Cyberscope

Audit Report

MoonLabs Token

April 2023

Network	ETH
Address	0x70B951F7d4Bb28BCE55Ee5374C95349324846833
Audited by	© cyberscope

Table of Contents

Review

Contract Name	MoonLabs
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	3500 runs
Explorer	https://etherscan.io/address/0x70b951f7d4bb28bce55ee5374c95349324846833
Address	0x70b951f7d4bb28bce55ee5374c95349324846833
Network	ETH
Symbol	MLAB
Decimals	9
Total Supply	100.000.000

Audit Updates

Initial Audit	24 Mar 2023 https://github.com/cyberscope-io/audits/blob/main/moonlabsd/v1/token.pdf
Corrected Phase 2	06 Apr 2023 https://github.com/cyberscope-io/audits/blob/main/moonlabsd/v2/token.pdf
Corrected Phase 3	20 Apr 2023

Source Files

Filename	SHA256
@openzeppelin/contracts/access/Ownable.sol	9353af89436556f7ba8abb3f37a6677249a a4df6024fbfaa94f79ab2f44f3231
@openzeppelin/contracts/token/ERC20/ERC20.sol	b39a02b7d35d9837f6f008bcdfe826c566c 1da17aa12fcbac12ebe79c6b1a9af
@openzeppelin/contracts/token/ERC20/extensions /IERC20Metadata.sol	af5c8a77965cc82c33b7ff844deb9826166 689e55dc037a7f2f790d057811990
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db800 3d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/token/ERC721/IERC721. sol	c7703068bac02fe1cdf109e38faf10399c66 eb411e4c9ae0d70c009eca4bf5ef
@openzeppelin/contracts/utils/Address.sol	8160a4242e8a7d487d940814e5279d934e 81f0436689132a4e73394bab084a6d
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a2 3a4baa0b5bd9add9fb6d6a1549814a
@openzeppelin/contracts/utils/introspection/IERC 165.sol	701e025d13ec6be09ae892eb029cd83b30 64325801d73654847a5fb11c58b1e5
@uniswap/v2-core/contracts/interfaces/IUniswapV 2Factory.sol	51d056199e3f5e41cb1a9f11ce581aa3e19 0cc982db5771ffeef8d8d1f962a0d
@uniswap/v2-periphery/contracts/interfaces/IUnis wapV2Router01.sol	0439ffe0fd4a5e1f4e22d71ddbda76d63d6 1679947d158cba4ee0a1da60cf663
@uniswap/v2-periphery/contracts/interfaces/IUnis wapV2Router02.sol	a2900701961cb0b6152fc073856b972564f 7c798797a4a044e83d2ab8f0e8d38
contracts/MoonLabs.sol	a4699ae8c5b8a8c1421f92246b745085aa 8636548b570b6f8ad3ed0746fa3cc1

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OCTD	Transfers Contract's Tokens	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	RCS	Redundant Code Statement	Unresolved
	MSC	Missing Sanity Check	Unresolved
	DDP	Decimal Division Precision	Unresolved
	MMSI	Max Mint Supply Inconsistency	Unresolved
	RAV	Reentrancy Attack Vulnerability	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L07	Missing Events Arithmetic	Unresolved
	L16	Validate Variable Setters	Unresolved

RCS - Redundant Code Statement

Criticality	Minor / Informative
Location	MoonLabs.sol#L255
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract is utilizing a redundant boolean variable. The functionality of distributed variable could be replaced with a pre existing variable.

```
bool rewardsSent = false;

for (uint i = 0; i < maxNftDistribution; i++) {
    /// Set distributed to true if not true
    if (!rewardsSent) rewardsSent = true;
    ...
}

/// Emit event if nft payout
if (rewardsSent) emit DistributeNftPayout(addressArray,
indexArray, nftPayout);
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

The contract could remove the distributed variable and use maxNftDistribution variable. For instance,

```
if (maxNftDistribution > 0) emit
DistributeNftPayout(addressArray, indexArray, nftPayout);
```

MSC - Missing Sanity Check

Criticality	Minor / Informative
Location	MoonLabs.sol#L269
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.

The contract could transfer `nftPayout` to zero addresses.

```
(bool sent, ) = payable(nftOwner).call{ value: nftPayout }("");
```

Recommendation

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

The variable `nftOwner` should be zero address.

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	MoonLabs.sol#L349
Status	Unresolved

Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

The taxes might not be splitted as expected.

```
uint addToLiquidityHalf = ((swapThreshold * liquidityTax) /
    totalTokenTax) / 2;

_swapTokens(swapThreshold - addToLiquidityHalf - burnTokenCut);

uint ethBalance = address(this).balance - nftBalance;

uint totalSellFee = (totalTokenTax - (liquidityTax / 2) -
burnTax);

/// Distribute to team and treasury
if (buyTax.treasuryTax + sellTax.treasuryTax > 0) {
    (treasuryWallet).call{
        value: (ethBalance *
            (buyTax.treasuryTax + sellTax.treasuryTax)) /
totalSellFee
    }("");
}

if (buyTax.teamTax + sellTax.teamTax > 0) {
    (teamWallet).call{
        value: (ethBalance * (buyTax.teamTax +
sellTax.teamTax)) /
            totalSellFee
    }("");
}

/// Add ETH to nft balance
nftBalance +=
    (ethBalance * (buyTax.nftTax + sellTax.nftTax)) /
    totalSellFee;

/// Add tokens to liquidity
if (addToLiquidityHalf > 0) {
    _addLiquidity(
        (addToLiquidityHalf),
        ((ethBalance * liquidityTax) / totalSellFee) / 2
    );
}
```

Recommendation

The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

MMSI - Max Mint Supply Inconsistency

Criticality	Minor / Informative
Location	MoonLabs.sol#L260
Status	Unresolved

Description

The contract uses a fixed value for the maximum mint amount instead of retrieving it from the linked NFT contract. This approach can lead to a discrepancy between the actual available max mint supply of NFTs and the preset fixed max mint supply.

```
if (nftIndex < 500) {
    nftIndex++;
} else {
    nftIndex = 1;
}
```

Recommendation

It is recommended to retrieve the maximum mint value from the linked NFT contract. This approach will ensure that the maximum mint value is updated according to the available supply of NFTs. Retrieving the maximum mint value from the NFT contract will also ensure that the contract's behavior is consistent with the linked NFT contract. The contract could initial the max mint amount on the contract constructor.

RAV - Reentrancy Attack Vulnerability

Criticality	Minor / Informative
Location	MoonLabs.sol#L269
Status	Unresolved

Description

The contract is vulnerable to a reentrancy attack, which can occur if a buyer initiates a trade using a contract address as the seller or by initiating a crypto trade from a contract. For instance,

A user tries to transfer tokens to manipulate NFT ownership in order to receive more rewards.

A reentrance attack will occur if the user implements a receive callback, they will have the ability to execute any method again within the same execution thread.

For instance, a user could exploit the NFT reward mechanism by manipulating NFT ownership through buying and selling transactions near the same NFT reward index, leading to potential rewards manipulation.

```
(bool sent, ) = payable(nftOwner).call{value: nftPayout}(
    ""
);
```

Recommendation

The contract could disallow the use of contract addresses for transfer transactions and the contract could aggregate all rewards distributions for the NFTs and execute them all at once at the end of the function.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	MoonLabs.sol#L129,133,139,145,150,155,160,165,205
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.

```
uint _nftPayout
uint8 _maxNftDistribution
address payable _treasuryWallet
address payable _teamWallet
address payable _liqWallet
address _address
bool _taxSwap
uint _swapThreshold
```

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	MoonLabs.sol#L130,135,207
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.

```
nftPayout = _nftPayout
maxNftDistribution = _maxNftDistribution
swapThreshold = _swapThreshold
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	MoonLabs.sol#L79,80,81
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
treasuryWallet = _treasuryWallet
teamWallet = _teamWallet
liqWallet = _liqWallet
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

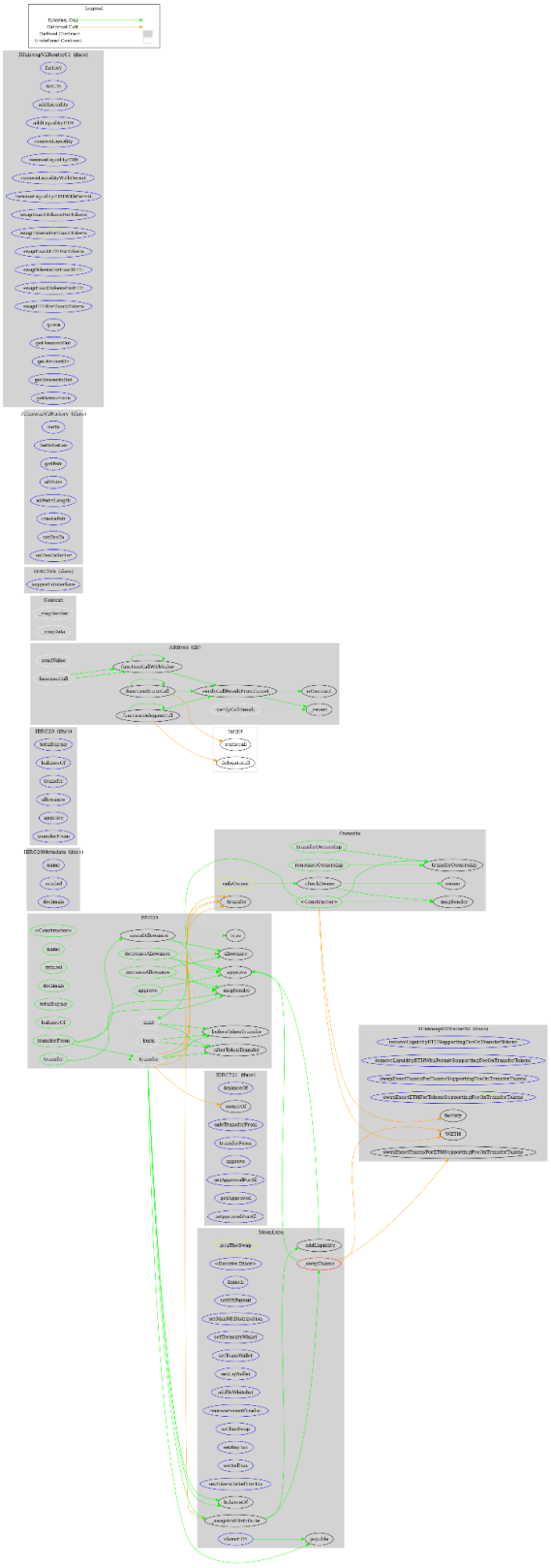
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
MoonLabs	Implementation	ERC20, Ownable		
		Public	✓	ERC20
		External	Payable	-
	launch	External	✓	onlyOwner
	setNftPayout	External	✓	onlyOwner
	setMaxNftDistribution	External	✓	onlyOwner
	setTreasuryWallet	External	✓	onlyOwner
	setTeamWallet	External	✓	onlyOwner
	setLiqWallet	External	✓	onlyOwner
	addToWhitelist	External	✓	onlyOwner
	removeFromWhitelist	External	✓	onlyOwner
	setTaxSwap	External	✓	onlyOwner
	setBuyTax	External	✓	onlyOwner
	setSellTax	External	✓	onlyOwner
	setTokensToSellForTax	External	✓	onlyOwner
	claimETH	External	✓	onlyOwner
	_transfer	Internal	✓	

	_swapTokens	Private	✓	lockTheSwap
	_swapAndDistribute	Private	✓	lockTheSwap
	_addLiquidity	Private	✓	lockTheSwap

Inheritance Graph

Flow Graph



Summary

Moonlabs contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. Moonlabs is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler errors or critical issues. The Contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions.

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