HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: KeyFi

Date: April 6th, 2021

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

Document

Name	Smart Contract Code Review and Security Analysis Report for KeyFi
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Token, Multisig Timelock, Token factory, Farming
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/KEYFIAI/keyfi-token
Commit	DA604E1926E2B360C3B7E75575A7471E5F3CCC42
Deployed contract	
Timeline	05 APR 2021 – 06 APR 2021
Changelog	06 APR 2021 – INITIAL AUDIT

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

Table of contents

Introduction 4
Scope 4
Executive Summary 5
Severity Definitions 7
AS-IS overview 8
Conclusion 37
Disclaimers 38

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

Introduction

Hacken OÜ (Consultant) was contracted by KeyFi (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 April 5th, 2021 – April 6th, 2021.

Scope

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

Repository:
https://github.com/KEYFIAI/keyfi-token
Commit:
da604e1926e2b360c3b7e75575a7471e5f3ccc42
Files:
Airdrop.sol
Governance.sol
KeyfiToken.sol
Migrations.sol
RewardPool.sol
Timelock.sol
TreasuryVester.sol
Whitelist.sol

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

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency

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 are secure.

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.

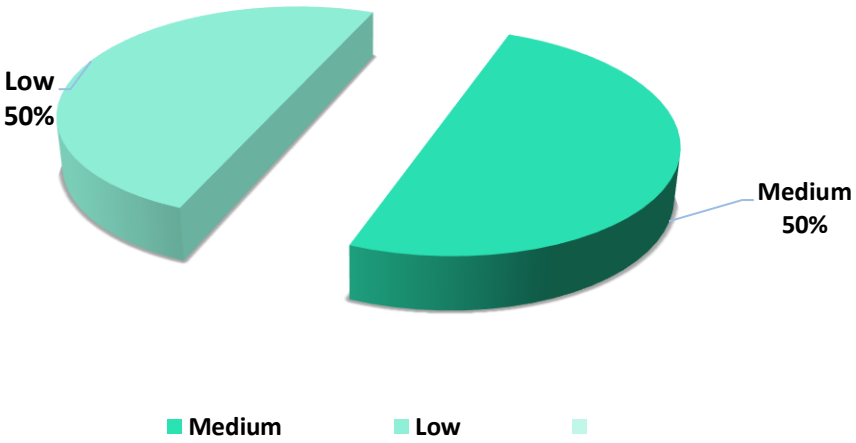
As a result of the **first audit**, security engineers found 2 medium and 2 low severity issues.

Notice:

1. New tokens may be minted by owners unlimitedly.

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

Graph 1. The distribution of vulnerabilities after the first review.



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

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution
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

KeyfiToken.sol

Description

KeyfiToken is an ERC-20 token with voting and minting functionality. Keyfi token has following parameters:

- Name: Keyfi Token
- Symbol: KEYFI
- Decimals: 18

Imports

KeyfiToken contract has following imports:

- import "@openzeppelin/contracts/access/Ownable.sol";
- import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
- import "@openzeppelin/contracts/math/SafeMath.sol";

Inheritance

KeyfiToken contract is IERC20, Ownable.

Usages

KeyfiToken contract has no custom usages.

Structs

KeyfiToken contract has following data structures:

- struct Checkpoint

Enums

KeyfiToken contract has no custom enums.

Events

KeyfiToken contract has following events:

- event MinterChanged(address minter, address newMinter);
- event MinimumMintGapChanged(uint32 previousMinimumGap, uint32 newMinimumGap);
- event MintCapChanged(uint8 previousCap, uint8 newCap);

- event DelegateChanged(address indexed delegator, address indexed fromDelegate, address indexed toDelegate);
- event DelegateVotesChanged(address indexed delegate, uint256 previousBalance, uint256 newBalance);

Modifiers

KeyfiToken has no custom modifiers.

Fields

KeyfiToken contract has following fields and constants:

- string public constant name = "Keyfi Token";
- string public constant symbol = "KEYFI";
- uint8 public constant decimals = 18;
- uint256 public override totalSupply = 1000000e18;
- mapping (address => mapping (address => uint256)) internal allowances;
- mapping (address => uint256) internal balances;
- mapping (address => address) public delegates;
- address public minter;
- uint256 public mintingAllowedAfter;
- uint32 public minimumMintGap = 1 days * 365;
- uint8 public mintCap = 2;
- mapping (address => mapping (uint256 => Checkpoint)) public checkpoints; mapping (address => uint256) public numCheckpoints;
- bytes32 public constant DOMAIN_TYPEHASH = keccak256("EIP712Domain(string name,uint256 chainId,address verifyingContract)");
- bytes32 public constant DELEGATION_TYPEHASH = keccak256("Delegation(address delegatee,uint256 nonce,uint256 expiry)");
- mapping (address => uint) public nonces;

Functions

KeyfiToken has following public and external functions:

- ***constructor***
  Description
  Assigns totalSupply of tokens to an account. Sets minter and minting delay.

Input parameters

- address account
- address _minter
- uint256 _mintingAllowedAfter

Constraints

None

Events emit

Emits Transfer and MinterChanged events.

Output

None

- ***setMinter, setMintCap, setMinimumMintGap***

  Description

  Setters functions restricted for admin use only.

- ***mint***

  Description

  Mint tokens

  Input parameters

  - address _to
  - uint256 _amount

  Constraints

  - Can only be called by the minter.
  - mintingAllowedAfter has reached.
  - _to address should not be 0
  - _amount should not exceed (totalSupply.mul(mintCap)).div(100)

  Events emit

  Emits Transfer event.

  Output

  None

- ***allowance***

  Description

  Get the number of tokens spender is approved to spend on behalf of account

  Input parameters

  - address account
  - address spender

  Constraints

  None

  Events emit

  None

  Output

- uint256 — The number of tokens approved.

- ***approve***
  Description
  Approve spender to transfer up to amount from src
  Input parameters
  - address spender
  - uint256 amount

  Constraints
  None
  Events emit
  None
  Output
  - bool — Whether or not the approval succeeded.
- ***approve***
  Description
  Approve spender to transfer up to amount from src.
  Input parameters
  - address spender
  - uint256 amount

  Constraints
  None
  Events emit
  Emits Approval event.
  Output
  - bool — Whether or not the approval succeeded.
- ***balanceOf***
  Description
  Get the number of tokens held by the account.
  Input parameters
  - address account

  Constraints
  None
  Events emit
  None
  Output
  - uint256 — The number of tokens held.
- ***transfer***
  Description
  Transfer amount tokens from msg.sender to dst.
  Input parameters

- address dst
- uint256 amount

Constraints

- A sender should have enough tokens.

Events emit

Emits Transfer event.

Output

- bool — Whether or not the transfer succeeded.

- ***transferFrom***

  Description

  Transfer amount tokens from src to dst.

  Input parameters

  - address src
  - address dst
  - uint256 amount

  Constraints

  - A message sender should have allowance to transfer tokens from src.

  Events emit

  Emits Approval and Transfer event.

  Output

  - bool — Whether or not the transfer succeeded.

- ***delegate***

  Description

  Delegate votes from msg.sender to delegatee

  Input parameters

  - address delegatee

  Constraints

  None

  Events emit

  Emits DelegateChanged event.

  Output

  None

- ***delegateBySig***

  Description

  Delegates votes from signatory to delegatee.

  Input parameters

  - address delegate
  - uint256 nonce
  - uint256 expiry

- uint8 v
- bytes32 r
- bytes32 s

Constraints
None
Events emit
Emits DelegateChanged event.
Output
None

- ***getCurrentVotes***
  Description
  Get current votes balance for account.
  Input parameters
  - address account

  Constraints
  None
  Events emit
  Emits DelegateChanged event.
  Output
  - uint256 — number of current votes for account.

- ***getCurrentVotes***
  Description
  Get current votes balance for account.
  Input parameters
  - address account

  Constraints
  None
  Events emit
  None
  Output
  - uint256 — number of current votes for account.

- ***getPriorVotes***
  Description
  Determine the prior number of votes for an account as of a block number.
  Input parameters
  - address account
  - uint256 blockNumber

  Constraints

None

Events emit

None

Output

- uint256 — number of votes the account had as of the given block.

Whitelist.sol

Description

Whitelist is a contract that provides whitelist functionality.

Imports

Whitelist contract has following imports:

- import "@openzeppelin/contracts/access/AccessControl.sol";

Inheritance

Whitelist contract is AccessControl.

Usages

Whitelist contract has no custom usages.

Structs

Whitelist contract has no custom data structures.

Enums

Whitelist contract has no custom enums.

Events

Whitelist contract has no custom events.

Modifiers

Whitelist has following modifiers:

- onlyWhitelistAdmin – checks whether a caller is whitelist admin.
- onlyWhitelisted – checks whether a caller is whitelisted.

Fields

Whitelist contract has following fields and constants:

- bytes32 public constant WHITELIST_ADMIN = keccak256("WHITELIST_ADMIN");
- bytes32 public constant WHITELISTED = keccak256("WHITELISTED");

Functions

Whitelist has following public functions:

- ***constructor***
  Description
  Inits the contract. Sets whitelist admin role to a message sender.
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  None
- ***addWhitelistAdmin***
  Description
  Adds new whitelist admin
  Input parameters
  - address account

  Constraints
  - Can only be called by the whitelist admin.

  Events emit
  None
  Output
  None
- ***removeWhitelistAdmin***
  Description
  Remove a whitelist admin
  Input parameters
  - address account

  Constraints
  - Can only be called by the whitelist admin.

  Events emit

None
Output
None

- ***addWhitelisted***
  Description
  Add an address to the whitelist.
  Input parameters
  - address account

  Constraints
  - Can only be called by the whitelist admin.

  Events emit
  None
  Output
  None
- ***removeWhitelisted***
  Description
  Remove an address from the whitelist.
  Input parameters
  - address account

  Constraints
  - Can only be called by the whitelist admin.

  Events emit
  None
  Output
  None
- ***removeWhitelisted***
  Description
  Check whether an address is whitelisted.
  Input parameters
  - address account

  Constraints
  None
  Events emit
  None
  Output
  None
- ***isWhitelistAdmin***
  Description
  Check whether an address is the whitelist admin.
  Input parameters

- address account

Constraints
None
Events emit
None
Output
None

RewardPool.sol

Description

RewardPool is a staking contract with rewards in reward tokens.

Imports

RewardPool contract has following imports:

- import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
- import "@openzeppelin/contracts/token/ERC20/SafeERC20.sol";
- import "@openzeppelin/contracts/math/SafeMath.sol";
- import "@openzeppelin/contracts/access/Ownable.sol";
- import "./KeyfiToken.sol";

Inheritance

RewardPool is Ownable

Usages

RewardPool contract following usages:

- SafeMath for uint256;
- SafeERC20 for IERC20;
- SafeERC20 for KeyfiToken;

Structs

RewardPool contract has following data structures:

- UserInfo
- StakingToken
- TokenIndex

Enums

RewardPool contract has no custom enums.

Events

RewardPool contract has following events:

- event TokenAdded(address indexed token, uint256 allocPoints);
- event TokenRemoved(address indexed token);
- event Deposit(address indexed user, uint256 indexed pid, uint256 amount);
- event Withdraw(address indexed user, uint256 indexed pid, uint256 amount);
- event WithdrawRewards(address indexed user, uint256 amount);
- event EmergencyWithdraw(address indexed user, uint256 indexed pid, uint256 amount);
- event RewardPerBlockChanged(uint256 previousRate, uint256 newRate);
- event SetAllocPoint(address token, uint256 allocPoints);
- event InsufficientRewardpool();
- event SetFeeAddressA(address indexed user, address indexed newAddress);
- event SetFeeAddressB(address indexed user, address indexed newAddress);

Modifiers

RewardPool has no custom modifiers.

Fields

RewardPool contract has following fields and constants:

- KeyfiToken public immutable rewardToken;
- uint256 public immutable bonusEndBlock;
- uint256 public immutable bonusMultiplier;
- uint256 public rewardPerBlock;
- uint256 public totalKeyfiStake;
- address public feeAddA = 0xBff76b1Ab7A545EdB58feB4068A5737AAf3a102c;
- address public feeAddB = 0x5BEBAFE58FC8b87a03Bd39bC147a7fb53e4FABd5;
- StakingToken[] public stakingTokens;
- mapping(address => TokenIndex) public stakingTokenIndexes;

- mapping (uint256 => mapping (address => UserInfo)) public userInfo;
- uint256 public totalAllocPoint = 0;
- uint256 public startBlock;
- uint256 public launchDate;

Functions
RewardPool has following public functions:

- ***constructor***
  Description
  Inits the contract and sets default parameters.
  Input parameters
  - KeyfiToken _rewardToken
  - uint256 _rewardPerBlock
  - uint256 _startBlock
  - uint256 _bonusEndBlock
  - uint256 _bonusMultiplier
  - uint256 _launchDate

  Constraints
  None
  Events emit
  None
  Output
  None
- ***stakingTokensCount, isStakingToken***
  Description
  Simple view functions.
- ***addStakingToken***
  Description
  Adds a token to the list of allowed staking tokens.
  Input parameters
  - uint256 _allocPoint
  - IERC20 _stakingToken
  - uint16 _depositFeeBP

  Constraints
  - Can only be called by the owner.
  - The _depositFeeBPshould should not exceed 400.

  Events emit
  Emits the TokenAdded event.
  Output

None
None

- ***setAllocPoint***
  Description
  Changes the weight allocation for a particular token.
  Input parameters
  - IERC20 _token
  - uint256 _allocPoint
  - uint16 _depositFeeBP.

  Constraints
  - Can only be called by the owner.
  - The _stakingToken should be added
  - The _depositFeeBPshould should not exceed 400.

  Events emit
  None
  Output
  None
- ***getMultiplier***
  Description
  Returns multiplier factor for possible bonuses within a period.
  Input parameters
  - uint256 _from — starting block
  - uint256 _to — last block of the period

  Constraints
  None
  Events emit
  None
  Output
  None
- ***getMultiplier***
  Description
  Returns multiplier factor for possible bonuses within a period.
  Input parameters
  - uint256 _from — starting block
  - uint256 _to — last block of the period

  Constraints
  None
  Events emit
  None
  Output

- uint256 – multiplier factor.

- ***pendingReward***
  Description
  Calculates pending reward for a given staking token and a user.
  Input parameters
  - IERC20 _token
  - address _user

  Constraints
  - A token should be set.

  Events emit
  None
  Output
  - uint256 – pending reward.

- ***massUpdateTokens***
  Description
  Invokes a checkpoint update on all staking tokens in the list.
  Input parameters
  None
  Constraints
  Non
  Events emit
  None
  Output
  None

- ***checkpoint***
  Description
  Calculates all reward rates for a specified token since last checkpoint.
  Visibility
  public
  Input parameters
  - uint256 _pid – a token id.

  Constraints
  - Token with _pid should exist.

  Events emit
  None
  Output
  None

- ***deposit***
  Description
  Deposit _amount into a given _token pool.

Input parameters

- IERC20 _token — the staking token to be deposited.
- uint256 _amount — The amount of tokens to be staked.

Constraints

- _token should be added.

Events emit

Emits Deposit event.

Output

None

- ***withdraw***

Description

Withdraw _amount of a given staking token.

Visibility

public

Input parameters

- IERC20 _token — the staking token to be withdrawn.
- uint256 _amount — the amount of tokens to be withdrawn.

Constraints

- A _token should be added.
- A message sender should have at least _amount of tokens to be deposited earlier.

Events emit

Emits Withdraw event.

Output

None

- ***emergencyWithdraw***

Description

Withdraw all specified staked tokens without a reward.

Visibility

public

Input parameters

- IERC20 _token — the staking token to be withdrawn.

Constraints

- A _token should be added.

Events emit

Emits EmergencyWithdraw event.

Output

None

- ***rewardBlocksLeft***

Description

Calculate remaining blocks left according to current reward supply and rate. **Visibility**
public view
Input parameters
None
Constraints
None
Events emit
None
Output

- uint256 – remaining blocks.

- ***setFeeAddressA, setFeeAddressB***
  Setters of fee addresses. Available only for owner.

GovernorAlpha.sol

Description

GovernorAlpha allows to vote for specific actions using KeyFi token votes.

Imports

GovernorAlpha has no imports.

Inheritance

GovernorAlpha does not inherit anything.

Usages

GovernorAlpha contract has no usages.

Structs

GovernorAlpha contract has following data structures:

- Proposal
- Receipt

Enums

GovernorAlpha contract has following enums:

- ProposalState

Events

GovernorAlpha contract has following events:

- event ProposalCreated(uint id, address proposer, address[] targets, uint[] values, string[] signatures, bytes[] calldatas, uint startBlock, uint endBlock, string description);
- event VoteCast(address voter, uint proposalId, bool support, uint votes);
- event ProposalCanceled(uint id);
- event ProposalQueued(uint id, uint eta);
- event ProposalExecuted(uint id);

Modifiers

GovernorAlpha has no modifiers.

Fields

GovernorAlpha contract has following fields and constants:

- string public constant name = "KeyFi Governance" – name of the contract
- TimelockInterface public timelock – the timelock address.
- TokenInterface public keyfi – the KeyFi token address.
- address public guardian – a guardian address.
- uint public proposalCount – the total number of proposals
- mapping (uint => Proposal) public proposals – all proposals.
- mapping (address => uint) public latestProposalIds – proposals of an address.
- bytes32 public constant DOMAIN_TYPEHASH = keccak256("EIP712Domain(string name,uint256 chainId,address verifyingContract)");
- bytes32 public constant BALLOT_TYPEHASH = keccak256("Ballot(uint256 proposalId,bool support)");

Functions

GovernorAlpha has following public functions:

- ***constructor***
  Description
  Inits the contract and sets default parameters.
  Visibility
  public
  Input parameters
  - address timelock_ - the time contract address.

- address keyfi_ - the KeyFi token address.
- address guardian_ - a guardian address.

Constraints

None

Events emit

None

Output

None

- ***propose***

  Description

  Propose a new vote.

  Visibility

  public

  Input parameters

  - address[] memory targets – transaction targets.
  - uint[] memory values – transaction values.
  - string[] memory signatures – functions to be executed.
  - bytes[] memory calldatas – calldata of calls.
  - string memory description – a propose description.

  Constraints

  - A proposer votes should be above the proposal threshold.
  - targets, values, signatures, calldatas should be of the same length.
  - targets length should be more than 0.
  - targets length should be less than proposalMaxOperations.
  - A proposer can have one live.

  Events emit

  Emits the ProposalCreated event.

  Output

  - uint – a propose id.

- ***queue***

  Description

  Queue a proposal for the execution.

  Visibility

  public

  Input parameters

  - uint proposalId – a proposal id.

  Constraints

  - A proposal can only be queued only if it is succeeded.
  - A proposal should not be queued yet.

  Events emit

Emits the ProposalQueued event.
Output
None

- ***execute***
  Description
  Execute a proposal.
  Visibility
  public
  Input parameters
  - uint proposalId – a proposal id.

  Constraints
  - A proposal should be queued.
  - A proposal should not be executed yet.

  Events emit
  Emits the ProposalExecuted event.
  Output
  None
- ***cancel***
  Description
  Cancel a proposal.
  Visibility
  public
  Input parameters
  - uint proposalId – a proposal id.

  Constraints
  - A proposal should not be executed yet.
  - A message sender should be a guardian or a proposer votes become less than threshold.

  Events emit
  Emits the ProposalCanceled event.
  Output
  None
- ***getActions***
  Description
  Get actions of a specified proposal.
  Visibility
  public view
  Input parameters
  - uint proposalId – a proposal id.

  Constraints

None

Events emit

None

Output

- address[] memory targets
- uint[] memory values
- string[] memory signatures
- bytes[] memory calldatas

- ***getReceipt***

  Description

  Get a vote results of a voter in a provided proposal.

  Visibility

  public view

  Input parameters

  - uint proposalId – a proposal id.
  - address voter – a voter address.

  Constraints

  None

  Events emit

  None

  Output

  - Receipt memory

- ***state***

  Description

  Get a state of a given proposal.

  Visibility

  public view

  Input parameters

  - uint proposalId – a proposal id.

  Constraints

  None

  Events emit

  None

  Output

  - ProposalState

- ***state***

  Description

  Get a state of a given proposal.

  Visibility

  public view

Input parameters

- uint proposalId – a proposal id.

Constraints

None

Events emit

None

Output

- ProposalState

- ***castVote***

Description

Participate in a vote.

Visibility

public

Input parameters

- uint proposalId – a proposal id.
- bool support

Constraints

- A proposal should be active.
- A voter should not participate yet.

Events emit

Emits the VoteCast event

Output

None

- ***castVoteBySig***

Description

Participate in a vote on behalf of another voter.

Visibility

public

Input parameters

- uint proposalId – a proposal id.
- bool support
- uint8 v – part of a signature.
- bytes32 r – part of a signature.
- bytes32 s – part of a signature.

Constraints

- A proposal should be active.
- A voter should not participate yet.

Events emit

Emits the VoteCast event

Output

None

- ***__acceptAdmin***
  Description
  Accept admin rights of the Timelock contract.
  Visibility
  public
  Input parameters
  None
  Constraints
  - A message sender should be a guardian.

  Events emit
  None
  Output
  None
- ***__abdicate***
  Description
  Removes a guardian address.
  Visibility
  public
  Input parameters
  None
  Constraints
  - A message sender should be a guardian.

  Events emit
  None
  Output
  None
- ***__queueSetTimelockPendingAdmin***
  Description
  Queue a new pending admin of the Timelock contract.
  Visibility
  public
  Input parameters
  - address newPendingAdmin – an address of the new admin.
  - uint eta – execution time.

  Constraints
  - A message sender should be a guardian.

  Events emit
  None
  Output

None

- ***executeSetTimelockPendingAdmin***
  Description
  Set a new pending admin of the Timelock contract.
  Visibility
  public
  Input parameters
  - address newPendingAdmin – an address of the new admin.
  - uint eta – execution time.

  Constraints
  - A message sender should be a guardian.
  - A transaction should be previously queued.

  Events emit
  None
  Output
  None

Timelock.sol

Description

Timelock queues and executes transactions.

Imports

Timelock has following imports:

- SafeMath.sol – from the OpenZeppelin.

Inheritance

Timelock does not inherit anything.

Usages

Timelock contract has following usages:

- SafeMath for uint.

Structs

Timelock contract has no data structures.

Enums

Timelock contract has no enums.

Events

Timelock contract has following events:

- event NewAdmin(address indexed newAdmin);
- event NewPendingAdmin(address indexed newPendingAdmin);
- event NewDelay(uint indexed newDelay);
- event CancelTransaction(bytes32 indexed txHash, address indexed target, uint value, string signature, bytes data, uint eta);
- event ExecuteTransaction(bytes32 indexed txHash, address indexed target, uint value, string signature, bytes data, uint eta);
- event QueueTransaction(bytes32 indexed txHash, address indexed target, uint value, string signature, bytes data, uint eta);

Modifiers
Timelock has no modifiers.

Fields

Timelock contract has following fields and constants:

- uint public constant GRACE_PERIOD = 14 days;
- uint public constant MINIMUM_DELAY = 2 days;
- uint public constant MAXIMUM_DELAY = 30 days;
- address public admin – an admin address.
- address public pendingAdmin – a pending adming.
- uint public delay – delay between a transaction queueing and execution.
- mapping (bytes32 => bool) public queuedTransactions – queued transactions.

Functions
Timelock has following public functions:

- ***constructor***
  Description
  Inits the contract and sets default parameters.
  Visibility
  public
  Input parameters
  - address admin_ - a contract admin.

- uint delay_ - delay between a transaction queuing and execution.

Constraints

- A delay_ value should be between DELAY and MAXIMUM_DELAY.

Events emit

None

Output

None

- ***receive***

Description

Allows to receive ETH.

- ***setDelay***

Description

Sets a delay.

Visibility

public

Input parameters

- uint delay_ - delay between a transaction queuing and execution.

Constraints

- A message sender should be the contract itself.
- A delay_ value should be between DELAY and MAXIMUM_DELAY.

Events emit

Emits the NewDelay event.

Output

None

- ***acceptAdmin***

Description

Accept the admin permissions.

Visibility

public

Input parameters

None

Constraints

- A message sender should be a pending admin.

Events emit

Emits the NewAdmin event.

Output

None

- ***setPendingAdmin***

Description

Accept the admin permissions.

Visibility
public
Input parameters
- address pendingAdmin_ - a pending admin address.

Constraints
- A message sender should be the contract itself.

Events emit
Emits the NewPendingAdmin event.
Output
None

- ***queueTransaction***

Description
Add a new transaction to the queue.
Visibility
public
Input parameters
- address target – a tx target.
- uint value – a tx value.
- string memory signature – a method signature.
- bytes memory data – a tx data.
- uint eta – a minimum delay between a tx queuing and execution.

Constraints
- A message sender should be admin.
- eta should be more than current time plus delay value.

Events emit
Emits the QueueTransaction event.
Output
bytes32 – a tx hash.

- ***cancelTransaction***

Description
Cancel a transaction.
Visibility
public
Input parameters
- address target – a tx target.
- uint value – a tx value.
- string memory signature – a method signature.
- bytes memory data – a tx data.

- uint eta – a minimum delay between a tx queuing and execution.

Constraints

- A message sender should be admin.

Events emit

Emits the CancelTransaction event.

Output

None

- ***cancelTransaction***

Description

Execute a transaction.

Visibility

public

Input parameters

- address target – a tx target.
- uint value – a tx value.
- string memory signature – a method signature.
- bytes memory data – a tx data.
- uint eta – a minimum delay between a tx queuing and execution.

Constraints

- A message sender should be admin.
- A transaction should be queued.
- Current timestamp should be between eta and eta + GRACE_PERIOD.

Events emit

Emits the ExecuteTransaction event.

Output

None

Airdrop.sol

Description

Simple airdrop contract that allows receive tokens for whitelisted users.

TreasuryVester.sol

Description

Simple vesting contract.

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. _cap parameter contains no upper limit validations. Unlimited amount of tokens can be minted as a result.

   Contract: RewardPool

   Function: setMintCap

   Recommendation: limit the _cap value.

2. _gap parametr contains no lower limits validation. Tokens can be minted very often.

   Contract: RewardPool

   Function: setMinimumMintGap

   Recommendation: limit the _gap value.

Low

1. Functions withdraws rewards when called but does not emits the WithdrawRewards event.

   Contract: RewardPool

   Functions: withdraw, deposit

   Recommendation: emit corresponding event.

2. Functions contains common rewards withdraw logic that can be moved to a separate function.

   Contract: RewardPool

   Functions: withdraw, deposit, withdrawRewards

Recommendation: emit corresponding event.

Informational / Code style / Best Practice

No informational issues were found.

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.

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

As a result of the **first audit**, security engineers found 2 medium and 2 low severity issues.

Notice:

1. New tokens may be minted by owners unlimitedly.

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

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

Technical Disclaimer

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