Audit Report

Circle Launchpad Airdrop

March 2022

Github	https://github.com/monkey-shanti/Circle-Launchpad
Commit	7f6f46693c2710c2dfc986618b6531561f0ddabb
Audited by	© cyberscope

Table of Contents

Contract Review

Repository	https://github.com/monkey-shanti/Circle-Launchpad
Commit	7f6f46693c2710c2dfc986618b6531561f0ddabb

Contract Name	Testing Deploy
AirdropMaster	https://testnet.bscscan.com/address/0x83D3fa666143A76298AC182215F9B3FAe185D227
AirdropManager	https://testnet.bscscan.com/address/0x4638F1273c1A8334230DD59B532C2e87dFFAA3b2
AirdropFactory	https://testnet.bscscan.com/address/0xF50Af5B6FA0A4be8FAB2B3f9259D677dE04cD66E

Audit Updates

Initial Audit	20 Dec 2022 https://github.com/cyberscope-io/audits/blob/main/circleLaunchpad/v1/airdrop.pdf
Corrected Phase 2	02 Jan 2023 https://github.com/cyberscope-io/audits/blob/main/circleLaunchpad/v2/airdrop.pdf
Corrected Phase 3	16 Mar 2023

Source Files

Filename	SHA256
AirdropFactory.sol	11fcd932a40b52d872031c85dab8fc4690d52e8c9f6cdfe5d2705fe4a9843bb0
AirdropMain.sol	55047579ee95b0123a5fcf0a0059a6423b0ffb9fe2a8685bddaac82105b0ce12
AirdropManager.sol	37d599916fd2747e34ff3bad99067741624577a58dbe3f514acbd1b36ecc69e2
launchpad/interfaces/IUniswapV2Pair.sol	5631411f67c8741031e9bfdd27fad3c81554c0c92a37dce6990b618ef634cd0a
launchpad/interfaces/PoolLibrary.sol	f209394b1e0c66187d6e6f86f5de7708930fe8f282c3fbfbe69e507dc5133939
launchpad/libraries/LibEnsureSafeTransfer.sol	a4c2990e467b6b694059f106497a2c31f4489c6723fa6470c076a919548cc7ca
Locker.sol	1fc523d7494f0dbaa480c2b2899a893cdab6d95c83cbe4ce14ec556318cbceb3
utils/Utility.sol	bb982ca156ddbd0ea26ba803843a755ff4ad6440addadc577d3ba8335f8e0705

Introduction

The Circle launchpad Airdrop contract implements a locker mechanism. It consists of a factory, a manager, and the master airdrop contract.

Airdrop Factory

The Airdrop Factory is responsible for creating new airdrops.

Roles

The contract has two roles.

Owner Role

The owner role has the authority to

- setMasterAddress
- setAdminWallet
- setPartnerFee
- setVersion
- setPoolOwner
- setPresalePoolPrice
- setPoolManager
- bnbLiquidity
- poolEmergencyWithdrawToken
- poolEmergencyWithdraw
- poolSetGovernance

User Role

The user has the authority to createSale.

Airdrop Manager

The Airdrop Manager is responsible for adding or removing factories. Additionally, it is responsible for monitoring airdrop factories and keeping registries about them.

Roles

The contract has three roles.

Owner Role

The Owner has the authority to

- addAdminPoolFactory
- addPoolFactories
- removePoolFactory
- bnbLiquidity

AllowedFactory Role

The Allowed Factories have the authority to

- addPoolFactory
- registerPool
- increaseTotalValueLocked
- decreaseTotalValueLocked
- recordContribution
- removePoolForToken

User Role

The users have the authority to

- view isPoolFactory
- view isPoolGenerated
- getPoolsOfLength
- getPoolsForTokenLength
- getPoolsOf
- getPoolsForToken
- getAllPools
- getPoolAt

- getTotalNumberOfPools
- getTotalNumberOfContributedPools
- getAllContributedPools
- getContributedPoolAtIndex
- getTotalNumberOfPools
- getPoolAt
- getCumulativePoolInfo
- getUserContributedPoolInfo

Airdrop Master

The Airdrop Master implements the core functionality of the airdrop.

Airdrop State

The Airdrop has 3 states

- inUse
- completed
- cancelled

Roles

The contract has 5 roles.

Owner Role

The Owner has the authority to

- emergencyWithdrawToken
- emergencyWithdraw
- setGovernance

Whitelisted Role

The Whitelisted role has the authority to

- claim

Operator Role

The Operator has the authority to

- initializeVesting
- addWhitelistedUsers
- addWhitelistedUser
- removeWhitelistedUsers
- isUserWhitelisted
- changeStartAt
- updatePoolDetails

Governance Role

The Governance role is not utilized on the contract implementation.

User Role

The users have the authority to

- getPoolInfo
- getNumberOfWhitelistedUsers
- getWhitelistedUsers
- getUpdatedState
- view userAvalibleClaim

Contract Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	ICN	Inappropriate Contract Naming	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L08	Tautology or Contradiction	Unresolved
●	L13	Divide before Multiply Operation	Unresolved

ICN - Inappropriate Contract Naming

Criticality	Minor / Informative
Location	AirdropMain.sol#L48 AirdropManager.sol#L15 AirdropFactory.sol#L41
Status	Unresolved

Description

The Airdrop ecosystem is implementing a Locker mechanism. Hence the contract naming is inappropriate.

```
contract AirdropMaster

contract AirdropManager

contract AirdropFactory
```

Recommendation

The team is advised to carefully check if the implementation follows the expected business logic and rename the contracts accordingly.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	AirdropManager.sol#L51,326 AirdropMain.sol#L54,56,128,129,130,131,132,154,172,182,210,334,372 AirdropFactory.sol#L50,60,61,62,63,64,76,81,86,90,97,98,99,100,101,116,117,118,119,156,161,166,170,208
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.

```
event sender(address sender);
address payable _receiver
uint256 _amount
uint256 public MAX_ALLOCATIONS = 500
uint8 public VERSION
address[3] memory _addrs
uint256[1] memory _saleInfo
string memory _poolDetails
address[3] memory _linkAddress
uint8 _version
uint256[3] memory _vestingInit
uint256[] memory _allocations
uint256 _allocation
uint256 _startTime

...
```

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.

L08 - Tautology or Contradiction

Criticality	Minor / Informative
Location	AirdropMain.sol#L158,159,160
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(_vestingInit[1] >= 0, "Invalid cycle")
require(_vestingInit[0] >= 0 && _vestingInit[0] < 10_000, "Invalid bips
for TGE")
require(_vestingInit[2] >= 0 && _vestingInit[2] < 10_000, "Invalid bips
for cycle")
```

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.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	AirdropMain.sol#L320,396
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 - startTime) / cycle) *
                    cycleReleaseAmount) +
                    tgeReleaseAmount
```

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.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IAirdropManager	Interface			
	addPoolFactory	External	✓	-
	increaseTotalValueLocked	External	✓	-
	decreaseTotalValueLocked	External	✓	-
	removePoolForToken	External	✓	-
	recordContribution	External	✓	-
	isPoolGenerated	External		-
	registerPool	External	✓	-
AirdropFactory	Implementation	OwnableUpgradeable, Utility		
	initialize	External	✓	validAddress validAddress validAmount initializer
		External	Payable	-
	setMasterAddress	Public	✓	onlyOwner
	setAdminWallet	Public	✓	onlyOwner
	setPartnerFee	Public	✓	onlyOwner
	setVersion	Public	✓	onlyOwner
	initializeClone	Internal	✓	validAddress validAddress validAddress
	createSale	External	Payable	-
	setPoolOwner	Public	✓	onlyOwner
	setPresalePoolPrice	Public	✓	onlyOwner validAmount

	setPoolManager	Public	✓	onlyOwner validAddress
	bnbLiquidity	Public	✓	onlyOwner validAddress validAmount
	transferAnyERC20Token	Public	✓	onlyOwner validAddress validAddress validAmount
	poolEmergencyWithdrawToken	Public	✓	onlyOwner validAddress validAddress validAddress validAmount
	poolEmergencyWithdraw	Public	✓	onlyOwner validAddress validAddress validAmount
	poolSetGovernance	Public	✓	onlyOwner validAddress validAddress
IERC20Info	Interface			
	decimals	External		-
	name	External		-
	symbol	External		-
	supply	External		-
IPoolFactory	Interface			
	increaseTotalValueLocked	External	✓	-
	decreaseTotalValueLocked	External	✓	-
	removePoolForToken	External	✓	-
	recordContribution	External	✓	-
IAirdrop	Interface			
	initialize	External	✓	-
	initializeVesting	External	✓	-

	setGovernance	External	✓	-
	emergencyWithdraw	External	✓	-
	emergencyWithdrawToken	External	✓	-
	getPoolInfo	External		-
AirdropMaster	Implementation	OwnableUpgradeable, IAirdrop, ReentrancyGuard, Utility		
	initialize	External	✓	validAddress validAddress validAddress initializer
	initializeVesting	External	✓	onlyOperator
	addWhitelistedUsers	External	✓	-
	addWhitelistedUser	External	✓	-
	removeWhitelistedUsers	External	✓	-
	removeWhitelistedUser	External	✓	-
	isUserWhitelisted	Public		-
	setWhitelist	Internal	✓	onlyOperator notInProgress validAddress
	getPoolInfo	External		-
	getNumberOfWhitelistedUsers	Public		-
	getWhitelistedUsers	Public		-
	claim	Public	✓	nonReentrant inProgress onlyWhitelisted
	_withdrawableTokens	Internal		
	changeStartAt	External	✓	onlyOperator notInProgress
	cancel	External	✓	onlyOperator notInProgress
	emergencyWithdrawToken	External	✓	onlyOwner

	emergencyWithdraw	External	✓	onlyOwner
	updatePoolDetails	External	✓	onlyOperator
	setGovernance	External	✓	onlyOwner validAddress
	getUpdatedState	Public		-
	userAvalibleClaim	Public		-
AirdropManager	Implementation	OwnableUpgradeable, IAirdropManager, Utility		
		External	Payable	-
	initialize	External	✓	initializer
	addPoolFactory	Public	✓	onlyAllowedFactory validAddress
	addAdminPoolFactory	Public	✓	onlyOwner validAddress
	addPoolFactories	External	✓	onlyOwner
	removePoolFactory	External	✓	onlyOwner validAddress
	isPoolFactory	Public		-
	isPoolGenerated	Public		-
	registerPool	External	✓	onlyAllowedFactory validAddress validAddress validAddress validAmount
	increaseTotalValueLocked	External	✓	onlyAllowedFactory
	decreaseTotalValueLocked	External	✓	onlyAllowedFactory
	recordContribution	External	✓	onlyAllowedFactory
	removePoolForToken	External	✓	onlyAllowedFactory
	getPoolsOfLength	Public		-
	getPoolsForTokenLength	Public		-

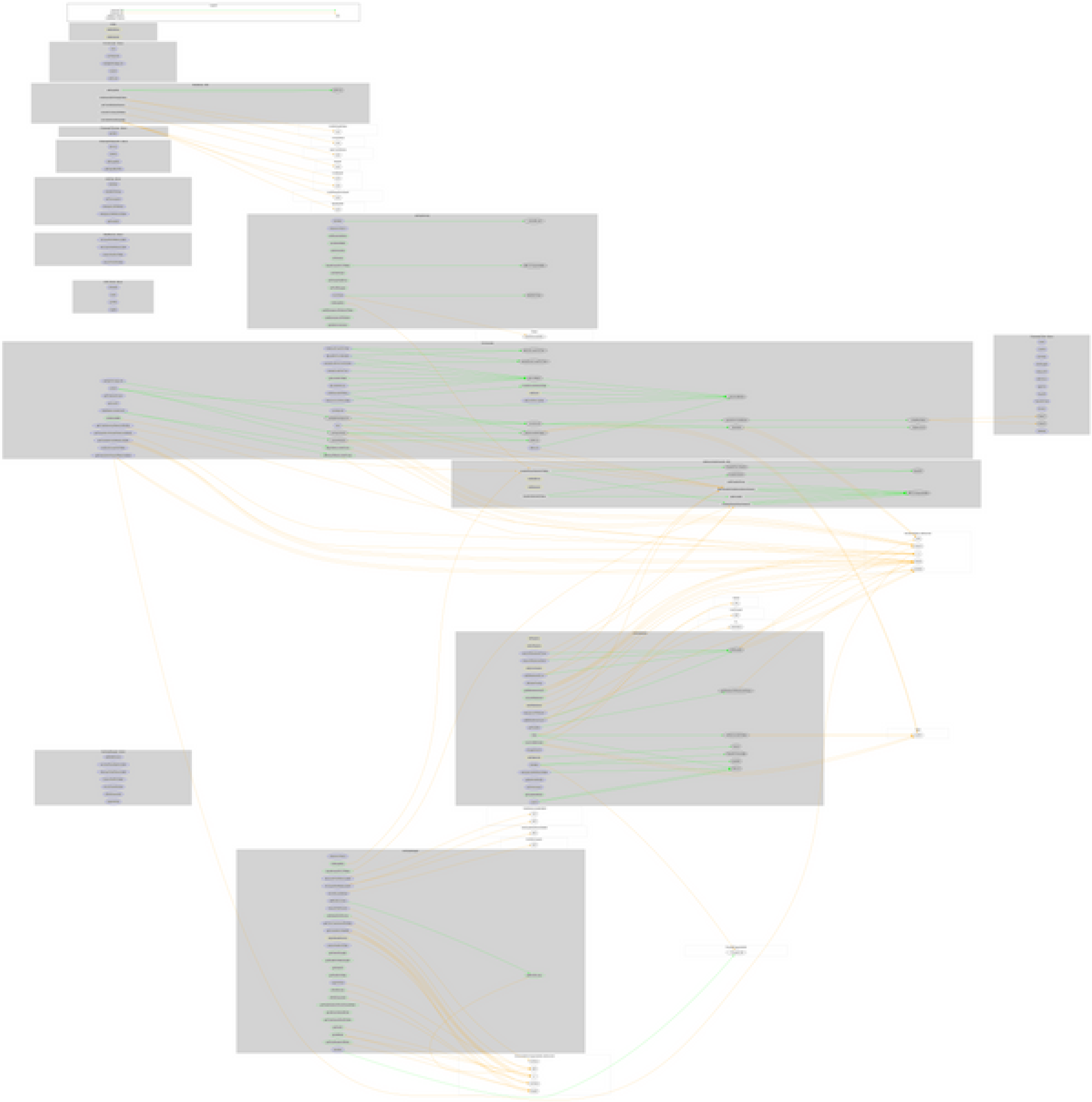
	getPoolsOf	Public		-
	getPoolsForToken	Public		-
	getAllPools	Public		-
	getPoolAt	Public		-
	getTotalNumberOfPools	Public		-
	getTotalNumberOfContributedPools	Public		-
	getAllContributedPools	Public		-
	getContributedPoolAtIndex	Public		-
	getTotalNumberOfPools	Public		-
	getPoolAt	Public		-
	getCumulativePoolInfo	External		-
	getUserContributedPoolInfo	External		-
	bnbLiquidity	Public	✓	onlyOwner validAddress validAmount
	transferAnyERC20Token	Public	✓	onlyOwner
IUniswapV2Pair	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	factory	External		-
	token0	External		-
	token1	External		-

	initialize	External	✓	-
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
IUniswapV2Router02	Interface	IUniswapV2Router01		
IUniswapV2Factory	Interface			
	getPair	External		-
PoolLibrary	Library			
	withdrawableVestingTokens	Internal		
	getContributionAmount	Internal		
	convertCurrencyToToken	Internal		
	addLiquidity	Internal	✓	
	calculateFeeAndLiquidity	Internal		
LibEnsureSafeTransfer	Library			
	safeTransferFromEnsureExactAmount	Internal	✓	validAddress validAddress validAddress validAmount
	transferEnsureExactAmount	Internal	✓	validAddress validAddress validAmount
	transferExactNativeOrToken	Internal	✓	
	transferExactNative	Internal	✓	validAddress validAmount

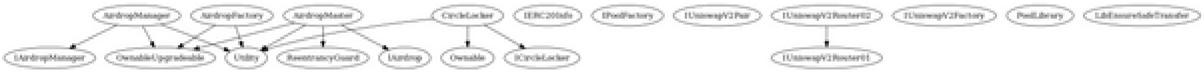
	safeTransferFrom	Internal	✓	validAddress validAddress validAddress validAmount
	safeTransfer	Internal	✓	validAddress validAddress validAmount
	transferNativeOrToken	Internal	✓	
	transferNative	Internal	✓	validAddress validAmount
ICircleLocker	Interface			
	lock	External	✓	-
	vestingLock	External	✓	-
	multipleVestingLock	External	✓	-
	unlock	External	✓	-
	editLock	External	✓	-
CircleLocker	Implementation	ICircleLocker, Ownable, Utility		
	lock	External	✓	-
	vestingLock	External	✓	-
	multipleVestingLock	External	✓	-
	_multipleVestingLock	Internal	✓	
	_createLock	Internal	✓	
	_lockToken	Private	✓	
	_registerLock	Private	✓	
	unlock	External	✓	validLock
	_normalUnlock	Internal	✓	
	_vestingUnlock	Internal	✓	
	withdrawableTokens	External		-
	_withdrawableTokens	Internal		
	editLock	External	✓	validLock

	editLockDescription	External	✓	validLock
	transferLockOwnership	Public	✓	validLock
	renounceLockOwnership	External	✓	-
	getTotalLockCount	External		-
	getLockAt	External		-
	getLockById	Public		-
	allLpTokenLockedCount	Public		-
	allNormalTokenLockedCount	Public		-
	getCumulativeLpTokenLockInfoAt	External		-
	getCumulativeNormalTokenLockInfoAt	External		-
	getCumulativeLpTokenLockInfo	External		-
	getCumulativeNormalTokenLockInfo	External		-
	totalTokenLockedCount	External		-
	lpLockCountForUser	Public		-
	lpLocksForUser	External		-
	lpLockForUserAtIndex	External		-
	normalLockCountForUser	Public		-
	normalLocksForUser	External		-
	normalLockForUserAtIndex	External		-
	totalLockCountForUser	External		-
	totalLockCountForToken	External		-
	getLocksForToken	Public		-
	_getActualIndex	Internal		
	_parseFactoryAddress	Internal		
	_isValidLpToken	Private		
	withdrawBNB	Public	✓	onlyOwner validAddress validAmount
Utility	Implementation			

Contract Flow



Inheritance Graph



Summary

The Airdrop ecosystem contracts implement a locker 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