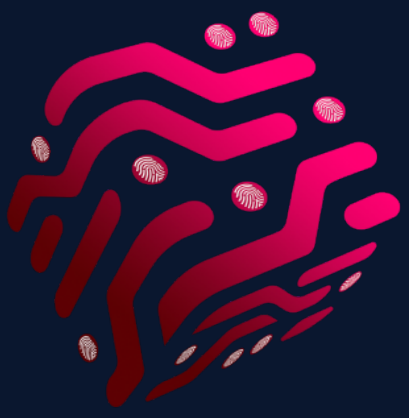
freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT



3N Rewards

$3NR

23/06/2024

TOKEN OVERVIEW

Fees

- Buy fees: 0%
- Sell fees: 0%

Fees privileges

- Can’t change fees

Ownership

- Owned

Minting

- No mint function

Max Tx Amount / Max Wallet Amount

- Can’t change max tx amount and / or max wallet amount

Blacklist

- Blacklist function not detected

Other privileges

- There are several improvements that should be implemented in the contract code to enhance safety and security
- The contract owner can block transfers to a specific address for 90 days

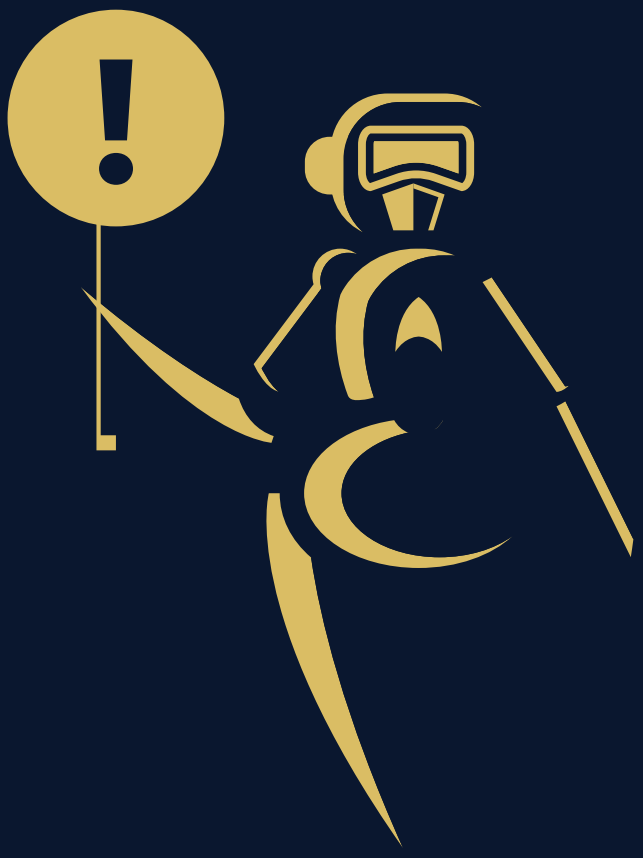
TABLE OF CONTENTS

DISCLAIMER

The information provided on this analysis document is only for general information and should not be used as a reason to invest.

FreshCoins Team will take no payment for manipulating the results of this audit.

The score and the result will stay on this project page information on our website https://freshcoins.io
FreshCoins Team does not guarantees that a project will not sell off team supply, or any other scam strategy (RUG or Honeypot etc)



INTRODUCTION

FreshCoins (Consultant) was contracted by
3N Rewards (Customer) to conduct a Smart Contract Code Review and Security Analysis.

0xF42ccC8C3aebB2aa4e64DDC58f215D9EAb7a250A

Network: Base Network (ETH)

This report presents the findings of the security assessment of Customer's smart contract and its code review conducted on 23/06/2024

WEBSITE DIAGNOSTIC

https://nnn.cloud


0-49


50-89


90-100

Performance

Accessibility

Best Practices

SEO

Progressive Web App

Socials


Twitter

https://twitter.com/nnn_onX


Telegram

https://t.me/join_nnn

AUDIT OVERVIEW

54

Security Score
HIGH RISK
Audit FAIL

93

Static Scan
Automatic scanning for common vulnerabilities

52

ERC Scan
Automatic checks for ERC's conformance

3	High
4	Medium
0	Low
0	Optimizations
0	Informational

No.	Issue description	Checking Status
1	Compiler Errors / Warnings	Passed
2	Reentrancy and Cross-function	Medium
3	Front running	Low
4	Timestamp dependence	Passed
5	Integer Overflow and Underflow	Passed
6	Reverted DoS	Passed
7	DoS with block gas limit	Low
8	Methods execution permissions	Passed
9	Exchange rate impact	Passed
10	Malicious Event	Passed
11	Scoping and Declarations	Passed
12	Uninitialized storage pointers	Passed
13	Design Logic	Passed
14	Safe Zeppelin module	Passed

OWNER PRIVILEGES

- Contract owner can't mint tokens after initial contract deploy
- Contract owner can't exclude addresses from transactions
- Contract owner can update URLs from urls list

  Instead of using delete urls followed by a loop to push new URLs, it can be more gas-efficient to simply update the array in place if possible

```
function updateUrls(string[] calldata newUrls) external onlyOwner {
    require(
      block.timestamp < restrictionEndTime,
      "Cannot update URLs after restriction end time"
    );
    delete urls;
    for (uint256 i = 0; i < newUrls.length; i++) {
      urls.push(newUrls[i]);
    }
}
```

- Contract owner can update PriceFeeds

  Emitting an event after updating the price feeds can help in tracking these changes also deleting the array and then reinitializing it might not be the most gas-efficient approach

  It's generally a good practice to check for zero addresses to prevent setting invalid price feeds

```
function updatePriceFeeds(
    address[] calldata newPriceFeeds
  ) external onlyOwner {
    delete priceFeeds;
    usePriceFeeds = newPriceFeeds.length > 0;
    for (uint256 i = 0; i < newPriceFeeds.length; i++) {
      priceFeeds.push(AggregatorV3Interface(newPriceFeeds[i]));
    }
  }
```

- Function commitPriceUpdate appears to work as intended

  Emitting an event after a price commit can help track changes

```
function commitPriceUpdate(bytes32 priceHash) external onlyOwner {
    require(priceCommits[_msgSender()].timestamp == 0, "Already committed");
    priceCommits[_msgSender()] = PriceCommit(priceHash, block.timestamp);
}
```

The revealPriceUpdate function requires improvements to ensure proper functionality and security

Emitting an event after a successful price update can help track changes

```
function revealPriceUpdate(
    int256 newPrice,
    string calldata secret
  ) external onlyOwner {
    PriceCommit memory commit = priceCommits[_msgSender()];
    require(commit.timestamp > 0, "No committed price update");
    require(
      block.timestamp >= commit.timestamp + REVEAL_PERIOD,
      "Reveal period has not passed"
    );

    bytes32 hash = keccak256(abi.encodePacked(newPrice, secret));
    require(hash == commit.priceHash, "Invalid price or secret");

    // Update the latest price
    latestPrice = newPrice;
    _updatePriceHistory(newPrice);

    // Clear the commit
    delete priceCommits[_msgSender()];
}
```

The calculateMedianPrice function requires multiple improvements

Using Insertion Sort for small arrays is fine, but for larger arrays, consider a more efficient sorting algorithm like QuickSort or MergeSort

At function sort line 2254, the while loop condition (j >= 0) with j as uint256 will always be true since j cannot be negative. This can lead to an infinite loop. Instead, use a different type or logic

```
function calculateMedianPrice(
    int256[] memory prices,
    uint256 count
  ) internal pure returns (int256) {
    int256[] memory sortedPrices = new int256[](count);
    for (uint256 i = 0; i < count; i++) {
      sortedPrices[i] = prices[i];
    }
    sort(sortedPrices);
    if (count % 2 == 1) {
      return sortedPrices[count / 2];
    } else {
      return
        (sortedPrices[(count / 2) - 1] + sortedPrices[count / 2]) / 2;
    }
}
```

Contract owner can set up the initial distribution of tokens to various addresses

The function ensures that the total supply of tokens is minted before initiating the distribution and calculates the distribution progress for each category

Function doesn't seem to make any external calls, it's generally a good practice to use a reentrancy guard on functions modifying the state extensively

Emitting events for critical actions is a good practice for better tracking and debugging

```
function startInitialDistribution(
        DistributionAddresses calldata _addresses
    ) external onlyOwner {
        require(
            totalSupply() == TOTAL_SUPPLY,
            "Total supply must be minted first"
        );

        _startInitialDistribution(
            0,
            NEXUS_SPRITE_SUPPLY,
            _addresses.nexusSpriteAddresses
        );
...
        _startInitialDistribution(
            8,
            NODE_REWARDS_SUPPLY,
            _addresses.nodeRewardsAddresses
        );
}
```

Contract owner can include/exclude wallet in isFoundingTeamAddress array

Adding a wallet to isFoundingTeamAddress can block transfers for 90 days (check _transfer function)

```
function addFoundingTeamAddress(
        address foundingTeamAddress
    ) external onlyOwner {
        isFoundingTeamAddress[foundingTeamAddress] = true;
}

function removeFoundingTeamAddress(
        address foundingTeamAddress
    ) external onlyOwner {
        isFoundingTeamAddress[foundingTeamAddress] = false;
}
```

The _transfer function appears to include the necessary logic for enforcing sale limits on founding team addresses and dynamically adjusting the allowed transfer percentage

Emitting events for critical actions can help with tracking and debugging

Ensure that the isFoundingTeamAddress mapping is necessary to update for the recipient in every transfer

```
function _transfer(
        address sender,
        address recipient,
        uint256 amount
    ) internal override {
        uint256 currentTimestamp = block.timestamp;

        // Dynamic adjustment of allowed percentage based on market conditions
        uint256 allowedPercentage = getDynamicAllowedPercentage();

        // Apply sale limits for founding team addresses post 90 days
        if (
            currentTimestamp >= restrictionEndTime &&
            isFoundingTeamAddress[sender]
        ) {
            uint256 allowedAmount = (allowedPercentage * balanceOf(sender)) /
                100;
            require(
                foundingTeamSoldAmount[sender] + amount <= allowedAmount,
                "Transfer amount exceeds the allowed limit for founding team addresses"
            );

            foundingTeamSoldAmount[sender] += amount;
            lastFoundingTeamSaleTimestamp[sender] = currentTimestamp;

            // Mark recipient as restricted if the sender is a founding team member
            isFoundingTeamAddress[recipient] = true;
        }

        super._transfer(sender, recipient, amount);
    }
```

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

CONCLUSION AND ANALYSIS

Smart Contracts within the scope were manually reviewed and analyzed with static tools.

Audit report overview contains all found security vulnerabilities and other issues in the reviewed code.

Found 3 HIGH issues during the first review.

TOKEN DETAILS

Details

Buy fees:	0%
Sell fees:	0%
Max TX:	N/A
Max Wallet:	N/A

Honeypot Risk

Ownership:	Owned
Blacklist:	Not detected
Modify Max TX:	Not detected
Modify Max Sell:	Not detected
Disable Trading:	Not detected

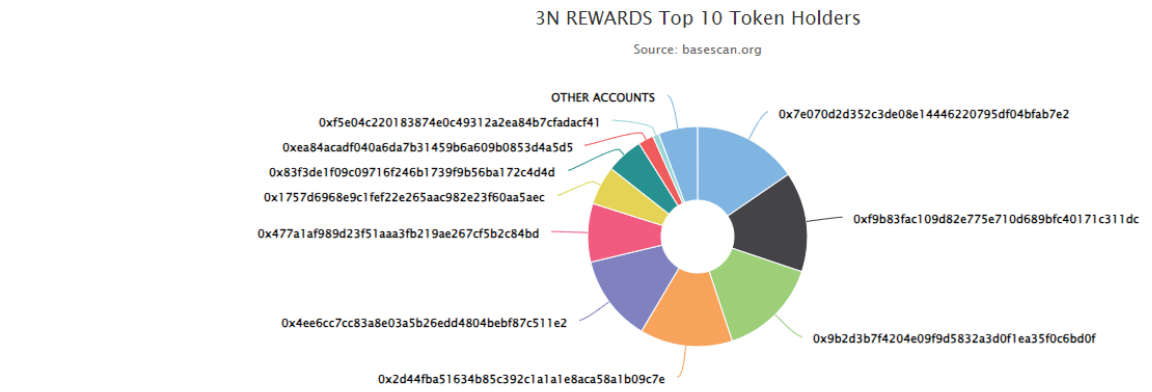
Rug Pull Risk

Liquidity:	N/A
Holders:	100% unlocked tokens

3NR TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 94.24% (1,036,657,845.00 Tokens) of 3N REWARDS

Token Total Supply: 1,100,000,000.00 Token | Total Token Holders: 1,809



(A total of 1,036,657,845.00 tokens held by the top 10 accounts from the total supply of 1,100,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0x7e070d2D...04BFAb7E2	169,899,000	15.4454%
2	0xF9B83faC...171C311dC	162,000,000	14.7273%
3	0x9b2D3B7F...5F0c6bd0f	162,000,000	14.7273%
4	0x2d44fba5...8A1B09c7e	150,000,000	13.6364%
5	0x4eE6cc7C...F87C511e2	140,000,000	12.7273%
6	0x477a1AF9...F5B2c84Bd	94,680,000	8.6073%
7	0x1757d696...f60Aa5aEC	63,078,845	5.7344%
8	0x83f3de1F...A172c4D4D	60,000,000	5.4545%
9	0xEa84ACAd...853d4A5d5	25,000,000	2.2727%
10	0xF5E04c22...CfADAcF41	10,000,000	0.9091%

TECHNICAL DISCLAIMER

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. The audit can't guarantee the explicit security of the audited project / smart contract.