Cyberscope

Audit Report

Myntflo Staking

January 2023

Github	https://github.com/oxalexa/myntflo-contracts
Commit	45104c691230fee4efe06cd9686bd0ee82365ca7
Audited by	© cyberscope

Table of Contents

Review

Contract Name	MyntfloStaking
Repository	https://github.com/oxalexa/myntflo-contracts
Commit	45104c691230fee4efe06cd9686bd0ee82365ca7
Audit Scope	MyntfloStaking.sol

Audit Updates

Initial Audit	06 Jan 2023

Source Files

Filename	SHA256
@openzeppelin/contracts/metatx/ERC2771Context.sol	350e132f5ebc838e000770ceee044e4541a598b05bf998e96285c859eea5d8ef
@openzeppelin/contracts/metatx/MinimalForwarder.sol	95a2f6b10918f410d143f27581a0a1c7603c9dd774c31899bb4cc20cc1619515
@openzeppelin/contracts/security/ReentrancyGuard.sol	aa73590d5265031c5bb64b5c0e7f84c44cf5f8539e6d8606b763adac784e8b2e
@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol	3e7aa0e0f69eec8f097ad664d525e7b3f0a3fda8dcdd97de5433ddb131db86ef
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db8003d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	fa36a21bd954262006d806b988e4495562e7b50420775e2aa0deecb596fd1902
@openzeppelin/contracts/token/ERC721/IERC721.sol	fde830ac73ef320f7e3ce977b8cf567173f1e479ba86d584498f8362a67a5dc0
@openzeppelin/contracts/utils/Address.sol	1e0922f6c0bf6b1b8b4d480dcabb691b1359195a297bde6dc5172e79f3a1f826
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol	fc0e6c5d7184bd03b8deae6ca9a48a1eaaecf9f5e4703611aabfb63401e6d43f
@openzeppelin/contracts/utils/cryptography/ECDSA.sol	4e45d53327d561848fbcf381262ec5c0ac91b2f1f06432210bf76db55279d945
@openzeppelin/contracts/utils/introspection/IERC165.sol	701e025d13ec6be09ae892eb029cd83b3064325801d73654847a5fb11c58b1e5

@openzeppelin/contracts/utils/Strings.sol	34127ad0054df5963b0fd694c1b313d17e9114a2f426b85526d6d976210298ab
contracts/testingDeploy/MyntfloStaking.sol	a2c2bb8abdaabb357696863600d82672b0efb673b99460de2ac4cabcb6bb6ab9
hardhat/console.sol	47a72fddde001a2977f460b759ce035f9788daa34c186ce1f9a10066236b3f75

Introduction

MyntfloStaking implements an NFT staking mechanism where users have the ability to stake NFTs in order to receive rewards. The elligible NFT collections are defined by the contract owner. Additionally the contract owner has tha authority to change the rewarded token address. The reward amount is calculated by a formula proportionally to the time period that have elapsed. The reward amount is redeems by an ERC20 token. The users have the ability to claim their rewards and unstack their holding any time.

The contract does not implement any mechanism that guarantees the rewards amount. This is a common methodology in many staking contracts since the staking period does not have an expiration date and the reward token does not implement a public mint method. The contract owner is responsible for keeping the reward reserves in a healthy amount.

Roles

Public Roles

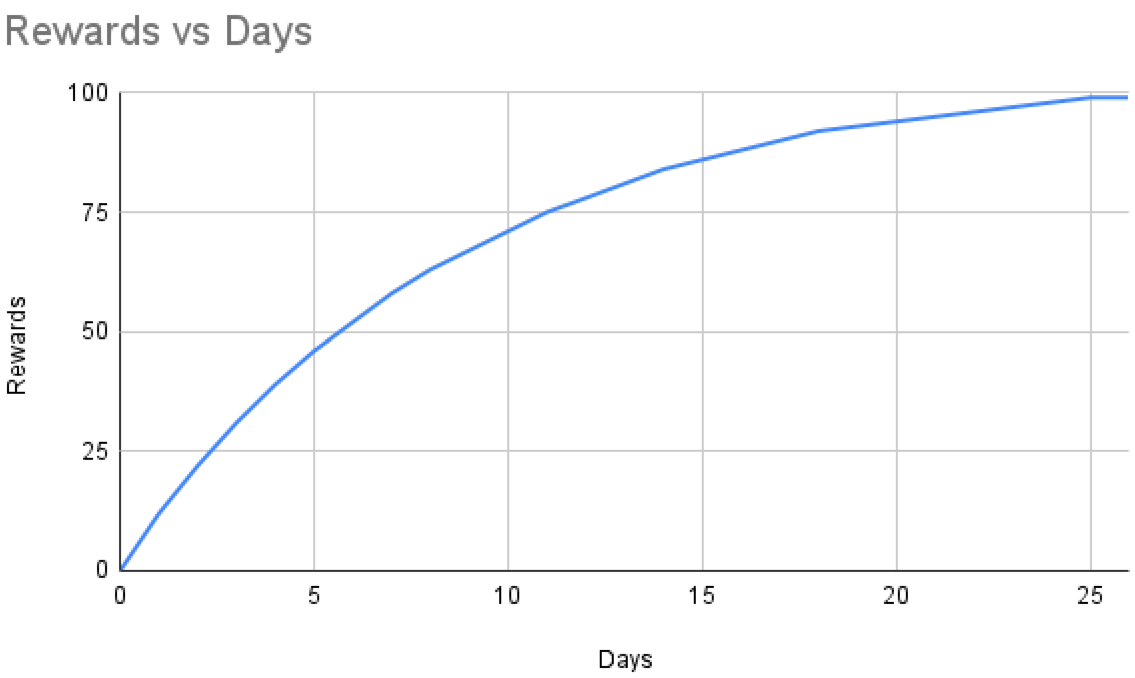
- stake()
- unstake()
- claimRewards()

Admin Roles

- setRewardsToken()
- setElegibleCollection()
- setElegibleCollections()

Rewards Formula

As integrite part of the audit report, the rewards algorithm was tested by providing applicable values. The following chart depicts the rewards (Y-Axis) that will be provided when a specific time period has been elapsed (X-Axis). For instance, if the time that has been elapsed since the last claim is 5 days, then the rewarded amount will be 46. If more than 25 days have been elapsed then the reward amount is stabilized to 99.



The following table depicts the rewarded amount across the period that has elapsed.

Days	Reward
1	12
2	22
3	31
4	39
5	46
6	52
7	58
8	63
9	67
10	71
11	75
12	78
13	81
14	84
15	86
16	88
17	90
18	92
19	93
20	94
21	95
22	96
23	97
24	98
25	99

Diagnostics

🔴 Critical 🟠 Medium ● Minor / Informative

Severity	Code	Description	Status
●	DPI	Decimals Precision Inconsistency	Unresolved
●	PDT	Performant Data Type	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L19	Stable Compiler Version	Unresolved

DPI - Decimals Precision Inconsistency

Criticality	Minor / Informative
Status	Unresolved

Description

The decimals field of a contract's ERC20 token can be used to specify the number of decimal places that the token uses. For example, if decimals is set to `8`, it means that the smallest unit of the token is `0.00000001`, and if decimals are set to `18`, it means that the smallest unit of the token is `0.000000000000000001`.

However, there is an inconsistency in the way that the decimals field is handled in some ERC20 contracts. The ERC20 specification does not specify how the decimals field should be implemented, and as a result, some contracts use different precision numbers.

This inconsistency can cause problems when interacting with these contracts, as it is not always clear how the decimals field should be interpreted. For example, if a contract expects the decimals field to be 18 digits, but the contract being interacted with uses 8 digits, the result of the interaction may not be what was expected.

The contract uses the `decay` method to calculate the rewarded amount. The `decay` is not taking into consideration the rewarded token decimals. As a result, the reward amount may vary according to the size of the token decimal. As a result, unexpected reward amounts may be produced.

```
uint256 rewards = decay(secondsPassed);
rewardsToken.safeTransfer(_msgSender(), rewards);
```

Recommendation

To avoid these issues, it is important to carefully review the implementation of the `decay` method. The team is advised to take into consideration the rewarded token decimals inside the `decay` algorithm.

The following example depicts the reward result of 4 tokens with different decimals precision. We assume that the `decay` will return the value `10`.

ERC20	Decimals	Reward
Token 1	6	3
Token 2	9	5
Token 3	18	10
Token 4	24	13

All the decimals could be normalized to 18 since it represents the ERC20 token with the greatest digits.

PDT - Performant Data Type

Criticality	Minor / Informative
Location	contracts/testingDeploy/MyntfloStaking.sol#L26
Status	Unresolved

Description

The `StakedToken` struct stores staker's address. The `StakedToken` struct is part of the `Staker` struct. The `Staker` struct is pointed by the staker's mapping. The contract does not contain any other structure that is pointing directly to the `StakedToken` struct. The `StakedToken.staker` variable is used by the contract to determine if the stack record is active. This requirement could also be archived by exploiting a boolean data type.

```
struct StakedToken {
    address staker;
    uint256 tokenId;
    uint256 timeStaked;
    uint256 timeOfLastUpdate;
    address contractAddress;
}
```

Recommendation

Since the staker's address is solely used as a boolean indicator. The team is advised to use a boolean indicator rather than an address since the boolean data type will reduce the storage size and improve the gas cost.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/testingDeploy/MyntfloStaking.sol#L61,89,131,135,139,172,178,225
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.

```
address _tokenContract
uint256 _tokenId
IERC20 _rewardsToken
address _collection
bool _elegible
bool[] memory _elegible
address[] memory _collections
address _staker
address _user
```

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.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/testingDeploy/MyntfloStaking.sol#L2
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.4;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
ERC2771Context	Implementation	Context		
		Public	✓	-
	isTrustedForwarder	Public		-
	_msgSender	Internal		
	_msgData	Internal		
MinimalForwarder	Implementation	EIP712		
		Public	✓	EIP712
	getNonce	Public		-
	verify	Public		-
	execute	Public	Payable	-
ReentrancyGuard	Implementation			
		Public	✓	-
IERC20Permit	Interface			
	permit	External	✓	-
	nonces	External		-
	DOMAIN_SEPARATOR	External		-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-

	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeERC20	Library			
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeApprove	Internal	✓	
	safeIncreaseAllowance	Internal	✓	
	safeDecreaseAllowance	Internal	✓	
	safePermit	Internal	✓	
	_callOptionalReturn	Private	✓	
IERC721	Interface	IERC165		
	balanceOf	External		-
	ownerOf	External		-
	safeTransferFrom	External	✓	-
	safeTransferFrom	External	✓	-
	transferFrom	External	✓	-
	approve	External	✓	-
	setApprovalForAll	External	✓	-
	getApproved	External		-
	isApprovedForAll	External		-
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	

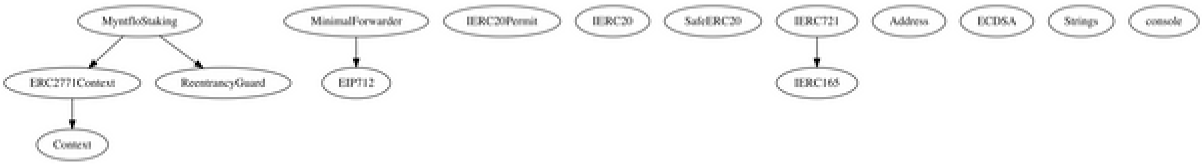
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	verifyCallResult	Internal		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
EIP712	Implementation			
		Public	✓	-
	_domainSeparatorV4	Internal		
	_buildDomainSeparator	Private		
	_hashTypedDataV4	Internal		
ECDSA	Library			
	_throwError	Private		
	tryRecover	Internal		
	recover	Internal		
	tryRecover	Internal		
	recover	Internal		
	tryRecover	Internal		
	recover	Internal		
	toEthSignedMessageHash	Internal		
	toEthSignedMessageHash	Internal		

	toTypedDataHash	Internal		
IERC165	Interface			
	supportsInterface	External		-
Strings	Library			
	toString	Internal		
	toHexString	Internal		
	toHexString	Internal		
	toHexString	Internal		
MyntfloStaking	Implementation	ReentrancyGuard, ERC2771Context		
		Public	✓	ERC2771Context
	stake	External	✓	nonReentrant
	unstake	External	✓	nonReentrant
	setRewardsToken	External	✓	onlyOwner
	setElegibleCollection	External	✓	onlyOwner
	setElegibleCollections	External	✓	onlyOwner
	claimRewards	External	✓	-
	availableRewards	Public		-
	getStakedTokens	Public		-
	decay	Internal		
	calculateRewards	Internal		
console	Library			
	_sendLogPayload	Private		
	log	Internal		
	logInt	Internal		

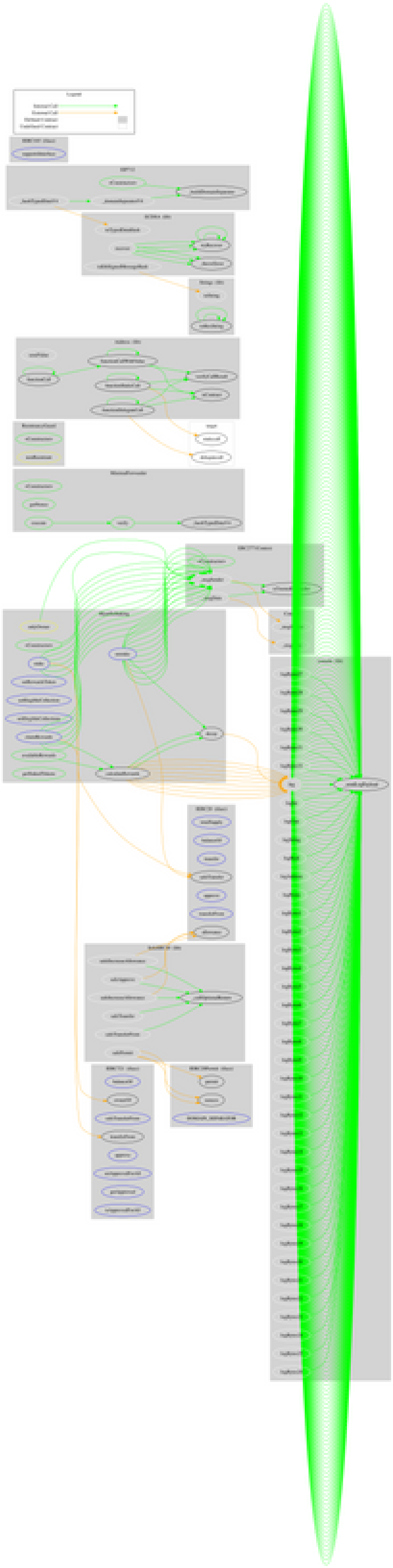
	logUint	Internal		
	logString	Internal		
	logBool	Internal		
	logAddress	Internal		
	logBytes	Internal		
	logBytes1	Internal		
	logBytes2	Internal		
	logBytes3	Internal		
	logBytes4	Internal		
	logBytes5	Internal		
	logBytes6	Internal		
	logBytes7	Internal		
	logBytes8	Internal		
	logBytes9	Internal		
	logBytes10	Internal		
	logBytes11	Internal		
	logBytes12	Internal		
	logBytes13	Internal		
	logBytes14	Internal		
	logBytes15	Internal		
	logBytes16	Internal		
	logBytes17	Internal		
	logBytes18	Internal		
	logBytes19	Internal		
	logBytes20	Internal		
	logBytes21	Internal		
	logBytes22	Internal		
	logBytes23	Internal		
	logBytes24	Internal		

	logBytes25	Internal		
	logBytes26	Internal		
	logBytes27	Internal		
	logBytes28	Internal		
	logBytes29	Internal		
	logBytes30	Internal		
	logBytes31	Internal		
	logBytes32	Internal		
	log	Internal		

Inheritance Graph



Flow Graph



Summary

Myntflo contract implement an NFT staking 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