freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

ShibaPixels 404 AI

$SHIBPIXEL

29/03/2024

TOKEN OVERVIEW

Fees

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

Fees privileges

- Can’t change / set fees

Ownership

- Owned

Minting

- Mint function detected

Max Tx Amount / Max Wallet Amount

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

Blacklist

- Blacklist function not detected

Other privileges

- N/A

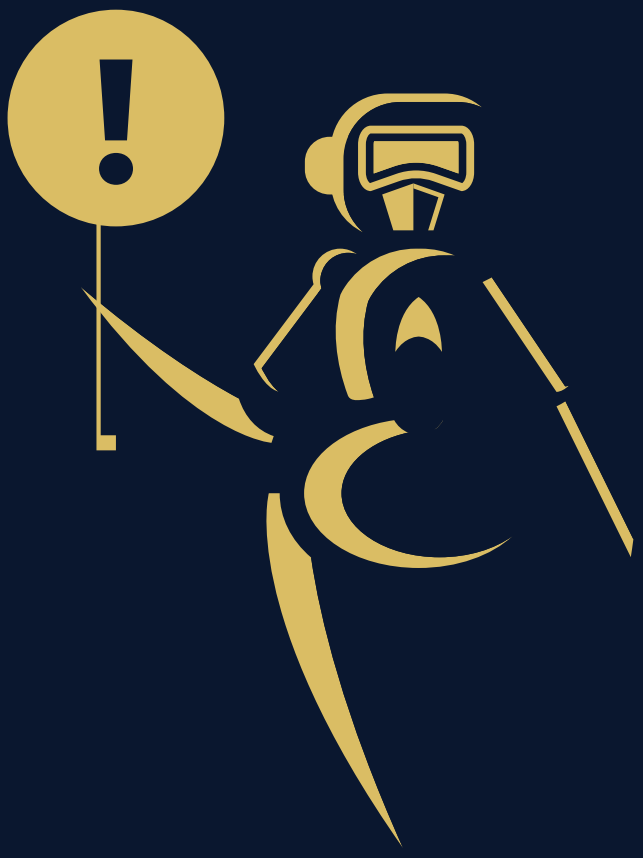
TABLE OF CONTENTS

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 ShibaPixels 404 AI (Customer) to conduct a Smart Contract Code Review and Security Analysis.

0x71FeDE67660f57455a30961Be33F67402D2e0DEa

Network: Binance Smart Chain (BSC)

This report presents the findings of the security assessment of Customer's smart contract and its code review conducted on 29/03/2024

WEBSITE DIAGNOSTIC

https://shibapixels.com/

0-49

50-89

90-100

Performance

Accessibility

Best Practices

SEO

Progressive Web App

Socials

Twitter

https://twitter.com/ShibaPixels404

Telegram

https://t.me/shibapixels

AUDIT OVERVIEW

67

Security Score
HIGH RISK
Audit FAIL

93

Static Scan
Automatic scanning for common vulnerabilities

62

ERC Scan
Automatic checks for ERC's conformance

1 High

1 Medium

0 Low

0 Optimizations

0 Informational

No.	Issue description	Checking Status
1	Compiler Errors / Warnings	Passed
2	Reentrancy and Cross-function	Passed
3	Front running	Medium
4	Timestamp dependence	Passed
5	Integer Overflow and Underflow	Passed
6	Reverted DoS	Low
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 change the token name and symbol, which may confuse users

  It is recommended to delete setNameSymbol function

```
function setNameSymbol(
        string memory _name,
        string memory _symbol
    ) public onlyOwner {
        _setNameSymbol(_name, _symbol);
}
```

- Contract owner holds significant responsibility for managing token-wide operations

  This includes tasks such as authorizing users to access certain features and configuring the user interface for token and data management. Below, we outline a key function that could be impacted by this privileged role.

```
function setDataURI(string memory _dataURI) public onlyOwner {
        dataURI = _dataURI;
}

function setIpfsDefault(string memory _ipfsDefault) public onlyOwner {
        ipfsDefault = _ipfsDefault;
}

function setTokenURI(string memory _tokenURI) public onlyOwner {
        baseTokenURI = _tokenURI;
}

function startMint() public onlyOwner {
        require(started == false, "Mint has already been initiated!");
        started = true;
}
```

Secure Management of ERC20-ERC721 Conversion Logic to Prevent Potential Exploitation

```
...
// Check if start
      if (started == true) {
        // Skip burn for certain addresses to save gas
        if (!exemptFee[from]) {
          uint256 tokens_to_burn = (balanceBeforeSender / unit) -
            (balanceOf[from] / unit);
          for (uint256 i = 0; i < tokens_to_burn; i++) {
            _burn(from);
          }
        }

        // Skip minting for certain addresses to save gas
        if (!exemptFee[to]) {
          uint256 tokens_to_mint = (balanceOf[to] / unit) -
            (balanceBeforeReceiver / unit);
          for (uint256 i = 0; i < tokens_to_mint; i++) {
            _mint(to);
          }
        }
}
...
```

The function includes additional logic to mint or burn ERC721 tokens based on the fractional ERC20 balance changes. This logic is contingent on the started state variable and the exemptFee mapping

While the balance updates themselves are protected against overflow/underflow, the minting and burning logic could potentially be manipulated if the exemptFee mapping is not managed correctly. An attacker with control over an address marked as exempt could transfer large amounts of ERC20 tokens without triggering the corresponding mint or burn of ERC721 tokens, potentially leading to an imbalance between the ERC20 and ERC721 representations

The calculation of tokens_to_burn and tokens_to_mint uses division, which in Solidity rounds down. This could lead to scenarios where small amounts of fractional tokens do not result in minting or burning of ERC721 tokens, creating a discrepancy over time

The loops for minting and burning are potentially gas-intensive and could fail if they iterate over a large number of tokens due to block gas limits

ShibaPixels Notes:

Inheritance Audit: Review the ERC404 contract for any security issues or non-standard implementations.

Randomness Source: The seed variable and its usage in tokenURI could be exploited; consider a more secure randomness source.

Input Validation: Add checks for constructor and setter inputs to prevent invalid configurations.

Owner Privileges: Evaluate the necessity of owner-only functions and consider decentralization aspects or timelocks for sensitive changes.

Metadata Integrity: Assess the risk of allowing the owner to change dataURI, baseTokenURI, and ipfsDefault after minting.

ERC Compliance: Verify that ERC404 adheres to ERC-20, ERC-721, or ERC-1155 standards, as applicable.

Upgrade Path: Consider adding an upgrade mechanism or immutable design patterns for critical logic.

Gas Efficiency: Optimize tokenURI for lower gas costs.

Error Reporting: Implement events and require messages for better transparency and user experience.

Tokenomics Review: Ensure the initial balance and supply match the project's economic model.

IPFS Fallback: Reconsider the use of a generic ipfsDefault for unique tokens.

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 1 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

SHIBPIXEL TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

There are no matching entries (ERC404)

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.