



Cyberscope

Audit Report

MoonLabsStakingCards

April 2023

Network ETH

Address 0xE6FBD188fFc0604E7Aa3290C303C9Dc11ec98D53

Audited by © cyberscope

Table of Contents

Table of Contents	1
Review	2
Audit Updates	2
Findings Breakdown	3
Introduction	4
Roles	4
Diagnostics	5
MT - Mints Tokens	6
Description	6
Recommendation	6
L02 - State Variables could be Declared Constant	7
Description	7
Recommendation	7
L04 - Conformance to Solidity Naming Conventions	8
Description	8
Recommendation	8
Functions Analysis	10
Inheritance Graph	17
Flow Graph	18
Summary	19
Disclaimer	20
About Cyberscope	21

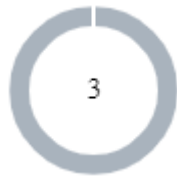
Review

Contract Name	MoonLabsStakingCards
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	3500 runs
Explorer	https://etherscan.io/address/0xe6fbd188ffc0604e7aa3290c303c9dc11ec98d53
Address	0xe6fbd188ffc0604e7aa3290c303c9dc11ec98d53
Network	ETH
Symbol	MLSC

Audit Updates

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

Findings Breakdown



● Critical	0
● Medium	0
● Minor / Informative	3

Severity	Unresolved	Acknowledged	Resolved	Other
● Critical	0	0	0	0
● Medium	0	0	0	0
● Minor / Informative	3	0	0	0

Introduction

The MoonLabsStakingCards contract extends ERC721Enumerable and Ownable contracts from the OpenZeppelin library. It allows for the creation and management of non-fungible tokens (NFTs). The contract has a maximum supply of 500 NFTs, which can be minted by the contract owner only. The contract also provides a function for getting all the token IDs owned by a particular address, setting the base token URI, and claiming all ETH held in the contract. The tokenURI function returns the compiled token URI for a given token ID.

Roles

The contract consists of the owner role. The owner is responsible for the base URI configuration and for minting tokens.

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	MT	Mints Tokens	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved

MT - Mints Tokens

Criticality	Minor / Informative
Status	Unresolved

Description

The contract owner has the authority to mint tokens.

```
function ownerMint(address _to, uint amount) external onlyOwner
{
    uint supply = totalSupply();
    require(supply + amount <= MAX_SUPPLY, "Max supply
reached");
    for (uint i = 1; i <= amount; i++) {
        _safeMint(_to, supply + i);
    }
}
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	contracts/MoonLabsStakingCards.sol#L29
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
string public BASE_EXTENSION = ".json";
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/MoonLabsStakingCards.sol#L29,39,54,87
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.

```
string public BASE_EXTENSION = ".json";
function getTokenIds(address _owner) external view returns
(uint[] memory)
function ownerMint(address _to, uint amount) external onlyOwner
uint _id

...
```

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>.

Functions Analysis

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
ERC721	Implementation	Context, ERC165, IERC721, IERC721Met adata		
		Public	✓	-
	supportsInterface	Public		-
	balanceOf	Public		-
	ownerOf	Public		-
	name	Public		-
	symbol	Public		-
	tokenURI	Public		-

	_baseURI	Internal		
	approve	Public	✓	-
	getApproved	Public		-
	setApprovalForAll	Public	✓	-
	isApprovedForAll	Public		-
	transferFrom	Public	✓	-
	safeTransferFrom	Public	✓	-
	safeTransferFrom	Public	✓	-
	_safeTransfer	Internal	✓	
	_ownerOf	Internal		
	_exists	Internal		
	_isApprovedOrOwner	Internal		
	_safeMint	Internal	✓	
	_safeMint	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_transfer	Internal	✓	
	_approve	Internal	✓	
	_setApprovalForAll	Internal	✓	
	_requireMinted	Internal		
	_checkOnERC721Received	Private	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	

	__unsafe_increaseBalance	Internal	✓	
ERC721Enumerable	Implementation	ERC721, IERC721Enumerable		
	supportsInterface	Public		-
	tokenOfOwnerByIndex	Public		-
	totalSupply	Public		-
	tokenByIndex	Public		-
	_beforeTokenTransfer	Internal	✓	
	_addTokenToOwnerEnumeration	Private	✓	
	_addTokenToAllTokensEnumeration	Private	✓	
	_removeTokenFromOwnerEnumeration	Private	✓	
	_removeTokenFromAllTokensEnumeration	Private	✓	
IERC721Enumerable	Interface	IERC721		
	totalSupply	External		-
	tokenOfOwnerByIndex	External		-
	tokenByIndex	External		-
IERC721Metadata	Interface	IERC721		
	name	External		-
	symbol	External		-

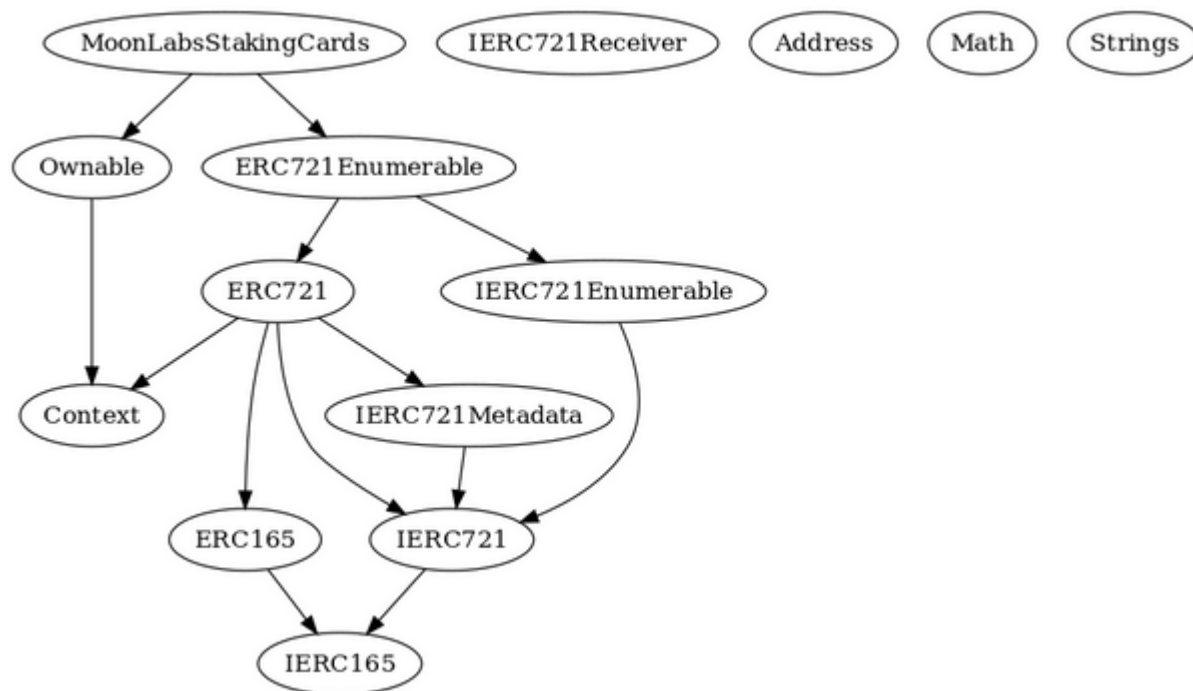
	tokenURI	External		-
IERC721	Interface	IERC165		
	balanceOf	External		-
	ownerOf	External		-
	safeTransferFrom	External	✓	-
	safeTransferFrom	External	✓	-
	transferFrom	External	✓	-
	approve	External	✓	-
	setApprovalForAll	External	✓	-
	getApproved	External		-
	isApprovedForAll	External		-
IERC721Receiver	Interface			
	onERC721Received	External	✓	-
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	

	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	verifyCallResultFromTarget	Internal		
	verifyCallResult	Internal		
	_revert	Private		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
ERC165	Implementation	IERC165		
	supportsInterface	Public		-
IERC165	Interface			
	supportsInterface	External		-
Math	Library			
	max	Internal		
	min	Internal		
	average	Internal		
	ceilDiv	Internal		

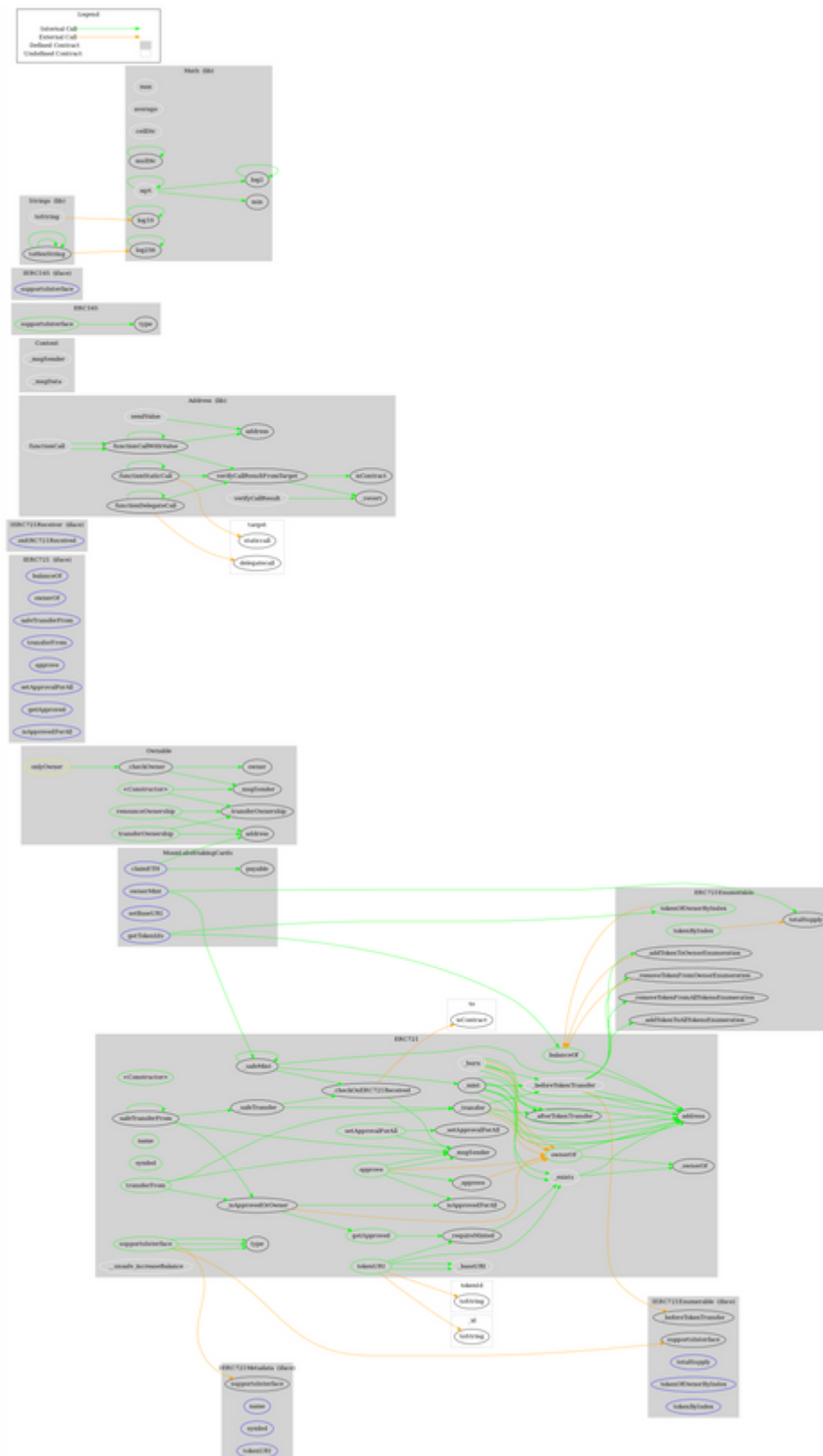
	mulDiv	Internal		
	mulDiv	Internal		
	sqrt	Internal		
	sqrt	Internal		
	log2	Internal		
	log2	Internal		
	log10	Internal		
	log10	Internal		
	log256	Internal		
	log256	Internal		
Strings	Library			
	toString	Internal		
	toHexString	Internal		
	toHexString	Internal		
	toHexString	Internal		
MoonLabsStakingCards	Implementation	ERC721Enumerable, Ownable		
		Public	✓	ERC721
	getTokenIds	External		-
	ownerMint	External	✓	onlyOwner
	setBaseURI	External	✓	onlyOwner

	claimETH	External	Payable	onlyOwner
	tokenURI	Public		-
	_baseURI	Internal		

Inheritance Graph



Flow Graph



Summary

MoonLabsStakingCards contract implements an nft 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>