Cyberscope

Audit Report

Meta Ape Militia

November 2022

Type	ERC20
Network	GOERLI ETH
Address	0x8a027d7172D820DFEA05CfB4f4069A11EE0E042D
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	BulkMamStake
Compiler Version	v0.8.11+commit.d7f03943
Optimization	200 runs
Licence	MIT
Explorer	https://goerli.etherscan.io/token/0x8a027d7172D820DFEA05CfB4f4069A11EE0E042D
Domain	https://www.metaapemilitia.com

Source Files

Filename	**SHA256**
contract.sol	03309145e38a193f8f32338af254b1455e831b8aa884d2e667cba3e62881b464

Audit Updates

Initial Audit	3rd November 2022
Corrected	

Introduction

The contract BulkMamStake implements an NFT staking contract. There are two types of NFT stakers mam and mutant stakers.

Roles

The contract has an owner role. The owner has the authority to

- Configure the addresses of the NFTs contracts.
- Configure the token address.
- Configure the dayRate of the reward.
- Add and remove users from the blacklist.
- Withdraw contract tokens and liquidity.

Users have the authority to

- Stake NFTs.
- Unstake NFTs.
- Collect staking rewards.
- Withdraw stuck NFTs if the owner enables this functionality.
- View if they are eligible to withdraw daily.
- View staked NFTs

Contract Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	ISAD	Incorrect Staked Amount Decrease	Unresolved
●	PAP	Payout Access Permissions	Unresolved
●	SVI	State Variables Inconsistency	Unresolved
●	OCTD	Transfers Contract's Tokens	Unresolved
●	BC	Blacklists Addresses	Unresolved
●	STC	Succeeded Transfer Check	Unresolved
●	DSI	Data Structure Improvement	Unresolved
●	CR	Code Repetition	Unresolved
●	ADU	Arbitrary Decimals Usage	Unresolved
●	DNV	Descriptive Numberic Values	Unresolved
●	RMA	Redundant Method Argument	Unresolved
●	MC	Missing Check	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved

●	L07	Missing Events Arithmetic	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved

ISAD - Incorrect Staked Amount Decrease

Criticality	critical
Location	contract.sol#L805,914
Status	Unresolved

Description

The contract decreases the mamStaked and numberMamStaked properties without checking if the tokenId has actually been transferred. One case like that would be if the caller adds an irrelevant tokenId as an argument.

```
function unstakeMamById(address _owner, uint16 tokenId) external nonReentrant
{

    ...
    mamStaked--;
    mamstaker[_owner].daily = true;
    mamstaker[_owner].tokenStakedAt = block.timestamp;
    mamstaker[_owner].numberMamStaked--;

    emit MamUnstaked(msg.sender, totalPayout);

}
```

Recommendation

The contract should not allow the decrease of mamStaked/numberMamStaked variables if the tokenId has not unstaked from the user's struct.

PAP - Payout Access Permissions

Criticality	critical
Location	contract.sol#L843,953
Status	Unresolved

Description

The payoutMutant and payoutMam methods can be called by any user that is not blacklisted. As a result, any user can execute the payout method from the staked amount of other users. This will result in state mutations that are not expected by the users that have staked. For instance, the `daily` variable will be set to true.

```
function payoutMutant(address _owner) external nonReentrant {
    require(!blacklistedUsers[msg.sender], "User is blacklisted");
```

Recommendation

The contract should allow the payout only to the users that are related to the staked account.

SVI - State Variables Inconsistency

Criticality	critical
Location	contract.sol#L1059
Status	Unresolved

Description

The method escapeHatchWithdrawal transfer all the user's staked assets back. The corresponding properties are not updated. As a result, it will create inconsistency between the state variables and the actual state.

```
function escapeHatchWithdrawal(address _owner, uint8 _type, uint16[] calldata
_ids) external {
    require(escapeHatchOpen, "Escape hatch is closed");
    if(_type == 2) {
        require(mutantstaker[_owner].owner == msg.sender, "Can't unstake
someone else's nft");
        for(uint8 i; i < _ids.length; i++) {
            mutant.transferFrom(address(this), _owner, _ids[i]);
        }
    } else {
        require(mamstaker[_owner].owner == msg.sender, "Can't unstake someone
else's nft");
        for(uint8 j; j < _ids.length; j++) {
            mam.transferFrom(address(this), _owner, _ids[j]);
        }
    }
}
```

Recommendation

The contract should update the state variables according to the withdrawal functionality.

OCTD - Transfers Contract's Tokens

Criticality	minor / informative
Location	contract.sol#L1131
Status	Unresolved

Description

The contract owner has the authority to claim all the balance of the contract. The owner may take advantage of it by calling the `emergencyTokenWithdraw` function.

```
function emergencyTokenWithdraw() external onlyOwner {
    uint256 balance = token.balanceOf(address(this));
    token.transfer(msg.sender, balance);
  }
```

Recommendation

The contract could keep a reasonable amount of tokens as a reserve for paying out the staked addresses.

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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

BC - Blacklists Addresses

Criticality	minor / informative
Location	contract.sol#L1123
Status	Unresolved

Description

The contract owner has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `blacklistUser` function.

```
function blacklistUser(address _owner) external onlyOwner {
    blacklistedUsers[_owner] = true;
 }
```

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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

STC - Succeeded Transfer Check

Criticality	minor / informative
Location	contract.sol
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
mam.transferFrom(address(this), msg.sender, tokenId);
…
token.transfer(msg.sender, totalPayoutPS);
…
token.transfer(_owner, totalPayoutPS);
…
```

Recommendation

The contract should check if the result of the transfer methods is successful.

DSI - Data Structure Improvement

Criticality	minor / informative
Location	contract.sol#L644,656
Status	Unresolved

Description

The structures MamStaker and MutantStaker have the same shape. The only difference is the name of the number of staked property. This diversion increases significantly the code size and complexity of the contract.

```
struct MamStaker {
    uint16[] ids;
    uint256 numberMamStaked;
    uint16[] remainingIds;
    uint256 tokenStakedAt;
    bool daily;
    address owner;
}

struct MutantStaker {
    uint16[] ids;
    uint256 numberMutantStaked;
    uint16[] remainingIds;
    uint256 tokenStakedAt;
    bool daily;
    address owner;

}
```

Recommendation

MamStaker could be the same struct with MutantStaker if the `numberMamStaked/numberMutantStaked` renamed to `numberStaked`. So all the mam and mutant staking methods could reuse all the internal functionality.

CR - Code Repetition

Criticality	minor / informative
Location	contract.sol
Status	Unresolved

Description

There are code segments that are repetitive in the contract. Those segments increase the code size of the contract unnecessarily.
`unstakeMamById` could reuse the same functionality with `unstakeMutantById`, `payoutMam` with `payoutMutant` and
`unstakeMam` with unstakeMutant.

These methods are implementing the same functionality. The only difference is the mamstaker/mutantstaker property and the payout rate.

Recommendation

The contract could create an internal function that contains the code segment and accepts the two abovementioned parameters and remove it from all the sections.

ADU - Arbitrary Decimals Usage

Criticality	minor / informative
Location	contract.sol
Status	Unresolved

Description

The contract multiplies the amount with 10**18 in order to calculate the tokens with the decimals precision. The token property is mutable. The contract owner has the authority to add any token with different amount of decimals. As a result the precision will be wrong.

```
uint256 totalPayout = payout *10**18;
uint256 totalPayoutPS = totalPayout.div(86400);
amountPaid+= totalPayoutPS;

token.transfer(msg.sender, totalPayoutPS);
```

Recommendation

The contract should get the decimals from the contract instead of adding a fixed 10**18 value.

DNV - Descriptive Numberic Values

Criticality	minor / informative
Location	contract.sol
Status	Unresolved

Description

The contract is using fixed numbers like 86400 in order to calculate time-related expressions. Solidity provides keywords that improve the readability of time-related values.

```
uint256 totalPayoutPS = totalPayout.div(86400);
```

Recommendation

The contract could use the unit `1 day` instead of a fixed value `86400`. This will increase the readability of the contract.

RMA - Redundant Method Argument

Criticality	minor / informative
Location	contract.sol
Status	Unresolved

Description

The methods unstakeMamById, unstakeMam, unstakeMutantById and unstakeMutant accepts the owner as argument. The contract checks if this argument is equal with the sender. As a result, the argument is redundant since the it could be only the owner.

```
function unstakeMutant(address _owner) external {
    require(!blacklistedUsers[msg.sender], "User is blacklisted");
    require(mutantstaker[_owner].owner == msg.sender, "Can't unstake someone
else's nfts");
```

Recommendation

The owner parameter could be eliminated from the methods parameters.

MC - Missing Check

Criticality	minor / informative
Location	contract.sol#L699,1031,1036,1040
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(address _mam, address _mutant, address _token)  {
        mam = IERC721(_mam);
        mutant = IERC721(_mutant);
        token = IERC20(_token);
    }
```

```
function setNFTAddress(address _newMamToken, address _newMutantToken) external
onlyOwner {
      mam = IERC721(_newMamToken);
      mutant = IERC721(_newMutantToken);
  }

  function setTokenAddress(address _newToken) external onlyOwner {
      token = IERC20(_newToken);
  }

function setDayRate(uint256 _newRate) external onlyOwner {
      dayRate = _newRate;
  }
```

Recommendation

The dayRate should not be zero because it will prevent the rewards redeem. The addresses should not be zero.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contract.sol#L839,1091,1031,1123,1036,1102,868,759,1040,1080,1050,949,1055,1119,801,787,1071,1115,1111,772,1106,1044,910,978
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.

```
_owner
_newMamToken
_newToken
_newMutantToken
_newRate
_state
_type
_ids
_time
...
```

Recommendation

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

L07 - Missing Events Arithmetic

Criticality	minor / informative
Location	contract.sol#L1040,1071
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.

```
dayRate = _newRate
period = _time
```

Recommendation

Emit an event for critical parameter changes.

L14 - Uninitialized Variables in Local Scope

Criticality	minor / informative
Location	contract.sol#L888,727,1064,998,991,814,750,1059,881,923
Status	Unresolved

Description

The are variables that are defined in the local scope and are not initialized.

```
i_scope_0
i
j
...
```

Recommendation

All the local scoped variables should be initialized.

L15 - Local Scope Variable Shadowing

Criticality	minor / informative
Location	contract.sol#L759,1123,1080,772,839,1102,1115,1119,1091,1055,787,801,910,949,1111,1106,978,868
Status	Unresolved

Description

The are variables that are defined in the local scope containing the same name from an upper scope.

```
_owner
...
```

Recommendation

The local variables should have different names from the upper scoped variables.

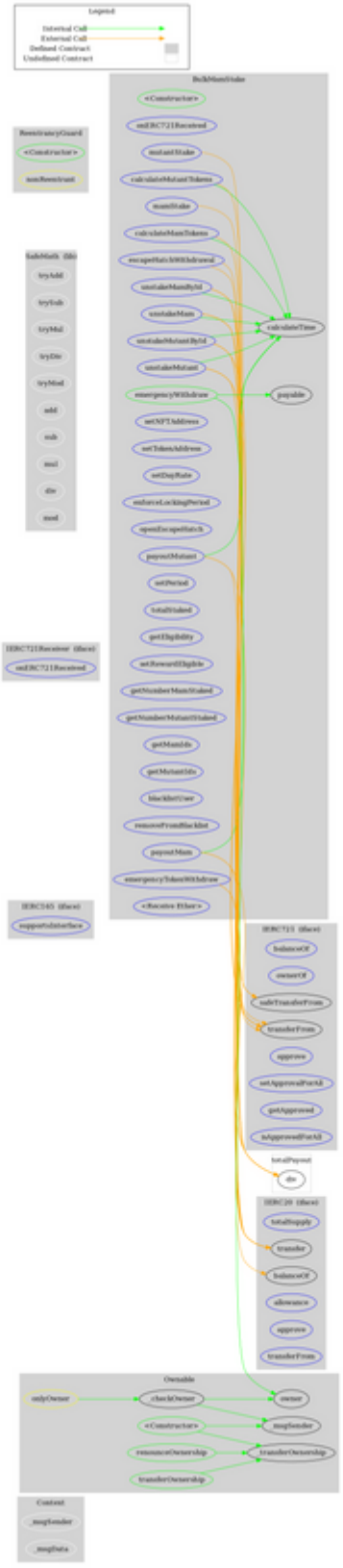
Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
	<Constructor>	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	✓	-
IERC165	Interface			
	supportsInterface	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	✓	-
SafeMath	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
ReentrancyGuard	Implementation			
	<Constructor>	Public	✓	-
BulkMamStake	Implementation	Ownable, IERC721Receiver, ReentrancyGuard		
	<Constructor>	Public	✓	-
	onERC721Received	External		-
	mamStake	External	✓	-

	mutantStake	External	✓	-
	calculateTime	Public		-
	calculateMamTokens	External		-
	calculateMutantTokens	External		-
	unstakeMamById	External	✓	nonReentrant
	payoutMam	External	✓	nonReentrant
	unstakeMam	External	✓	nonReentrant
	unstakeMutantById	External	✓	nonReentrant
	payoutMutant	External	✓	nonReentrant
	unstakeMutant	External	✓	-
	setNFTAddress	External	✓	onlyOwner
	setTokenAddress	External	✓	onlyOwner
	setDayRate	External	✓	onlyOwner
	enforceLockingPeriod	External	✓	onlyOwner
	openEscapeHatch	External	✓	onlyOwner
	escapeHatchWithdrawal	External	✓	-
	setPeriod	External	✓	onlyOwner
	totalStaked	External		-
	getEligibility	External		-
	setRewardEligible	External	✓	onlyOwner
	getNumberMamStaked	External		-
	getNumberMutantStaked	External		-
	getMamIds	External		-
	getMutantIds	External		-
	blacklistUser	External	✓	onlyOwner
	removeFromBlacklist	External	✓	onlyOwner
	emergencyTokenWithdraw	External	✓	onlyOwner
	emergencyWithdraw	Public	✓	onlyOwner nonReentrant
	<Receive Ether>	External	Payable	-

Contract Flow



Domain Info

Domain Name	metaapemilitia.com
Registry Domain ID	2667001015_DOMAIN_COM-VRSN
Creation Date	2022-01-09T09:25:37Z
Updated Date	2022-01-09T09:25:37Z
Registry Expiry Date	2024-01-09T09:25:37Z
Registrar WHOIS Server	whois.godaddy.com
Registrar URL	https://www.godaddy.com
Registrar	GoDaddy.com, LLC
Registrar IANA ID	146

The domain was created 42 weeks and 5 days before the creation of the audit. It will expire in 24 months.

There is no public billing information, the creator is protected by the privacy settings.

Summary

Meta Ape Militia implements a staking mechanism. The audit focuses in security vulnerabilities, business logic concerns and potential improvements.

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