Cyberscope

Audit Report

# Illumi

June 2022

| | |
|---|---|
| Github | https://github.com/liron757/illumiContract/blob/main/contracts/illumiShareTokenUpgradeable.sol |
| Commit | c145e48264b11472304c8fd65953b9a5c3a2e23a |
| Audited by | © cyberscope |

# Table of Contents

## Contract Review

| | |
|---|---|
| **Contract Name** | illumiShareTokenUpgradeable |
| **Github** | https://github.com/liron757/illumiContract/blob/main/contracts/illumiShareTokenUpgradeable.sol |
| **Commit** | c145e48264b11472304c8fd65953b9a5c3a2e23a |
| **Testing Deploy** | https://testnet.bscscan.com/address/0xf66a40f5c4e0665EACCB65b636976B656fd73f9d |

## Audit Updates

| | |
|---|---|
| **Initial Audit** | 24th June 2022 |
| **Corrected** | |

## Source Files

| Filename | SHA256 |
|---|---|
| AddressUpgradeable.sol | 44edc4d7099c781d11421cea2d82a52948e738f5f6191c8ad01dfc0f9858549c |
| ContextUpgradeable.sol | 80eae6c1e176f3e54242b950459016a2e38609b03c8f560da1ddc0a299f1af86 |
| ERC20BurnableUpgradeable.sol | ad5e221e6c0358d37a77fcbd7fe4383d0bcba94001cb800952aa9e0f09733df7 |
| ERC20LockableUpgradeable.sol | c3d91e19d6b5417d498a75c2683f00c6ac64cd406c4c8e94b6f5dc182613ceb3 |
| ERC20Upgradeable.sol | e3f00a2c180cdf4384a8c993adafafb1c56d00692412dff5914a218cd6897dc2 |
| IERC20MetadataUpgradeable.sol | 665b132315c4d4787cc1c64638297509c98b71075a0b037a0134553839f2e6d8 |
| IERC20Upgradeable.sol | db1d80b38061ba675444e6ad861a621d9966604295027 8d6cdeae9a108afdd17 |
| illumiShareTokenUpgradeable.sol | ce7dcebb0b260da857ce1fc090520c20d19df136d8a330057f0e1c1e78861786 |
| Initializable.sol | a796e04c1879286779386a2d9bb1f1ed696befc9a125b6c62a41fc37e3083865 |
| OwnableUpgradeable.sol | ef2ebdd2f4ab7efc886c91264bf27cdbebad55e236b1199732bdd08dc374eef7 |

## Contract Analysis

🔴 Critical 🟠 Medium ⚪ Minor 🔵 Pass

| 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 |
| 🔵 | OTUT | Owner Transfer User's Tokens |
| 🔵 | ELFM | Contract Owner is not able to increase fees more than a reasonable percent (25%) |
| 🔵 | ULTW | Contract Owner is not able to increase the amount of liquidity taken by dev wallet more than a reasonable percent |
| 🔴 | MT | Contract Owner is not able to mint new tokens |
| 🔵 | BT | Contract Owner is not able to burn tokens from specific wallet |
| 🔵 | BC | Contract Owner is not able to blacklist wallets from selling |

## MT - Mint Tokens

| | |
|---|---|
| **Criticality** | critical |
| **Location** | illumiShareTokenUpgradeable.sol#L24 |

### Description

The contract owner has the authority to mint tokens. The owner may take advantage of it by calling the `mint` function. As a result the contract tokens will be highly inflated.

```
function mint(address to, uint256 amount) onlyOwner public {
    _mint(to, amount);
}
```

### Recommendation

The owner should carefully manage the credentials of the owner's account. We advised considering an extra-strong security mechanism that the actions may be quarantined by many users instead of one. The owner could also renounce the contract ownership for a period of time or pass the access to the zero address.

## Contract Diagnostics

● Critical ● Medium ● Minor

| Severity | Code | Description |
|---|---|---|
| ● | BLC | Business Logic Concern |
| ● | CR | Code Repetition |
| ● | L01 | Public Function could be Declared External |
| ● | L04 | Conformance to Solidity Naming Conventions |
| ● | L05 | Unused State Variable |
| ● | L09 | Dead Code Elimination |

## BLC - Business Logic Concern

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract.sol#L69 |

### Description

The method transferWithLock locks an amount of tokens to a specific address. This method can be called by any user. As a result, a user can execute this method, providing 1 token as an amount, for all the token holders and prevent them from locking.

```
function transferWithLock(address _to, bytes32 _reason, uint256 _amount, uint256
_time)
    public
    returns (bool)
{
    uint256 validUntil = block.timestamp + _time; //solhint-disable-line

    require(tokensLocked(_to, _reason) == 0, ALREADY_LOCKED);
    require(_amount != 0, AMOUNT_ZERO);

    if (locked[_to][_reason].amount == 0)
        lockReason[_to].push(_reason);

    transfer(address(this), _amount);

    locked[_to][_reason] = lockToken(_amount, validUntil, false);

    emit Locked(_to, _reason, _amount, validUntil);
    return true;
}
```

### Recommendation

The `transferWithLock()` method could implement a functionality similar to the allowance so the users will not be able to abuse the locking mechanism.

## CR - Code Repetition

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract.sol#L66,96 |

### Description

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

The methods ‘transferWithLock()’ and ‘lock()’ contain the same statement. The only difference is the target address (_to).

### Recommendation

The ‘lock()’ method could internally call the ‘transferWithLock()’ method providing the msg.sender as the _to variant.

## L01 - Public Function could be Declared External

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract/ERC20LockableUpgradeable.sol#L66,96,140,153,170,187,218,241<br><br>contract/illumiShareTokenUpgradeable.sol#L34 |

### Description

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

```
mint
getUnlockableTokens
unlock
increaseLockAmount
extendLock
totalBalanceOf
tokensLockedAtTime
transferWithLock
lock
...
```

### Recommendation

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

## L04 - Conformance to Solidity Naming Conventions

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract/ERC20LockableUpgradeable.sol#L18,49,66,96,123,140,153,170,187,205,218,241,256<br><br>contract/illumiShareTokenUpgradeable.sol#L8 |

### 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.

```
illumiShareTokenUpgradeable
__gap
_of
_reason
_amount
_time
_to
__ERC20Lockable_init
lockToken
...
```

### Recommendation

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

## L05 - Unused State Variable

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract/ERC20LockableUpgradeable.sol#L256 |

### Description

There are segments that contain unused state variables.

```
__gap
```

### Recommendation

Remove unused state variables.

## L09 - Dead Code Elimination

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract/ERC20LockableUpgradeable.sol#L49 |

### Description

Functions that are not used in the contract, and make the code's size bigger.

```
__ERC20Lockable_init
```

### Recommendation

Remove unused functions.

# Contract Functions

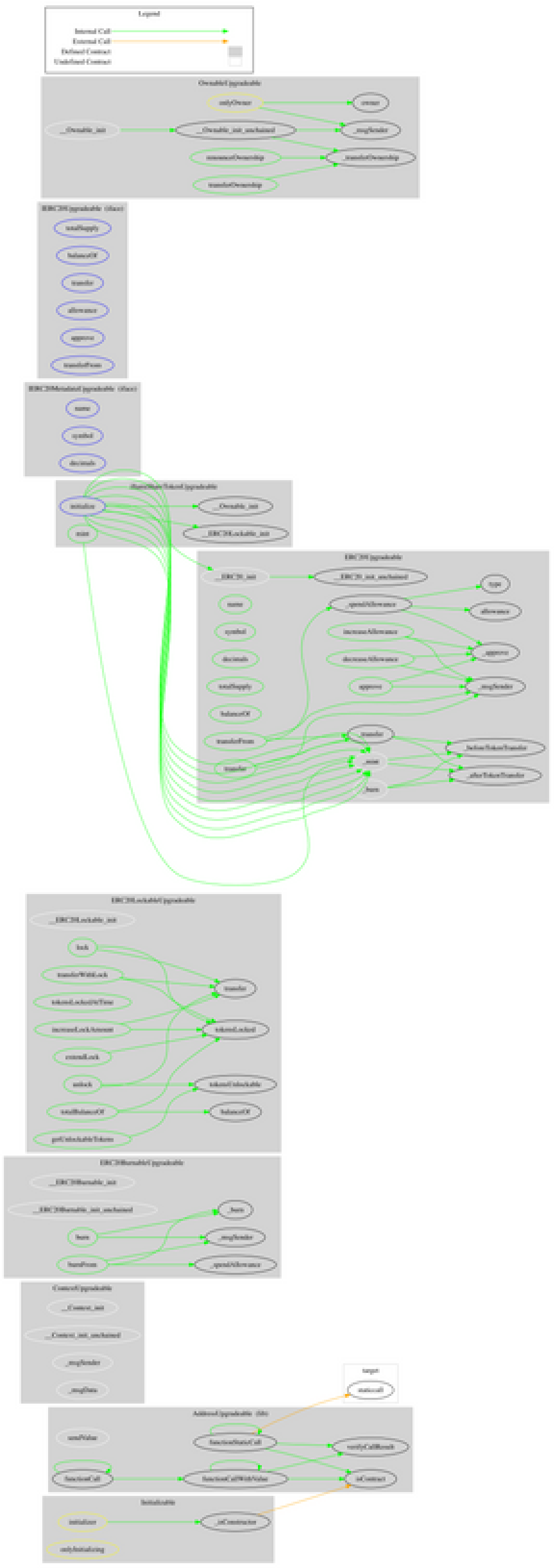
| Contract | Type | Bases | | |
|---|---|---|---|---|
| | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| | | | | |
| **AddressUpgradeable** | Library | | | |
| | isContract | Internal | | |
| | sendValue | Internal | ✓ | |
| | functionCall | Internal | ✓ | |
| | functionCall | Internal | ✓ | |
| | functionCallWithValue | Internal | ✓ | |
| | functionCallWithValue | Internal | ✓ | |
| | functionStaticCall | Internal | | |
| | functionStaticCall | Internal | | |
| | verifyCallResult | Internal | | |
| | | | | |
| **ContextUpgradeable** | Implementation | Initializable | | |
| | __Context_init | Internal | ✓ | onlyInitializing |
| | __Context_init_unchained | Internal | ✓ | onlyInitializing |
| | _msgSender | Internal | | |
| | _msgData | Internal | | |
| | | | | |
| **ERC20BurnableUpgradeable** | Implementation | Initializable, ContextUpgradeable, ERC20Upgradeable | | |
| | __ERC20Burnable_init | Internal | ✓ | onlyInitializing |
| | __ERC20Burnable_init_unchained | Internal | ✓ | onlyInitializing |
| | burn | Public | ✓ | - |
| | burnFrom | Public | ✓ | - |
| | | | | |

| | | | | |
|---|---|---|---|---|
| **ERC20LockableUpgradeable** | Implementation | Initializable, ContextUpgradeable, ERC20BurnableUpgradeable | | |
| | __ERC20Lockable_init | Internal | ✓ | onlyInitializing |
| | lock | Public | ✓ | - |
| | transferWithLock | Public | ✓ | - |
| | tokensLocked | Public | | - |
| | tokensLockedAtTime | Public | | - |
| | totalBalanceOf | Public | | - |
| | extendLock | Public | ✓ | - |
| | increaseLockAmount | Public | ✓ | - |
| | tokensUnlockable | Public | | - |
| | unlock | Public | ✓ | - |
| | getUnlockableTokens | Public | | - |
| | | | | |
| **ERC20Upgradeable** | Implementation | Initializable, ContextUpgradeable, IERC20Upgradeable, IERC20MetadataUpgradeable | | |
| | __ERC20_init | Internal | ✓ | onlyInitializing |
| | __ERC20_init_unchained | Internal | ✓ | onlyInitializing |
| | name | Public | | - |
| | symbol | Public | | - |
| | decimals | Public | | - |
| | totalSupply | Public | | - |
| | balanceOf | Public | | - |
| | transfer | Public | ✓ | - |
| | allowance | Public | | - |
| | approve | Public | ✓ | - |
| | transferFrom | Public | ✓ | - |
| | increaseAllowance | Public | ✓ | - |
| | decreaseAllowance | Public | ✓ | - |

| | | | | |
|---|---|---|---|---|
| | _transfer | Internal | ✓ | |
| | _mint | Internal | ✓ | |
| | _burn | Internal | ✓ | |
| | _approve | Internal | ✓ | |
| | _spendAllowance | Internal | ✓ | |
| | _beforeTokenTransfer | Internal | ✓ | |
| | _afterTokenTransfer | Internal | ✓ | |
| | | | | |
| **IERC20MetadataUpgradeable** | Interface | IERC20Upgradeable | | |
| | name | External | | - |
| | symbol | External | | - |
| | decimals | External | | - |
| | | | | |
| **IERC20Upgradeable** | Interface | | | |
| | totalSupply | External | | - |
| | balanceOf | External | | - |
| | transfer | External | ✓ | - |
| | allowance | External | | - |
| | approve | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | | | | |
| **illumiShareTokenUpgradeable** | Implementation | Initializable, ERC20LockableUpgradeable, OwnableUpgradeable | | |
| | initialize | External | ✓ | initializer |
| | mint | Public | ✓ | onlyOwner |
| | | | | |
| **Initializable** | Implementation | | | |
| | _isConstructor | Private | | |
| | | | | |
| **OwnableUpgradeable** | Implementation | Initializable, ContextUpgradeable | | |

| | | | | |
|---|---|---|---|---|
| | __Ownable_init | Internal | ✓ | onlyInitializing |
| | __Ownable_init_unchained | Internal | ✓ | onlyInitializing |
| | owner | Public | | - |
| | renounceOwnership | Public | ✓ | onlyOwner |
| | transferOwnership | Public | ✓ | onlyOwner |
| | _transferOwnership | Internal | ✓ | |

# Contract Flow



## Summary

Illumi Token is a contract that implements a token enriched with a locked mechanism. The Smart Contract analysis reported one critical severity issue. The contract owner has the authority to mint tokens. if the contract owner abuses the mint functionality, then the contract will be highly inflated. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats. There is also a limit of max 25% fees.

## 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