Cyberscope

Audit Report

AirDrop

December 2022

Github https://github.com/Deeplink-Network/Staking

Commit ab56a7e7cde209bdad1c70a24ce8ce257c04413d

Audited by © cyberscope

Table of Contents

Contract Review

Contract Name	AirDrop
Testing Deploy	https://testnet.bscscan.com/token/0x16bb2b875ba05ae992cf1e8151251145fb049d47

Audit Updates

Initial Audit	15 Dec 2022

Source Files

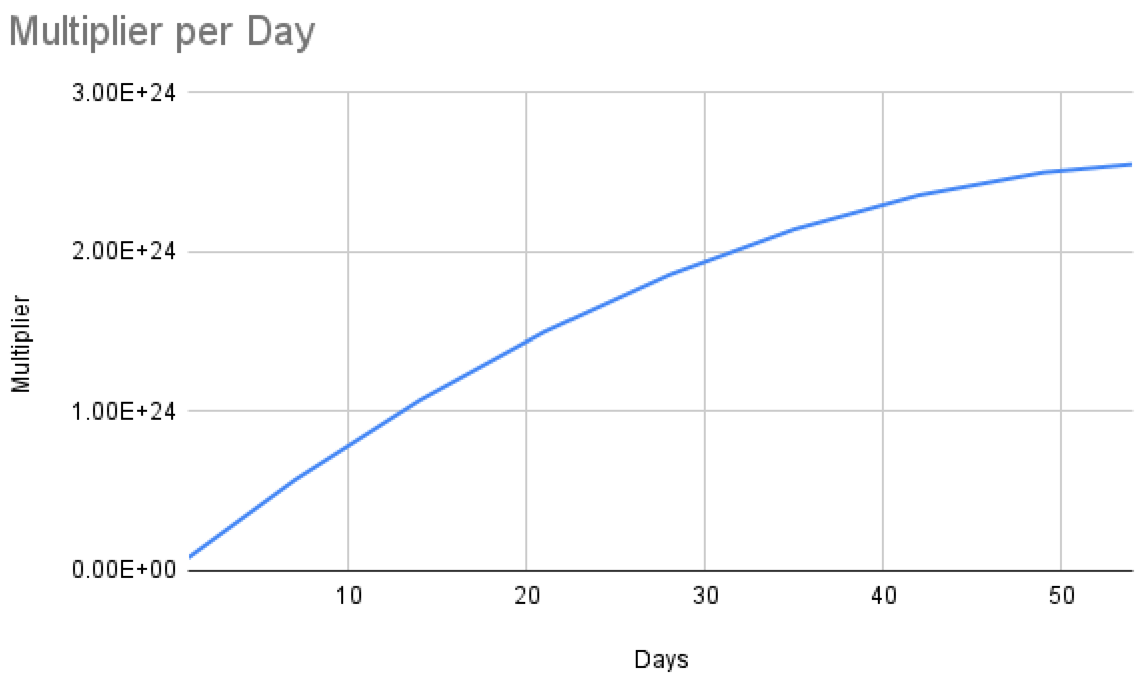
Filename	SHA256
@openzeppelin/contracts/access/Ownable.sol	9353af89436556f7ba8abb3f37a6677249aa4df6024fbfaa94f79ab2f44f3231
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db8003d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
contracts/Airdrop.sol	44198537964a122a54df0d1e35516baae9d9a3494b2415ad57ee5a83ed305ec4
contracts/Interface/IDeepToken.sol	4271d346dd077ad51065f40716dc98a65c87eda77e3a647d7269b0a3ddc30b7b
contracts/Interface/IDKeeperEscrow.sol	6df308b29f088764ba315afd89dda58a6769a2afa5be16cd97ab6e0df4fd3892

Introduction

The AirDrop contract implements an airdrop mechanism. It is responsible for the distribution of rewards to the airdrop winners. The airdrop winners are added or removed by the contract owner. One user cannot be added more than once. The allocation amount can be between 1-10. The users are responsible for claiming their rewards.

Rewards Formula

The rewards calculation follows a logarithmic distribution from day one until the end of the staking period (8 weeks). The rewards file contains the details about the calculations. The Y-Axis depicts the rewards multiplier. Each user that participated in the staking contract, receives the proportional amount of rewards.



Scenario

Admin adds 100 airdrop applicable addresses with 5 allocation points for each address. That means `100 * 5 = 500` allocation points in total.

The total amount of Deep tokens that will be minted is `Multiplier / Decimals = 2,571,428 * 10 ^ 18 / 10 ^ 18 ≈ 2,571,428` tokens.

Each user will receive `2,551,020 Tokens / 100 Points = 2,571` tokens.

Roles

The contract has 2 roles.

Owner Role

The contract owner has the authority to

- addAirdropWallets
- removeAirdropWallets
- setEscrow address

User Role

The contract users have the authority to

- View pendingDeep
- updatePool
- claim rewards

Contract Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	MC	Missing Check	unresolved
●	TUU	Time Units Usage	unresolved
●	ADU	Arbitrary Decimals Usage	unresolved
●	SRAI	Sufficient Reward Amount Issue	unresolved
●	L04	Conformance to Solidity Naming Conventions	unresolved
●	L07	Missing Events Arithmetic	unresolved
●	L10	State Variables in Loop	unresolved

MC - Missing Check

Criticality	minor / informative
Location	contracts/Airdrop.sol#L44
Status	unresolved

Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

```
constructor(
    IDeepToken _deep,
    uint256 _startTime,
    uint256 _endTime
) public {
    require(_endTime >= _startTime && block.timestamp <= _startTime,
"Invalid timestamp");
    deepToken = _deep;
    startTime = _startTime;
    endTime = _endTime;

    totalAllocPoint = 0;
    lastRewardTime = _startTime;
}
```

Recommendation

The contract should properly check the variables according to the required specifications.

- The address _deep should not be set to zero address.

TUU - Time Units Usage

Criticality	minor / informative
Location	contracts/Airdrop.sol#L52
Status	unresolved

Description

The contract is using arbitrary numbers to form time-related values. As a result, it decreases the readability of the codebase and prevents the compiler to optimize the source code.

```
uint256 public constant WEEK = 3600 * 24 * 7;
```

Recommendation

It is a good practice to use the time units reserved keywords like seconds, minutes, hours, days, weeks, and years to process time-related calculations.

ADU - Arbitrary Decimals Usage

Criticality	minor / informative
Location	contracts/Airdrop.sol#L21,121
Status	unresolved

Description

The contract calculates the rewards assuming the token's decimals are fixed. The token property is mutable. The contract owner has the authority to add any token with different amounts of decimals. As a result, the precision will be wrong.

```
function claim() public {
    UserInfo storage user = userInfo[msg.sender];
    require(user.alloc != 0, "Not allocated with this account.");
    updatePool();

    uint256 pending = (user.alloc * accTokenPerShare) / 1e6 -
user.rewardDebt;
    if (pending > 0) {
        safeDeepTransfer(msg.sender, pending);
        user.lastClaimTime = block.timestamp;
        emit Claimed(msg.sender, pending);
    }

    user.rewardDebt = (user.alloc * accTokenPerShare) / 1e6;
}

function getRewardRatio(uint256 _time) internal view returns (uint256)
{
    if (8 < (_time - startTime) / WEEK) return 0;

    return (((2e24 * (8 - (_time - startTime) / WEEK)) / 8 / 35) * 10)
/ WEEK;
}
```

Recommendation

The contract could calculate the reward ratio with the corresponding tokens decimals `ERC20.decimals()` instead of adding a fixed value.

SRAI - Sufficient Reward Amount Issue

Criticality	minor / informative
Location	contracts/Airdrop.sol#L88,130
Status	unresolved

Description

The contract is distributing rewards without checking if the contract's balance is sufficient to cover the reward amount. As a result, the expected rewards might not be transferred.

```
function claim() public {
    UserInfo storage user = userInfo[msg.sender];
    require(user.alloc != 0, "Not allocated with this account.");
    updatePool();
    uint256 pending = (user.alloc * accTokenPerShare) / 1e6 -
user.rewardDebt;
    if (pending > 0) {
        safeDeepTransfer(msg.sender, pending);
        user.lastClaimTime = block.timestamp;
        emit Claimed(msg.sender, pending);
    }
    user.rewardDebt = (user.alloc * accTokenPerShare) / 1e6;
}
```

Recommendation

The contract could check if the contract's balance is sufficient to cover the reward amount. If it is not sufficient then it could return a descriptive message. A possible solution could be to check if there is sufficient balance prior to adding airdrop wallets.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contracts/Airdrop.sol#L159,130,146,130,104,109,104,109,121,59
Status	unresolved

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.

```
_escrow
_account
s
_account
s
_allocs
_amount
_to
_to
_from
_time
_user
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-conventions.

L07 - Missing Events Arithmetic

Criticality	minor / informative
Location	contracts/Airdrop.sol#L130
Status	unresolved

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.

```
totalAllocPoint +=
_allocs[i]
```

Recommendation

Emit an event for critical parameter changes.

L10 - State Variables in Loop

Criticality	minor / informative
Location	contracts/Airdrop.sol#L146,146,130
Status	unresolved

Description

Costly operations inside a loop might waste gas, so optimizations are justified. Incrementing state variables in a loop incurs a lot of gas because of expensive SSTOREs, which might lead to an out-of-gas.

```
totalAllocPoint -=
userInfo[_accounts[i]].alloc
delete userInfo[_accounts[i]]
totalAllocPoint += _allocs[i]
```

Recommendation

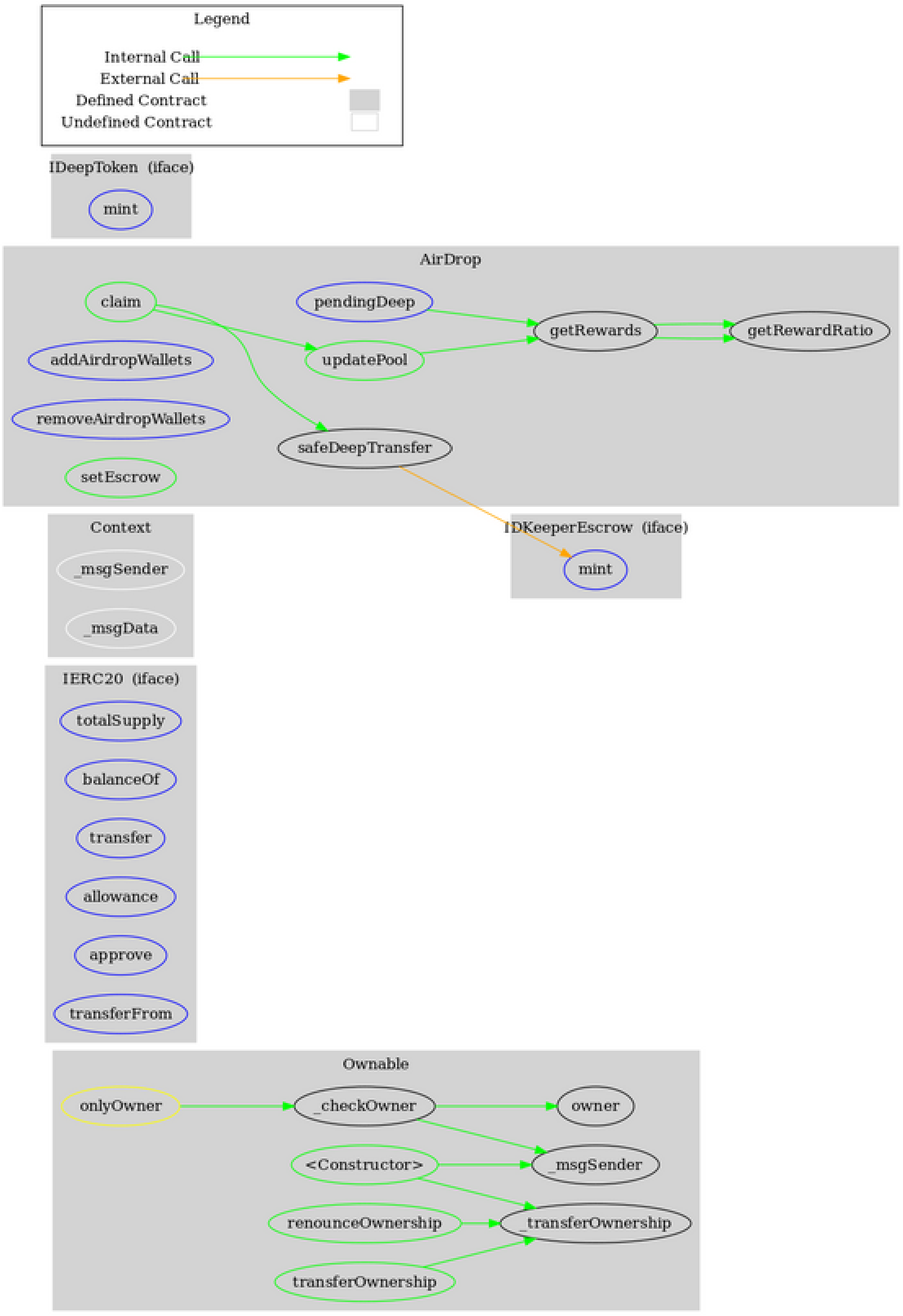
Use a local variable to hold the loop computation result.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
AirDrop	Implementation	Ownable		
		Public	✓	-
	pendingDeep	External		-
	updatePool	Public	✓	-

	claim	Public	✓	-
	safeDeepTransfer	Internal	✓	
	getRewards	Internal		
	getRewardRatio	Internal		
	addAirdropWallets	External	✓	onlyOwner
	removeAirdropWallets	External	✓	onlyOwner
	setEscrow	Public	✓	onlyOwner
IDeepToken	Interface	IERC20		
	mint	External	✓	-
IDKeeperEscrow	Interface			
	mint	External	✓	-

Contract Flow



Summary

AirDrop contract implements an airdrop 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