



Cyberscope

Audit Report

Circle Launchpad Pool

January 2022

Github <https://github.com/monkey-shanti/Circle-Launchpad>

Commit [915604357d2528c77dbdb6672471c2bcc9b8bb2a](https://github.com/monkey-shanti/Circle-Launchpad/commit/915604357d2528c77dbdb6672471c2bcc9b8bb2a)

Audited by © cyberscope

Table of Contents

Table of Contents	1
Contract Review	4
Audit Updates	4
Source Files	5
Introduction	7
CirclePoolFactory	7
Roles	7
Owner Role	7
User Role	8
CirclePoolManager	9
Roles	9
Owner Role	9
AllowedFactory Role	9
User Role	9
CircleFairPool	11
Pool States	11
Roles	11
Owner Role	11
Operator Role	11
Governance Role	11
User Role	11
CirclePrivatePool	13
Pool States	13
Roles	13
Owner Role	13
Operator Role	13
Governance Role	13
Whitelisted Role	14
User Role	14
CirclePresalePool	15
Pool States	15
Roles	15

Owner Role	15
Operator Role	15
Governance Role	16
User Role	16
Contract Diagnostics	17
CRI - Code Readability Issue	18
Description	18
Recommendation	18
MTV - Missing Token Validation	19
Description	19
Recommendation	19
RDS - Redundant Data Structure	20
Description	20
Recommendation	20
L04 - Conformance to Solidity Naming Conventions	21
Description	21
Recommendation	22
L07 - Missing Events Arithmetic	23
Description	23
Recommendation	23
L08 - Tautology or Contradiction	24
Description	24
Recommendation	24
L11 - Unnecessary Boolean equality	26
Description	26
Recommendation	26
L12 - Using Variables before Declaration	27
Description	27
Recommendation	27
L13 - Divide before Multiply Operation	28
Description	28
Recommendation	28
L14 - Uninitialized Variables in Local Scope	29
Description	29
Recommendation	29

Contract Functions	30
Contract Flow	42
Inheritance Graph	43
Summary	43
Disclaimer	44
About Cyberscope	46

Contract Review

Repository	https://github.com/monkey-shanti/Circle-Launchpad
Commit	915604357d2528c77dbdb6672471c2bcc9b8bb2a

Contract Name	Testing Deploy
CirclePoolManager	https://testnet.bscscan.com/address/0xE0508740f095501d793aBfb6A47F0b73e921b132
CirclePoolFactory	https://testnet.bscscan.com/address/0x8eBc03B07914a0fABb086104F2ce28e07840150B
CircleFairPool	https://testnet.bscscan.com/address/0x6A972E15554f1421Ba7A446c0AB6737f81a0E292
CirclePrivatePool	https://testnet.bscscan.com/address/0xB677fd0F7BFD0B81ef354930A094F44967Ff5454
CirclePresalePool	https://testnet.bscscan.com/address/0xB0CEb88b15Cc59aAE03f04B81da3ee3Cc2FB3988

Audit Updates

Initial Audit	20 Dec 2022 https://github.com/cyberscope-io/audits/blob/main/circleLaunchpad/v1/pool.pdf
Corrected Phase 2	02 Jan 2023

Source Files

Filename	SHA256
FairPool.sol	0612b023512e66e3dcebc338c1541fd90446dd8c2817a639687c9c941c35fa2b
interfaces/IERC20Info.sol	1ca3d43543f838b7014dc2021dc5338492249523f0d9038eff73befa3f15ed1b
interfaces/IFairPool.sol	2bf91078f72b43b4775e8288dd81ab57c79d11b142b068891fd679630220e9a2
interfaces/IPool.sol	c015d0fd8888dade1d9e7a492e49ccdfa4d68a3c328d68a52575bb4764e073b2
interfaces/IPoolFactory.sol	4ecdb74fc7eedf0ca1d31a5099a89e970b8ad199f566bfff62195ab7a014b8752
interfaces/IPoolManager.sol	c2ac0c23d4936946b8558f86bd57c7f602a6912e013af00a49c48d3532f48654
interfaces/IPrivatePool.sol	7d0a66447b55878209e03717a70e424c4e7149dd35f610920ae3dc7bf4c2ca6b
interfaces/IUniswapV2Pair.sol	797d818f0b1ce3bccbdec7c5158babae5d4082d7feab73dc92fed7d661dc926
interfaces/libraries/LibEnsureSafeTransfer.sol	03f9e4a97f22127d967d3064a874192bc5ff6c69f8eaf5a4c16c9d58bdb4d661
interfaces/libraries/LibPresale.sol	af71027b0d7f1e8ca1a81c8c5fa77ca6d71e665ea8cfbb63875a94750384e962
interfaces/libraries/LibTier.sol	4aedc295fb6e752b1beea948f23aac1f29f885eaa9891df7388c59d2110a4d3b
interfaces/PoolLibrary.sol	bea8cc1b2ccf49e29be71baa6e9e105a67c56a3094cefbdd271b5104c7026aa23
libraries/LibEnsureSafeTransfer.sol	03f9e4a97f22127d967d3064a874192bc5ff6c69f8eaf5a4c16c9d58bdb4d661
libraries/LibPresale.sol	af71027b0d7f1e8ca1a81c8c5fa77ca6d71e665ea8cfbb63875a94750384e962
libraries/LibTier.sol	4aedc295fb6e752b1beea948f23aac1f29f885eaa9891df7388c59d2110a4d3b

PoolFactory.sol	a7b143816b72894d3cd049e26a52c5866 e50b75ca13a9e99806c2f7e48e24e5b
PoolManager.sol	b13f1a5ad88f304976a1f4982438e70e08 a4f2875256365a08febd82ec42f1c7
PresalePool.sol	a3f89fc7e9f2f6d0c5788c120bac7f9ab78 a2baa1edf313be675f6f6ccc63bf0
PrivatePool.sol	f6789e5ddb6d02e9fb8515b562757bf4b7 9599398aea49ae9677c45a9d35f8b3
utils/Utility.sol	bb982ca156ddbd0ea26ba803843a755ff 4ad6440addadc577d3ba8335f8e0705

Introduction

The Circle launchpad pool implements a presale mechanism. It consists of a factory, a manager, a fair pool, a presale pool, and a private pool contract.

CirclePoolFactory

The Circle PoolFactory is responsible for creating new presales.

Roles

The contract has two roles.

Owner Role

The owner role has the authority to

- setMasterAddress
- setFairAddress
- setPrivateAddress
- setAdminWallet
- setPartnerFee
- setVersion
- setFees
- setPaymentCurrency
- setPoolOwner
- setkycPrice
- setAuditPrice
- setPresalePoolPrice
- setPrivatePoolPrice
- setFairPoolPrice
- setPoolManager
- bnbLiquidity
- transferAnyERC20Token
- poolEmergencyWithdrawLiquidity

- poolEmergencyWithdrawToken
- poolEmergencyWithdraw
- poolSetGovernance

User Role

The user has the authority to

- getFee
- getPaymentCurrency
- createSale
- createPrivateSale
- createFairSale

CirclePoolManager

The Circle Pool Manager is responsible for adding or removing the presale. Additionally, it's responsible for monitoring pool factories and keeping registries about them.

Roles

The contract has three roles.

Owner Role

The owner role has the authority to

- addAdminPoolFactory
- addPoolFactories
- removePoolFactory
- initializeTopPools
- bnbLiquidity
- transferAnyERC20Token

AllowedFactory Role

The Operator has the authority to

- addPoolFactory
- registerPool
- registerPrivatePool
- increaseTotalValueLocked
- decreaseTotalValueLocked
- recordContribution
- removePoolForToken
- removePrivatePoolForToken
- addTopPool
- removeTopPool

User Role

The user has the authority to

- isPoolGenerated

- poolForToken
- privatePoolForToken
- getPoolsOf
- getAllPools
- getPoolAt
- getTotalNumberOfContributedPools
- getAllContributedPools
- getContributedPoolAtIndex
- getTotalNumberOfPools
- getPoolAt
- getTopPool
- getCumulativePoolInfo

CircleFairPool

The Circle FairPool implements a fair launch mechanism.

Pool States

The pool has 4 states.

- inProgress
- notInProgress

Roles

The contract has two roles.

Owner Role

The owner role has the authority to

- emergencyWithdrawLiquidity
- emergencyWithdrawToken
- emergencyWithdraw
- updateCompletedKyc
- setGovernance

Operator Role

The Operator has the authority to

- finalize
- cancel
- withdrawLeftovers
- withdrawLiquidity
- updatePoolDetails

Governance Role

The Governance role is not utilized on the contract implementation.

User Role

The user has the authority to

- contribute
- claim

- withdrawContribution
- getPrice
- getPoolInfo
- convert
- View userAvalibleClaim

CirclePrivatePool

The Circle PrivatePool implements a private presale.

Pool States

The pool has 3 states.

- inUse
- completed
- cancelled

Roles

The contract has two roles.

Owner Role

The owner role has the authority to

- emergencyWithdrawToken
- emergencyWithdraw
- setGovernance

Operator Role

The Operator has the authority to

- setWhitelist
- finalize
- cancel
- withdrawLeftovers
- updatePoolDetails
- startPublicSaleNow
- startTier2SaleNow
- changeWhitelist
- changeTierDates

Governance Role

The Governance role is not utilized in the contract implementation.

Whitelisted Role

The Whitelisted role is not utilized in the contract implementation.

User Role

The user has the authority to

- `getPoolInfo`
- `getNumberOfWhitelistedUsers`
- `getWhitelistedUsers`
- `contribute`
- `withdrawContribution`
- `claim`
- `getContributionAmount`
- `remainingContribution`
- `userAvalibleClaim`
- `getTier`

CirclePresalePool

The Circle Presale rPool implements a regular presale mechanism.

Pool States

The pool has 3 states.

- inUse
- completed
- cancelled

Roles

The contract has two roles.

Owner Role

The owner role has the authority to

- emergencyWithdrawLiquidity
- emergencyWithdrawToken
- emergencyWithdraw
- setGovernance

Operator Role

The Operator has the authority to

- addWhitelistedUsers
- removeWhitelistedUsers
- finalize
- cancel
- withdrawLiquidity
- updatePoolDetails
- startPublicSaleNow
- changeWhitelist
- startTier2SaleNow
- changeTierDates

Governance Role

The Governance role is not utilized on the contract implementation.

User Role

The user has the authority to

- `getNumberOfWhitelistedUsers`
- `getWhitelistedUsers`
- `getPoolInfo`
- `contribute`
- `claim`
- `withdrawContribution`
- `getContributionAmount`
- `remainingContribution`
- `userAvailableClaim`

Contract Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	CRI	Code Readability Issue	Unresolved
●	MTV	Missing Token Validation	Unresolved
●	RDS	Redundant Data Structure	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L08	Tautology or Contradiction	Unresolved
●	L11	Unnecessary Boolean equality	Unresolved
●	L12	Using Variables before Declaration	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved

CRI - Code Readability Issue

Criticality	Minor / Informative
Location	PoolManager.sol#L132,145,339,416
Status	Unresolved

Description

There are code segments that are hard to read. It is unclear for the reader to distinguish which pool state is utilized `poolState[pool] = 1`.

```
_poolState[pool] = 1;
```

Recommendation

The team is advised to utilize descriptive values like an enumeration, which can make the contract easier to read and maintain.

MTV - Missing Token Validation

Criticality	Minor / Informative
Location	PrivatePool.sol#L165
Status	Unresolved

Description

Payment tokens are not validated in Private Pool like the Presale Pool and Fair Pool. The contract is processing payment tokens that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

```
address public paymentToken;
```

Recommendation

The team is advised to properly check the variables according to the required specifications.

RDS - Redundant Data Structure

Criticality	Minor / Informative
Location	PresalePool.sol#L96,97 PrivatePool.sol#L36,37
Status	Unresolved

Description

The contract utilizes two data structures with the same information.

- One mapping to keep a registry of the amount that the users contributed.
- One mapping that keeps a registry of the amount that the users purchased.

The contribution amount is proportional to the purchased amount. The rate between these two amounts is fixed. Storing two state variables increases gas consumption and decreases readability.

```
mapping(address => uint256) public contributionOf;  
mapping(address => uint256) public purchasedOf;
```

Recommendation

The team is advised to remove the purchasedOf data structure. The information could be reshaped by using the proportional contributionOf value. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	PrivatePool.sol#L26,166,167,168,316,522,583,588,609 PresalePool.sol#L25,26,29,191,192,193,350,638,648,657 PoolManager.sol#L67,412 PoolFactory.sol#L39,65,66,67,68,69,70,71,72,73,98,102,106,110,114,118,123,129,152,167,168,238,239,285,286,413,418,423,428,433,438,485 FairPool.sol#L25,26,40,153,154,155,213,458
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
uint8 public VERSION
uint256[2] memory _fees
address[3] memory _linkAddress
uint8 _version
uint256 _funds
address _userAddress
bool _whitelist
uint256 _endTime
uint256 _tier
uint public MINIMUM_LOCK_DAYS = 30 days

...
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation

<https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention>.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	PoolFactory.sol#L125
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
feeIndex = _index
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L08 - Tautology or Contradiction

Criticality	Minor / Informative
Location	PrivatePool.sol#L175,183,184,185,590 PresalePool.sol#L204,216,217,218,659 PoolFactory.sol#L124,130 FairPool.sol#L167
Status	Unresolved

Description

A tautology is a logical statement that is always true, regardless of the values of its variables. A contradiction is a logical statement that is always false, regardless of the values of its variables.

Using tautologies or contradictions can lead to unintended behavior and can make the code harder to understand and maintain. It is generally considered good practice to avoid tautologies and contradictions in the code.

```
require(  
    _fees[1] >= 0 &&  
    _fees[1] <= 100 &&  
    _fees[0] >= 0 &&  
    _fees[0] <= 100,  
    "Invalid fee settings. Must be percentage (0 -> 100)"  
)  
require(presale.cycle >= 0, "Invalid cycle")  
require(presale.tgeBps >= 0 && presale.tgeBps < 10_000, "Invalid bips  
for TGE")  
require(presale.cycleBps >= 0 && presale.cycleBps < 10_000, "Invalid  
bips for cycle")  
_endTime >= 0  
require(_index >= 0, "Invalid Fee Index")  
  
...
```

Recommendation

The team is advised to carefully consider the logical conditions is using in the code and ensure that it is well-defined and make sense in the context of the smart contract.

L11 - Unnecessary Boolean equality

Criticality	Minor / Informative
Location	PoolFactory.sol#L194,258,320
Status	Unresolved

Description

Boolean equality is unnecessary when comparing two boolean values. This is because a boolean value is either true or false, and there is no need to compare two values that are already known to be either true or false.

it's important to be aware of the types of variables and expressions that are being used in the contract's code, as this can affect the contract's behavior and performance. 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(paymentCurrencies[presale.paymentToken] == true || address(0)  
== presale.paymentToken, "Invalid payment token")
```

Recommendation

Using the boolean value itself is clearer and more concise, and it is generally considered good practice to avoid unnecessary boolean equalities in Solidity code.

L12 - Using Variables before Declaration

Criticality	Minor / Informative
Location	PoolManager.sol#L355
Status	Unresolved

Description

The contract is using a variable before the declaration. This is usually happening either if it has not been declared yet or if the variable has been declared in a different scope. It is not a good practice to use a local variable before it has been declared.

```
string memory poolDetails
```

Recommendation

By declaring local variables before using them, contract ensures that it operates correctly. It's important to be aware of this rule when working with local variables, as using a variable before it has been declared can lead to unexpected behavior and can be difficult to debug.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	PrivatePool.sol#L421,546 PresalePool.sol#L427 PoolFactory.sol#L392,395
Status	Unresolved

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of prediction.

```
currentTotal =  
    (((block.timestamp - tgeDate) / cycle) *  
    cycleReleaseAmount) +  
    tgeReleaseAmount  
uint256 currentTotal = (((block.timestamp - tgeDate) / cycle) *  
    cycleReleaseAmount) + tgeReleaseAmount  
uint256 totalFees = (((((_rate * _hardcap) / 10**decimals)) * _fees) /  
100)
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	PoolManager.sol#L350,353,354,382,383
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
address token
string memory name
string memory symbol
string memory poolDetails
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
CircleFairPool	Implementation	OwnableUp gradeable, IFairPool, Reentrancy Guard, Utility		
		External	Payable	-
	initialize	External	✓	initializer
	getDecimal	Public		-
	contribute	Public	Payable	inProgress
	claim	Public	✓	nonReentrant
	withdrawContribution	External	✓	nonReentrant
	finalize	External	✓	onlyOperator nonReentrant
	getPrice	Public		-
	getPoolInfo	External		-
	cancel	External	✓	onlyOperator
	withdrawLeftovers	External	✓	onlyOperator
	withdrawLiquidity	External	✓	onlyOperator
	emergencyWithdrawLiquidity	External	✓	onlyOwner
	emergencyWithdrawToken	External	✓	onlyOwner
	emergencyWithdraw	External	✓	onlyOwner
	updatePoolDetails	External	✓	onlyOperator
	updateCompletedKyc	External	✓	onlyOwner
	setGovernance	External	✓	onlyOwner validAddress

	userAvalibleClaim	Public		-
IERC20Info	Interface			
	decimals	External		-
	name	External		-
	symbol	External		-
	supply	External		-
IFairPool	Interface			
	initialize	External	✓	-
	emergencyWithdrawLiquidity	External	✓	-
	emergencyWithdraw	External	✓	-
	setGovernance	External	✓	-
	emergencyWithdrawToken	External	✓	-
	getPoolInfo	External		-
IPool	Interface			
	initialize	External	✓	-
	emergencyWithdrawLiquidity	External	✓	-
	emergencyWithdraw	External	✓	-
	setGovernance	External	✓	-
	emergencyWithdrawToken	External	✓	-
	getPoolInfo	External		-
IPoolFactory	Interface			
	increaseTotalValueLocked	External	✓	-
	decreaseTotalValueLocked	External	✓	-
	removePoolForToken	External	✓	-
	recordContribution	External	✓	-

	addTopPool	External	✓	-
	removeTopPool	External	✓	-
	removePrivatePoolForToken	External	✓	-
IPoolManager	Interface			
	increaseTotalValueLocked	External	✓	-
	decreaseTotalValueLocked	External	✓	-
	removePoolForToken	External	✓	-
	removePrivatePoolForToken	External	✓	-
	recordContribution	External	✓	-
	isPoolGenerated	External		-
	addTopPool	External	✓	-
	removeTopPool	External	✓	-
	registerPool	External	✓	-
	registerPrivatePool	External	✓	-
	poolForToken	External		-
	privatePoolForToken	External		-
	addPoolFactory	External	✓	-
IPrivatePool	Interface			
	initialize	External	✓	-
	getPoolInfo	External		-
	emergencyWithdraw	External	✓	-
	setGovernance	External	✓	-
	emergencyWithdrawToken	External	✓	-
IUniswapV2Pair	Interface			
	name	External		-
	symbol	External		-

	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	mint	External	✓	-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-

	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IUniswapV2Factory	Interface			
	feeTo	External		-

	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
LibEnsureSafe Transfer	Library			
	safeTransferFromEnsureExactAmount	Internal	✓	validAddress validAddress validAddress validAmount
	transferEnsureExactAmount	Internal	✓	validAddress validAddress validAmount
	transferNativeOrToken	Internal	✓	
	transferNative	Internal	✓	validAddress validAmount
LibPresale	Library			
LibTier	Library			
PoolLibrary	Library			
	withdrawableVestingTokens	Internal		
	getContributionAmount	Internal		
	convertCurrencyToToken	Internal		
	addLiquidity	Internal	✓	
	calculateFeeAndLiquidity	Internal		
LibEnsureSafe Transfer	Library			

	safeTransferFromEnsureExactAmount	Internal	✓	validAddress validAddress validAddress validAmount
	transferEnsureExactAmount	Internal	✓	validAddress validAddress validAmount
	transferNativeOrToken	Internal	✓	
	transferNative	Internal	✓	validAddress validAmount
LibPresale	Library			
LibTier	Library			
CirclePoolFactory	Implementation	OwnableUp gradeable, Utility		
	initialize	External	✓	validAddress validAddress validAddress validAddress initializer
		External	Payable	-
	setMasterAddress	Public	✓	onlyOwner validAddress
	setFairAddress	Public	✓	onlyOwner validAddress
	setPrivateAddress	Public	✓	onlyOwner validAddress
	setAdminWallet	Public	✓	onlyOwner validAddress
	setPartnerFee	Public	✓	onlyOwner
	setVersion	Public	✓	onlyOwner
	setFeeIndex	Public	✓	onlyOwner
	setFees	Public	✓	onlyOwner
	getFee	Public		-
	setPaymentCurrency	Public	✓	onlyOwner

	getPaymentCurrency	Public		-
	initializeClone	Internal	✓	
	createSale	External	Payable	-
	initializePrivateClone	Internal	✓	
	createPrivateSale	External	Payable	-
	initializeFairClone	Internal	✓	
	createFairSale	External	Payable	-
	checkFees	Internal	✓	
	fairFees	Internal	✓	
	checkPrivateSalefees	Internal	✓	
	_feesCount	Internal		
	_feesFairCount	Internal		
	setPoolOwner	Public	✓	onlyOwner
	setPresalePoolPrice	Public	✓	onlyOwner validAmount
	setPrivatePoolPrice	Public	✓	onlyOwner validAmount
	setFairPoolPrice	Public	✓	onlyOwner validAmount
	setPoolManager	Public	✓	onlyOwner
	bnbLiquidity	Public	✓	onlyOwner validAddress validAmount
	transferAnyERC20Token	Public	✓	onlyOwner
	poolEmergencyWithdrawLiquidity	Public	✓	onlyOwner
	poolEmergencyWithdrawToken	Public	✓	onlyOwner
	poolEmergencyWithdraw	Public	✓	onlyOwner
	poolSetGovernance	Public	✓	onlyOwner
CirclePoolManager	Implementation	OwnableUp gradeable, IPoolManager, Utility		
		External	Payable	-

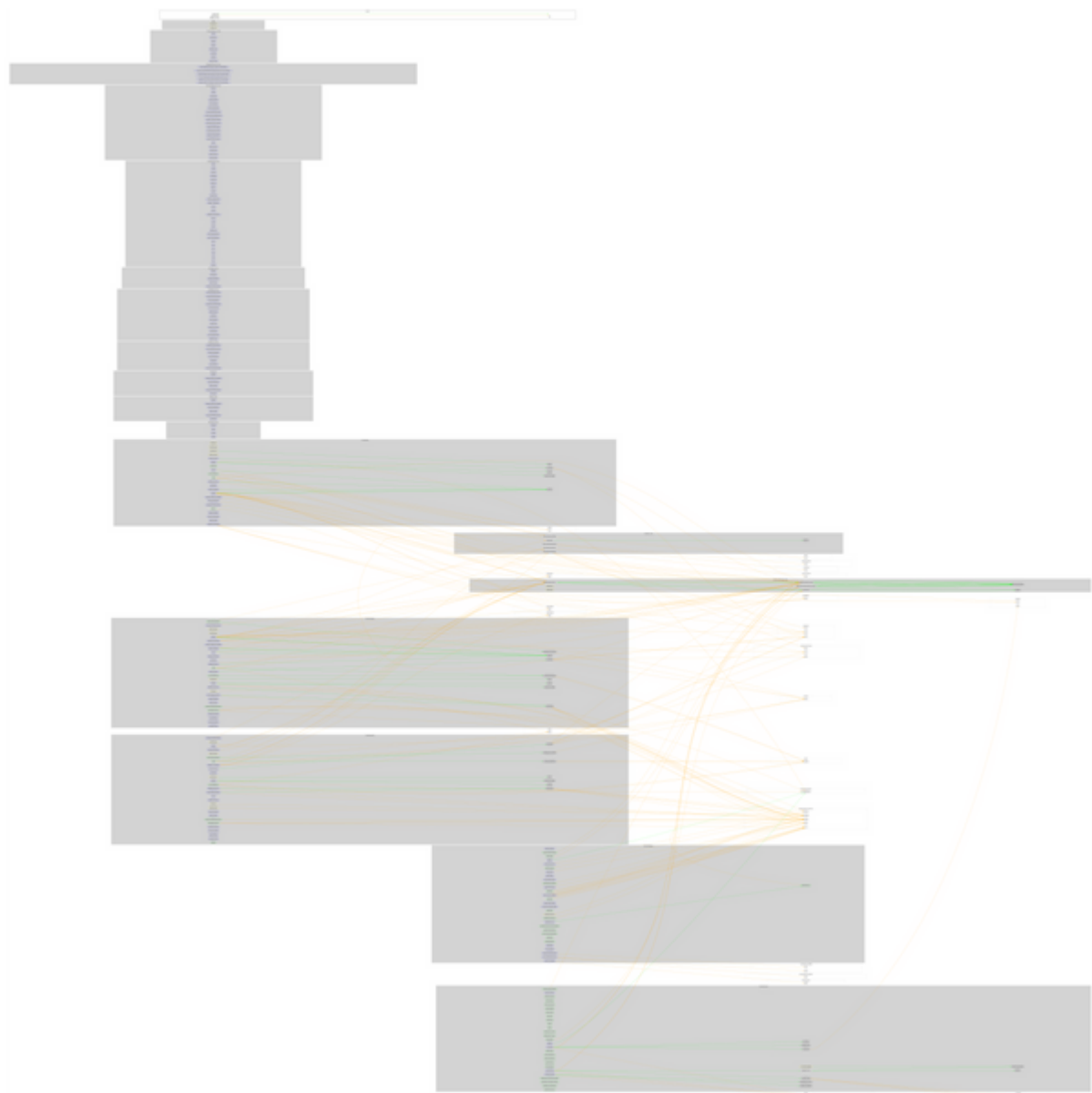
	initialize	External	✓	initializer
	addPoolFactory	Public	✓	onlyAllowedFactory
	addAdminPoolFactory	Public	✓	onlyOwner
	addPoolFactories	External	✓	onlyOwner
	removePoolFactory	External	✓	onlyOwner
	isPoolGenerated	Public		-
	poolForToken	External		-
	privatePoolForToken	External		-
	registerPool	External	✓	onlyAllowedFactory
	registerPrivatePool	External	✓	onlyAllowedFactory
	increaseTotalValueLocked	External	✓	onlyAllowedFactory
	decreaseTotalValueLocked	External	✓	onlyAllowedFactory
	recordContribution	External	✓	onlyAllowedFactory
	removePoolForToken	External	✓	onlyAllowedFactory
	removePrivatePoolForToken	External	✓	onlyAllowedFactory
	getPoolsOf	Public		-
	getAllPools	Public		-
	getPoolAt	Public		-
	getTotalNumberOfPools	Public		-
	getTotalNumberOfContributedPools	Public		-
	getAllContributedPools	Public		-
	getContributedPoolAtIndex	Public		-
	getTotalNumberOfPools	Public		-
	getPoolAt	Public		-
	getTopPool	Public		-
	initializeTopPools	Public	✓	onlyOwner

	addTopPool	External	✓	onlyAllowedFactory
	removeTopPool	External	✓	onlyAllowedFactory
	getCumulativePoolInfo	External		-
	bnbLiquidity	Public	✓	onlyOwner validAddress validAmount
	transferAnyERC20Token	Public	✓	onlyOwner validAddress validAddress validAmount
CirclePresalePool	Implementation	OwnableUp gradeable, IPool, Reentrancy Guard, Utility		
		External	Payable	-
	initialize	External	✓	initializer
	addWhitelistedUsers	External	✓	-
	removeWhitelistedUsers	External	✓	-
	setWhitelist	Internal	✓	onlyOperator
	getNumberOfWhitelistedUsers	Public		-
	getWhitelistedUsers	Public		-
	getPoolInfo	External		-
	contribute	Public	Payable	inProgress
	claim	Public	✓	nonReentrant
	_withdrawableTokens	Internal		
	withdrawContribution	External	✓	nonReentrant
	finalize	External	✓	onlyOperator nonReentrant
	cancel	External	✓	onlyOperator
	withdrawLeftovers	External	✓	onlyOperator
	withdrawLiquidity	External	✓	onlyOperator

	emergencyWithdrawLiquidity	External	✓	onlyOwner
	emergencyWithdrawToken	External	✓	onlyOwner
	emergencyWithdraw	External	✓	onlyOwner
	updatePoolDetails	External	✓	onlyOperator
	setGovernance	External	✓	onlyOwner validAddress
	getContributionAmount	Public		-
	remainingContribution	Public		-
	userAvailableClaim	Public		-
	startPublicSaleNow	External	✓	onlyOperator inProgress
	changeWhitelist	External	✓	onlyOperator
	startTier2SaleNow	External	✓	onlyOperator inProgress
	changeTierDates	External	✓	onlyOperator notInProgress
CirclePrivatePool	Implementation	OwnableUp gradeable, IPrivatePool, Reentrancy Guard, Utility		
		External	Payable	-
	initialize	External	✓	initializer
	addWhitelistedUsers	External	✓	-
	removeWhitelistedUsers	External	✓	-
	setWhitelist	Internal	✓	onlyOperator
	getPoolInfo	External		-
	getNumberOfWhitelistedUsers	Public		-
	getWhitelistedUsers	Public		-
	contribute	Public	Payable	inProgress
	withdrawContribution	External	✓	nonReentrant
	claim	Public	✓	nonReentrant onlyGovernanc

				e
	_withdrawableTokens	Internal		
	finalize	External	✓	onlyOperator nonReentrant
	cancel	External	✓	onlyOperator
	withdrawLeftovers	External	✓	onlyOperator nonReentrant
	emergencyWithdrawToken	External	✓	onlyOwner
	emergencyWithdraw	External	✓	onlyOwner
	updatePoolDetails	External	✓	onlyOperator
	setGovernance	External	✓	onlyOwner validAddress
	getContributionAmount	Public		-
	remainingContribution	Public		-
	userAvalibleClaim	Public		-
	startPublicSaleNow	External	✓	onlyOperator inProgress
	startTier2SaleNow	External	✓	onlyOperator inProgress
	changeWhitelist	External	✓	onlyOperator
	changeTierDates	External	✓	onlyOperator notInProgress
	getTier	Public		-
Utility	Implementation			

Contract Flow



Inheritance Graph



Summary

The Pool ecosystem contracts implement a pool mechanism. This audit investigates security issues, business logic concerns, and potential improvements.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security. Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.



The Cyberscope team

<https://www.cyberscope.io>