SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: YVS

Date: December 15th, 2020

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 YVS (122 pages)
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Token, Vaults, Staking
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Git	HTTPS://GITHUB.COM/YVS-FINANCE/YVS-PROTOCOL
Commit	A5EAA61EC1B231D5ACB88070FF8EB21FFE2C5D4A
Timeline	7TH DEC 2020 – 15TH DEC 2020
Changelog	15TH DEC 2020 - Initial Audit 15TH DEC 2020 - Second Audit

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by YVS (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between December 7th, 2020 – December 15th, 2020.

Scope

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

HTTPS://GITHUB.COM/YVS-FINANCE/YVS-PROTOCOL
(1) A5EAA61EC1B231D5ACB88070FF8EB21FFE2C5D4A
(2) E76AFF47F5B613E02D25A6C49D62D764C70DCE04

Files in scope of review
./contracts/controller.sol
./contracts/payment-splitter.sol
./contracts/pool.sol
./contracts/pool-liquidity.sol
./contracts/pool-staking.sol
./contracts/presale.sol
./contracts/tax-collector.sol
./contracts/timelock.sol
./contracts/token.sol
./contracts/token-timelock.sol
./contracts/vault.sol
./contracts/strategies/strategy-base.sol
./contracts/strategies/strategy-curve-base.sol
./contracts/strategies/curve/strategy-curve-rencrv-v1.sol
./contracts/strategies/curve/strategy-curve-scrv-v1.sol
./contracts/strategies/curve/strategy-curve-tbtccrv-v1.sol
./contracts/strategies/curve/strategy-curve-usdncrv-v1.sol
./contracts/token/erc20.sol

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

Category	Check Item
Code review	▪ Reentrancy

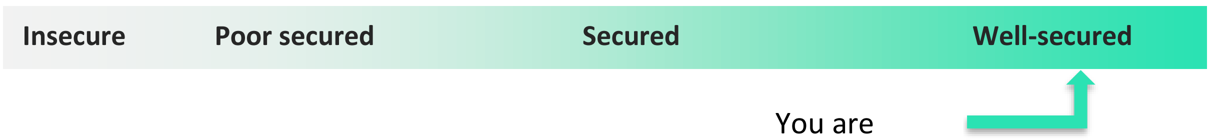
	▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility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

Executive Summary

According to the assessment, the Customer's smart contracts have critical vulnerabilities and can not be considered secure. Fixes are required.

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

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

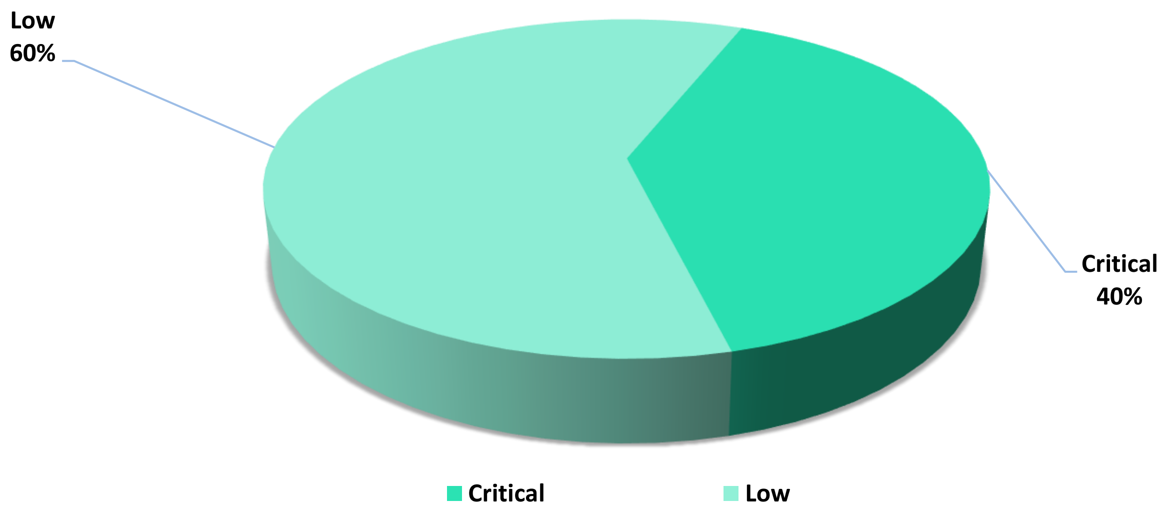


[1] Look for details and justification in conclusion section

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 **2** critical and **3** low severity issues during the audit.

Graph 1. The distribution of vulnerabilities.



Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are essential to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored.

AS-IS overview

token.sol

Description

YvsToken is an ERC20 token contract with custom functions that are used by the Owner to manage the whitelist and tax address.

Imports

YvsToken contract has 2 imports:

- *ownable.sol* — from project files;
- *erc20.sol* — from project files;

Inheritance

YvsToken contract inherits *ERC20* and *Ownable*.

Functions

YvsToken has 3 functions:

- ***setTaxAddress***

  Description

  Wrapper for *_setupTaxAddress* function.

  Visibility

  public

  Input parameters

  - *address taxAddress_* — a tax address;

  Constraints

  - Only the Owner can call it.

  Events emit

  None

  Output

None

- ***addWhitelistedAddress***

Description

Used by the Owner to add an address to the whitelist.

Visibility

public

Input parameters

 - *address _address* — an address;

Constraints

 - Only the Owner can call it.

Events emit

None

Output

None

- ***removeWhitelistedAddress***

Description

Used by the Owner to remove an address from the whitelist.

Visibility

public

Input parameters

 - *address _address* — an address;

Constraints

 - Only the Owner can call it.

Events emit

None

Output

None

erc20.sol

Description

ERC20 is an ERC20 token contract with custom functions that are used by the Owner to manage the whitelist and tax address.

Imports

ERC20 contract has 4 imports:

- *context.sol* — from project files;
- *safe-math.sol* — from project files;
- *address.sol* — from project files;
- *ierc20.sol* — from project files;

Inheritance

ERC20 contract inherits *Context* and *IERC20*.

Usings

ERC20 contract use:

- *SafeMath* for *uint256*;
- *Address* for *address*;

Fields

ERC20 contract has 12 fields:

- *mapping (address => bool) public _whitelistedAddresses* — a map that tracks whitelisted addresses;
- *mapping (address => uint256) private _balances* — a map that tracks balances;
- *mapping (address => mapping (address => uint256)) private _allowances* — a map that tracks allowance;
- *uint256 private _totalSupply* — total supply;
- *uint256 private _burnedSupply* — burned supply;
- *uint256 private _taxedSupply* — taxed supply;
- *uint256 private _taxRate* — tax rate;
- *uint256 private _taxRateBase* — a base for tax rate;

- *address private _taxAddress* — an address for taxes;
- *string private _name* — a name of the token;
- *string private _symbol* — a symbol of the token;
- *uint256 private _decimals* — a decimals of the token;

Default OpenZeppelin Functions

ERC20 has 15 functions originally from OpenZeppelin:

- *name*;
- *symbol*;
- *decimals*;
- *totalSupply*;
- *balanceOf*;
- *transfer*;
- *burn*;
- *allowance*;
- *approve*;
- *transferFrom*;
- *increaseAllowance*;
- *decreaseAllowance*;
- *_mint*;
- *_approve*;
- *_beforeTokenTransfer*;

Custom Functions

ERC20 has 8 custom functions:

- ***constructor***

  Description

  Initializes the contract. Mints init supply.

  Visibility

  public

  Input parameters

  - *string memory name* — a name of the token;
  - *string memory symbol* — a symbol of the token;
  - *uint256 decimals* — a decimals of the token;
  - *uint256 initSupply* — the number of tokens to be minted;

  Constraints

None

Events emit

None

Output

None

- ***burnedSupply***

Description

Used to get the amount of burned tokens.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns the amount of burned tokens.

- ***taxedSupply***

Description

Used to get the amount of taxed tokens.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns the amount of taxed tokens.

- ***taxRate***

  Description

  Used to get the tax rate.

  Visibility

  public view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns the tax rate.

- ***_transfer***

  Description

  Used to move tokens from *sender* to *receiver*. If someone from
  the *sender* or *recipient* is not on the white list, taxes will be deducted from
  the *amount*.

Visibility

internal virtual

Input parameters

- *address sender* — an address of the sender;
- *address recipient* — an address of the recipient;
- *uint256 amount* — an amount of tokens;

Constraints

- *sender* should not be zero address.
- *recipient* should not be zero address.
- *amount* must be less than or equal to the balance of the *sender*.
- Taxes must be calculated correctly.

Events emit

- *Transfer(sender, recipient, amount);*

Output

None

- ***_burn***

Description

Used to burn tokens. Keeps track of how many tokens have been burned.

Visibility

internal virtual

Input parameters

- *address account* — an address of the account from which tokens will be burned;
- *uint256 amount* — an amount of tokens to burn;

Constraints

- *account* should not be zero address.
- *amount* must be less than or equal to the balance of the *account*.

Events emit

- *Transfer(account, address(0), amount);*

Output

None

- ***_tax***

Description

Blabla

Visibility

public

Input parameters

- *address account* — an address of the account from which taxes will be deducted;
- *uint256 amount* — an amount of tokens to deduct;

Constraints

- *account* should not be zero address.
- *_taxAddress* should not be zero address.
- *amount* must be less than or equal to the balance of the *account*.

Events emit

- *Transfer(account, _taxAddress, amount);*

Output

None

- ***_setupTaxAddress***

Description

Used to set the tax address.

Visibility

internal virtual

Input parameters

- *address taxAddress_* — the new address for receiving taxes;

Constraints

- The tax address can only be set once.

Events emit

None

Output

None

presale.sol

Description

YvsPresale is a contract that is responsible for collecting Wei and distributing its own tokens without any administrative control (for all eth / token related activities).

Imports

YvsPresale contract has 7 imports:

- *burnable.sol* — from project files;
- *uniswap-v2.sol* — from project files;
- *controller.sol* — from project files;
- *token-timelock.sol* — from project files;
- *ownable.sol* — from project files;
- *erc20.sol* — from project files;
- *safe-math.sol* — from project files;

Inheritance

YvsPresale contract inherits *Ownable*.

Usings

YvsPresale contract use:

- *SafeMath* for *uint256*;
- *SafeERC20* for *IERC20*;

Modifiers

YvsPresale contract has 2 modifiers:

- *distributed* — checks if tokens distributed;
- *active* — checks if presale is active;

Fields

YvsPresale contract has 35 fields:

- *IERC20 public token* — token that is for sale;
- *address payable public team* — an address of the team;
- *address payable public marketing* — an address of the marketing;
- *address payable public listing* — an address of the listing;
- *address public controller* — an address of the controller contract;
- *address payable public treasury* — an address of the treasury contract;
- *address public timelock* — an address of the timelock contract;
- *uint256 public start* — a timestep of the start pre-selling;
- *uint256 public duration* — duration of pre-sale;
- *uint256 public grace* — duration of the grace period;
- *uint256 public cap* — token max cap;
- *uint256 public threshold* — the presale threshold to close;
- *uint256 public total* — total to be distributed;
- *uint256 public deposited* — total wei deposited;
- *uint256 public depositors* — total number of depositors;
- *uint256 public min* — the minimum amount of ETH for investment;
- *uint256 public max* — the maximum amount of ETH for investment;
- *uint256 public rate* — the token exchange rate for the base amount;
- *uint256 public referralRate* — referral bonus rate;
- *uint256 public referralRateReferrer* — referral bonus rate for referrer;
- *uint256 public referralRateDepositor* — referral bonus rate for depositor;
- *uint256 public referralRateBase* — referral bonus rate base;
- *string public contact* — public contact information;
- *bool public finalized* — indicates if the presale is finalized;
- *bool public completed* — indicates if the distribution is finished;
- *bool public cancelled* — indicates if the presale is cancelled;
- *bool public closed* — indicates if the presale is closed;
- *mapping(address => uint256) public deposits* — a mapping for deposits;
- *mapping(address => uint256) public balances* — a mapping for balances;
- *mapping(address => uint256) public bonus* — a mapping for bonuses;
- *mapping(bytes12 => address) public referrals* — a mapping for referrals;
- *mapping(address => bool) public registered* — a mapping for track registered referrals;
- *UniswapRouterV2 internal uniswap* — Uniswap Router;
- *UniswapV2Factory internal factory* — Uniswap Factory
- *address internal weth* — an address of WETH9 contract;

Functions

YvsPresale has 20 functions:

- ***constructor***

  Description

  Initializes the contract.

  Visibility

  public

  Input parameters

  - *address _token* — an address of the token for sale;
  - *address _timelock* — an address of the timelock contract;
  - *uint256 _start* — a timestep of the start pre-selling;
  - *string memory _contact* — public contact information;

  Constraints

  - *_start* must be greater than or equal to *block.timestamp*.

  Events emit

  None

  Output

  None

- ***receive***

  Description

  Fallback function to enter presale.

  Visibility

  external payable

  Input parameters

  None

  Constraints

  None

  Events emit

None

Output

None

- ***enter***

Description

Used to enter presale.

Visibility

public payable

Input parameters

  - *uint256 _amount* — the wei amount;

Constraints

  - Presale must be active.
  - *msg.value* must be equal to *_amount*.
  - *msg.sender* must not be zero address.
  - The amount of wei must be within the specified limits.
  - There must be enough tokens for distribution.

Events emit

  - *DailyBonusEarned(msg.sender, dailyBonus);*
  - *PresaleEntered(msg.sender, amount, distribution);*

Output

None

- ***enter***

Description

Used to enter presale with referral.

Visibility

public payable

Input parameters

- *uint256 _amount* — the wei amount;
- *bytes12 _code* — referral code;

Constraints

- The presale must be active.
- *msg.value* must be equal to *_amount*.
- *msg.sender* must not be zero address.
- The amount of wei must be within the specified limits.
- Referral code must be valid.
- The referral code must belong to an account other than *msg.sender*.
- There must be enough tokens for distribution.

Events emit

- *DailyBonusEarned(msg.sender, dailyBonus);*
- *ReferrerEarned(referrer, msg.sender, referrerBonus);*
- *DepositorEarned(msg.sender, depositorBonus);*
- *PresaleEntered(msg.sender, amount, distribution);*

Output

None

- ***refund***

Description

Used to refund collected eth from the user if presale is canceled.

Visibility

external

Input parameters

None

Constraints

- The presale must be canceled.
- *msg.sender* must have deposit.

Events emit

- *Refunded(msg.sender, deposits[msg.sender]);*

Output

None

- ***claim***

Description

Used to claim tokens after presale is distributed.

Visibility

external

Input parameters

None

Constraints

- Tokens were distributed.
- *msg.sender* must have tokens on the balance.

Events emit

- *Claimed(msg.sender, balances[msg.sender]);*

Output

None

- ***referral***

Description

Sets a referral code for an address.

Visibility

external

Input parameters

- *bytes12 code* — referral code;

Constraints

- This referral code must not be set before.

- *msg.sender* must not have a registered code.

Events emit

- *ReferralSet(msg.sender, code);*

Output

None

- ***distribute***

Description

Used to distribute wei, create a liquidity pair, and an initial reward after the end of the presale.

Visibility

external

Input parameters

None

Constraints

- The presale must be concluded.
- The balance of the contract must be greater than or equal to deposited amount.

Events emit

- *LiquidityAddedAndLocked(added, timelock);*
- *Completed();*

Output

None

- ***salvage***

Description

Salvages unrelated tokens to presale.

Visibility

external

Input parameters

- *address _token* — an address of token to salvage;

Constraints

- Tokens were distributed.
- Only owner can call it.
- The presale token can not be salvage.

Events emit

- *Salvaged(_token, balance);*

Output

None

- ***collect_dust***

Description

Used to collect wei left as dust on contract after grace period.

Visibility

external

Input parameters

None

Constraints

- Tokens were distributed.
- Only owner can call it.
- The presale must not be canceled.
- The grace period must over.

Events emit

- *DustCollected(treasury, balance);*

Output

None

- ***destroy***

  Description

  Used to burn leftover tokens from presale.

  Visibility

  external

  Input parameters

  None

  Constraints

  - Tokens were distributed.
  - Only owner can call it.
  - The presale must not be canceled.
  - The grace period must over.

  Events emit

  - *Destroyed(balance);*

  Output

  None

- ***set_controller***

  Description

  Used to set controller contract.

  Visibility

  external

  Input parameters

  - *address _controller* — an address of the controller contract;

  Constraints

  - Only owner can call it.
  - The controller address must not be zero.

Events emit

None

Output

None

- ***update***

Description

Updates public contact information.

Visibility

external

Input parameters

- *string memory _contact* — text to set as contact information;

Constraints

- Only owner can call it.

Events emit

None

Output

None

- ***cancel***

Description

Used to cancel presale, stop accepting wei and enable refunds.

Visibility

external

Input parameters

None

Constraints

- Only owner can call it.

Events emit

None

Output

None

- ***close***

Description

Used to close presale if threshold is reached.

Visibility

external

Input parameters

None

Constraints

- Only owner can call it.
- The the threshold must be reached.

Events emit

None

Output

None

- ***claimable***

Description

Used to get claimable amount for address.

Visibility

external view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns claimable amount for address.

- ***valid***

Description

Checks if wei amount is within limits.

Visibility

internal view

Input parameters

  - *address account* — an address of the account;
  - *uint256 amount* — wei amount;

Constraints

None

Events emit

None

Output

Returns *true* if wei amount is within limits.

- ***distributable***

Description

Checks if token amount can be distributed.

Visibility

internal view

Input parameters

- *uint256 amount* — an amount of the tokens;

Constraints

None

Events emit

None

Output

Returns *true* if token amount can be distributed.

- ***concluded***

Description

Checks if the presale is concluded.

Visibility

internal view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns *true* if the presale is concluded.

- ***reached***

Description

Checks if threshold is reached.

Visibility

internal view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns *true* if threshold is reached.

controller.sol

Description

YvsController is a contract that is responsible for starting rewards after presale is concluded, for distribution after initial rewards are finished, and for communicating with the vaults/strategies.

Imports

YvsController contract has 7 imports:

- *burnable.sol* — from project files;
- *pool.sol* — from project files;
- *vault.sol* — from project files;
- *strategy.sol* — from project files;
- *ownable.sol* — from project files;
- *erc20.sol* — from project files;
- *safe-math.sol* — from project files;

Inheritance

YvsController contract inherits *Ownable*.

Usings

YvsController contract use:

- *SafeMath* for *uint256*;
- *SafeERC20* for *IERC20*;

Modifiers

YvsController contract has 4 modifiers:

- *onlyPresale* — checks if a caller is presale contract;
- *restricted* — checks if a caller is presale contract or owner;
- *distributed* — checks if the distribution is finished;
- *started* — checks if presale started;

Fields

YvsController contract has 20 fields:

- *IERC20 public token* — token that is for sale;
- *address public presale* — presale contract address;
- *address public staking_pool* — an address of the staking pool;
- *address public liquidity_pool* — an address of the liquidity pool;
- *address public vault_btc_pool* — an address of the vault pool (btc);
- *address public vault_stables_pool* — an address of the vault pool (stablecoin)
- *address public vault_btc* — an address of the btc vault;
- *address public vault_stables* — an address of the stablecoin address
- *uint256 public stakingPercentage* — percentage of tokens for the staking pool;
- *uint256 public liquidityPercentage* — percentage of tokens for the liquidity pool;
- *uint256 public baseRate* — base rate;
- *uint256 public last_distribution* — a timestamp of the last distribution;
- *uint256 public last_harvest* — a timestamp of the last harvest;
- *uint256 public start* — a timestamp of the start;
- *uint256 public grace* — duration of the grace period;
- *uint256 public interval* — interval after grace period;
- *uint256 public harvest_interval* — interval for harvesting;
- *address private uniswap_pair* — an address of the Uniswap liquidity pair;
- *bool private ready* — token distribution indicator;
- *bool private first_run* — indicates whether this is an initial presale call or not;

Functions

YvsController has 13 functions:

- ***constructor***

Description

Initializes the contract.

Visibility

public

Input parameters

- *address _token* — an address of the token for sale;
- *address _presale* — an address of the presale contract;
- *uint256 _start* — a timestep of the start pre-selling;

Constraints

None

Events emit

None

Output

None

- ***set_pair***

Description

Sets Uniswap liquidity pair.

Visibility

public

Input parameters

- *address pair* — an address of the pair;

Constraints

- Only presale contract can call it.
- *uniswap_pair* is not set before.

Events emit

None

Output

None

- ***set_ready***

Description

Sets ready indicator to allow rewards to start.

Visibility

public

Input parameters

- *bool _ready* — a true/false value to signal the start;

Constraints

- Only presale contract can call it.

Events emit

None

Output

None

- ***set_harvest_interval***

Description

Sets harvest interval (how often rewards are collected).

Visibility

public

Input parameters

- *uint256 _harvest_interval* — interval in seconds;

Constraints

- Tokens were distributed.
- Only owner or presale contract can call it.

- It must be an initial presale call.
- *uniswap_pair* must be set.

Events emit

None

Output

None

- ***notify***

Description

Notifies reward amounts to pools after presale distribution.

Visibility

external

Input parameters

None

Constraints

- Only owner or presale contract can call it.

Events emit

None

Output

None

- ***_notify***

Description

Internal notify function to signal rewards.

Visibility

internal

Input parameters

None

Constraints

None

Events emit

- *NotifyRewards();*

Output

None

- ***distribute***

Description

Used to distribute tokens after the grace period.

Visibility

external

Input parameters

None

Constraints

- There must be an initial presale call before this.
- The grace period must over.
- If *last_distribution* is greater than 0, the interval should end.

Events emit

None

Output

None

- ***_distribute***

Description

Internal distribution method that allocates tokens.

Visibility

internal

Input parameters

None

Constraints

- This contract must have tokens to distribute.

Events emit

- *Distributed(balance);*

Output

None

- ***vaults_earn***

Description

Used to to start earning rewards in vaults.

Visibility

public

Input parameters

None

Constraints

- The presale must be started.

Events emit

None

Output

None

- ***vaults_harvest***

Description

Used to harvest rewards in vault.

Visibility

public

Input parameters

None

Constraints

- The presale must be started.
- If *last_harvest* is greater than 0, the harvest interval should end.

Events emit

None

Output

None

- ***vaults_collect***

Description

Used to collect purchases in vaults.

Visibility

public

Input parameters

None

Constraints

- The presale must be started.

Events emit

None

Output

None

- ***salvage***

Description

Salvages non-native tokens from the contract.

Visibility

external

Input parameters

- *address _token* — an address of token to salvage;
- *address recipient* — an address of the tokens recipient;

Constraints

- Only owner can call it.
- The native token can not be salvage.

Events emit

- *Salvaged(_token, balance);*

Output

None

- ***next***

Description

Used to get the next distribution timestamp.

Visibility

external view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns the next distribution timestamp.

payment-splitter.sol

Description

YvsPaymentSplitter is a fork from OpenZeppelin PaymentSplitter contract with a changed name only. (https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/payment/PaymentSplitter.sol)

timelock.sol

Description

YvsTimelock is a fork from Compound Timelock contract with a changed name and fallback function changed to receive. (https://github.com/compound-finance/compound-protocol/blob/master/contracts/Timelock.sol).

pool.sol

Description

YvsPool is a staking pool contract.

Imports

YvsPool contract has 4 imports:

- *reentrancy-guard.sol* — from project files;
- *pausable.sol* — from project files;
- *erc20.sol* — from project files;
- *safe-math.sol* — from project files;

Inheritance

YvsPool contract inherits *ReentrancyGuard* and *Pausable*.

Usings

YvsPool contract use:

- *SafeMath* for *uint256*;
- *SafeERC20* for *IERC20*;

Modifiers

YvsPool contract has 2 modifiers:

- *updateReward* — updates reward;
- *restricted* — checks if a caller is controller contract or owner;

Fields

YvsPool contract has 12 fields:

- *address public controller* — an address of the controller contract;
- *IERC20 public rewardsToken* — rewards token;
- *IERC20 public stakingToken* — staking token;
- *uint256 public periodFinish* — a timestamp of the period finish;
- *uint256 public rewardRate* — reward rate;
- *uint256 public rewardsDuration* — reward period duration;
- *uint256 public lastUpdateTime* — a timestamp of the last update;
- *uint256 public rewardPerTokenStored* — reward per token;
- *mapping(address => uint256) public userRewardPerTokenPaid* — a mapping for reward per token;
- *mapping(address => uint256) public rewards* — a mapping for rewards;
- *uint256 private _totalSupply* — total supply;
- *mapping(address => uint256) private _balances* — a mapping for balances;

Functions

YvsPool has 17 functions:

- ***constructor***

  Description

  Initializes the contract.

  Visibility

  public

  Input parameters

  - *address _rewardsToken* — an address of the rewards token;
  - *address _stakingToken* — an address of the staking token;

Constraints

None

Events emit

None

Output

None

- ***totalSupply***

Description

Used to get total supply.

Visibility

external view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns total supply.

- ***balanceOf***

Description

Used to get balance of account.

Visibility

external view

Input parameters

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

Constraints

None

Events emit

None

Output

Returns balance.

- ***lastTimeRewardApplicable***

Description

Used to get the minimum value for *block.timestamp* and *periodFinish*.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns the minimum value for *block.timestamp* and *periodFinish*.

- ***rewardPerToken***

Description

Used to get reward per token.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns reward per token.

- ***earned***

Description

Used to calculate reward for account.

Visibility

public view

Input parameters

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

Constraints

None

Events emit

None

Output

Returns reward for account.

- ***getRewardForDuration***

Description

Used to calculate reward for duration.

Visibility

external view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns reward for duration.

- ***min***

Description

Used to get min from two *uint256*.

Visibility

public pure

Input parameters

- *uint256 a*;
- *uint256 b*;

Constraints

None

Events emit

None

Output

Returns min value.

- ***stake***

Description

Used to stake tokens.

Visibility

external

Input parameters

  - *uint256 amount* — an amount of tokens;

Constraints

  - It cannot be used for reentrancy.
  - The contract must not be paused.
  - *amount* should be greater than 0.

Events emit

  - *Staked(msg.sender, actualReceived);*

Output

None

- ***withdraw***

Description

Used to withdraw tokens.

Visibility

public

Input parameters

  - *uint256 amount* — an amount of tokens;

Constraints

  - It cannot be used for reentrancy.
  - *amount* should be greater than 0.

Events emit

- *Withdrawn(msg.sender, amount);*

Output

None

- ***getReward***

Description

Used to withdraw reward.

Visibility

public

Input parameters

None

Constraints

- It cannot be used for reentrancy.

Events emit

- *RewardPaid(msg.sender, reward);*

Output

None

- ***exit***

Description

Used by the user to withdraw all tokens of his account.

Visibility

external

Input parameters

None

Constraints

None

Events emit

None

Output

None

- ***setStakingToken***

Description

Used to set staking token.

Visibility

external

Input parameters

- *address _stakingToken* — an address of the staking token;

Constraints

- Only owner or the controller contract can call it.
- *_stakingToken* cannot be zero address.

Events emit

None

Output

None

- ***setController***

Description

Used to set the controller contract.

Visibility

external

Input parameters

- *address _controller* — an address of the controller contract;

Constraints

- Only owner or the controller contract can call it.
- *_controller* cannot be zero address.

Events emit

None

Output

None

- ***notifyRewardAmount***

Description

Used to notify reward amount.

Visibility

external

Input parameters

- *uint256 reward* — reward amount;

Constraints

- Only owner or the controller contract can call it.
- Provided reward amount must be less than or equal to the contract balance.

Events emit

- *RewardAdded(reward);*

Output

None

- ***recoverERC20***

Description

Used to recover tokens.

Visibility

external

Input parameters

- *address tokenAddress* — an address of the token;
- *uint256 tokenAmount* — an amount of the token;

Constraints

- Only owner can call it.
- *tokenAddress* cannot be staking token address or rewards token address.

Events emit

- *Recovered(tokenAddress, tokenAmount);*

Output

None

- ***setRewardsDuration***

Description

Used to set rewards duration.

Visibility

external

Input parameters

- *uint256 _rewardsDuration* — rewards duration;

Constraints

- Only owner or the controller contract can call it.
- Previous rewards period must be complete before changing the duration for the new period.

Events emit

- *RewardsDurationUpdated(rewardsDuration);*

Output

None

pool-liquidity.sol

Description

YvsLiquidityPool is a staking pool contract for Uniswap liquidity pool tokens.

Imports

YvsLiquidityPool contract has 4 imports:

- *reentrancy-guard.sol* — from project files;
- *pausable.sol* — from project files;
- *erc20.sol* — from project files;
- *safe-math.sol* — from project files;

Inheritance

YvsLiquidityPool contract inherits *ReentrancyGuard* and *Pausable*.

Usings

YvsLiquidityPool contract use:

- *SafeMath* for *uint256*;
- *SafeERC20* for *IERC20*;

Modifiers

YvsLiquidityPool contract has 2 modifiers:

- *updateReward* — updates reward;
- *restricted* — checks if a caller is controller contract or owner;

Fields

YvsLiquidityPool contract has 14 fields:

- *address public controller* — an address of the controller contract;
- *IERC20 public rewardsToken* — rewards token;
- *IERC20 public stakingToken* — staking token;
- *uint256 public periodFinish* — a timestamp of the period finish;
- *uint256 public rewardRate* — reward rate;
- *uint256 public rewardsDuration* — reward period duration;

- *uint256 public lastUpdateTime* — a timestamp of the last update;
- *uint256 public rewardPerTokenStored* — reward per token;
- *mapping(address => uint256) public userRewardPerTokenPaid* — a mapping for reward per token;
- *mapping(address => uint256) public rewards* — a mapping for rewards;
- *uint256 private _totalSupply* — total supply;
- *uint256 private _totalLocked* — total locked tokens;
- *mapping(address => uint256) private _balances* — a mapping for balances;
- *mapping(address => uint256) private _locked* — a mapping for locked tokens;

Functions

YvsLiquidityPool has 19 functions:

- ***constructor***

  Description

  Initializes the contract.

  Visibility

  public

  Input parameters

  - *address _rewardsToken* — an address of the rewards token;
  - *address _stakingToken* — an address of the staking token;

  Constraints

  None

  Events emit

  None

  Output

  None

- ***totalSupply***

  Description

  Used to get total supply.

Visibility

external view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns total supply.

- ***totalLocked***

Description

Used to get total locked tokens.

Visibility

external view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns total locked tokens.

- ***balanceOf***

Description

Used to get balance of account.

Visibility

external view

Input parameters

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

Constraints

None

Events emit

None

Output

Returns balance.

- ***withdrawable***

Description

Used to get the number of tokens on the account that can be withdrawn.

Visibility

external view

Input parameters

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

Constraints

None

Events emit

None

Output

Returns the number of tokens.

- ***lastTimeRewardApplicable***

Description

Used to get the minimum value for *block.timestamp* and *periodFinish*.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns the minimum value for *block.timestamp* and *periodFinish*.

- ***rewardPerToken***

Description

Used to get reward per token.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns reward per token.

- ***earned***

Description

Used to calculate reward for account.

Visibility

public view

Input parameters

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

Constraints

None

Events emit

None

Output

Returns reward for account.

- ***getRewardForDuration***

Description

Used to calculate reward for duration.

Visibility

external view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns reward for duration.

- ***min***

Description

Used to get min from two *uint256*.

Visibility

public pure

Input parameters

- *uint256 a;*
- *uint256 b;*

Constraints

None

Events emit

None

Output

Returns min value.

- ***stake***

Description

Used to stake tokens. 50% of deposited liquidity is permanently locked.

Visibility

external

Input parameters

- *uint256 amount* — an amount of tokens;

Constraints

- It cannot be used for reentrancy.
- The contract must not be paused.
- *amount* should be greater than 0.

Events emit

- *Staked(msg.sender, amount);*

Output

None

- ***withdraw***

Description

Used to withdraw tokens.

Visibility

public

Input parameters

- *uint256 amount* — an amount of tokens;

Constraints

- It cannot be used for reentrancy.
- *amount* should be greater than 0.
- Cannot withdraw locked tokens.

Events emit

- *Withdrawn(msg.sender, amount);*

Output

None

- ***getReward***

Description

Used to withdraw reward.

Visibility

public

Input parameters

None

Constraints

- It cannot be used for reentrancy.

Events emit

- *RewardPaid(msg.sender, reward);*

Output

None

- ***exit***

Description

Used by the user to withdraw all tokens of his account.

Visibility

external

Input parameters

None

Constraints

None

Events emit

None

Output

None

- ***setStakingToken***

  Description

  Used to set staking token.

  Visibility

  external

  Input parameters

  - *address _stakingToken* — an address of the staking token;

  Constraints

  - Only owner or the controller contract can call it.
  - *_stakingToken* cannot be zero address.

  Events emit

  None

  Output

  None

- ***setController***

  Description

  Used to set the controller contract.

  Visibility

  external

  Input parameters

  - *address _controller* — an address of the controller contract;

  Constraints

  - Only owner or the controller contract can call it.
  - *_controller* cannot be zero address.

  Events emit

None

Output

None

- ***notifyRewardAmount***

Description

Used to notify reward amount.

Visibility

external

Input parameters

- *uint256 reward* — reward amount;

Constraints

- Only owner or the controller contract can call it.
- Provided reward amount must be less than or equal to the contract balance.

Events emit

- *RewardAdded(reward);*

Output

None

- ***recoverERC20***

Description

Used to recover tokens.

Visibility

external

Input parameters

- *address tokenAddress* — an address of the token;
- *uint256 tokenAmount* — an amount of the token;

Constraints

- Only owner can call it.
- *tokenAddress* cannot be staking token address or rewards token address.

Events emit

- *Recovered(tokenAddress, tokenAmount);*

Output

None

- ***setRewardsDuration***

Description

Used to set rewards duration.

Visibility

external

Input parameters

- *uint256 _rewardsDuration* — rewards duration;

Constraints

- Only owner or the controller contract can call it.
- Previous rewards period must be complete before changing the duration for the new period.

Events emit

- *RewardsDurationUpdated(rewardsDuration);*

Output

None

pool-staking.sol

Description

YvsStakingPool is a staking pool contract for native tokens.

Imports

YvsStakingPool contract has 4 imports:

- *reentrancy-guard.sol* — from project files;
- *pausable.sol* — from project files;
- *erc20.sol* — from project files;
- *safe-math.sol* — from project files;

Inheritance

YvsStakingPool contract inherits *ReentrancyGuard* and *Pausable*.

Usings

YvsStakingPool contract use:

- *SafeMath* for *uint256*;
- *SafeERC20* for *IERC20*;

Modifiers

YvsStakingPool contract has 2 modifiers:

- *updateReward* — updates reward;
- *restricted* — checks if a caller is controller contract or owner;

Fields

YvsStakingPool contract has 17 fields:

- *address public controller* — an address of the controller contract;
- *IERC20 public rewardsToken* — rewards token;
- *IERC20 public stakingToken* — staking token;
- *uint256 public periodFinish* — a timestamp of the period finish;
- *uint256 public rewardRate* — reward rate;
- *uint256 public rewardsDuration* — reward period duration;
- *uint256 public lastUpdateTime* — a timestamp of the last update;
- *uint256 public rewardPerTokenStored* — reward per token;
- *mapping(address => uint256) public userRewardPerTokenPaid* — a mapping for reward per token;
- *mapping(address => uint256) public rewards* — a mapping for rewards;
- *uint256 private _totalSupply* — total supply;
- *uint256 private _totalDeposited* — total deposited tokens;
- *mapping(address => uint256) private _balances* — a mapping for balances;
- *mapping(address => uint256) private _deposits* — a mapping for deposits;
- *mapping(address => uint256) private _periods* — a mapping for periods;
- *mapping(address => uint256) private _locks* — a mapping for locks;
- *uint256 private constant multiplierBase* — multiplier base;

Functions

YvsStakingPool has 22 functions:

- ***constructor***

  Description

  Initializes the contract.

  Visibility

  public

  Input parameters

  - *address _rewardsToken* — an address of the rewards token;
  - *address _stakingToken* — an address of the staking token;

  Constraints

  None

  Events emit

  None

  Output

  None

- ***totalSupply***

  Description

  Used to get total supply.

  Visibility

  external view

  Input parameters

  None

  Constraints

  None

Events emit

None

Output

Returns total supply.

- ***totalDeposited***

  Description

  Used to get total deposited tokens.

  Visibility

  external view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns total deposited tokens.

- ***balanceOf***

  Description

  Used to get balance of account.

  Visibility

  external view

  Input parameters

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

Constraints

None

Events emit

None

Output

Returns balance.

- ***depositOf***

  Description

  Used to get the number of tokens on the account that were deposited.

  Visibility

  external view

  Input parameters

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

  Constraints

  None

  Events emit

  None

  Output

  Returns the number of tokens.

- ***unlockedAt***

  Description

  Used to get a timestamp when tokens will be unlocked for an account.

  Visibility

  external view

Input parameters

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

Constraints

None

Events emit

None

Output

Returns a timestamp.

- ***lockedFor***

Description

Used to get the token lockout period for an account.

Visibility

external view

Input parameters

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

Constraints

None

Events emit

None

Output

Returns a period.

- ***lastTimeRewardApplicable***

Description

Used to get the minimum value for *block.timestamp* and *periodFinish*.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns the minimum value for *block.timestamp* and *periodFinish*.

- ***rewardPerToken***

Description

Used to get reward per token.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns reward per token.

- ***earned***

Description

Used to calculate reward for account.

Visibility

public view

Input parameters

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

Constraints

None

Events emit

None

Output

Returns reward for account.

- ***getRewardForDuration***

Description

Used to calculate reward for duration.

Visibility

external view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns reward for duration.

- ***min***

  Description

  Used to get min from two *uint256*.

  Visibility

  public pure

  Input parameters

  - *uint256 a*;
  - *uint256 b*;

  Constraints

  None

  Events emit

  None

  Output

  Returns min value.

- ***stake***

  Description

  Used to stake tokens.

  Visibility

  external

  Input parameters

  - *uint256 amount* — an amount of tokens;
  - *uint256 lockPeriod* — lock period;

  Constraints

  - It cannot be used for reentrancy.
  - The contract must not be paused.

  - *amount* should be greater than 0.
  - The lock period must not be less than 2 weeks.
  - If the user already has a deposit, the lock period should be greater than or equal to the previous.

Events emit

  - *Staked(msg.sender, actualReceived, lockPeriod);*

Output

None

- ***extend***

Description

Used to extend the lock period.

Visibility

external

Input parameters

  - *uint256 lockPeriod* — lock period;

Constraints

  - It cannot be used for reentrancy.
  - The user must have a deposit.
  - The lock period should be greater than the previous.

Events emit

  - *Extended(msg.sender, _periods[msg.sender], lockPeriod);*

Output

None

- ***withdraw***

Description

Used to withdraw tokens.

Visibility

public

Input parameters

- *uint256 amount* — an amount of tokens;

Constraints

- It cannot be used for reentrancy.
- *amount* should be greater than 0.
- The user must have a deposit.
- The lock period should be finished.

Events emit

- *Withdrawn(msg.sender, amount);*

Output

None

- ***getReward***

Description

Used to withdraw reward.

Visibility

public

Input parameters

None

Constraints

- It cannot be used for reentrancy.

Events emit

- *RewardPaid(msg.sender, reward);*

Output

None

- ***exit***

Description

Used by the user to withdraw all tokens of his account.

Visibility

external

Input parameters

None

Constraints

None

Events emit

None

Output

None

- ***setStakingToken***

Description

Used to set staking token.

Visibility

external

Input parameters

- *address _stakingToken* — an address of the staking token;

Constraints

- Only owner or the controller contract can call it.
- *_stakingToken* cannot be zero address.

Events emit

None

Output

None

- ***setController***

Description

Used to set the controller contract.

Visibility

external

Input parameters

  - *address _controller* — an address of the controller contract;

Constraints

  - Only owner or the controller contract can call it.
  - *_controller* cannot be zero address.

Events emit

None

Output

None

- ***notifyRewardAmount***

Description

Used to notify reward amount.

Visibility

external

Input parameters

  - *uint256 reward* — reward amount;

Constraints

  - Only owner or the controller contract can call it.
  - Provided reward amount must be less than or equal to the contract balance.

Events emit

- *RewardAdded(reward);*

Output

None

- ***recoverERC20***

  Description

  Used to recover tokens.

  Visibility

  external

  Input parameters

  - *address tokenAddress* — an address of the token;
  - *uint256 tokenAmount* — an amount of the token;

  Constraints

  - Only owner can call it.
  - *tokenAddress* cannot be staking token address or rewards token address.

  Events emit

  - *Recovered(tokenAddress, tokenAmount);*

  Output

  None

- ***setRewardsDuration***

  Description

  Used to set rewards duration.

  Visibility

  external

  Input parameters

- *uint256 _rewardsDuration* — rewards duration;

Constraints

- Only owner or the controller contract can call it.
- Previous rewards period must be complete before changing the duration for the new period.

Events emit

- *RewardsDurationUpdated(rewardsDuration);*

Output

None

tax-collector.sol

Description

YvsTaxCollector is a contract that collects and distributes the tax amount.

Imports

YvsTaxCollector contract has 4 imports:

- *burnable.sol* — from project files;
- *ownable.sol* — from project files;
- *erc20.sol* — from project files;
- *safe-math.sol* — from project files;

Inheritance

YvsTaxCollector contract inherits *Ownable*.

Usings

YvsTaxCollector contract use:

- *SafeMath* for *uint256*;
- *SafeERC20* for *IERC20*;

Fields

YvsTaxCollector contract has 9 fields:

- *IERC20 public token* — the token for which tax is collected;

- *address public controller* — an address of the controller contract;
- *address public treasury* — an address of the treasury;
- *uint256 public controllerRate* — controller rate;
- *uint256 public treasuryRate* — treasury rate;
- *uint256 public baseRate* — base rate;
- *uint256 public lastDistribution* — a timestamp of the last distribution;
- *uint256 public lastBurn* — a timestamp of the last burn;
- *uint256 public burnable* — an amount of tokens to be burned;

Functions

YvsTaxCollector has 5 functions:

- ***constructor***

  Description

  Initializes the contract.

  Visibility

  public

  Input parameters

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

  Constraints

  None

  Events emit

  None

  Output

  None

- ***set_controller***

  Description

  Used to set the controller contract.

  Visibility

external

Input parameters

- *address _controller* — an address of the controller;

Constraints

- Only owner can call it.
- The controller address must not be zero.

Events emit

None

Output

None

- ***distribute***

Description

Used to distribute tokens.

Visibility

external

Input parameters

None

Constraints

None

Events emit

- *Distributed(_controller, _treasury);*

Output

None

- ***burn***

Description

Used to burn tokens.

Visibility

external

Input parameters

None

Constraints

- Burnable tokens amount must be greater than 0.

Events emit

- *Burned(burnable);*

Output

None

- ***salvage***

Description

Salvages unrelated tokens.

Visibility

external

Input parameters

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

Constraints

- Only owner can call it.
- The main token can not be salvage.

Events emit

- *Salvaged(_token, balance);*

Output

None

token-timelock.sol

Description

YvsTokenTimelock is 1 year locking contract for Uniswap LP tokens.

Imports

YvsTokenTimelock contract has 3 imports:

- *erc20.sol* — from project files;
- *safe-math.sol* — from project files;
- *ownable.sol* — from project files;

Inheritance

YvsTokenTimelock contract inherits *Ownable*.

Usings

YvsTokenTimelock contract use:

- *SafeMath* for *uint256*;
- *SafeERC20* for *IERC20*;

Modifiers

YvsStakingPool contract has 1 modifier:

- *restricted* — checks if a caller is presale contract or owner;

Fields

YvsTokenTimelock contract has 9 fields:

- *IERC20 public token* — basic token;
- *address private _presale* — an address of the presale contract;
- *address private _beneficiary* — an address of the beneficiary;
- *uint256 private _releaseTime* — a timestamp when the tokens are released;
- *uint256 private _minReleaseTime* — minimum release time;

Functions

YvsTokenTimelock has 7 functions:

- ***constructor***

Description

Initializes the contract.

Visibility

public

Input parameters

- *address beneficiary_* — an address of the beneficiary;
- *uint256 releaseTime_* — a timestamp when token release is enabled;

Constraints

- *releaseTime_* should be greater than *block.timestamp*.
- *releaseTime_* should be greater than minimum.

Events emit

None

Output

None

- ***token***

Description

Used to get token.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns token.

- ***beneficiary***

Description

Used to get beneficiary.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns beneficiary.

- ***releaseTime***

Description

Used to get a timestamp when the tokens are released.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns timestamp.

- ***set_token***

Description

Sets the token held by the timelock.

Visibility

public

Input parameters

  - *address token_* — an address of the token;

Constraints

  - Only owner and the presale contract can call it.

Events emit

None

Output

None

- ***set_presale***

Description

Sets the presale contract for the timelock.

Visibility

public

Input parameters

  - *address presale_* — an address of the presale contract;

Constraints

- Only owner and the presale contract can call it.

Events emit

None

Output

None

- ***release***

Description

Transfers tokens held by timelock to beneficiary.

Visibility

public

Input parameters

None

Constraints

- Token release time has already come.
- The contract must have tokens to release.

Events emit

None

Output

None

vault.sol

Description

YvsVault is vault contract.

Imports

YvsVault contract has 6 imports:

- *erc20.sol* — from project files;
- *safe-math.sol* — from project files;
- *reentrancy-guard.sol* — from project files;
- *burnable.sol* — from project files;
- *strategy.sol* — from project files;
- *timelock.sol* — from project files;

Inheritance

YvsVault contract inherits *ERC20* and *ReentrancyGuard*.

Usings

YvsVault contract use:

- *SafeERC20* for *IERC20*;
- *Address* for *address*;
- *SafeMath* for *uint256*;

Modifiers

YvsStakingPool contract has 3 modifier:

- *restricted* — checks if a caller is timelock, governance or *tx.origin*;
- *isTimelock* — chacks if a caller is timelock;
- *isGovernance* — chacks if a caller is governance;

Fields

YvsVault contract has 28 fields:

- *IERC20 internal token* — the underlying token;
- *IERC20 internal yvs* — the yvs token;
- *address public underlying* — an address of the underlying token;
- *address public controller* — an address of the controller contract;
- *uint256 public min* — the minimum amount for investment;
- *uint256 public constant max* — the maximum amount for investment;
- *uint256 public burnFee* — burn fee;
- *uint256 public constant burnFeeMax* — maximum burn fee;
- *uint256 public constant burnFeeMin* — minimum burn fee;
- *uint256 public constant burnFeeBase* — base burn fee;
- *uint256 public withdrawalFee* — withdrawal fee;
- *uint256 public constant withdrawalFeeMax* — maximum withdrawal fee;
- *uint256 public constant withdrawalFeeBase* — base withdrawal fee;
- *uint256 public minDepositPeriod* — minimum deposit period;
- *bool public isActive* — indicates if strategy is active;
- *address public governance* — an address of the governance;

- *address public treasury* — an address of the treasury;
- *address public timelock* — an address of the timelock;
- *address public strategy* — an address of the strategy;
- *uint256 public constant minTimelockInterval* — minimum timelock interval;
- *mapping(address => uint256) public depositBlocks* — a mapping for deposit blocks;
- *mapping(address => uint256) public deposits* — a mapping for deposits;
- *mapping(address => uint256) public issued* — a mapping for issued;
- *mapping(address => uint256) public tiers* — a mapping for tiers;
- *uint256[] public multiplierCosts* — costs multipliers;
- *uint256 internal constant tierMultiplier* — tier multiplier;
- *uint256 internal constant tierBase* — tier base;
- *uint256 public totalDeposited* — total deposited;

Functions

YvsVault has 28 functions:

- ***constructor***

  Description

  Initializes the contract.

  Visibility

  public

  Input parameters

  - *address _underlying* — an address of the underlying token;
  - *address _yvs* — an address of the yvs token;
  - *address _governance* — an address of the governance;
  - *address _treasury* — an address of the treasury;
  - *address _timelock* — an address of the timelock;

  Constraints

  - *_underlying* cannot be equal to *_yvs*.
  - The timelock contract delay must be greater than or equal to *minTimelockInterval*.

  Events emit

  None

  Output

None

- ***balance***

Description

Used to get the total underlying token balance.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns the total underlying token balance.

- ***setActive***

Description

Sets whether deposits are accepted by the vault.

Visibility

external

Input parameters

  - *bool _isActive* — true or false value;

Constraints

  - Only governance can call it.

Events emit

None

Output

None

- ***setMin***

Description

Sets the minimum percentage of tokens that can be deposited to earn.

Visibility

external

Input parameters

  - *uint256 _min* — the minimum percentage of tokens;

Constraints

  - Only governance can call it.
  - *_min* should be less than or equal to *max*.

Events emit

None

Output

None

- ***setGovernance***

Description

Sets a new governance address.

Visibility

external

Input parameters

  - *address _governance* — a new governance address;

Constraints

   - Only governance can call it.

Events emit

None

Output

None

- ***setTreasury***

Description

Sets a new treasury address.

Visibility

external

Input parameters

   - *address _treasury* —

Constraints

   - Only governance can call it.

Events emit

None

Output

None

- ***setTimelock***

Description

Sets the timelock address.

Visibility

external

Input parameters

- *address _timelock* — an address of the timelock;

Constraints

- Only timelock can call it.
- The timelock contract delay must be greater than or equal to *minTimelockInterval*.

Events emit

None

Output

None

- ***setStrategy***

Description

Sets a new strategy address.

Visibility

external

Input parameters

- *address _strategy* — an address of the new strategy;

Constraints

- Only timelock can call it.
- The new strategy should support underlying token.

Events emit

None

Output

None

- ***setController***

Description

Sets the controller address.

Visibility

external

Input parameters

- *address _controller* — an address of the controller;

Constraints

- Only governance can call it.
- Can only be set once after deployment.

Events emit

None

Output

None

- ***setBurnFee***

Description

Sets the burn fee for multipliers.

Visibility

public

Input parameters

- *uint256 _burnFee* — burn fee;

Constraints

- Only timelock can call it.
- *_burnFee* should be less than or equal to *burnFeeMax*.
- *_burnFee* should be greater than or equal to *burnFeeMin*.

Events emit

None

Output

None

- ***setWithdrawalFee***

  Description

  Sets withdrawal fee for the vault.

  Visibility

  external

  Input parameters

  - *uint256 _withdrawalFee* — withdrawal fee;

  Constraints

  - Only timelock can call it.
  - *_withdrawalFee* should be less than or equal to *withdrawalFeeMax*.

  Events emit

  None

  Output

  None

- ***addMultiplier***

  Description

  Adds a new multplier with the selected cost.

  Visibility

  public

  Input parameters

  - *uint256 _cost* — a cost;

  Constraints

  - Only timelock can call it.

  Events emit

  None

Output

Returns an index of the new multiplier.

- ***setMultiplier***

Description

Sets a new cost for multiplier.

Visibility

public

Input parameters

- *uint256 index* — an index;
- *uint256 _cost* — a cost;

Constraints

- Only timelock can call it.

Events emit

None

Output

None

- ***available***

Description

Used to get how much of the underlying asset can be deposited.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns how much of the underlying asset can be deposited.

- ***earn***

Description

Deposits collected underlying assets into the strategy and starts earning.

Visibility

public

Input parameters

None

Constraints

- The vault should be active.
- The strategy must be set.

Events emit

None

Output

None

- ***deposit***

Description

Deposits underlying assets from the user into the vault contract.

Visibility

public

Input parameters

- *uint256 _amount* an amount of tokens;

Constraints

- It cannot be used for reentrancy.
- *msg.sender* cannot be a contract.
- The vault should be active.
- The strategy must be set.

Events emit

- *Deposit(msg.sender, _amount);*
- *SharesIssued(msg.sender, shares);*

Output

None

- ***depositAll***

Description

Deposits all the funds of the user.

Visibility

external

Input parameters

None

Constraints

None

Events emit

None

Output

None

- ***withdraw***

Description

Used to withdraw tokens.

Visibility

public

Input parameters

- *uint256 _amount* — an amount of tokens;

Constraints

- It cannot be used for reentrancy.
- *msg.sender* cannot be a contract.
- The deposit period must be ended.
- _*amount* should be greater than 0.
- _*amount* should be less than or equal to user's deposit.
- The user must have deposit.
- The user must have enough shares.

Events emit

- *Withdraw(msg.sender, _amount);*
- *SharesPurged(msg.sender, r);*
- *ClaimRewards(msg.sender, userRewards);*

Output

None

- ***withdrawAll***

Description

Withdraws all underlying assets belonging to the user.

Visibility

external

Input parameters

None

Constraints

None

Events emit

None

Output

None

- ***pendingRewards***

  Description

  Calculates the amount of rewards the user has gained.

  Visibility

  external view

  Input parameters

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

  Constraints

  None

  Events emit

  None

  Output

  Returns the amount of rewards the user has gained.

- ***purchaseMultiplier***

  Description

  Used to purchase a multiplier tier for the user.

  Visibility

  external

  Input parameters

  - *uint256 _tiers* — the number of tiers;

  Constraints

- The vault should be active.
- The strategy must be set.
- *_tiers* should be greater than 0.
- The new tier shuld be less than or equal to multipliers length.
- The user must have enough YVS tokens to purchase.

Events emit

- *MultiplierPurchased(msg.sender, _tiers, totalCost);*

Output

Returns a new multiplier tier.

- ***distribute***

Description

Distributes the YVS tokens collected by the multiplier purchases.

Visibility

external

Input parameters

None

Constraints

- Only governance, controller or not a contract can call it.

Events emit

None

Output

None

- ***salvage***

Description

Used to salvage any non-underlying assets to treasury.

Visibility

external

Input parameters

- *address reserve* — an address of the token;
- *uint256 amount* — an amount of tokens;

Constraints

- Only governance can call it.
- *reserve* must not be underlying token.
- *reserve* must not be YVS token.

Events emit

None

Output

None

- ***setMultiplier***

Description

Sets a new multiplier to any account by governance.

Visibility

external

Input parameters

- *address account* — an address of the account;
- *uint256 multiplier* — a multiplier;

Constraints

- Only governance can call it.
- The multiplier must be less than or equal to multipliers length.

Events emit

None

Output

None

- ***getMultiplier***

  Description

  Used to get the current multiplier tier for the user.

  Visibility

  external view

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns the current multiplier tier for the user.

- ***getNextMultiplierCost***

  Description

  Used to get the next multiplier tier cost for the user.

  Visibility

  external view

  Input parameters

  None

  Constraints

  - The multiplier must be less than multipliers length.

  Events emit

  None

Output

Returns the next multiplier tier cost for the user.

- ***getCountOfMultipliers***

Description

Used to get the total number of multipliers.

Visibility

external view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns the total number of multipliers.

- ***getRatio***

Description

Used to get the current ratio between earned assets and deposited assets.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns the current ratio between earned assets and deposited assets.

strategy-base.sol

Description

YvsStrategyBase is abstract strategy base contract.

Imports

YvsStrategyBase contract has 5 imports:

- *erc20.sol* — from project files;
- *safe-math.sol* — from project files;
- *vault.sol* — from project files;
- *timelock.sol* — from project files;
- *uniswap-v2.sol* — from project files;

Usings

YvsStrategyBase contract use:

- *SafeERC20* for *IERC20*;
- *Address* for *address*;
- *SafeMath* for *uint256*;

Modifiers

YvsStakingPool contract has 3 modifier:

- *restricted* — checks if a caller is timelock, governance or *tx.origin*;
- *isTimelock* — chacks if a caller is timelock;
- *isGovernance* — chacks if a caller is governance;

Fields

YvsStrategyBase contract has 14 fields:

- *uint256 public strategyFee* — strategy fee;
- *uint256 public constant strategyFeeMax* — strategy max fee;
- *uint256 public constant strategyFeeBase* — strategy fee base;

- *address public underlying* — an address of the underlying token;
- *address public constant weth* — an address of weth;
- *address public constant wbtc* — an address of wbtc;
- *address public treasury* — an address of the treasury;
- *address public governance* — an address of the governance;
- *address public strategist* — an address of the strategist;
- *address public timelock* — an address of the timelock;
- *address public vault* — an address of the vault;
- *address public controller* — an address of the controller;
- *uint256 public constant minTimelockInterval* — minimum timelock interval;
- *address public univ2Router2* — an address of *UniswapV2Router02*;

Functions

YvsStrategyBase has 22 functions:

- ***constructor***

  Description

  Initializes the contract.

  Visibility

  public

  Input parameters

  - *address _underlying* — an address of the underlying token;
  - *address _governance* — an address of the governance;
  - *address _strategist* — an address of the strategist;
  - *address _timelock* — an address of the timelock;
  - *address _vault* — an address of the vault;

  Constraints

  - The address of the underlying token cannot be zero.
  - The address of the governance cannot be zero.
  - The address of the strategist cannot be zero.
  - The address of the timelock cannot be zero.
  - The address of the vault cannot be zero.
  - The timelock contract delay must be greater than or equal to *minTimelockInterval*.

  Events emit

  None

Output

None

- ***balanceOfUnderlying***

Description

Used to get a balance of the underlying token.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns a balance of the underlying token.

- ***balanceOfPool***

Description

Used to get the pool balance.

Visibility

public virtual view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns the pool balance.

- ***balanceOf***

Description

Used to get a balance.

Visibility

public view

Input parameters

None

Constraints

None

Events emit

None

Output

Returns a balance.

- ***getName***

Description

Used to get a name.

Visibility

external virtual pure

Input parameters

None

Constraints

None

Events emit

None

Output

Returns a name.

- ***setStrategyFee***

Description

Sets the strategy fee.

Visibility

external

Input parameters

  - *uint256 _strategyFee* — strategy fee;

Constraints

  - Only timelock can call it.
  - *_strategyFee* should be less than or equal to *strategyFeeMax*.

Events emit

None

Output

None

- ***setStrategist***

Description

Sets the strategist.

Visibility

external

Input parameters

- *address _strategist* — an address of the strategist;

Constraints

- Only governance can call it.

Events emit

None

Output

None

- ***setGovernance***

Description

Sets the governance.

Visibility

external

Input parameters

- *address _governance* — an address of the governance;

Constraints

- Only governance can call it.

Events emit

None

Output

None

- ***setTreasury***

Description

Sets the treasury.

Visibility

external

Input parameters

- *address _treasury* — an address of the treasury;

Constraints

- Only governance can call it.

Events emit

None

Output

None

- ***setTimelock***

Description

Sets the timelock.

Visibility

external

Input parameters

- *address _timelock* — an address of the timelock;

Constraints

- Only timelock can call it.
- The timelock contract delay must be greater than or equal to *minTimelockInterval*.

Events emit

None

Output

None

- ***setVault***

  Description

  Sets the vault.

  Visibility

  external

  Input parameters

  - *address _vault* — an address of the vault;

  Constraints

  - Only timelock can call it.
  - The new vault should support the underlying token.

  Events emit

  None

  Output

  None

- ***setController***

  Description

  Sets the controller.

  Visibility

  external

  Input parameters

  - *address _controller* — an address of the controller;

  Constraints

  - Only governance can call it.
  - The controller address cannot be zero.

  Events emit

None

Output

None

- ***deposit***

Description

Used to deposit.

Visibility

public virtual

Input parameters

None

Constraints

None

Events emit

None

Output

None

- ***salvage***

Description

Used to salvage non-underlying assets.

Visibility

external

Input parameters

- *IERC20 _asset* — an asset;

Constraints

- Only governance can call it.
- *_asset* cannot be the underlying token.

Events emit

None

Output

Returns a balance of the asset.

- ***withdraw***

Description

Used to withdraw partial funds.

Visibility

external

Input parameters

- *uint256 _amount* — an amount of tokens;

Constraints

- *vault* should be set.
- Only vault can call it.

Events emit

None

Output

None

- ***withdrawAll***

Description

Used to withdraw all funds.

Visibility

external

Input parameters

None

Constraints

- Only governance, strategist, controller or not a contract can call it.
- *vault* should be set.

Events emit

None

Output

Returns the underlying token balance of this contract.

- ***_withdrawAll***

Description

Used to withdraw all funds.

Visibility

internal

Input parameters

None

Constraints

None

Events emit

None

Output

None

- ***_withdrawSome***

Description

Used to withdraw.

Visibility

internal virtual

Input parameters

- *uint256 _amount* — an amount of tokens;

Constraints

None

Events emit

None

Output

Returns amount of tokens to withdraw.

- ***harvest***

Description

Used to harvest.

Visibility

public virtual

Input parameters

None

Constraints

None

Events emit

None

Output

None

- ***_distributeAndDeposit***

Description

Used to deposit tokens with a fee to strategist.

Visibility

internal

Input parameters

None

Constraints

None

Events emit

None

Output

None

- ***execute***

Description

Emergency function

Visibility

public payable

Input parameters

- *address _target* — an address;
- *bytes memory _data* — a data;

Constraints

- Only timelock can call it.
- *_target* cannot be zero.

Events emit

None

Output

None

- ***_swapUniswap***

Description

Used to swap tokens with Uniswap.

Visibility

internal

Input parameters

- *address _from* — an address from;
- *address _to* — an address to;
- *uint256 _amount* — an amount;

Constraints

- *_to* cannot be zero address.

Events emit

None

Output

None

strategy-curve-base.sol

Description

YvsStrategyCurveBase is abstract strategy curve base contract.

Imports

YvsStrategyCurveBase contract has 2 imports:

- *curve.sol* — from project files;
- *strategy-base.sol* — from project files;

Inheritance

YvsStrategyCurveBase contract inherits *YvsStrategyBase*.

Fields

YvsStrategyCurveBase contract has 14 fields:

- *address public curve* — an address of curve;
- *address public gauge* — an address of gauge;
- *address public mintr* — an address of mintr;
- *address public dai* — an address of dai;
- *address public usdc* — an address of usdc;
- *address public usdt* — an address of usdt;
- *address public susd* — an address of susd;
- *address public renbtc* — an address of renbtc;
- *address public crv* — an address of crv;
- *address public keep* — an address of keep;
- *address public keep_rewards* — an address of keep_rewards;
- *address public snx* — an address of snx;
- *uint256 public keepCRV* — an amount of CRV tokens to keep;
- *uint256 public keepCRVMax* — maximum amount of CRV tokens to keep;

Functions

YvsStrategyCurveBase has 7 functions:

- ***constructor***

  Description

  Initializes the contract.

  Visibility

  public

  Input parameters

  - *address _curve* — an address of curve;
  - *address _gauge* — an address of gauge;
  - *address _underlying* — an address of the underlying token;
  - *address _governance* — an address of the governance;
  - *address _strategist* — an address of the strategist;
  - *address _timelock* — an address of the timelock;
  - *address _vault* — an address of the vault;

  Constraints

  None

Events emit

None

Output

None

- ***balanceOfPool***

Description

Used to get balance of pool.

Visibility

public override

Input parameters

None

Constraints

None

Events emit

None

Output

Returns balance of pool.

- ***getHarvestable***

Description

Used to get harvestable tokens amount.

Visibility

external

Input parameters

None

Constraints

None

Events emit

None

Output

Returns harvestable tokens amount.

- ***getMostPremium***

Description

Used to get most premium token.

Visibility

public virtual view

Input parameters

None

Constraints

None

Events emit

None

Output

Return an address and a position.

- ***setKeepCRV***

Description

Used to set an amount of CRV tokens to keep.

Visibility

external

Input parameters

- *uint256 _keepCRV* — an amount of CRV tokens to keep;

Constraints

- Only governance can call it.

Events emit

None

Output

None

- ***deposit***

Description

Used to deposit.

Visibility

public override

Input parameters

None

Constraints

None

Events emit

None

Output

None

- ***_withdrawSome***

Description

Used to withdraw.

Visibility

internal override

Input parameters

- *uint256 _amount* — an amount of tokens;

Constraints

None

Events emit

None

Output

Returns amount of tokens to withdraw.

strategy-curve-rencrv-v1.sol

Description

YvsStrategyCurveRenCRV is strategy curve contract.

Imports

YvsStrategyCurveRenCRV contract has 6 imports:

- *erc20.sol* — from project files;
- *safe-math.sol* — from project files;
- *vault.sol* — from project files;
- *uniswap-v2.sol* — from project files;
- *curve-rencrv.sol* — from project files;
- *strategy-curve-base.sol* — from project files;

Inheritance

YvsStrategyCurveRenCRV contract inherits *YvsStrategyCurveBase*.

Fields

YvsStrategyCurveRenCRV contract has 3 fields:

- *address public ren_pool* — an address of the pool;
- *address public ren_gauge* — an address of the gauge;

- *address public ren_crv* — an address of the underlying token;

Functions

YvsStrategyCurveRenCRV has 4 functions:

- ***constructor***

  Description

  Initializes the contract.

  Visibility

  public

  Input parameters

  - *address _governance* — an address of the governance;
  - *address _strategist* — an address of the strategist;
  - *address _timelock* — an address of the timelock;
  - *address _vault* — an address of the vault;

  Constraints

  None

  Events emit

  None

  Output

  None

- ***getMostPremium***

  Description

  Used to get most premium token.

  Visibility

  public override view

  Input parameters

  None

Constraints

None

Events emit

None

Output

Return an address and a position.

- ***getName***

Description

Used to get a name.

Visibility

external override pure

Input parameters

None

Constraints

None

Events emit

None

Output

Returns a name.

- ***harvest***

Description

Used to harvest.

Visibility

public virtual

Input parameters

None

Constraints

None

Events emit

- *Harvested(to, _to);*

Output

None

strategy-curve-scrv-v1.sol

Description

YvsStrategyCurveSCRV is strategy curve contract.

Imports

YvsStrategyCurveSCRV contract has 6 imports:

- *erc20.sol* — from project files;
- *safe-math.sol* — from project files;
- *vault.sol* — from project files;
- *uniswap-v2.sol* — from project files;
- *curve-rencrv.sol* — from project files;
- *strategy-curve-base.sol* — from project files;

Inheritance

YvsStrategyCurveSCRV contract inherits *YvsStrategyCurveBase*.

Fields

YvsStrategyCurveSCRV contract has 3 fields:

- *address public susdv2_pool* — an address of the pool;
- *address public susdv2_gauge* — an address of the gauge;
- *address public scrv* — an address of the underlying token;

Functions

YvsStrategyCurveSCRV has 4 functions:

- ***constructor***

  Description

  Initializes the contract.

  Visibility

  public

  Input parameters

  - *address _governance* — an address of the governance;
  - *address _strategist* — an address of the strategist;
  - *address _timelock* — an address of the timelock;
  - *address _vault* — an address of the vault;

  Constraints

  None

  Events emit

  None

  Output

  None

- ***getMostPremium***

  Description

  Used to get most premium token.

  Visibility

  public override view

  Input parameters

  None

  Constraints

  None

Events emit

None

Output

Return an address and a position.

- ***getName***

Description

Used to get a name.

Visibility

external override pure

Input parameters

None

Constraints

None

Events emit

None

Output

Returns a name.

- ***harvest***

Description

Used to harvest.

Visibility

public virtual

Input parameters

None

Constraints

None

Events emit

- *Harvested(to, _to);*

Output

None

strategy-curve-tbtccrv-v1.sol

Description

YvsStrategyCurveTBTC is strategy curve contract.

Imports

YvsStrategyCurveTBTC contract has 6 imports:

- *erc20.sol* — from project files;
- *safe-math.sol* — from project files;
- *vault.sol* — from project files;
- *uniswap-v2.sol* — from project files;
- *curve-rencrv.sol* — from project files;
- *strategy-curve-base.sol* — from project files;

Inheritance

YvsStrategyCurveTBTC contract inherits *YvsStrategyCurveBase*.

Fields

YvsStrategyCurveTBTC contract has 3 fields:

- *address public tbtc_pool* — an address of the pool;
- *address public tbtc_gauge* — an address of the gauge;
- *address public tbtc_crv* — an address of the underlying token;

Functions

YvsStrategyCurveTBTC has 4 functions:

- ***constructor***

Description

Initializes the contract.

Visibility

public

Input parameters

- *address _governance* — an address of the governance;
- *address _strategist* — an address of the strategist;
- *address _timelock* — an address of the timelock;
- *address _vault* — an address of the vault;

Constraints

None

Events emit

None

Output

None

- ***getMostPremium***

Description

Used to get most premium token.

Visibility

public override view

Input parameters

None

Constraints

None

Events emit

None

Output

Return an address and a position.

- ***getName***

Description

Used to get a name.

Visibility

external override pure

Input parameters

None

Constraints

None

Events emit

None

Output

Returns a name.

- ***harvest***

Description

Used to harvest.

Visibility

public virtual

Input parameters

None

Constraints

None

Events emit

- *Harvested(to, _to);*

Output

None

strategy-curve-usdncrv-v1.sol

Description

YvsStrategyCurveUSDN is strategy curve contract.

Imports

YvsStrategyCurveUSDN contract has 6 imports:

- *erc20.sol* — from project files;
- *safe-math.sol* — from project files;
- *vault.sol* — from project files;
- *uniswap-v2.sol* — from project files;
- *curve-rencrv.sol* — from project files;
- *strategy-curve-base.sol* — from project files;

Inheritance

YvsStrategyCurveUSDN contract inherits *YvsStrategyCurveBase*.

Fields

YvsStrategyCurveUSDN contract has 3 fields:

- *address public usdn_pool* — an address of the pool;
- *address public usdn_gauge* — an address of the gauge;
- *address public usdn_crv* — an address of the underlying token;

Functions

YvsStrategyCurveUSDN has 4 functions:

- ***constructor***

  Description

  Initializes the contract.

Visibility

public

Input parameters

- *address _governance* — an address of the governance;
- *address _strategist* — an address of the strategist;
- *address _timelock* — an address of the timelock;
- *address _vault* — an address of the vault;

Constraints

None

Events emit

None

Output

None

- ***getMostPremium***

Description

Used to get most premium token.

Visibility

public override view

Input parameters

None

Constraints

None

Events emit

None

Output

Return an address and a position.

- ***getName***

  Description

  Used to get a name.

  Visibility

  external override pure

  Input parameters

  None

  Constraints

  None

  Events emit

  None

  Output

  Returns a name.

- ***harvest***

  Description

  Used to harvest.

  Visibility

  public virtual

  Input parameters

  None

  Constraints

  None

  Events emit

  - *Harvested(to, _to);*

Output

None

Audit overview

■■■■ Critical

1. The *refund* function of the *YvsPresale* contract has a re-entry vulnerability.

   Fixed during second audit. The function has been reformed to follow the checks-effects-interaction policy to prevent re-entry.

2. The *setMultiplier* (line 340) function of the *YvsVault* contract allows to change user tier.

   Fixed during second audit. The function has been completely removed.

■■■ High

No high issues were found.

■■ Medium

No medium issues were found.

■ Low

1. Both *YvsPresale* contract *enter* (lines 209, 259) functions have the same code for calculating daily bonuses, which can be reused as a separate function.

   This code has been moved into separate function to calculate daily bonus.

2. The *_transfer* function of *ERC20* contract has code duplicates.

   This function has been refactored to remove duplicated code.

3. The *pendingRewards* function of the *YvsVault* contract does not have a default return statement.

   A default return value has been added.

■ Lowest / Code style / Best Practice

No lowest severity issues were found.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, high-level description of functionality was presented in As-is overview section of the report.

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

Security engineers found **2** critical and **3** low severity issues during the audit.

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

Category	Check Item	Comments
Code review	▪ Functionality Checks	▪ Both *YvsPresale* contract *enter* (lines 209, 259) functions have the same code for calculating daily bonuses, which can be reused as a separate function. ▪ The *_transfer* function of *ERC20* contract has code duplicates. ▪ The *pendingRewards* function of the *YvsVault* contract does not have a default return statement.
	▪ Reentrancy	▪ The refund function of the YvsPresale contract has a re-entry vulnerability.
Functional review	▪ User Balances manipulation	▪ The setMultiplier (line 340) function of the YvsVault contract allows to change user tier.

Disclaimers

Hacken Disclaimer

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

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

Technical Disclaimer

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