SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: ACryptos Protocol
Date: November 16th, 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 ACryptos Protocol.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Token, Governance, TimeLock, Defi, Strategies
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/acryptos/acryptos-protocol/
Commit	6B670C6BB3E8068EA3310506B4450535D6E8E6D1
Timeline	11 MARCH 2021 – 16 NOVEMBER 2021
Changelog	18 MARCH 2021- INITIAL AUDIT 16 NOVEMBER 2021 – SECOND REVIEW

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
Audit overview 56
Conclusion 57
Check the updates (11/16/2021) 58

Introduction

Hacken OÜ (Consultant) was contracted by ACryptos Protocol (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 March 11th, 2021 - March 18th, 2021.

Second review conducted on November 16th, 2021.

Scope

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

```
Repository: https://github.com/acryptos/acryptos-protocol/
Commit: 6B670C6BB3E8068EA3310506B4450535D6E8E6D1
File:
      /strategies/StrategyACryptoSVenusLeverageV2.sol
      /strategies/StrategyACryptoSVenusLeverageBnb.sol
      /strategies/StrategyACryptoSVenusLeverageXvs.sol
      /strategies/StrategyACryptoS0V3.sol
      /strategies/StrategyACryptoSCakeBnbToken.sol
      /strategies/StrategyACryptoSCakeBnbCake.sol
      /strategies/StrategyACryptoSCakeV2b.sol
      /strategies/StrategyACryptoSVenusVAI.sol
      /farms/MasterChef.sol
      /farms/MasterChefV2.sol
      /governance/Duplicate/acsACS_ACSI.sol
      /governance/acsACS.sol
      /controllers/Unchanged/Controller.sol
      /governance/Unchanged/Timelock6H.sol
      /governance/ACS.sol
      /governance/Duplicate/ACS_ACSI.sol
      /vaults/ACryptoSVaultBnb.sol
      /vaults/ACryptoSVault0V2_ACSI.sol
      /vaults/ACryptoSVault.sol
      /vaults/ACryptoSVault0.sol
New files:
      https://github.com/acryptos/acryptos-
protocol/blob/main/farms/ACryptoSFarmV3.sol

      https://github.com/acryptos/acryptos-
protocol/blob/main/strategies/StrategyACryptoSVenusLeverageUGV6.sol
```

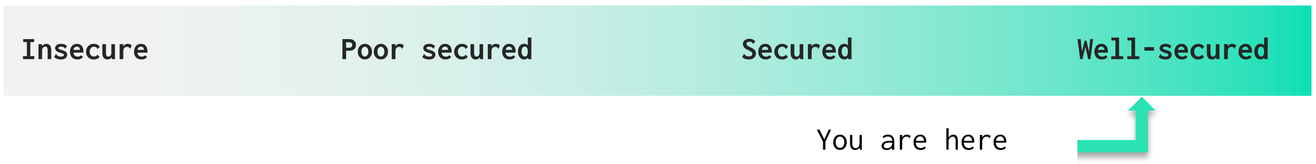
We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

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 ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts are well-secured.



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

Security engineers found **4** informational issue during the audit.

Notice:

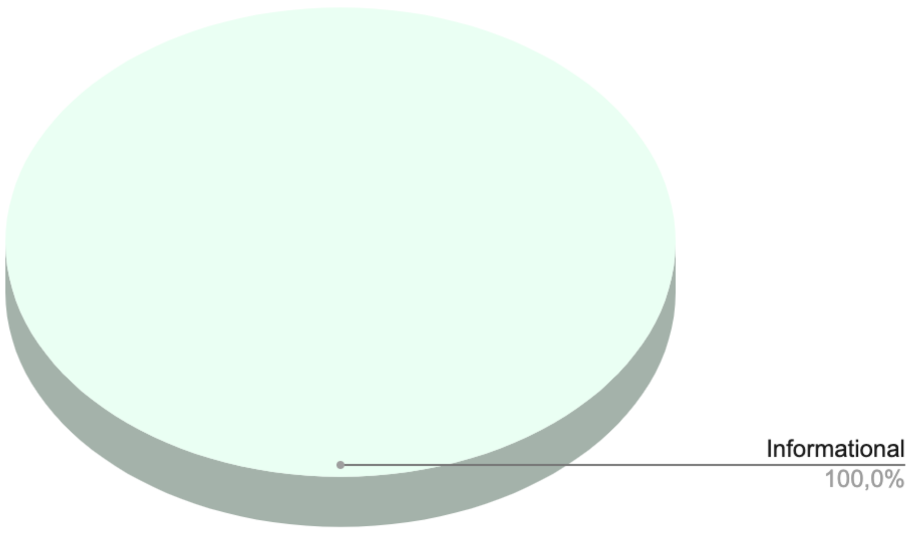
1. Lots of redundant operations, which will cost more gas, that is needed. We strongly recommend you to be more thoughtful and pay more attention to gas efficiency.

2. We can't fully audit these contracts, because of unclear functionality. We can't investigate the implementation of some third-party key entities during the audit.

3. Some operations like depositing or harvesting implemented and controlled in an async way. It means there are some not-blockchain

third-party like bots and time triggers that can affect the contract work but were not audited by our team.

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



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.

AS-IS overview

StrategyACryptoSVenusLeverageV2.sol

Description

Contract to define one of the investment strategies in the ACryptos protocol.

Imports

StrategyACryptoSVenusLeverageV2 has following imports:

- IERC20.sol – from the OpenZeppelin.
- SafeMath.sol – from the OpenZeppelin.
- Math.sol – from the OpenZeppelin.
- Address.sol – from the OpenZeppelin.
- SafeERC20.sol – from the OpenZeppelin.
- interfaces/yearn/IController.sol – Custom project interrface
- interfaces/yearn/Token.sol – Custom project interrface

Inheritance

StrategyACryptoSVenusLeverageV2 does not inherit anything.

Usages

StrategyACryptoSVenusLeverageV2 contract has following usages:

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

Structs

StrategyACryptoSVenusLeverageV2 contract has no data structures.

Enums

StrategyACryptoSVenusLeverageV2 contract has no enums.

Events

StrategyACryptoSVenusLeverageV2 contract has no events.

Modifiers

StrategyACryptoSVenusLeverageV2 has no modifiers.

Fields

StrategyACryptoSVenusLeverageV2 contract has following fields and constants:

- address public constant xvs = address(0xcF6BB5389c92Bdda8a3747Ddb454cB7a64626C63);
- address public constant wbnb = address(0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c);
- address public constant venusComptroller = address(0xfD36E2c2a6789Db23113685031d7F16329158384);
- address public constant uniswapRouter = address(0x05fF2B0DB69458A0750badebc4f9e13aDd608C7F);
- address public want;
- address public vToken;
- uint256 public targetBorrowLimit;
- uint256 public targetBorrowLimitHysteresis;
- address public governance;
- address public controller;
- address public strategist;
- uint256 public performanceFee = 450;
- uint256 public strategistReward = 50;
- uint256 public withdrawalFee = 50;
- uint256 public harvesterReward = 30;
- uint256 public constant FEE_DENOMINATOR = 10000;

Functions

StrategyACryptoSVenusLeverageV2 has following public functions:

- *constructor*
  Description
  Inits the contract and sets default parameters.
  Visibility
  public
  Input parameters
  - address _controller,
  - address _want,
  - address _vToken,

  - uint256 _targetBorrowLimit,
  - uint256 _targetBorrowLimitHysteresis.

  Constraints
  None
  Events emit
  None
  Output
  None
- *getName*
  Description
  Return the name of the contract.
  Visibility
  External pure
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  string memory
- *deposit*
  Description
  Transfer funds to CakeChefPid.
  Visibility
  public
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  None
- *withdraw*
  Description
  Controller only function for creating additional rewards from dust. Withdraw reward funds to controller wallet.
  Visibility
  External
  Input parameters
  uint256 balance
  Constraints
  - require(msg.sender == controller, "!controller");
  - require(want != address(_asset), "want");

  Events emit

None
Output
uint256 balance

- *withdraw*
  Description
  Withdraw partial funds, normally used with a vault withdrawal
  Visibility
  external
  Input parameters
    - uint256 _amount.
  Constraints
    - require(msg.sender == controller, "!controller").
  Events emit
  None.
  Output
  None
- *withdrawAll*
  Description
  Withdraw all funds, normally used when migrating strategies
  Visibility
  external
  Input parameters
  None
  Constraints
    - require(msg.sender == controller, "!controller");
  Events emit
  None
  Output
  uint256 balance.
- *harvest*
  Description
  Calculate and send reward to user.
  Visibility
  public
  Input parameters
  None.
  Constraints
    - require(msg.sender == tx.origin, "not eoa");
  Events emit
  None
  Output
  uint harvesterRewarded
- *balanceOfWant, balanceOfStakedWant¸ balanceOf, setGovernance, setController, setStrategist, setPerformanceFee, setStrategistReward, setWithdrawalFee, setHarvesterReward*

Description
Simple get info and set params functions. All set functions works only with governance account.
Visibility
external

- *pause*
  Description
  Function to stop strategy activity and withdraw all funds to controller address.
- *unpause*
  Description
  Function to stop strategy activity and withdraw all funds to controller address.

StrategyACryptoSVenusLeverageBnb.sol

Description

Contract to define one of the investment strategies in the ACryptos protocol.

Imports

StrategyACryptoSVenusLeverageBnb has following imports:

- IERC20.sol – from the OpenZeppelin.
- SafeMath.sol – from the OpenZeppelin.
- Math.sol – from the OpenZeppelin.
- Address.sol – from the OpenZeppelin.
- SafeERC20.sol – from the OpenZeppelin.
- interfaces/yearn/IController.sol
- interfaces/yearn/Token.sol
- interfaces/weth/WETH.sol

Inheritance

StrategyACryptoSVenusLeverageBnb does not inherit anything.

Usages

StrategyACryptoSVenusLeverageBnb contract has following usages:

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

Structs

StrategyACryptoSVenusLeverageBnb contract has no data structures.

Enums

StrategyACryptoSVenusLeverageBnb contract has no enums.

Events

StrategyACryptoSVenusLeverageBnb contract has no events.

Modifiers

StrategyACryptoSVenusLeverageBnb has no modifiers.

Fields

StrategyACryptoSVenusLeverageBnb contract has following fields and constants:

- address public constant xvs = address(0xcF6BB5389c92Bdda8a3747Ddb454cB7a64626C63);
- address public constant wbnb = address(0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c);
- address public constant venusComptroller = address(0xfD36E2c2a6789Db23113685031d7F16329158384);
- address public constant uniswapRouter = address(0x05fF2B0DB69458A0750badebc4f9e13aDd608C7F);
- address public want;
- address public vToken;
- uint256 public targetBorrowLimit;
- uint256 public targetBorrowLimitHysteresis;
- address public governance;
- address public controller;
- address public strategist;
- uint256 public performanceFee = 450;
- uint256 public strategistReward = 50;
- uint256 public withdrawalFee = 50;
- uint256 public harvesterReward = 30;
- uint256 public constant FEE_DENOMINATOR = 10000;

Functions

StrategyACryptoSVenusLeverageBnb has following public functions:

- *constructor*
  Description

Inits the contract and sets default parameters.
Visibility
public
Input parameters
- address _controller,
- address _want,
- address _vToken,
- uint256 _ _targetBorrowLimit,
- uint256 _targetBorrowLimitHysteresis.

Constraints
None
Events emit
None
Output
None

- *getName*
  Description
  Return the name of the contract.
  Visibility
  External pure
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  string memory
- *deposit*
  Description
  Transfer funds to CakeChefPid.
  Visibility
  public
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  None
- *withdraw*
  Description

Controller only function for creating additional rewards from dust. Withdraw reward funds to controller wallet.
Visibility
External
Input parameters
uint256 balance
Constraints
  - require(msg.sender == controller, "!controller");
  - require(want != address(_asset), "want");

Events emit
None
Output
uint256 balance

- *withdraw*
  Description
  Withdraw partial funds, normally used with a vault withdrawal
  Visibility
  external
  Input parameters
    - uint256 _amount.

  Constraints
    - require(msg.sender == controller, "!controller").

  Events emit
  None.
  Output
  None
- *withdrawAll*
  Description
  Withdraw all funds, normally used when migrating strategies
  Visibility
  external
  Input parameters
  None
  Constraints
    - require(msg.sender == controller, "!controller");

  Events emit
  None
  Output
  uint256 balance.
- *harvest*
  Description
  Calculate and send reward to user.
  Visibility
  public
  Input parameters
  None.

Constraints
- require(msg.sender == tx.origin, "not eoa");

Events emit
None
Output
uint harvesterRewarded

- *balanceOfWant, balanceOfStakedWant¸ balanceOf, setGovernance, setController, setStrategist, setPerformanceFee, setStrategistReward, setWithdrawalFee, setHarvesterReward*
  Description
  Simple get info and set params functions. All set functions works only with governance account.
  Visibility
  external
- *pause*
  Description
  Function to stop strategy activity and withdraw all funds to controller address.
- *unpause*
  Description
  Function to stop strategy activity and withdraw all funds to controller address.

StrategyACryptoSVenusLeverageXvs.sol

Description

Contract to define one of the investment strategy in ACryptos protocol.

Imports

StrategyACryptoSVenusLeverageXvs has following imports:

- IERC20.sol – from the OpenZeppelin.
- SafeMath.sol – from the OpenZeppelin.
- Math.sol – from the OpenZeppelin.
- Address.sol – from the OpenZeppelin.
- SafeERC20.sol – from the OpenZeppelin.
- interfaces/yearn/IController.sol – Custom project interrface
- interfaces/yearn/Token.sol – Custom project interrface

Inheritance

StrategyACryptoSVenusLeverageXvs does not inherit anything.

Usages

StrategyACryptoSVenusLeverageXvs contract has following usages:

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

Structs

StrategyACryptoSVenusLeverageXvs contract has no data structures.

Enums

StrategyACryptoSVenusLeverageXvs contract has no enums.

Events

StrategyACryptoSVenusLeverageXvs contract has no events.

Modifiers

StrategyACryptoSVenusLeverageXvs has no modifiers.

Fields

StrategyACryptoSVenusLeverageXvs contract has following fields and constants:

- address public constant xvs = address(0xcF6BB5389c92Bdda8a3747Ddb454cB7a64626C63);
- address public constant wbnb = address(0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c);
- address public constant venusComptroller = address(0xfD36E2c2a6789Db23113685031d7F16329158384);
- address public want;
- address public vToken;
- uint256 public targetBorrowLimit;
- uint256 public targetBorrowLimitHysteresis;
- address public governance;
- address public controller;
- address public strategist;
- uint256 public performanceFee = 450;

- uint256 public strategistReward = 50;
- uint256 public withdrawalFee = 50;
- uint256 public harvesterReward = 30;
- uint256 public constant FEE_DENOMINATOR = 10000;

Functions

StrategyACryptoSVenusLeverageXvs has following public functions:

- *constructor*
  Description
  Inits the contract and sets default parameters.
  Visibility
  public
  Input parameters
    - address _controller,
    - address _want,
    - address _vToken,
    - uint256 _ _targetBorrowLimit,
    - uint256 _targetBorrowLimitHysteresis.

  Constraints
  None
  Events emit
  None
  Output
  None
- *getName*
  Description
  Return the name of the contract.
  Visibility
  External pure
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  string memory
- *deposit*
  Description
  Transfer funds to CakeChefPid.
  Visibility
  public

Input parameters
None
Constraints
None
Events emit
None
Output
None

- *withdraw*
  Description
  Controller only function for creating additional rewards from dust. Withdraw reward funds to controller wallet.
  Visibility
  External
  Input parameters
  uint256 balance
  Constraints
    - require(msg.sender == controller, "!controller");
    - require(want != address(_asset), "want");
  Events emit
  None
  Output
  uint256 balance
- *withdraw*
  Description
  Withdraw partial funds, normally used with a vault withdrawal
  Visibility
  external
  Input parameters
    - uint256 _amount.
  Constraints
    - require(msg.sender == controller, "!controller").
  Events emit
  None.
  Output
  None
- *withdrawAll*
  Description
  Withdraw all funds, normally used when migrating strategies
  Visibility
  external
  Input parameters
  None
  Constraints
    - require(msg.sender == controller, "!controller");
  Events emit

None
Output
uint256 balance.

- *harvest*
  Description
  Calculate and send reward to user.
  Visibility
  public
  Input parameters
  None.
  Constraints
    - require(msg.sender == tx.origin, "not eoa");
  Events emit
  None
  Output
  uint harvesterRewarded
- *balanceOfWant, balanceOfStakedWant¸ balanceOf, setGovernance, setController, setStrategist, setPerformanceFee, setStrategistReward, setWithdrawalFee, setHarvesterReward*
  Description
  Simple get info and set params functions. All set functions works only with governance account.
  Visibility
  external
- *pause*
  Description
  Function to stop strategy activity and withdraw all funds to controller address.
- *unpause*
  Description
  Function to stop strategy activity and withdraw all funds to controller address.

StrategyACryptoS0V3.sol

Description

Contract to define one of the investment strategy in ACryptos protocol.

Imports

StrategyACryptoS0V3 has following imports:

- IERC20.sol – from the OpenZeppelin.

- SafeMath.sol – from the OpenZeppelin.
- Math.sol – from the OpenZeppelin.
- Address.sol – from the OpenZeppelin.
- SafeERC20.sol – from the OpenZeppelin.
- interfaces/yearn/IController.sol – Custom project interrface
- interfaces/yearn/Token.sol – Custom project interrface

Inheritance

StrategyACryptoS0V3 does not inherit anything.

Usages

StrategyACryptoS0V3 contract has following usages:

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

Structs

StrategyACryptoS0V3 contract has following data structures:

- PairToLiquidate
- TokenToSwap

Enums

StrategyACryptoS0V3 contract has no enums.

Events

StrategyACryptoS0V3 contract has no events.

Modifiers

StrategyACryptoS0V3 has no modifiers.

Fields

StrategyACryptoS0V3 contract has following fields and constants:

- address public constant want = address(0x4197C6EF3879a08cD51e5560da5064B773aa1d29); //ACS
- address public constant pancakeSwapRouter = address(0x05fF2B0DB69458A0750badebc4f9e13aDd608C7F);

- address[] public ssToWithdraw; //StableSwap pools to withdraw admin fees from
- PairToLiquidate[] public pairsToLiquidate;
- TokenToSwap[] public tokensToSwap0;
- TokenToSwap[] public tokensToSwap1;
- address public governance;
- address public controller;
- address public strategist;
- uint256 public withdrawalFee = 1000; //10%
- uint256 public harvesterReward = 30;
- uint256 public constant FEE_DENOMINATOR = 10000;

Functions

StrategyACryptoS0V3 has following public functions:

- *getName*
  Description
  Return the name of the contract.
  Visibility
  External pure
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  string memory
- *deposit*
  Description
  Transfer funds to CakeChefPid.
  Visibility
  public
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output

None

- *withdraw*
  Description
  Controller only function for creating additional rewards from dust. Withdraw reward funds to controller wallet.
  Visibility
  External
  Input parameters
  uint256 balance
  Constraints
    - require(msg.sender == controller, "!controller");
    - require(want != address(_asset), "want");

  Events emit
  None
  Output
  uint256 balance
- *withdraw*
  Description
  Withdraw partial funds, normally used with a vault withdrawal
  Visibility
  external
  Input parameters
    - uint256 _amount.

  Constraints
    - require(msg.sender == controller, "!controller").

  Events emit
  None.
  Output
  None
- *withdrawAll*
  Description
  Withdraw all funds, normally used when migrating strategies
  Visibility
  external
  Input parameters
  None
  Constraints
    - require(msg.sender == controller, "!controller");

  Events emit
  None
  Output
  uint256 balance.
- *harvest*
  Description
  Calculate and send reward to user.
  Visibility

public
Input parameters
None.
Constraints
  - require(msg.sender == tx.origin, "not eoa");

Events emit
None
Output
uint harvesterRewarded

- *balanceOfWant, balanceOfStakedWant¸ balanceOf, setGovernance, setController, setStrategist, setPerformanceFee,setStrategistReward, setWithdrawalFee, setHarvesterReward*
  Description
  Simple get info and set params functions. All set functions works only with governance account.
  Visibility
  external
- *pause*
  Description
  Function to stop strategy activity and withdraw all funds to controller address.
- *unpause*
  Description
  Function to stop strategy activity and withdraw all funds to controller address.

StrategyACryptoSCakeBnbToken.sol
Description

Contract to define one of the investment strategies in ACryptos protocol.

Imports

StrategyACryptoSCakeBnbToken has following imports:

- IERC20.sol – from the OpenZeppelin.
- SafeMath.sol – from the OpenZeppelin.
- Math.sol – from the OpenZeppelin.

- Address.sol – from the OpenZeppelin.
- SafeERC20.sol – from the OpenZeppelin.
- interfaces/yearn/IController.sol – Custom project interrface
- interfaces/yearn/Token.sol – Custom project interrface

Inheritance

StrategyACryptoSCakeBnbToken does not inherit anything.

Usages

StrategyACryptoSCakeBnbToken contract has following usages:

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

Structs

StrategyACryptoSCakeBnbToken contract has no data structures.

Enums

StrategyACryptoSCakeBnbToken contract has no enums.

Events

StrategyACryptoSCakeBnbToken contract has no events.

Modifiers

StrategyACryptoSCakeBnbToken has no modifiers.

Fields

StrategyACryptoSCakeBnbToken contract has following fields and constants:

- address public constant cake = address(0x0E09FaBB73Bd3Ade0a17ECC321fD13a19e81cE82);
- address public constant cakeChef = address(0x73feaa1eE314F8c655E354234017bE2193C9E24E);
- address public constant pancakeSwapRouter = address(0x05fF2B0DB69458A0750badebc4f9e13aDd608C7F);
- address public want;
- address public tokenA;

- address public tokenB;
- uint256 public cakeChefPid;
- address[] public cakeToTokenAPath;
- address public governance;
- address public controller;
- address public strategist;
- uint256 public performanceFee = 450;
- uint256 public strategistReward = 50;
- uint256 public withdrawalFee = 50;
- uint256 public harvesterReward = 30;
- uint256 public constant FEE_DENOMINATOR = 10000;

Functions

StrategyACryptoSCakeBnbToken has following public functions:

- *constructor*
  Description
  Inits the contract and sets default parameters.
  Visibility
  public
  Input parameters
  - address _controller,
  - address _want,
  - address _tokenB,
  - uint256 _cakeChefPid,

  Constraints
  None
  Events emit
  None
  Output
  None
- *getName*
  Description
  Return the name of the contract.
  Visibility
  External pure
  Input parameters
  None

Constraints
None
Events emit
None
Output
string memory

- *deposit*
  Description
  Transfer funds to CakeChefPid.
  Visibility
  public
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  None
- *withdraw*
  Description
  Controller only function for creating additional rewards from dust. Withdraw reward funds to controller wallet.
  Visibility
  External
  Input parameters
  uint256 balance
  Constraints
    - require(msg.sender == controller, "!controller");
    - require(want != address(_asset), "want");

  Events emit
  None
  Output
  uint256 balance
- *withdraw*
  Description
  Withdraw partial funds, normally used with a vault withdrawal
  Visibility
  external
  Input parameters
    - uint256 _amount.

  Constraints
    - require(msg.sender == controller, "!controller").

  Events emit
  None.
  Output

None

- *withdrawAll*
  Description
  Withdraw all funds, normally used when migrating strategies
  Visibility
  external
  Input parameters
  None
  Constraints
    - require(msg.sender == controller, "!controller");
  Events emit
  None
  Output
  uint256 balance.
- *harvest*
  Description
  Calculate and send reward to user.
  Visibility
  public
  Input parameters
  None.
  Constraints
    - require(msg.sender == tx.origin, "not eoa");
  Events emit
  None
  Output
  uint harvesterRewarded
- *balanceOfWant, balanceOfStakedWant, balanceOf, setGovernance, setController, setStrategist, setPerformanceFee, setStrategistReward, setWithdrawalFee, setHarvesterReward*
  Description
  Simple get info and set params functions. All set functions works only with governance account.
  Visibility
  external
- *pause*
  Description
  Function to stop strategy activity and withdraw all funds to controller address.
- *unpause*
  Description
  Function to stop strategy activity and withdraw all funds to controller address.

StrategyACryptoSCakeBnbCake.sol

Description

Contract to define one of the investment strategy in ACryptos protocol.

Imports

StrategyACryptoSCakeBnbCake has following imports:

- IERC20.sol – from the OpenZeppelin.
- SafeMath.sol – from the OpenZeppelin.
- Math.sol – from the OpenZeppelin.
- Address.sol – from the OpenZeppelin.
- SafeERC20.sol – from the OpenZeppelin.
- interfaces/yearn/IController.sol – Custom project interrface
- interfaces/yearn/Token.sol – Custom project interrface

Inheritance

StrategyACryptoSCakeBnbCake does not inherit anything.

Usages

StrategyACryptoSCakeBnbCake contract has following usages:

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

Structs

StrategyACryptoSCakeBnbCake contract has no data structures.

Enums

StrategyACryptoSCakeBnbCake contract has no enums.

Events

StrategyACryptoSCakeBnbCake contract has no events.

Modifiers

StrategyACryptoSCakeBnbCake has no modifiers.

Fields

StrategyACryptoSCakeBnbCake contract has following fields and constants:

- address public constant cake = address(0x0E09FaBB73Bd3Ade0a17ECC321fD13a19e81cE82);
- address public constant cake = address(0x0E09FaBB73Bd3Ade0a17ECC321fD13a19e81cE82);
- address public constant wbnb = address(0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c);
- address public constant cakeChef = address(0x73feaa1eE314F8c655E354234017bE2193C9E24E);
- address public constant pancakeSwapRouter = address(0x05fF2B0DB69458A0750badebc4f9e13aDd608C7F);
- address public want;
- uint256 public cakeChefPid;
- address public governance;
- address public controller;
- address public strategist;
- uint256 public performanceFee = 450;
- uint256 public strategistReward = 50;
- uint256 public withdrawalFee = 50;
- uint256 public harvesterReward = 30;
- uint256 public constant FEE_DENOMINATOR = 10000;

Functions

StrategyACryptoSCakeBnbCake has following public functions:

- *constructor*
  Description
  Inits the contract and sets default parameters.
  Visibility
  public
  Input parameters
  - address _controller,
  - address _want,
  - uint256 _cakeChefPid,

Constraints
None
Events emit
None
Output
None

- *getName*
Description
Return the name of the contract.
Visibility
External pure
Input parameters
None
Constraints
None
Events emit
None
Output
string memory
- *deposit*
Description
Transfer funds to CakeChefPid.
Visibility
public
Input parameters
None
Constraints
None
Events emit
None
Output
None
- *withdraw*
Description
Controller only function for creating additional rewards from dust. Withdraw reward funds to controller wallet.
Visibility
External
Input parameters
uint256 balance
Constraints
    - require(msg.sender == controller, "!controller");
    - require(want != address(_asset), "want");

  Events emit
None
Output

uint256 balance

- *withdraw*
  Description
  Withdraw partial funds, normally used with a vault withdrawal
  Visibility
  external
  Input parameters
    - uint256 _amount.
  Constraints
    - require(msg.sender == controller, "!controller").
  Events emit
  None.
  Output
  None
- *withdrawAll*
  Description
  Withdraw all funds, normally used when migrating strategies
  Visibility
  external
  Input parameters
  None
  Constraints
    - require(msg.sender == controller, "!controller");
  Events emit
  None
  Output
  uint256 balance.
- *harvest*
  Description
  Calculate and send reward to user.
  Visibility
  public
  Input parameters
  None.
  Constraints
    - require(msg.sender == tx.origin, "not eoa");
  Events emit
  None
  Output
  uint harvesterRewarded
- *balanceOfWant, balanceOfStakedWant¸ balanceOf, setGovernance, setController, setStrategist, setPerformanceFee,setStrategistReward, setWithdrawalFee, setHarvesterReward*
  Description

Simple get info and set params functions. All set functions work only with governance account.
Visibility
external

- *pause*
  Description
  Function to stop strategy activity and withdraw all funds to controller address.
- *unpause*
  Description
  Function to stop strategy activity and withdraw all funds to controller address.

StrategyACryptoSCakeV2b.sol

Description

Contract to define one of the investment strategies in ACryptos protocol.

Imports

StrategyACryptoSCakeV2b has following imports:

- IERC20.sol – from the OpenZeppelin.
- SafeMath.sol – from the OpenZeppelin.
- Math.sol – from the OpenZeppelin.
- Address.sol – from the OpenZeppelin.
- SafeERC20.sol – from the OpenZeppelin.
- interfaces/yearn/IController.sol – Custom project interrface
- interfaces/yearn/Token.sol – Custom project interrface

Inheritance

StrategyACryptoSCakeV2b does not inherit anything.

Usages

StrategyACryptoSCakeV2b contract has following usages:

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

Structs

StrategyACryptoSCakeV2b contract has no data structures.

Enums

StrategyACryptoSCakeV2b contract has no enums.

Events

StrategyACryptoSCakeV2b contract has no events.

Modifiers

StrategyACryptoSCakeV2b has no modifiers.

Fields

StrategyACryptoSCakeV2b contract has following fields and constants:

- address public constant want = address(0x0E09FaBB73Bd3Ade0a17ECC321fD13a19e81cE82);
- address public constant wbnb = address(0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c);
- address public constant cakeChef = address(0x73feaa1eE314F8c655E354234017bE2193C9E24E);
- address public constant pancakeSwapRouter = address(0x05fF2B0DB69458A0750badebc4f9e13aDd608C7F);
- address public governance;
- address public controller;
- address public strategist;
- uint256 public performanceFee = 450;
- uint256 public strategistReward = 50;
- uint256 public withdrawalFee = 50;
- uint256 public harvesterReward = 30;
- uint256 public constant FEE_DENOMINATOR = 10000;

Functions

StrategyACryptoSCakeV2b has following public functions:

- *constructor*
  Description
  Inits the contract and sets default parameters.

Visibility
public
Input parameters
- address _controller,

Constraints
None
Events emit
None
Output
None

- *getName*
  Description
  Return the name of the contract.
  Visibility
  External pure
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  string memory
- *deposit*
  Description
  Transfer funds to CakeChefPid.
  Visibility
  public
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  None
- *withdraw*
  Description
  Controller only function for creating additional rewards from dust. Withdraw reward funds to controller wallet.
  Visibility
  External
  Input parameters
  uint256 balance
  Constraints
  - require(msg.sender == controller, "!controller");

  - require(want != address(_asset), "want");

Events emit
None
Output
uint256 balance

- *withdraw*
  Description
  Withdraw partial funds, normally used with a vault withdrawal
  Visibility
  external
  Input parameters
  - uint256 _amount.

  Constraints
  - require(msg.sender == controller, "!controller").

  Events emit
  None.
  Output
  None
- *withdrawAll*
  Description
  Withdraw all funds, normally used when migrating strategies
  Visibility
  external
  Input parameters
  None
  Constraints
  - require(msg.sender == controller, "!controller");

  Events emit
  None
  Output
  uint256 balance.
- *harvest*
  Description
  Calculate and send reward to user.
  Visibility
  public
  Input parameters
  None.
  Constraints
  - require(msg.sender == tx.origin, "not eoa");

  Events emit
  None
  Output
  uint harvesterRewarded
- *balanceOfWant, balanceOfStakedWant¸ balanceOf, setGovernance, setController, setStrategist,*

setPerformanceFee,setStrategistReward, setWithdrawalFee, setHarvesterReward
Description
Simple get info and set params functions. All set functions works only with governance account.
Visibility
external

- *pause*
  Description
  Function to stop strategy activity and withdraw all funds to controller address.
- *unpause*
  Description

Function to stop strategy activity and withdraw all funds to controller address.

StrategyACryptoSVenusVAI.sol

Description

Contract to define one of the investment strategy in ACryptos protocol.

Imports

StrategyACryptoSVenusVAI has following imports:

- IERC20.sol – from the OpenZeppelin.
- SafeMath.sol – from the OpenZeppelin.
- Math.sol – from the OpenZeppelin.
- Address.sol – from the OpenZeppelin.
- SafeERC20.sol – from the OpenZeppelin.
- interfaces/yearn/IController.sol – Custom project interrface
- interfaces/yearn/Token.sol – Custom project interrface

Inheritance

StrategyACryptoSVenusVAI does not inherit anything.

Usages

StrategyACryptoSVenusVAI contract has following usages:

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

Structs

StrategyACryptoSVenusVAI contract has no data structures.

Enums

StrategyACryptoSVenusVAI contract has no enums.

Events

StrategyACryptoSVenusVAI contract has no events.

Modifiers

StrategyACryptoSVenusVAI has no modifiers.

Fields

StrategyACryptoSVenusVAI contract has following fields and constants:

- address public constant xvs = address(0xcF6BB5389c92Bdda8a3747Ddb454cB7a64626C63);
- address public constant wbnb = address(0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c);
- address public constant busd = address(0xe9e7CEA3DedcA5984780Bafc599bD69ADd087D56);
- address public constant uniswapRouter = address(0x05fF2B0DB69458A0750badebc4f9e13aDd608C7F);
- address public constant ssPool = address(0x191409D5A4EfFe25b0f4240557BA2192D18a191e);
- address public constant venusVAIVault = address(0x0667Eed0a0aAb930af74a3dfeDD263A73994f216);
- address public want;
- address public governance;
- address public controller;
- address public strategist;
- uint256 public performanceFee = 450;
- uint256 public strategistReward = 50;
- uint256 public withdrawalFee = 50;
- uint256 public harvesterReward = 30;
- uint256 public constant FEE_DENOMINATOR = 10000;

Functions

StrategyACryptoSVenusVAI has following public functions:

- *constructor*
  Description
  Inits the contract and sets default parameters.
  Visibility
  public
  Input parameters
    - address _controller,
    - address _want,

  Constraints
  None
  Events emit
  None
  Output
  None
- *getName*
  Description
  Return the name of the contract.
  Visibility
  External pure
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  string memory
- *deposit*
  Description
  Transfer funds to CakeChefPid.
  Visibility
  public
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  None
- *withdraw*

Description
Controller only function for creating additional rewards from dust. Withdraw reward funds to controller wallet.
Visibility
External
Input parameters
uint256 balance
Constraints
- require(msg.sender == controller, "!controller");
- require(want != address(_asset), "want");

Events emit
None
Output
uint256 balance

- *withdraw*
  Description
  Withdraw partial funds, normally used with a vault withdrawal
  Visibility
  external
  Input parameters
  - uint256 _amount.

  Constraints
  - require(msg.sender == controller, "!controller").

  Events emit
  None.
  Output
  None
- *withdrawAll*
  Description
  Withdraw all funds, normally used when migrating strategies
  Visibility
  external
  Input parameters
  None
  Constraints
  - require(msg.sender == controller, "!controller");

  Events emit
  None
  Output
  uint256 balance.
- *harvest*
  Description
  Calculate and send reward to user.
  Visibility
  public
  Input parameters

None.
Constraints
  ○ require(msg.sender == tx.origin, "not eoa");
Events emit
None
Output
uint harvesterRewarded

- *balanceOfWant, balanceOfStakedWant¸ balanceOf, setGovernance, setController, setStrategist, setPerformanceFee, setStrategistReward, setWithdrawalFee, setHarvesterReward*
  Description
  Simple get info and set params functions. All set functions works only with governance account.
  Visibility
  external
- *pause*
  Description
  Function to stop strategy activity and withdraw all funds to controller address.
- *unpause*
  Description
  Function to stop strategy activity and withdraw all funds to controller address.

MasterChef.sol

Imports

MasterChef has following imports:

- IERC20.sol – from the OpenZeppelin.
- SafeMath.sol – from the OpenZeppelin.
- SafeERC20.sol – from the OpenZeppelin.
- Ownable – from the OpenZeppelin.
- EnumerableSet – from the OpenZeppelin.

Inheritance

MasterChef is Ownable.

Usages

MasterChef contract has following usages:

- SafeERC20 for IERC20;
- SafeMath for uint256;

Structs

MasterChef contract has following data structures:

- UserInfo
- PoolInfo

Enums

MasterChef contract has no enums.

Events

MasterChef contract has following events:

- event Deposit(address indexed user, uint256 indexed pid, uint256 amount);
- event Withdraw(address indexed user, uint256 indexed pid, uint256 amount);
- event EmergencyWithdraw(address indexed user, uint256 indexed pid, uint256 amount);

Modifiers

MasterChef has no custom modifiers.

Fields

MasterChef contract has following fields and constants:

- ERC20Mintable public sushi;
- address public devaddr;
- uint256 public bonusEndBlock;
- uint256 public sushiPerBlock;
- uint256 public constant BONUS_MULTIPLIER = 10;
- address public acsACS;
- uint256 public withdrawalFee = 10e18;
- uint256 public devReward = 1000;
- uint256 public acsACSReward = 3333;
- uint256 public constant REWARD_DENOMINATOR = 10000;

- PoolInfo[] public poolInfo;
- mapping (uint256 => mapping (address => UserInfo)) public userInfo;
- uint256 public totalAllocPoint = 0;
- uint256 public startBlock;

MasterChef has following differences between the origin:

- Migration functionality was removed
- Add ability to change reward parameters:
  - sushiPerBlock
  - acsACS
  - withdrawalFee
  - devReward
  - acsACSReward
- Change math to calculate sushi mint amount
- Add reward to devaddr address during the mint process
- Add withdrawal fees to deposit

MasterChefV2.sol

Imports

MasterChefV2 has following imports:

- IERC20.sol – from the OpenZeppelin.
- SafeMath.sol – from the OpenZeppelin.
- SafeERC20.sol – from the OpenZeppelin.
- Ownable – from the OpenZeppelin.
- EnumerableSet – from the OpenZeppelin.

Inheritance

MasterChefV2 is Ownable.

Usages

MasterChef contract has following usages:

- SafeERC20 for IERC20;

- SafeMath for uint256;

Structs

MasterChefV2 contract has following data structures:

- UserInfo
- PoolInfo

Enums

MasterChefV2 contract has no enums.

Events

MasterChefV2 contract has following events:

- event Deposit(address indexed user, uint256 indexed pid, uint256 amount);
- event Withdraw(address indexed user, uint256 indexed pid, uint256 amount);
- event EmergencyWithdraw(address indexed user, uint256 indexed pid, uint256 amount);

Modifiers

MasterChefV2 has no custom modifiers.

Fields

MasterChefV2 contract has following fields and constants:

- ERC20Mintable public sushi;
- address public devaddr;
- uint256 public bonusEndBlock;
- uint256 public sushiPerBlock;
- uint256 public constant BONUS_MULTIPLIER = 10;
- address public acsACS;
- address public vault;
- bool public disableHarvest=false; //allow harvest to vault only
- uint256 public withdrawalFee = 10e18;

- uint256 public devReward = 1000;
- uint256 public acsACSReward = 3333;
- uint256 public constant REWARD_DENOMINATOR = 10000;
- PoolInfo[] public poolInfo;
- mapping (uint256 => mapping (address => UserInfo)) public userInfo;
- uint256 public totalAllocPoint = 0;
- uint256 public startBlock;

MasterChefV2 has following differences between the MasterChef:

- Add rewardCredit to userInfo struct
- Add Vault address and functionality
- add ability to disableHarvest
- Change withdraw pending logic amount to rewardCreedit.
- Add harvest functionality
- add harvestToVault function
- add safeSushiToVault function
- add setVault function
- setDisableHarvest function

Controller.sol

This contract is equal to origin.

Timelock6H.sol

This contract is equal to origin.

acsACS.sol

Imports

acsACS contract has following imports:

- SafeMath.sol – from the OpenZeppelin.

Inheritance

acsACS contract has no parents.

Usages

acsACS contract has no usages.

Structs

acsACS contract has following data structures:

- Checkpoint

Enums

acsACS contract has no custom enums.

Events

acsACS contract has following custom evets:

- event DelegateChanged(address indexed delegator, address indexed fromDelegate, address indexed toDelegate);
- event DelegateVotesChanged(address indexed delegate, uint previousBalance, uint newBalance);
- event Transfer(address indexed from, address indexed to, uint256 amount);
- event Approval(address indexed owner, address indexed spender, uint256 amount);

Modifiers

acsACS has no custom modifiers.

Fields

acsACS contract has following fields and constants:

- string public constant name = "ACryptoS ACryptoS";
- string public constant symbol = "acsACS";
- uint8 public constant decimals = 18;
- uint public totalSupply = 0;
- mapping (address => mapping (address => uint96)) internal allowances;
- mapping (address => uint96) internal balances;
- mapping (address => address) public delegates;

- mapping (address => mapping (uint32 => Checkpoint)) public checkpoints;
- mapping (address => uint32) 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)");
- bytes32 public constant PERMIT_TYPEHASH = keccak256("Permit(address owner,address spender,uint256 value,uint256 nonce,uint256 deadline)");
- mapping (address => uint) public nonces;

acsACS has following differences between the fork original

- Minter functionality was removed
- Constructor was removed
- Make contract untransferable

acsACSI.sol

Description

Forked from:

https://raw.githubusercontent.com/Uniswap/governance/ab22c084bacb2636a1aebf9759890063eb6e4946/contracts/Uni.sol

Imports

acsACSI contract has following imports:

- SafeMath.sol – from the OpenZeppelin.

Inheritance

acsACSI contract has no parents.

Usages

acsACSI contract has no usages.

Structs

acsACSI contract has following data structures:

- Checkpoint

Enums

acsACSI contract has no custom enums.

Events

acsACSI contract has following custom evets:

- event DelegateChanged(address indexed delegator, address indexed fromDelegate, address indexed toDelegate);
- event DelegateVotesChanged(address indexed delegate, uint previousBalance, uint newBalance);
- event Transfer(address indexed from, address indexed to, uint256 amount);
- event Approval(address indexed owner, address indexed spender, uint256 amount);

Modifiers

acsACSI has no custom modifiers.

Fields

acsACSI contract has following fields and constants:

- string public constant name = "ACryptoS ACryptoS";
- string public constant symbol = "acsACS";
- uint8 public constant decimals = 18;
- uint public totalSupply = 0;
- mapping (address => mapping (address => uint96)) internal allowances;
- mapping (address => uint96) internal balances;
- mapping (address => address) public delegates;
- mapping (address => mapping (uint32 => Checkpoint)) public checkpoints;
- mapping (address => uint32) 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)");
- bytes32 public constant PERMIT_TYPEHASH = keccak256("Permit(address owner,address spender,uint256 value,uint256 nonce,uint256 deadline)");
- mapping (address => uint) public nonces;

acsACSI has following differences between the fork original

- Minter functionality was removed
- Constructor was removed
- Make contract untransferable

ACryptoSVaultBnb.sol

Description

Forked from:

https://github.com/iearn-finance/yearn-protocol/blob/f06e448d3f11a3c181fa181c782067f5f81deae9/contracts/vaults/yWETH.sol

Imports

ACryptoSVaultBnb contract has following imports:

- IERC20.sol – from the OpenZeppelin.
- SafeMath.sol – from the OpenZeppelin.
- Address.sol – from the OpenZeppelin.
- SafeERC20.sol – from the OpenZeppelin.
- ERC20.sol – from the OpenZeppelin.
- ERC20Detailed.sol – from the OpenZeppelin.
- Ownable.sol – from the OpenZeppelin.

Inheritance

ACryptoSVaultBnb contract has following parents:

- ERC20

- ERC20Detailed

Usages

ACryptoSVaultBnb contract has following usages:

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

Structs

ACryptoSVaultBnb contract has no data structures.

Enums

ACryptoSVaultBnb contract has no custom enums.

Events

ACryptoSVaultBnb contract has no custom evets.

Modifiers

ACryptoSVaultBnb has no custom modifiers.

Fields

ACryptoSVaultBnb contract has following fields and constants:

- IERC20 public token;
- uint256 public min = 10000;
- uint256 public constant max = 10000;
- address public governance;
- address public controller;

ACryptoSVaultBnb has following differences between the fork original

- Add earn function call in scope of deposit and depositETH functions

ACryptoSVault0V2_ACSI.sol

Description

Forked from:

https://github.com/iearn-finance/yearn-protocol/blob/f06e448d3f11a3c181fa181c782067f5f81deae9/contracts/vaults/yVault.sol

Imports

ACryptoSVault0V2_ACSI contract has following imports:

- IERC20.sol – from the OpenZeppelin.
- SafeMath.sol – from the OpenZeppelin.
- Address.sol – from the OpenZeppelin.
- SafeERC20.sol – from the OpenZeppelin.
- governance/acsACS_ACSI.sol
- Ownable.sol – from the OpenZeppelin.
- interfaces/yearn/IController.sol";

Inheritance

ACryptoSVault0V2_ACSI contract has following parents:

- acsACSI

Usages

ACryptoSVault0V2_ACSI contract has following usages:

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

Structs

ACryptoSVault0V2_ACSI contract has no data structures.

Enums

ACryptoSVault0V2_ACSI contract has no custom enums.

Events

ACryptoSVault0V2_ACSI contract has no custom evets.

Modifiers

ACryptoSVault0V2_ACSI has no custom modifiers.

Fields

ACryptoSVault0V2_ACSI contract has following fields and constants:

- IERC20 public token;
- uint256 public min = 10000;
- uint256 public constant max = 10000;
- address public governance;
- address public controller;

ACryptoSVault0V2_ACSI has following differences with the original

- Add earn function call in scope of deposit and depositETH functions
- Uses modified untransferable Uniswap governance token as LP token.

ACryptoSVault0.sol

Description

Forked from:

https://github.com/iearn-finance/yearn-protocol/blob/f06e448d3f11a3c181fa181c782067f5f81deae9/contracts/vaults/yVault.sol

Imports

ACryptoSVault0 contract has following imports:

- IERC20.sol – from the OpenZeppelin.
- SafeMath.sol – from the OpenZeppelin.
- Address.sol – from the OpenZeppelin.
- SafeERC20.sol – from the OpenZeppelin.
- governance/ acsACS.sol
- Ownable.sol – from the OpenZeppelin.
- interfaces/yearn/IController.sol

Inheritance

ACryptoSVault0 contract has following parents:

- acsACSI

Usages

ACryptoSVault0 contract has following usages:

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

Structs

ACryptoSVault0 contract has no data structures.

Enums

ACryptoSVault0 contract has no custom enums.

Events

ACryptoSVault0 contract has no custom evets.

Modifiers

ACryptoSVault0 has no custom modifiers.

Fields

ACryptoSVault0 contract has following fields and constants:

- IERC20 public token;
- uint256 public min = 10000;
- uint256 public constant max = 10000;
- address public governance;
- address public controller;

ACryptoSVault0 has following differences with the original

- Add earn function call in scope of deposit and depositETH functions
- Uses modified untransferable Uniswap governance token as LP token.

ACryptoSVault.sol

Description

Forked from:

https://github.com/iearn-finance/yearn-protocol/blob/f06e448d3f11a3c181fa181c782067f5f81deae9/contracts/vaults/yVault.sol

Imports

ACryptoSVault contract has following imports:

- IERC20.sol – from the OpenZeppelin.
- SafeMath.sol – from the OpenZeppelin.
- Address.sol – from the OpenZeppelin.
- SafeERC20.sol – from the OpenZeppelin.
- governance/acsACS.sol
- Ownable.sol – from the OpenZeppelin.
- interfaces/yearn/IController.sol

Inheritance

ACryptoSVault contract has following parents:

- ERC20
- ERC20Detailed

Usages

ACryptoSVault contract has following usages:

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

Structs

ACryptoSVault contract has no data structures.

Enums

ACryptoSVault contract has no custom enums.

Events

ACryptoSVault contract has no custom evets.

Modifiers

ACryptoSVault has no custom modifiers.

Fields

ACryptoSVault contract has following fields and constants:

- IERC20 public token;
- uint256 public min = 10000;
- uint256 public constant max = 10000;
- address public governance;
- address public controller;

ACryptoSVault has following differences with the original

- Add earn function call in scope of deposit and depositETH functions
- Uses modified untransferable Uniswap governance token as LP token.

Audit overview

■ Informational / Code style / Best Practice

1. Some code style issues were found by the static code analyzers.
2. We recommend you extract recurring validations into modifiers:
    a. require(msg.sender == controller, "!controller");
    b. require(msg.sender == controller || msg.sender == strategist || msg.sender == governance, "!authorized");
    c. require(msg.sender == strategist || msg.sender == governance, "!authorized");
3. We recommend you extract direct values from code into constants.
4. *ERC20.safeApprove()* function called twice each time it is used. A simple approve function may be used to decrease gas consumption. If you have to be sure the already approved amount is less than logic need, it is better to check it. But while you use approved only for work with protocols, there is a low probability that approval will not be used.

   Customers` notice: We use SafeERC20 library as it “is a wrapper around the interface that eliminates the need to handle boolean return values”, this allows our contracts to work with a variety of tokens and handle most edge cases without modification. We also approve each time we transfer so the underlying protocols we farm do not have infinite approval on our contract tokens, which leads to some marginal improvement on risk.

   Hacken notice: ERC20.safeApprove() function called twice each time it is used. A simple approve function may be used to decrease gas consumption. If you have to be sure the already approved amount is less than logic need, it is better to check it. But while you use approved only for work with protocols, there is a low probability that approval will not be used.

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.

Security engineers found **4** informational issue during the audit.

Notice:

1. Lots of redundant operations, which will cost more gas, that is needed. We strongly recommend you to be more thoughtful and pay more attention to gas efficiency.

2. We can't fully audit these contracts, because of unclear functionality. We can't investigate the implementation of some third-party key entities during the audit.

3. Some operations like depositing or harvesting implemented and controlled in an async way. It means there are some not-blockchain third-party like bots and time triggers that can affect the contract work but were not audited by our team.

Check the updates (11/16/2021)

Description:

The customer creates a new ACryptoSFarmV3 contract as a changed version of contract ACryptoSFarmV2b_ACSI. The changes were mostly related to reward logic and contract initialization function.

Changes:

1. https://github.com/acryptos/acryptos-protocol/blob/main/farms/ACryptoSFarmV3.sol

   AdditionalPoolReward amount was changed to rewardPerBlock. The addAdditionalReward and addAdditionalPoolReward functions become allowed to set more than one at a time.

2. https://github.com/acryptos/acryptos-protocol/blob/main/strategies/StrategyACryptoSVenusLeverageUGV6.sol

   New redemption fee was accommodated from Venus Protocol. Contract changed to proxy upgradeable.

Conclusion:

New logic was audited and tested. During the test team have no information about new business logic. But no logical issues were found due to audit. But we strongly recommend update the compiler version.

Category	**Check Item**	**Comments**
Code review	▪ Third-party dependencies	▪ Some of important part of functionality located not in the blockchain
	▪ Optimization	▪ Gas consuming is not optimal
	▪ Logical issues	▪ Unused blocks of codes and illogical function behavior
	▪ Deployment Consistency	▪ No build and deploy scripts provided

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.