2-10 rounds average price.
- *uint public override launchedAfter;* - stores how many seconds passed between sale end and product launch.
- *mapping(uint => mapping(address => HolderRound)) internal _holderRounds;* - rounds participants storage.
- *mapping(address => UserInfo) internal _usersInfo;* - stores user information.
- *mapping(address => BalanceInfo) internal _balanceInfo;* - balances information.
- *mapping(LockType => LockConfig) public lockConfigs;* - lock configs.

- *mapping(LockType => mapping(address => Lock)) public locks;* - map of locks.
- *mapping(address => mapping(ForceReleaseType => bool)) public forceReleased*; - released stakes.
- *uint constant ROUND_DURATION_SEC = 86400;* - token-sale round duration in seconds.
- *uint constant TOTAL_ROUNDS = 22;* - total token-sale rounds.
- *mapping(uint => Round) public rounds;* - round information. Total contributions and exchange rate (USDC to DEX).
- *uint public constant tokensaleStartDate = 1600603200;* - token-sale start day.
- *uint public override constant tokensaleEndDate = tokensaleStartDate + ROUND_DURATION_SEC * TOTAL_ROUNDS;* - token-sale end day.

Events

Dexe contract has 2 events:

- *event NoteDeposit(address sender, uint value, bytes data);*
- *event Note(address sender, bytes data);*

Enums

Dexe contract has 3 enums:

- *LockType*
- *ForceReleaseType*
- *HolderRoundStatus*

Modifiers

Dexe contract has 2 custom modifiers:

- *noteDeposit()*
- *note()*

Functions

Dexe has 50 functions:

- ***constructor***
  Description
  Initializes contract. Sets a LockConfig's and specify the treasury address.
  Visibility
  public
  Input parameters

- *address _distributor* - an address where all locked funds are accrued.

Constraints
None
Events emit
None
Output
None

- ***setUSDTTokenAddress, setUSDCTokenAddress, setUSDTFeed, setDEXEFeed, setETHFeed, setTreasury***

Description
Setter functions used to set values of the corresponding fields.
Visibility
external
Input parameters
- *address* of a corresponding contract.

Constraints
- Can only be called by the owner.

Events emit
- Emits *note* event.

Output
None

- ***addToWhitelist***

Description
Add an address to the whitelist. Only whitelisted addresses can participate in the token sale.
Visibility
external
Input parameters
- address _address - whitelisted address.
- uint _limit - max allowed contribution sum.

Constraints
- Can only be called by the owner.

Events emit
- Emits *note* event.

Output
None

- ***removeFromWhitelist***

Description
Remove an address from the whitelist.
Visibility
external
Input parameters
- address _address - an address that should be removed from the whitelist.

Constraints
- Can only be called by the owner.

Events emit
- Emits *note* event.

Output
None

- ***_updateWhitelist***

Description
Add or remove an address to the whitelist.
Visibility
internal
Input parameters
- address _address - an address that should be added to or removed from the whitelist.
- uint _limit - max allowed contribution sum.

Constraints
- Can only be called by the owner.

Events emit
None
Output
None

- ***getAllRounds***

Description
Get all rounds info.
Visibility
external view
Input parameters
None
Constraints
None
Events emit
None
Output
- Round[22] memory - all rounds info.

- ***getFullHolderInfo***

Description
Get holder info.
Visibility
external view
Input parameters
- address _holder - a holder address.

Constraints
None
Events emit
None
Output

- UserInfo memory _info - user info.
- HolderRound[22] memory _rounds - tokensale rounds where the holder participated at.
- Lock[6] memory _locks - locks info.
- bool _isWhitelisted - whitlist status.
- bool[4] memory _forceReleases - swap releases.
- uint _balance - DEXE balance.

- ***prepareDistributionPrecise***
  Description
  Prepares distributions with lower and upper price limits.
  Visibility
  external
  Input parameters
  - uint _round - token-sale round.
  - uint _botPriceLimit - lower price limit.
  - uint _topPriceLimit - upper price limit.

  Constraints
  - Can only be called by the owner.

  Events emit
  - Emits *note* event.

  Output
  None
- ***prepareDistribution***
  Description
  Prepares distributions.
  Visibility
  external
  Input parameters
  - uint _round - token-sale round.

  Constraints
  - Can only be called by the owner.

  Events emit
  - Emits *note* event.

  Output
  None
- ***_prepareDistribution***
  Description
  Prepares distributions.
  Visibility
  private
  Input parameters
  - uint _round - token-sale round.

  Constraints
  - The round should be finished

  Events emit
  None

Output
None

- ***receiveAll***
  Description
  Receive tokens/rewards for all processed rounds.
  Visibility
  public
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  None
- ***_receiveAll***
  Description
  Receive tokens/rewards for all processed rounds.
  Visibility
  private
  Input parameters
    - address _holder - a holder address whose reward will be processed.

  Constraints
    - Unprocessed rewards should exist.
    - The holder should participate in the first round of the token-sale.
    - Token-sale should be active.

  Events emit
  None
  Output
  None
- ***_receiveDistribution***
  Description
  Receive tokens based on the deposit.
  Visibility
  private
  Input parameters
    - uint _round - round to be processed.
    - address _holder - an address to be processed for.
    - Round memory _localRound - round info.

  Constraints
  None
  Events emit
  None
  Output

None

- ***_receiveRewards***
  Description
  Receive rewards based on the last round balance.
  Visibility
  private
  Input parameters
  - uint _round - round to be processed.
  - address _holder - an address to be processed for.
  - Round memory _localRound - round info.

  Constraints
  - Round should be less than or equal to 21.

  Events emit
  None
  Output
  None

- ***depositUSDT, depositUSDC***
  Description
  Deposit in USDT or USDC tokens.
  Visibility
  external
  Input parameters
  - uint _amount - deposit sum.

  Constraints
  None
  Events emit
  - Emits *note* event.

  Output
  None
- ***depositETH, receive()***
  Description
  Deposit in ETH. All ETH that are send to the contract will be assumed as deposit.
  Visibility
  External
  Input parameters
  None
  Constraints
  None
  Events emit
  - Emits *noteDeposit* event.

  Output
  None
- ***_depositETH***
  Description

Transfer incoming ETH to the treasury and process deposit
Visibility
Private
Input parameters
None
Constraints
None
Events emit
None
Output
None

- ***_deposit***
  Description
  Process deposit
  Visibility
  Private
  Input parameters
    - uint _amount - amount to process.
  Constraints
    - if a deposit is the first round, a caller should be allowed to participate.
    - if a deposit is the first round, the amount should not exceed a limit.
    - amount should be equal or more than 1 USDC.
  Events emit
  None
  Output
  None
- ***currentRound***
  Description
  Get current token-sale round.
  Visibility
  public view
  Input parameters
  None
  Constraints
    - token-sale should be active.
  Events emit
  None
  Output
    - uint - current round.

- ***depositRound***
  Description
  Get current deposit token-sale round. Deposit rounds ends 1 hour before the end of each round.
  Visibility
  public view
  Input parameters
  None
  Constraints
  - token-sale should be active.

  Events emit
  None
  Output
  None
- ***isRoundDepositsEnded***
  Description
  Check if a provided deposit round ended.
  Visibility
  public view
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  - bool - true if the deposit round ended.
- ***isRoundPrepared***
  Description
  Check if a provided round prepared
  Visibility
  public view
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  - bool - true if the round prepared.
- ***currentPrice***
  Description
  Get current DEXE price in USDC.
  Visibility
  public view
  Input parameters

None
Constraints
None
Events emit
None
Output
- uint - current DEXE price in USDC.

- ***updateAndGetCurrentPrice***
Description
Update and get current DEXE price in USDC.
Visibility
public view
Input parameters
None
Constraints
None
Events emit
None
Output
  - uint - current DEXE price in USDC.
- ***_passed***
Description
Check if a provided time passed.
Visibility
private view
Input parameters
None
Constraints
None
Events emit
None
Output
  - bool - true if passed.
- ***_notPassed***
Description
Check if a provided time not passed.
Visibility
private view
Input parameters
None
Constraints
None
Events emit
None
Output
  - bool - true if not passed.

- ***_not***
  Description
  Revert a provided condition.
  Visibility
  private view
  Input parameters
    - bool _condition - a condition to be reverted.
  Constraints
  None
  Events emit
  None
  Output
    - bool - true if _condition is false and vice versa.
- ***releaseLock***
  Description
  Release locked tokens.
  Visibility
  external
  Input parameters
    - LockType _lock - lock type to be released.
  Constraints
    - Lock type should be allowed to be released.
  Events emit
    - Emits *note* event.
  Output
  None
- ***_release***
  Description
  Release locked tokens.
  Visibility
  private
  Input parameters
    - LockType _lock - lock type to be released.
    - address _holder - holder address.
  Constraints
    - Lock type should be allowed to be released.
  Events emit
  None
  Output
  None
- ***transferLock***
  Description
  Transfer locked tokens.
  Visibility
  external
  Input parameters

- LockType _lock - lock type to be released.
- address _to - recipient address.
- uint _amount - amount to transfer.

Constraints

- The sender should have corresponding amount of locked tokens.

Events emit

- Emits *note* event.

Output

None

- ***forceReleaseStaking***

Description

Release staking.

Visibility

external

Input parameters

- ForceReleaseType _forceReleaseType - release type.

Constraints

- Round of the token-sale should exceed 10.
- The sender should have locked staking balance.
- Staking should not be released yet.

Events emit

- Emits *note* event.

Output

None

- ***launchProduct***

Description

Mark product as launched.

Visibility

external

Input parameters

- ForceReleaseType _forceReleaseType - release type.

Constraints

- Can only be called by the owner.
- Token-sale should be passed and processed.
- The product should not be launched yet.

Events emit

- Emits *note* event.

Output

None

- ***isTokensaleProcessed***

Description

Check if the token-sale is processed

Visibility

private view

Input parameters

None
Constraints
None
Events emit
None
Output
- bool - true if processed.

- ***_isHolder***
Description
Check if a provided address is holder.
Visibility
private view
Input parameters
- address _addr - address to check.

Constraints
None
Events emit
None
Output
- bool - true if holder.

- ***_beforeTokenTransfer***
Description
Recalculates average balance. Called before every token transfer.
Visibility
internal
Input parameters
- address _from - sender address.
- address _to - recipient address.
- uint _amount - sent amount.

Constraints
None
Events emit
None
Output
None

- ***_since***
Description
Calculate how much time passed since provided time.
Visibility
private view
Input parameters
- uint _timestamp - timestamp to calculate from.

Constraints
None
Events emit

None
Output
- uint - passed time.

- ***launchDate***
Description
Get product launch date.
Visibility
public view
Input parameters
None
Constraints
None
Events emit
None
Output
  - uint - product launch timestamp.
- ***_calculateBalanceAverage***
Description
Calculates average balance of a holder.
Visibility
private view
Input parameters
  - address _holder - the holder address.
Constraints
None
Events emit
None
Output
  - BalanceInfo memory - balance info of the holder.
- ***_updateBalanceAverage***
Description
Updates average balance of a holder.
Visibility
private
Input parameters
  - address _holder - the holder address.
Constraints
None
Events emit
None
Output
None

- ***getAverageBalance***
  Description
  Get average balance of a holder.
  Visibility
  private view.
  Input parameters
    - address _holder - the holder address.

  Constraints
  None
  Events emit
  None
  Output
    - uint - holders average balance.
- ***firstBalanceChange***
  Description
  Get first balance change time.
  Visibility
  external view
  Input parameters
    - address _holder - the holder address.

  Constraints
  None
  Events emit
  None
  Output
    - uint - holders first balance change time.
- ***holderRounds***
  Description
  Get a holder token-sale round info.
  Visibility
  external view
  Input parameters
    - uint _round - round to get.
    - address _holder - the holder address.

  Constraints
  None
  Events emit
  None
  Output
    - HolderRound memory - holders' round info

- ***eusersInfo***
  Description
  Get a holder info.
  Visibility
  external view
  Input parameters
    - address _holder - the holder address.
  Constraints
  None
  Events emit
  None
  Output
    - UserInfo memory - holders' info
- ***withdrawLocked***
  Description
  Withdraw token in a case when they've been sent to the contract by mistake.
  Visibility
  external
  Input parameters
    - *IERC20 _token* - token to withdraw.
    - *address _receiver* - receiver address.
    - *uint _amount* - amount to withdraw.
  Constraints
    - Can only be called by the owner.
  Events emit
  Emits *note* event.
  Output
  None

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. DISTRIBUTOR_LIMIT field is set to 10 billion tokens. That is more than a total supply.

   Fixed before the secondary audit.

Low

1. priceFeed field of the Dexe contract is never used. It's recommended to remove unused fields and variables.

   Fixed before the secondary audit.

Lowest / Code style / Best Practice

No lowest severity issues were found.

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.

Security engineers found 1 medium and 1 low severity issue during the initial audit. ***All the issues have been fixed before the secondary audit.***

The code is well-tested and works as described in the whitepaper.

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.

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

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.