Cyberscope

Audit Report

vPoolv5

July 2022

SHA256 cfb4238fdc8200737adea6710c6303bc25f5672fb53ee47096d238ea38e4f751

Audited by © cyberscope

Table of Contents

Contract Review

Contract Name	vPoolv5
Explorer	https://testnet.bscscan.com/address/0x6b0a03c7Bd8441e0F8959394761E29Bd6afbfDf7
Domain	https://defilabs.farm

Audit Updates

Initial Audit	19th July 2022
Corrected	

Source Files

Filename	SHA256
@openzeppelin/contracts/access/Ownable.sol	4249cf7ca3111e505f92c0f23c5afcf3ec757055b0bc2babb7a637563ec7ddfd
@openzeppelin/contracts/GSN/Context.sol	0d30be46514c535ec373b07a5041ac484ca66a18fc66feb2b68c0e21103f1c01
@openzeppelin/contracts/math/Math.sol	188b811abb02569bcef02b41119b7653008c97d0609bbb1151e7e18c0ccc310e
@openzeppelin/contracts/math/SafeMath.sol	665f1eab7288dc1142b1330d74a42cf18bb24d1d9fbf1efbb17e0acb46a278dd
@openzeppelin/contracts/token/ERC20/IERC20.sol	d63052248b744c9a434cfb6feb4ac10aa5e4b9b852f728439777240b6af46b6d
@openzeppelin/contracts/token/ERC20/SafeERC20.sol	c4068e540290b83c45a8e52f2747de6b6ada924696f2499454df35633a4d4171
@openzeppelin/contracts/utils/Address.sol	23abcffc4cb1dc1d471500a17ba601ac92319f2d152ace49add276819d78d8cd
@openzeppelin/contracts/utils/ReentrancyGuard.sol	0bfdff81c9989c48ac53b71f89802cc37ee4774acaa6a6a33cab06195774fb26
contracts/interfaces/IEIP20.sol	8239f179f6b0e97e0588964cfe7b2ab2b8c7233caa692a5188fec144a2ff63d9
contracts/interfaces/IOracle.sol	876aff9492ecdae57325277ba4319b3f4364bf21b69cb19bbc0ecaaee3052f92

contracts/vPoolv5.sol	cfb4238fdc8200737adea6710c6303bc25f5672fb53ee47096d238ea38e4f751

Introduction

The vPoolv5 is a staking contract. Users can stake, restake, earn and exit from the pools and collect their rewards.

Stake

- Users have three options for locking their funds 1, 30, 90 Days.
- Users benefit index is calculated in relation with the funds that are invested in the pool.
- The bonus is calculated in relation with current lock duration since the last update on the stake and the full lock duration of the stake.
- Users reward index is updated on each stake with a random number.
- The contract keeps the user's stake state.

Restake

- Users can restake their funds with different stake options.
- Users earn rewards on each stake. The bonus is calculated in the same manner.
- The contract keeps the user's restake state.

Earn

- Users can earn bonuses at any moment. However they can not withdraw their bonus unless they exit the pool.

Exit

- Users can exit the pool and collect their bonus and staked amount. If there is no token address provided for the transfer, balance from the contact is transfered.

Operators have the ability to add new pools and change pool's benefits.

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description
Medium	ULTW	Contract Owner is not able to increase the amount of liquidity taken by dev wallet more than a reasonable percent
Minor	RV	Randomization Vulnerability
Minor	BLC	Business Logic Concern
Minor	CO	Code Optimization
Minor	CR	Code Repetition
Minor	RAV	Rewarded Amount Volatile
Minor	APYC	APY Calculation
Minor	L01	Public Function could be Declared External
Minor	L02	State Variables could be Declared Constant
Minor	L04	Conformance to Solidity Naming Conventions
Minor	L06	Missing Events Access Control
Minor	L11	Unnecessary Boolean equality
Minor	L13	Divide before Multiply Operation

ULTW - Unlimited Liquidity to Team Wallet

Criticality	medium
Location	contract.sol#L104

Description

The contract owner has the authority to transfer funds without limit to the team wallet. These funds have been accumulated from fees collected from the contract. The owner may take advantage of it by calling the withdrawFunds method.

```
function withdrawFunds(address _token, uint256 _amount) public payable nonReentrant {
        require(funder == msg.sender, "vPool: invalid recipient");

        if(address(_token) != address(0)) {
            IERC20(_token).safeTransfer(msg.sender, _amount);
        } else {
            safeTransferETH(msg.sender, _amount);
        }
    }
```

Recommendation

The contract could embody a check for the maximum amount of funds that can be swapped. Since a huge amount may volatile the token’s price.

The team should carefully manage the private keys of the owner’s account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

RV - Randomization Vulnerability

Criticality	minor
Location	contract.sol#L373

Description

The contract is using an on chain technique in order to determine random numbers. The blockchain runtime environment is fully deterministic, as a result, the pseudo-random numbers could be predicted. In addition, benefits are calculated in relation to the random. As a result the number can be zero so the user will not earn benefits.

```
uint256 random = doRand();
uint256 randBenefit = calcBenefit(benefit, random);
```

Recommendation

The contract could use an advanced randomization technique that quarandees an acceptable randomization factor. For instance, the Chainlink VRF (Verifiable Random Function). https://docs.chain.link/docs/chainlink-vrf/

BLC - Business Logic Concern

Criticality	minor
Location	contract.sol#L25

Description

The business logic seems peculiar. The implementation may not follow the expected behavior.

The variable isOpen is initialised as true. This variable is not modified anywhere and there is not a function that can modify it so it is redundant.

```
struct PoolInfo {
        uint256 pid;
        address token;
        bool isOpen;
        BenefitType benefitType;
        uint256 totalSupply;
        uint256 minAmount;
        uint256 maxbenefit; //apr of day
        mapping(address => uint256) balances;
        mapping(address => mapping(StakeType => uint256)) balanceOfStakedType;
        mapping(StakeType => uint256[]) benefits; //0.01 = 100/10000
        uint256 createdAt;
    }
```

Recommendation

The team is advised to carefully check if the implementation follows the expected business logic.

CO - Code Optimization

Criticality	minor
Location	contract.sol#L148,L166,L178

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

These code segments can be optimized. Every operation after `&&` is redundant.

```
function getBenefitIndex_7(uint256 amount_) public view returns (uint256) {
        if (amount_ >= oneUsdt.mul(20000)) {
            return 6;
        } else if (amount_ >= oneUsdt.mul(10000) && amount_ < oneUsdt.mul(20000)) {

function getBenefitIndex_4(uint256 amount_) public view returns (uint256) {
        if (amount_ >= oneUsdt.mul(20000)) {
            return 6;
        } else if (amount_ >= oneUsdt.mul(10000) && amount_ < oneUsdt.mul(20000)) {
            return 5;

 function getBenefitIndex_5(uint256 amount_) public view returns (uint256) {
        if (amount_ >= oneUsdt.mul(20000)) {
            return 6;
        } else if (amount_ >= oneUsdt.mul(10000) && amount_ < oneUsdt.mul(20000)) {
```

Recommendation

Rewrite some code segments so the runtime will be more performant.

CR - Code Repetition

Criticality	minor
Location	contract.sol#L148

Description

There are code segments that are repetitive in the contract. Those segments increase the code size of the contract unnecessarily.

The methods `getBenefitIndex_4`, `getBenefitIndex_5` could reuse the `getBenefitIndex_7`.

For `getBenefitIndex_4`, if the return value is < 3 then return 3

For `getBenefitIndex_4`, if the return value is < 2 then return 2

```
function getBenefitIndex_7(uint256 amount_) public view returns (uint256) {
        if (amount_ >= oneUsdt.mul(20000)) {
            return 6;
        } else if (amount_ >= oneUsdt.mul(10000) && amount_ < oneUsdt.mul(20000)) {
            return 5;
        }  else if (amount_ >= oneUsdt.mul(5000) && amount_ < oneUsdt.mul(10000)) {
            return 4;
        } else if (amount_ >= oneUsdt.mul(3000) && amount_ < oneUsdt.mul(5000)) {
            return 3;
        } else if (amount_ >= oneUsdt.mul(1000) && amount_ < oneUsdt.mul(3000)) {
            return 2;
        } else if (amount_ >= oneUsdt.mul(500) && amount_ < oneUsdt.mul(1000)) {
            return 1;
        } else {
            return 0; //[1, 500)
        }
}
```

Recommendation

Create an internal function that contains the code segment and remove it from all the sections.

RAV - Rewarded Amount Volatile

Criticality	minor
Location	contract.sol#L313

Description

The reward amount is calculated based on the USDT price of the initial stake. That means that during the exit, the reward funds will potentially be heavily volatile.

```
function calcBonus(uint256 _pid, StakeType _stakeType, address _account) public view
returns(uint256) {
        UserInfo memory userInfo = userInfos[_pid][_stakeType][_account];
        uint256 lockDuration = userInfo.lockDuration;
        if (lockDuration == 0) {
            return 0;
        }

        uint256 lastUpdateTime = userInfo.lastUpdateTime;
        uint256 unLockedTime = userInfo.unLockedTime;
        uint256 apy = userInfo.apy;
        uint256 totalBonus = userInfo.totalValue.mul(apy).div(denominator);
        uint256 bonusRate = totalBonus.div(lockDuration);

        if (block.timestamp > unLockedTime) {
            return bonusRate.mul(unLockedTime - lastUpdateTime);
        } else {
            return bonusRate.mul(block.timestamp - lastUpdateTime);
        }
    }
```

Recommendation

It is recommended to use fixed values on the calculation of the benefits in order to avoid heavily volite rewards.

APY - APY calculation

Criticality	minor
Location	contract.sol#L314

Description

Since apy is divided by the lockDuration, it's not an apy, it is a rate. Annual Percentage Yield (APY) should be divided by the year.

```
function calcBonus(uint256 _pid, StakeType _stakeType, address _account) public view
returns(uint256) {
        UserInfo memory userInfo = userInfos[_pid][_stakeType][_account];
        uint256 lockDuration = userInfo.lockDuration;
        if (lockDuration == 0) {
            return 0;
        }

        uint256 lastUpdateTime = userInfo.lastUpdateTime;
        uint256 unLockedTime = userInfo.unLockedTime;
        uint256 apy = userInfo.apy;
        uint256 totalBonus = userInfo.totalValue.mul(apy).div(denominator);
        uint256 bonusRate = totalBonus.div(lockDuration);

        if (block.timestamp > unLockedTime) {
            return bonusRate.mul(unLockedTime - lastUpdateTime);
        } else {
            return bonusRate.mul(block.timestamp - lastUpdateTime);
        }
    }
```

Recommendation

It is recommended to use precise definitions on the variable names. In order to make the code readable.

L01 - Public Function could be Declared External

Criticality	minor
Location	contracts/vPoolv5.sol#L249,128,465,271,351,284,132,428,297,104,263,259,267,114

Description

Public functions that are never called by the contract should be declared external to save gas.

```
provideFunds
totalSupply
balanceOf
getBalanceOfStakedType
withdrawFunds
earned
reStake
getStakedType
getRecordInfo
...
```

Recommendation

Use the external attribute for functions never called from the contract.

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contracts/vPoolv5.sol#L84

Description

Constant state variables should be declared constant to save gas.

```
bonusFeeRate
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contracts/vPoolv5.sol#L136,71,303,259,65,104,70,271,323,267,72,128,339,178,220,236,351,114,297,263,148,69,465,67,216,249,428,66,166,14,208,212,62,284,64

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
_pid
_amount
_account
C_Day1
_stakeType
_recordId
denominator
_newOracle
_operator
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions.

L06 - Missing Events Access Control

Criticality	minor
Location	contracts/vPoolv5.sol#L208

Description

Detected missing events for critical access control parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.

```
policyOperator = _operator
```

Recommendation

Emit an event for critical parameter changes.

L11 - Unnecessary Boolean equality

Criticality	minor
Location	contracts/vPoolv5.sol#L351,465,428

Description

The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
require(bool,string)(poolInfo.isOpen == true,vPool: pool is not opened)
```

Recommendation

Remove the equality to the boolean constant.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contracts/vPoolv5.sol#L303,323

Description

Performing divisions before multiplications may cause lose of prediction.

```
b = _benefit.mul(19).div(100)
bonusRate = totalBonus.div(lockDuration)
```

Recommendation

The multiplications should be prior to the divisions.

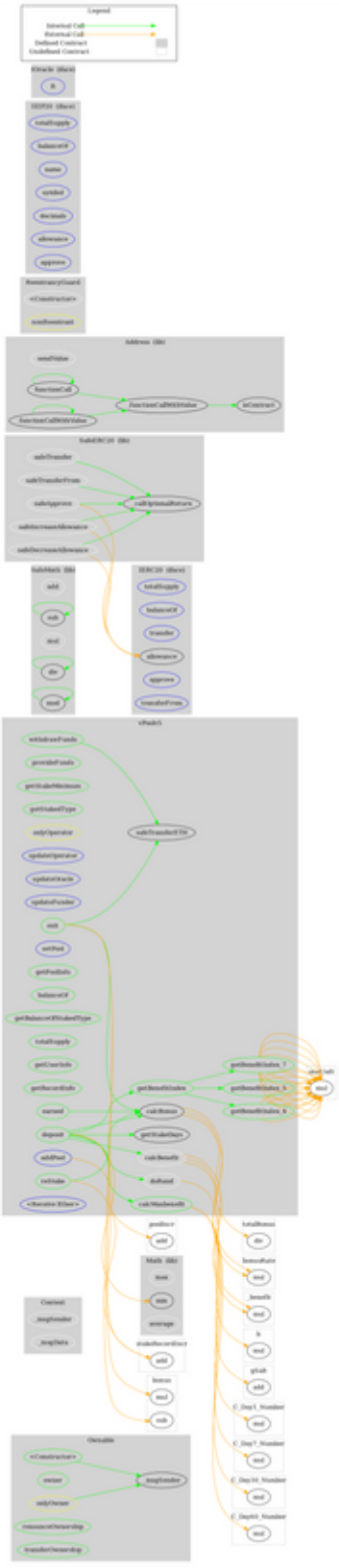
Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
Ownable	Implementation	Context		
	<Constructor>	Internal	✓	
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Math	Library			
	max	Internal		
	min	Internal		
	average	Internal		
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-

	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeERC20	Library			
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeApprove	Internal	✓	
	safeIncreaseAllowance	Internal	✓	
	safeDecreaseAllowance	Internal	✓	
	_callOptionalReturn	Private	✓	
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	_functionCallWithValue	Private	✓	
ReentrancyGuard	Implementation			
	<Constructor>	Internal	✓	
IEIP20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	name	External		-
	symbol	External		-
	decimals	External		-
	allowance	External		-
	approve	External	✓	-
IOracle	Interface			

	R	External		-
vPoolv5	Implementation	Ownable, Reentrancy Guard		
	<Constructor>	Public	✓	-
	withdrawFunds	Public	Payable	nonReentrant
	provideFunds	Public	Payable	nonReentrant
	getStakeMinimum	Public		-
	getStakedType	Public		-
	getStakeDays	Public		-
	getBenefitIndex_7	Public		-
	getBenefitIndex_4	Public		-
	getBenefitIndex_5	Public		-
	getBenefitIndex	Public		-
	updateOperator	External	✓	onlyOwner
	updateOracle	External	✓	onlyOwner
	updateFunder	External	✓	onlyOwner
	addPool	External	✓	onlyOperator
	setPool	External	✓	onlyOperator
	getPoolInfo	Public		-
	balanceOf	Public		-
	getBalanceOfStakedType	Public		-
	totalSupply	Public		-
	getUserInfo	Public		-
	getRecordInfo	Public		-
	earned	Public		-
	calcBonus	Public		-
	calcBenefit	Internal		
	doRand	Internal	✓	
	calcMaxbenefit	Public		-
	deposit	Public	Payable	nonReentrant
	reStake	Public	✓	nonReentrant
	exit	Public	✓	nonReentrant
	safeTransferETH	Internal	✓	
	<Receive Ether>	External	Payable	-

Contract Flow



Domain Info

Domain Name	defilabs.farm
Registry Domain ID	d44f7165186c43e6ab7e5570545b2f9e-DONUTS
Creation Date	2021-09-23T12:54:45Z
Updated Date	2022-07-18T09:44:52Z
Registry Expiry Date	2024-09-23T12:54:45Z
Registrar WHOIS Server	http://whois.cloudflare.com
Registrar URL	http://cloudflare.com
Registrar	Cloudflare, Inc
Registrar IANA ID	1910

The domain has been created in about 2 years before the creation of the audit.

There is no public billing information, the creator is protected by the privacy settings.

Summary

The vPool5 contract works like a staking contract where you can invest funds to earn interest. The Smart Contract analysis reported one critical severity issue. The contract owner has the authority to transfer funds to the team's wallet. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

Disclaimer

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment.

Cyberscope team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed.

The Cyberscope team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Cyberscope receive a payment to manipulate those results or change the awarding badge that we will be adding in our website.

Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token.

The Cyberscope team disclaims any liability for the resulting losses.

About Cyberscope

Coinscope audit and K.Y.C. service has been rebranded to Cyberscope.

Coinscope is the leading early coin listing, voting and auditing authority firm. The audit process is analyzing and monitoring many aspects of the project. That way, it gives the community a good sense of security using an informative report and a generic score.

Cyberscope and Coinscope are aiming to make crypto discoverable and efficient globally. They provide all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io