Cyberscope

Audit Report

Puli Runners

April 2022

Type	ERC20
Network	RINKEBY
Address	0x5341e6fccc30Cba7b9bf032C9164913009Bd8ED0
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	PuliRunners
Compiler Version	v0.8.1+commit.df193b15
Optimization	200 runs
Licence	MIT
Explorer	https://bscscan.com/token/0x5341e6fccc30Cba7b9bf032C9164913009Bd8ED0
Symbol	PRNR
Decimals	0
Total Supply	2
Domain	pulitoken.net

Source Files

Filename	**SHA256**
contract.sol	7a7e84ded3a7542f7e53f9b97e15ad2030068a57a05d69405a17793d4265f6f8

Audit Updates

Initial Audit	11th April 2022
Corrected	

Contract Analysis

- The contract owner can whitelist addresses to buy nfts.
- A whitelist describes the quantity and the price that each whitelisted user can buy.
- The price can be either tokens or the native currency.
- The whitelisted users can buy NFTs by either paying in the native currency or in specific tokens that have been defined by the whitelist descriptor.
- The non-whitelisted users can buy NFTs by either paying in the native currency or in specific tokens that have been defined by the general price list.

Contract Owner Privileges

- The contract owner has the authority to mint NFTs to any address.
- The contract owner has the authority to mint the remaining whitelisted NFTs to the owner's address.

Whitelisted Maximum Quantity Check

Criticality	minor
Location	contract.sol#L1941

Description

The contract does not check if the quantity of NFTs that is going to be minted is less than the maximum allowed. As a result, the whitelisted users counter will underflow.

```
whiteListMap[batchDescription].users[_msgSender()]--;
```

Recommendation

The contract should check if the quantity that is going to be minted by the whitelisted users is less than the maximum allowed.

Total Supply Inflation

Criticality	minor
Location	contract.sol#L1971,1987

Description

The contract contains a maximum amount of the total NFTs that can be minted. This number is fixed to 2010. The regular buys that are not whitelisted do not check if the total supply has reached the maximum allowed supply.

```
function mint(uint256 quantity) public payable {
    require(price[NATIVE] > 0, "Price is not set for minting");
    require(
        msg.value >= price[NATIVE] * quantity,
        "Insufficient amount, cannot mint"
    );

    for (uint256 index = 0; index < quantity; index++) {
        _safeMint(_msgSender(), tokenIdCounter);
        totalSupply++;
        emit Mint(msg.sender, tokenIdCounter++);
    }
}
```

Recommendation

The contract should check if the maximum allowed supply has been reached.

Remove Whitelist Naming Convention

Criticality	minor
Location	contract.sol#L2024

Description

The removeWhitelist method mints all the remaining NFTs of a whitelisted user to the contract owner. These NFTs have not minted yet, it is registered to the user's whitelist as a potential buy. Hence, the method does not solely remove the whitelist from the users, but it does mint the NFTs to the contract owner as well.

```
_safeMint(_msgSender(), tokenId); //Remove whitelist and send to owner
```

Recommendation

The method could have a more representative name according to the business logic or the implementation could mirror the expected functionality.

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description
	CO	Code Optimization
	MC	Missing Check
	L01	Public Function could be Declared External
	L04	Conformance to Solidity Naming Conventions
	L09	Dead Code Elimination
	L12	Using Variables before Declaration
	L15	Local Scope Variable Shadowing

CO - Code Optimization

Criticality	minor
Location	contract.sol#L1958,1980,1994,2047

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 contains loops that increase the variables on every iteration. This process produces `O(n)` total computation at the end of the loop.

```
for (uint256 index = 0; index < quantity; index++) {
    uint256 tokenId = whiteListMap[batchDescription].start +
        whiteListMap[batchDescription].counter;

    _safeMint(_msgSender(), tokenId);

    whiteListMap[batchDescription].users[_msgSender()]--; // Decrement counter
for specific user
    whiteListMap[batchDescription].counter++; // Increment batch counter
    totalSupply++;
    emit WhiteListMint(msg.sender, tokenId);
}
```

Recommendation

The total increased value could be added once outside the loop scope. As a result, complexity will be decreased to `O(1)` .

In this example the improvement could be applied to the `whiteListMap[batchDescription].users[_msgSender()]`, `whiteListMap[batchDescription].counter` and `totalSupply` variables.

MC - Missing Check

Criticality	minor
Location	contract.sol#L1937,1978,2176

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.

```
token.transferFrom(msgSender(), address(this), quantity * _price);
```

Recommendation

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

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L737,754,761,768,787,811,825,839,1179,1187,1637,1689,1785,1787,1828,1857,1881,1902,1922,1971,1987,2001,2007,2024,2130,2144,2150

Description

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

```
withdraw
setWithdrawalAddress
setBaseTokenURI
removeWhitelist
getWhitelistPrice
getPrice
mint
mintByToken
mintWhitelistToken
...
```

Recommendation

Use the external attribute for functions never called from the contract

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L854,1787,1816,2130,2144

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.

```
_withdrawalAddress
_baseToken
_tokenId
_to
_data
```

Recommendation

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

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L406,416,495,505,468,478,1226,1236,1230,1244,780,979,613,629

Description

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

```
toHexString
_burn
_baseURI
reset
increment
decrement
current
functionStaticCall
functionDelegateCall
...
```

Recommendation

Remove unused functions.

L12 - Using Variables before Declaration

Criticality	minor
Location	contract.sol#L1070,1072

Description

The contract is using a variable before the declaration. This is usually happening either if it has not been declared yet or the variable has been declared in a different scope.

```
reason
retval
```

Recommendation

The variables should be declared before any usage of them.

L15 - Local Scope Variable Shadowing

Criticality	minor
Location	contract.sol#L1772,1773

Description

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

```
symbol
name
```

Recommendation

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

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
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	✓	-
	transferFrom	External	✓	-
	approve	External	✓	-
	getApproved	External		-
	setApprovalForAll	External	✓	-
	isApprovedForAll	External		-
	safeTransferFrom	External	✓	-
IERC721Receiver	Interface			
	onERC721Received	External	✓	-
IERC721Metadata	Interface	IERC721		
	name	External		-

	symbol	External		-
	tokenURI	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		
Strings	Library			
	toString	Internal		
	toHexString	Internal		
	toHexString	Internal		
ERC165	Implementation	IERC165		
	supportsInterface	Public		-
ERC721	Implementation	Context, ERC165, IERC721, IERC721Metadata		
	<Constructor>	Public	✓	-
	supportsInterface	Public		-
	balanceOf	Public		-
	ownerOf	Public		-

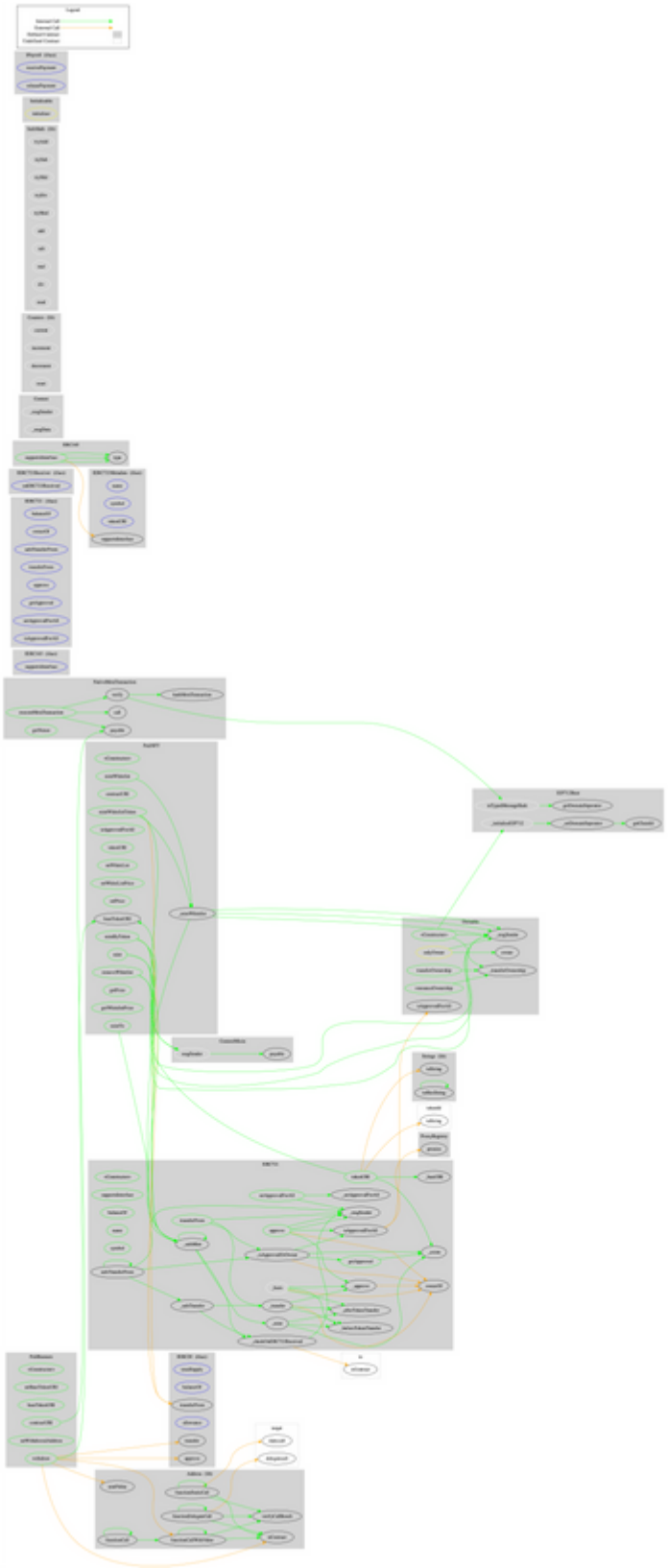
	name	Public		-
	symbol	Public		-
	tokenURI	Public		-
	_baseURI	Internal		
	approve	Public	✓	-
	getApproved	Public		-
	setApprovalForAll	Public	✓	-
	isApprovedForAll	Public		-
	transferFrom	Public	✓	-
	safeTransferFrom	Public	✓	-
	safeTransferFrom	Public	✓	-
	_safeTransfer	Internal	✓	
	_exists	Internal		
	_isApprovedOrOwner	Internal		
	_safeMint	Internal	✓	
	_safeMint	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_transfer	Internal	✓	
	_approve	Internal	✓	
	_setApprovalForAll	Internal	✓	
	_checkOnERC721Received	Private	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
Counters	Library			
	current	Internal		
	increment	Internal	✓	

	decrement	Internal	✓	
	reset	Internal	✓	
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		
ContextMixin	Implementation			
	msgSender	Internal		
Initializable	Implementation			
EIP712Base	Implementation	Initializable		
	_initializeEIP712	Internal	✓	initializer
	_setDomainSeperator	Internal	✓	
	getDomainSeperator	Public		-
	getChainId	Public		-
	toTypedMessageHash	Internal		
NativeMetaTransaction	Implementation	EIP712Base		
	executeMetaTransaction	Public	Payable	-
	hashMetaTransaction	Internal		
	getNonce	Public		-
	verify	Internal		

OwnableDelegateProxy	Implementation			
ProxyRegistry	Implementation			
PuliNFT	Implementation	ERC721, ContextMixin, NativeMetaTransaction, Ownable		
	<Constructor>	Public	✓	ERC721
	baseTokenURI	Public		-
	contractURI	Public		-
	mintTo	Public	✓	onlyOwner
	isApprovedForAll	Public		-
	tokenURI	Public		-
	setWhiteList	Public	✓	onlyOwner
	setWhiteListPrice	Public	✓	onlyOwner
	setPrice	Public	✓	onlyOwner
	mintWhitelist	Public	Payable	-
	mintWhitelistToken	Public	✓	-
	_mintWhitelist	Internal	✓	
	mintByToken	Public	✓	-
	mint	Public	Payable	-
	getPrice	Public		-
	getWhitelistPrice	Public		-
	removeWhitelist	Public	✓	onlyOwner
IPayroll	Interface			
	receivePayment	External	Payable	-
	receivePayment	External	✓	-
	releasePayment	External	Payable	-
	releasePayment	External	✓	-
PuliRunners	Implementation	PuliNFT		

	<Constructor>	Public	✓	PuliNFT
	setBaseTokenURI	Public	✓	onlyOwner
	baseTokenURI	Public		-
	contractURI	Public		-
	setWithdrawalAddress	Public	✓	onlyOwner
	withdraw	Public	Payable	onlyOwner

Contract Flow



Summary

Puli Runners is an NFT contract that gives the ability to the users to buy NFT under a specific price. There are whitelisted users that can buy a specific amount of NFTs at a different price. The price could be either in the native currency or in tokens. This audit focuses on the business logic flow, security 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 provides all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io