Cyberscope

Audit Report

InfinityPool

July 2022

SHA256 48546381fd72feec1265191edc9864de4ffb8dcd00ea90ef7d9b966b0df09c94

Audited by © cyberscope

Table of Contents

Table of Contents 1
Contract Review 3
Audit Updates 3
Source Files 4
Introduction 6
    Buy 6
    Sale 6
Contract Diagnostics 7
    ST - Stop Transactions 8
        Description 8
        Recommendation 8
    OCTD - Owner Contract Tokens Drain 9
        Description 9
        Recommendation 9
    ZD - Zero Division 10
        Description 10
        Recommendation 10
    BLC - Business Logic Concern 11
        Description 11
        Recommendation 11
    MC - Missing Check 12
        Description 12
        Recommendation 12
    L04 - Conformance to Solidity Naming Conventions 13
        Description 13
        Recommendation 13

L07 - Missing Events Arithmetic 14
Description 14
Recommendation 14
Contract Functions 15
Contract Flow 18
Domain Info 19
Summary 20
Disclaimer 21
About Cyberscope 22

Contract Review

Contract Name	InfinityPool
Test Deploy	https://testnet.bscscan.com/address/0xF63499a77c0d96AE8BEc3367FfCa0fB58AbF8B5a
Domain	https://hyfinance.net

Audit Updates

Initial Audit	15th July 2022
Corrected	

Source Files

Filename	SHA256
@openzeppelin/contracts/access/Ownable.sol	754825f501dd014526eee0c415687b0f6c600533adfc872f7d45edb4f8b3b053
@openzeppelin/contracts/math/SafeMath.sol	f6d6214aa03f8dd6d6d14b7c15ffa387b3f1ce38ba3a215177baa132a44636e2
@openzeppelin/contracts/token/ERC20/IERC20.sol	c4b741712b8dc93ab3945205554a3ba2f80953e64d684e752d5a0fd07fc93f22
@openzeppelin/contracts/token/ERC20/SafeERC20.sol	74e10f4538df92e1c89140f16654914be8d7e9a66b24d6272ff0f28f89f8728b
@openzeppelin/contracts/utils/Address.sol	a22903d00a93aa211164d90ad11f01ccc7d34648114be89ec38c859fdea0f8d4
@openzeppelin/contracts/utils/Context.sol	eafb62c654640a07832b56e00902b4bf249633346585331af311c738b1c23bc5
@openzeppelin/contracts/utils/Pausable.sol	e59e348bb0a6a4a7f5f88896f6a1b9f151b9857bf362bb2aa431b910ee579eea
@openzeppelin/contracts/utils/ReentrancyGuard.sol	a84a635e520d932183fc216c6f0ec109f8578149b15a91c728557a370430882a
contracts/InfinityPool.sol	48546381fd72feec1265191edc9864de4ffb8dcd00ea90ef7d9b966b0df09c94
contracts/interfaces/IERC20Meta.sol	6d83cc8a7eb156aec4ac633bfe9d8bcc330654dddbecc6601f78bfafe9abb064

contracts/interfaces/ITokenSwap.sol	8625a61d08e26e782e40db0d5d0db6fa8e70363972ea56c919c61dffa35b9b69

Introduction

The InfinityPool contract core functionality is to exchange USDC or USDP tokens to Hybrid Finance version 2 tokens.

The price of Hybrid Finance token version 2 depends on the total supply in relation to the balance of USDC and USDP of the InfinityPool. That means that InfinityPool works as a liquidity pool provider. If the contract does not contain USDC/USDP tokens, then it will not be able to perform transactions.

Buy

- The buy value is 15% decrease in relation to the sell value.
- The user takes a proportional amount of token in correlation with the current token price.
- The investment is distributed to the treasury (10%) and to InfinityPool balance (90%).
- The contract keeps balance between the USDC and the USDP tokens. The USDP is 10 times more than the USDC. This is achieved by swapping USDC to USDP when the ratio is changing.

Sale

- The user has the ability to exchange tokens for USDC.
- If the contract USDC balance is not sufficient for the transaction the contract exchanges USDP to USDC.

Contract Diagnostics

🔴 Critical 🟠 Medium ⚪ Minor

Severity	Code	Description
⚪	ST	Contract Owner is not able to stop or pause transactions
🟠	OCTD	Contract Owner is not able to transfer tokens from specific address
⚪	ZD	Zero Division
⚪	BLC	Business Logic Concern
⚪	MC	Missing Check
⚪	L04	Conformance to Solidity Naming Conventions
⚪	L07	Missing Events Arithmetic

ST - Stop Transactions

Criticality	minor
Location	contract.sol#L110,L128

Description

The contract owner has the authority to pause transactions for all users. The owner may take advantage of it by using the `pause` function.

```
function sell(uint256 tokenAmount) external whenNotPaused nonReentrant

function buy(address token, uint256 usdAmount) external whenNotPaused nonReentrant
```

Recommendation

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.

OCTD - Owner Contract Tokens Drain

Criticality	medium
Location	contract.sol#L62

Description

The contract owner has the authority to claim all the balance of the contract. The owner may take advantage of it by calling the adminWithdraw function.

```
function adminWithdraw(address token, uint256 amount) external onlyOwner {
        require(IERC20Meta(token).balanceOf(address(this)) >= amount, "Amount too high");
        IERC20Meta(token).safeTransfer(msg.sender, amount);
}
```

Recommendation

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.

ZD - Zero Division

Criticality	minor
Location	contracts#L114

Description

The contract is using variables that may be set to zero as denominators. As a result, the transactions will revert.

The getPrice calculates the price in relation with USDC and USDP price. That means that the balances of USDC and USDP have to be greater than zero initially in order to avoid zero division.

```
function buy(address token, uint256 usdAmount) external whenNotPaused nonReentrant {
        require(token == usdc || token == usdp, "Token not allowed");
        require(usdAmount > 0, "Cannot be 0");
        uint256 price = getPrice();
        uint256 tokenAmount = usdAmount.mul(1e18).mul(85).div(price).div(100);
```

Recommendation

The contract should prevent those variables to be set to zero or should not allow them to execute the corresponding statements.

BLC - Business Logic Concern

Criticality	minor
Location	contracts#L10,L140

Description

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

The buy method accumulates USDC/USDP tokens. It keeps the ratio 1/10 with the USDP tokens by swapping USDP with USDC. The distribute() method followers the same pattern. It accumulates tokens but it does not keep the 1/10 ratio.

```
IERC20Meta(token).safeTransferFrom(msg.sender, address(this), poolAmount);
if (token == usdc) {
    _handleUsdcBuy(poolAmount);
}
```

Recommendation

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

MC - Missing Check

Criticality	minor
Location	contracts

Description

The contract could early check if the user has the necessary balance before it performs transactions. The contract should not rely on that the external methods will fail if the user's balance is not sufficient. For instance, the sell method does not check if the user's balance is enough. It rallies in the fact that the `burnFrom` will handle this issue. The early check could be added in the buy, sell, distribute and release methods.

```
function sell(uint256 tokenAmount) external whenNotPaused nonReentrant {
    require(tokenAmount > 0, "Cannot be 0");
    uint256 price = getPrice();
    uint256 usdAmount = tokenAmount.mul(price).div(1e18);

    _handleSell(usdAmount);
    hybridv2.burnFrom(msg.sender, tokenAmount);
    IERC20Meta(usdc).safeTransfer(msg.sender, usdAmount);

    emit Sold(msg.sender, price, tokenAmount, usdAmount);
}
```

Recommendation

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

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contracts/InfinityPool.sol#L45,58,54,50

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.

```
_maxSwapRatio
_usdpRatio
_swapContract
_treasury
```

Recommendation

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

L07 - Missing Events Arithmetic

Criticality	minor
Location	contracts/InfinityPool.sol#L50

Description

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

```
maxSwapRatio = _maxSwapRatio
```

Recommendation

Emit an event for critical parameter changes.

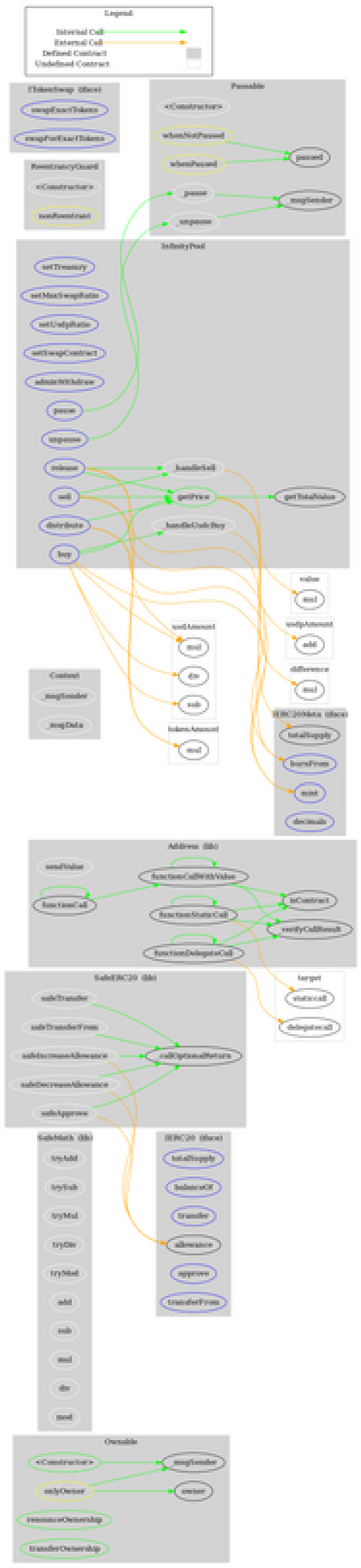
Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Ownable	Implementation	Context		
	<Constructor>	Internal	✓	
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
SafeMath	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	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	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	_verifyCallResult	Private		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Pausable	Implementation	Context		
	<Constructor>	Internal	✓	
	paused	Public		-
	_pause	Internal	✓	whenNotPaused
	_unpause	Internal	✓	whenPaused
ReentrancyGuard	Implementation			
	<Constructor>	Internal	✓	

InfinityPool	Implementation	Ownable, Pausable, Reentrancy Guard		
	<Constructor>	Public	✓	-
	setTreasury	External	✓	onlyOwner
	setMaxSwapRatio	External	✓	onlyOwner
	setUsdpRatio	External	✓	onlyOwner
	setSwapContract	External	✓	onlyOwner
	adminWithdraw	External	✓	onlyOwner
	pause	External	✓	onlyOwner
	unpause	External	✓	onlyOwner
	getPrice	Public		-
	getTotalValue	Public		-
	_handleUsdcBuy	Internal	✓	
	_handleSell	Internal	✓	
	buy	External	✓	whenNotPaused nonReentrant
	sell	External	✓	whenNotPaused nonReentrant
	distribute	External	✓	nonReentrant
	release	External	✓	nonReentrant
IERC20Meta	Interface	IERC20		
	decimals	External		-
	burnFrom	External	✓	-
	mint	External	✓	-
ITokenSwap	Interface			
	swapExactTokens	External	✓	-
	swapForExactTokens	External	✓	-

Contract Flow



Domain Info

Domain Name	hyfinance.net
Registry Domain ID	2683607355_DOMAIN_NET-VRSN
Creation Date	2022-03-22T21:24:53.00Z
Updated Date	0001-01-01T00:00:00.00Z
Registry Expiry Date	2023-03-22T21:24:53.00Z
Registrar WHOIS Server	whois.namecheap.com
Registrar URL	http://www.namecheap.com
Registrar	NAMECHEAP INC
Registrar IANA ID	1068

The domain has been created in 8 months before the creation of the audit.

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

Summary

The InfinityPool works similar to a DAO market maker mechanism. It exchanges USDT/USDP for Hybrid Finance tokens. There are some functions that can be abused by the owner like stopping transactions and transferring tokens to the team’s wallet. We state that the owner privileges are necessary and required for proper protocol operations. Thus, we emphasise the contract owner to be extra careful with the credentials.

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