RugFreeCoins Audit

Nut Gain NFT Marketplace

Smart Contract Security Audit

May 17, 2022

Contents

Audit details

Audited project
NUTG NFT Marketplace

Contract Address
0x937cD45350164EC3802eb3fA979a92a331804094

Client contact
NUTGAIN Team

Blockchain
Binance smart chain

Project website
https://nft.nutgain.io/

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Rugfreecoins and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Rugfreecoins) owe no duty of care towards you or any other person, nor does Rugfreecoins make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Rugfreecoins hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Rugfreecoins hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Rugfreecoins, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Background

Rugfreecoins was commissioned by the NUTGAIN Team to perform an audit of the smart contract.

https://bscscan.com/address/0x937cD45350164EC3802eb3fA979a92a331804094

The focus of this audit is to verify that the smart contract is secure, resilient, and working according to the specifications.

The information in this report should be used to understand the risk exposure of the smart contract, project feasibility, and long-term sustainability, and as a guide to improving the security posture of the smart contract by remediating the issues that were identified.

.

About the project

In the current stage

$NUTGV2 Token holders: every NFT purchase using $NUTGV2 would be subject to up to a 15% discount off the list price.

In the next stages

1.Raffle Price: By the use of $NUTGV2 token, buyers would be able to unlock surprisingly low prices. The innovative algorithm would allow for massive savings and a chance to earn big on NFTs.

Polygon Price - sales would be made at the list price using $Matic.

BNB Price - sales would be made at the list price using $BNB.

Contract details

Token contract details for 17th May 2022

Contract name	NUTG NFT Marketplace
Contract address	0x937cD45350164EC3802eb3fA979a92a331804094
Token supply	0
Token ticker	NUTG
Transaction count	0
Contract deployer address	0x02b373bb3513C9Ff881db2c1fd789F4355aD9254
Contract's current owner address	0x5eB93f1b0b3E1Fd0f99118e39684f087a84d40Ec

Contract code function details

No	Category	Item	Result
1	Coding conventions	BRC20 Token standards	pass
		compile errors	pass
		Compiler version security	pass
		visibility specifiers	pass
		Gas consumption	pass
		SafeMath features	pass
		Fallback usage	pass
		tx.origin usage	pass
		deprecated items	pass
		Redundant code	pass
		Overriding variables	pass
2	Function call audit	Authorization of function call	pass
		Low level function (call/delegate call) security	pass
		Returned value security	pass
		Selfdestruct function security	pass
3	Business security	Access control of owners	High
		Business logics	pass
		Business implementations	pass
4	Integer overflow/underflow		pass
5	Reentrancy		pass
6	Exceptional reachable state		pass
7	Transaction ordering dependence		pass
8	Block properties dependence		pass
9	Pseudo random number generator (PRNG)		pass
10	DoS (Denial of Service)		pass
11	Token vesting implementation		pass

12	**Fake deposit**		**pass**
13	**Event security**		**pass**

Contract description table

The below table represents the summary of the contracts and methods in the token contract. We scanned the whole contract and listed down all the Interfaces, functions, and implementations with their visibility and mutability.

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
NUTMARKET	**Implementation**	**ERC721URI Storage, Ownable, ReentrancyGuard**		
L		Public ❗	🛑	ERC721
L	takeCommission	Private 🔐	🛑	
L	mintTokenNUTG	Public ❗	🛑	nonReentrant
L	mintToken	Public ❗	💵	nonReentrant
L	buyToken	Public ❗	💵	nonReentrant
L	buyTokenNUTG	Public ❗	🛑	nonReentrant
L	resellToken	Public ❗	🛑	nonReentrant
L	delistItem	Public ❗	🛑	NO❗
L	updateListingPrice	Public ❗	🛑	NO❗
L	getListingPrice	Public ❗		NO❗
L	getMarketItem	Public ❗		NO❗
L	changePrice	Public ❗	🛑	NO❗
L	fetchMarketItems	Public ❗		NO❗

L	fetchMyNFTs	Public ❗		NO❗
L	fetchItemsListed	Public ❗		NO❗
L	renounceOwnership	Public ❗		NO❗
Counters	**Library**			
L	current	Internal 🔒		
L	increment	Internal 🔒	🛑	
L	decrement	Internal 🔒	🛑	
L	reset	Internal 🔒	🛑	
ERC721	**Implementation**	**Context, ERC165, IERC721, IERC721 Metadata**		
L		Public ❗	🛑	NO❗
L	supportsInterface	Public ❗		NO❗
L	balanceOf	Public ❗		NO❗
L	ownerOf	Public ❗		NO❗
L	name	Public ❗		NO❗
L	symbol	Public ❗		NO❗
L	tokenURI	Public ❗		NO❗
L	_baseURI	Internal 🔒		
L	approve	Public ❗	🛑	NO❗
L	getApproved	Public ❗		NO❗

L	setApprovalForAll	Public ❗	🛑	NO❗
L	isApprovedForAll	Public ❗		NO❗
L	transferFrom	Public ❗	🛑	NO❗
L	safeTransferFrom	Public ❗	🛑	NO❗
L	safeTransferFrom	Public ❗	🛑	NO❗
L	_safeTransfer	Internal 🔒	🛑	
L	_exists	Internal 🔒		
L	_isApprovedOrOwner	Internal 🔒		
L	_safeMint	Internal 🔒	🛑	
L	_safeMint	Internal 🔒	🛑	
L	_mint	Internal 🔒	🛑	
L	_burn	Internal 🔒	🛑	
L	_transfer	Internal 🔒	🛑	
L	_approve	Internal 🔒	🛑	
L	_setApprovalForAll	Internal 🔒	🛑	
L	_checkOnERC721Received	Private 🔐	🛑	
L	_beforeTokenTransfer	Internal 🔒	🛑	
IERC721	**Interface**	**IERC165**		
L	balanceOf	External ❗		NO❗
L	ownerOf	External ❗		NO❗

L	safeTransferFrom	External ❗	🛑	NO❗
L	transferFrom	External ❗	🛑	NO❗
L	approve	External ❗	🛑	NO❗
L	getApproved	External ❗		NO❗
L	setApprovalForAll	External ❗	🛑	NO❗
L	isApprovedForAll	External ❗		NO❗
L	safeTransferFrom	External ❗	🛑	NO❗
IERC165	**Interface**			
L	supportsInterface	External ❗		NO❗
IERC721 Receiver	**Interface**			
L	onERC721Received	External ❗	🛑	NO❗
IERC721 Metadata	**Interface**	**IERC721**		
L	name	External ❗		NO❗
L	symbol	External ❗		NO❗
L	tokenURI	External ❗		NO❗
Address	**Library**			
L	isContract	Internal 🔒		
L	sendValue	Internal 🔒	🛑	

L	functionCall	Internal 🔒	🛑	
L	functionCall	Internal 🔒	🛑	
L	functionCallWithValue	Internal 🔒	🛑	
L	functionCallWithValue	Internal 🔒	🛑	
L	functionStaticCall	Internal 🔒		
L	functionStaticCall	Internal 🔒		
L	functionDelegateCall	Internal 🔒	🛑	
L	functionDelegateCall	Internal 🔒	🛑	
L	verifyCallResult	Internal 🔒		
Context	**Implementation**			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		
Strings	**Library**			
L	toString	Internal 🔒		
L	toHexString	Internal 🔒		
L	toHexString	Internal 🔒		
ERC165	**Implementation**	**IERC165**		
L	supportsInterface	Public ❗		NO❗

ERC20	Implementation	Context, IERC20, IERC20Metadata		
L		Public ❗	🛑	NO❗
L	name	Public ❗		NO❗
L	symbol	Public ❗		NO❗
L	decimals	Public ❗		NO❗
L	totalSupply	Public ❗		NO❗
L	balanceOf	Public ❗		NO❗
L	transfer	Public ❗	🛑	NO❗
L	allowance	Public ❗		NO❗
L	approve	Public ❗	🛑	NO❗
L	transferFrom	Public ❗	🛑	NO❗
L	increaseAllowance	Public ❗	🛑	NO❗
L	decreaseAllowance	Public ❗	🛑	NO❗
L	_transfer	Internal 🔒	🛑	
L	_mint	Internal 🔒	🛑	
L	_burn	Internal 🔒	🛑	
L	_approve	Internal 🔒	🛑	
L	_beforeTokenTransfer	Internal 🔒	🛑	
L	_afterTokenTransfer	Internal 🔒	🛑	
IERC20	**Interface**			

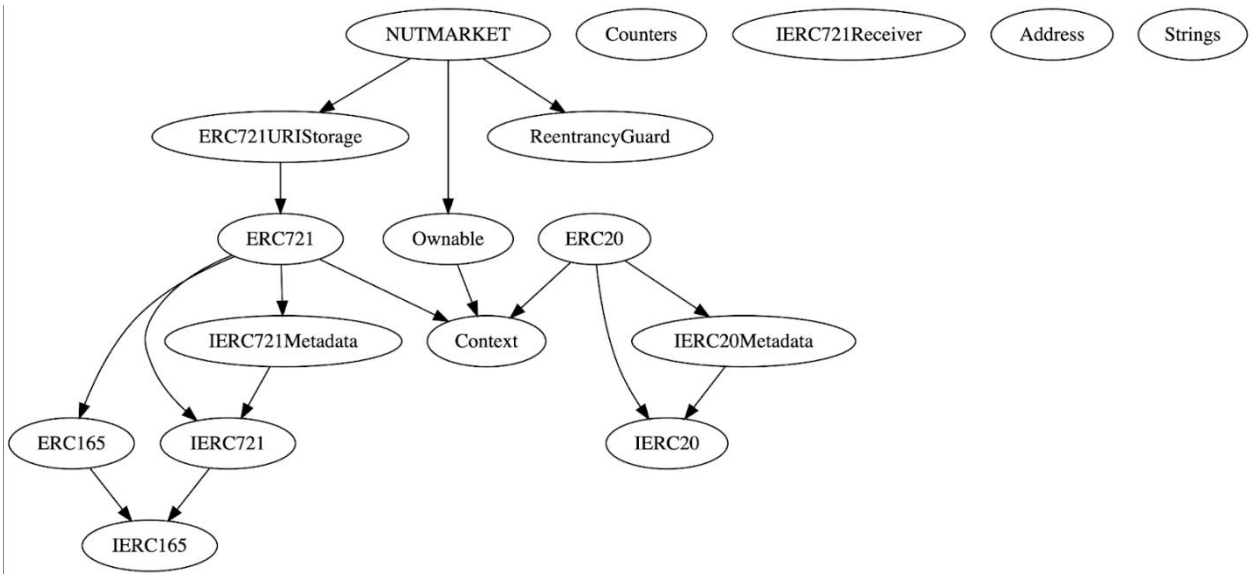
L	totalSupply	External ❗		NO❗
L	balanceOf	External ❗		NO❗
L	transfer	External ❗	🛑	NO❗
L	allowance	External ❗		NO❗
L	approve	External ❗	🛑	NO❗
L	transferFrom	External ❗	🛑	NO❗
IERC20 Metadata	**Interface**	**IERC20**		
L	name	External ❗		NO❗
L	symbol	External ❗		NO❗
L	decimals	External ❗		NO❗
Ownable	**Implementation**	**Context**		
L		Public ❗	🛑	NO❗
L	owner	Public ❗		NO❗
L	renounceOwnership	Public ❗	🛑	onlyOwner
L	transferOwnership	Public ❗	🛑	onlyOwner
L	_transferOwnership	Internal 🔒	🛑	
Reentrancy Guard	**Implementation**			
L		Public ❗	🛑	NO❗

ERC721URI Storage	Implementation	ERC721		
L	tokenURI	Public ❗		NO❗
L	_setTokenURI	Internal 🔒	🛑	
L	_burn	Internal 🔒	🛑	

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Inheritance Hierarchy



Security issue checking status

- **High severity issues**

  No High severity issues found.

- **Medium severity issues**

  No medium severity issues found

- **Low severity issues**

  No low severity issues found

- **Centralization risk**

  No Centralization issues found

Function details

- The platform will take commission maximum up to 50%

```
ftrace | funcSig
function takeCommission(
    address seller,
    address platform,
    uint256 amountPaid,
    uint256 commissionPercentage
) private {
    //validate royalty value
    require(
        commissionPercentage <= 500,
        "Value Overflow: Steted Value Is Above 50 percent"
    );

    // divide by 1000 because commission percentage is expressed as a uint * 10
    uint256 platformFee = (amountPaid * commissionPercentage) / 1000;

    amountPaid -= platformFee;
    payable(platform).transfer(platformFee);
    payable(seller).transfer(amountPaid);
}
```

❖ Users can mint new NFTs

```
function mintToken(
    string memory tokenURI,
    address creator,
    uint256 price,
    uint256 tokens
) public payable nonReentrant {
    require(price > 0, "Price must be at least 1 wei");
    require(
        msg.value == price,
        "Please submit the asking price in order to complete the purchase"
    );

    if (_tokenActive == true) {
        require(tokens > 0, "Token Price Must Be Greater Than Zero");
    }

    _tokenIds.increment();
    uint256 newTokenId = _tokenIds.current();
    _mint(msg.sender, newTokenId);
    _setTokenURI(newTokenId, tokenURI);
    _itemsSold.increment();

    //add token to market items
    idToMarketItem[newTokenId] = MarketItem(
        newTokenId,
        payable(address(creator)),
        payable(msg.sender),
        payable(creator),
        price,
        tokens,
        true
    );

    //pay all parties
    takeCommission(creator, owner(), price, listingPrice);

    // emit asset creation event
    emit MarketItemCreated(
        newTokenId,
        creator,
        msg.sender,
        price,
        tokens,
        true
    );
}
```

❖ Users can buy new NFTs

```
/* allows someone to purchase a listed token */
ftrace | funcSig
function buyToken(uint256 tokenId) public payable nonReentrant {
    uint256 price = idToMarketItem[tokenId].price;
    address seller = idToMarketItem[tokenId].seller;
    address previousOwner = idToMarketItem[tokenId].owner;

    require(
        msg.value == price,
        "Please submit the asking price in order to complete the purchase"
    );

    idToMarketItem[tokenId].sold = true;
    idToMarketItem[tokenId].owner = payable(msg.sender);
    idToMarketItem[tokenId].seller = payable(address(this));

    _itemsSold.increment();

    _transfer(address(this), msg.sender, tokenId);

    //finish transaction and pay respective parties
    takeCommission(seller, owner(), price, listingPrice);

    //emit market sales event
    emit TokenTransferred(previousOwner, msg.sender, tokenId);
}
```

❖ Users can resell new NFTs

```
function resellToken(
    uint256 tokenId,
    uint256 price,
    uint256 tokens
) public nonReentrant {
    require(
        idToMarketItem[tokenId].owner == msg.sender,
        "Only item owner can perform this operation"
    );
    require(price > 0, "Price must be at least 1 wei");

    if (_tokenActive == true) {
        require(tokens > 0, "Token Price Must Be Greater Than Zero");
    }

    idToMarketItem[tokenId].sold = false;
    idToMarketItem[tokenId].price = price;
    idToMarketItem[tokenId].tokens = tokens;
    idToMarketItem[tokenId].seller = payable(msg.sender);
    idToMarketItem[tokenId].owner = payable(address(this));
    _itemsSold.decrement();

    //transfer token to marketplace
    _transfer(msg.sender, address(this), tokenId);

    //emit market item add event
    emit MarketItemListed(tokenId, msg.sender, price, tokens);
}
```

❖ Users can delist listed NFTs from the market place

```
function delistItem(uint256 tokenId) public {
    require(
        idToMarketItem[tokenId].seller == msg.sender,
        "Only item owner can perform this operation"
    );
    idToMarketItem[tokenId].sold = false;
    idToMarketItem[tokenId].seller = payable(address(this));
    idToMarketItem[tokenId].owner = payable(msg.sender);
    _itemsSold.increment();

    //transfer token from market to user
    _transfer(address(this), msg.sender, tokenId);

    //emit item removal event
    emit MarketItemRemoved(tokenId);
}
```

❖ The owner can update market place listing fee

```
function updateListingPrice(uint256 _listingPrice) public {
    require(
        owner() == msg.sender,
        "Only marketplace owner can update listing price."
    );
    listingPrice = _listingPrice;
}
```

❖ Users can update listed NFT prices

```
function changePrice(
    uint256 tokenId,
    uint256 _price,
    uint256 _tokens
) public {
    require(
        idToMarketItem[tokenId].seller == msg.sender,
        "Only item owner can perform this operation"
    );
    if (_tokenActive == true) {
        require(_tokens > 0, "Token Price Must Be Greater Than Zero");
    }

    idToMarketItem[tokenId].price = _price;
    idToMarketItem[tokenId].tokens = _tokens;
}
```

Audit conclusion

RugFreeCoins team has performed in-depth testings, line by line manual code review, and automated audit of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, manipulations, and hacks. According to the smart contract audit.

Smart contract functional Status: **PASSED**

Number of risk issues: **NONE**

Solidity code functional issue level: **PASSED**

Number of owner privileges: **7**

Centralization risk correlated to the active owner: **LOW**

Smart contract active ownership: **YES**