freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

V_1 FUTURE CREATIVES

$FUCR

20/12/2023

TOKEN OVERVIEW

Fees

- Buy fees: 5%
- Sell fees: 5%
- Transfer fees 5%

Fees privileges

- Can change buy fees up to 5%, sell fees up to 5% and transfer fees up to 5%

Ownership

- Owned

Minting

- No mint function

Max Tx Amount / Max Wallet Amount

- Can change max tx amount and max wallet amount (with threshold)

Blacklist

- Blacklist function not detected

Other privileges

- Can exclude / include from fees
- Contract owner has to call enableTrading function to enable trade

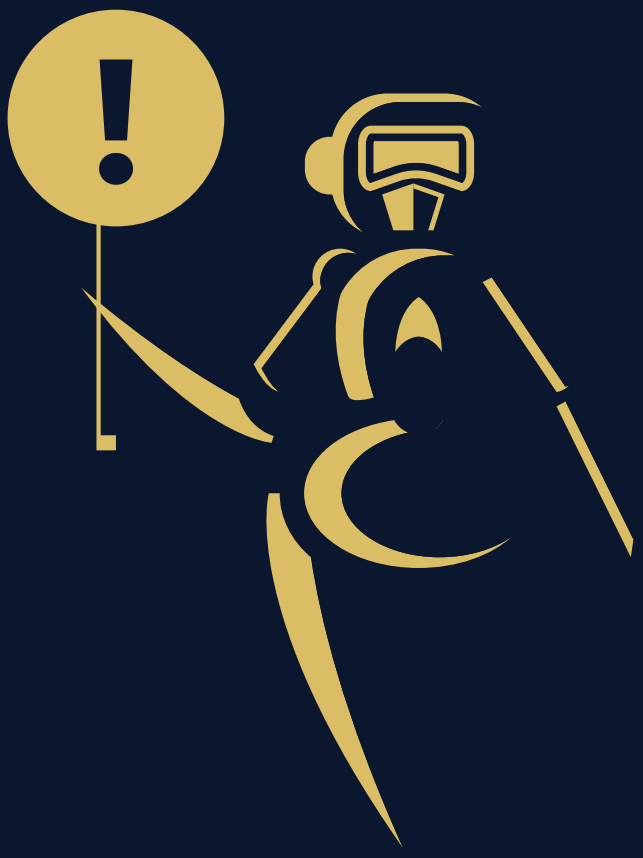
TABLE OF CONTENTS

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
V_1 FUTURE CREATIVES (Customer) to conduct a Smart Contract Code Review and Security Analysis.

0x090D693A079979aD2b48C4897cce9828D3374262

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 20/12/2023

WEBSITE DIAGNOSTIC

https://futurecreatives.ai/

0-49

50-89

90-100

Performance

Accessibility

Best Practices

SEO

Progressive Web App

Socials

Twitter

https://twitter.com/fucr_1

Telegram

https://t.me/Futurecreatives

AUDIT OVERVIEW

84

Security Score

98

Static Scan
Automatic scanning for common vulnerabilities

82

ERC Scan
Automatic checks for ERC's conformance

1 High

0 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	Passed
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 an address from transactions
- Contract owner can exclude/include wallet from tax

```
function excludeFromFees(address account, bool excluded) external onlyOwner{
    require(_isExcludedFromFees[account] != excluded,"Account is already the value of 'excluded'");
    _isExcludedFromFees[account] = excluded;

    emit ExcludeFromFees(account, excluded);
}
```

- Contract owner can exclude/include wallet from tx limitations

```
function excludeFromMaxTransactionLimit(address account, bool exclude) external onlyOwner {
    require( _isExcludedFromMaxTxLimit[account] != exclude, "Account is already set to that state");
    require(account != address(this), "Can't set this address.");

    _isExcludedFromMaxTxLimit[account] = exclude;

    emit ExcludedFromMaxTransactionLimit(account, exclude);
}
```

- Contract owner has to call `enableTrading` function to enable trade

Please note that any wallet excluded from fees retains the ability to engage in trading, even in situations where trading has been disabled

```
function enableTrading() external onlyOwner{
    require(!tradingEnabled, "Trading already enabled.");
    tradingEnabled = true;
    swapEnabled = true;

    emit TradingEnabled();
}

_transferFrom function line 664
.
.
.
require(tradingEnabled || _isExcludedFromFees[from] || _isExcludedFromFees[to], "Trading not yet enabled!");
.
.
.
```

Contract owner can change swap settings

```
function setSwapEnabled(bool _enabled) external onlyOwner{
    require(swapEnabled != _enabled, "swapEnabled already at this state.");
    swapEnabled = _enabled;
}

function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner{
    require(newAmount > totalSupply() / 1_000_000, "SwapTokensAtAmount must be greater than 0.0001%
of total supply");
    swapTokensAtAmount = newAmount;

    emit SwapTokensAtAmountUpdated(swapTokensAtAmount);
}
```

Contract owner has ability to retrieve any token held by the contract

Native tokens excluded

```
function claimStuckTokens(address token) external onlyOwner {
    require(token != address(this), "Owner cannot claim contract's balance of its own tokens");
    if (token == address(0x0)) {
        payable(msg.sender).sendValue(address(this).balance);
        return;
    }
    IERC20 ERC20token = IERC20(token);
    uint256 balance = ERC20token.balanceOf(address(this));
    ERC20token.transfer(msg.sender, balance);
}
```

Contract owner can enable/disable tx limitations

```
function setEnableMaxTransactionLimit(bool enable) external onlyOwner {
    require(enable != maxTransactionLimitEnabled, "Max transaction limit is already set to that state");
    maxTransactionLimitEnabled = enable;

    emit MaxTransactionLimitStateChanged(maxTransactionLimitEnabled);
}
```

Contract owner can change tx limitations (with threshold)

```
function setMaxTransactionAmounts(uint256 _maxTransactionAmountBuy, uint256 _maxTransaction-
AmountSell) external onlyOwner {
    require(
        _maxTransactionAmountBuy  >= (totalSupply() / (10 ** decimals())) / 1_000 &&
        _maxTransactionAmountSell >= (totalSupply() / (10 ** decimals())) / 1_000,
        "Max Transaction limis cannot be lower than 0.1% of total supply"
    );
    maxTransactionAmountBuy  = _maxTransactionAmountBuy  * (10 ** decimals());
    maxTransactionAmountSell = _maxTransactionAmountSell * (10 ** decimals());

    emit MaxTransactionLimitAmountChanged(maxTransactionAmountBuy, maxTransactionAmountSell);
}
```

Contract owner can change marketingWallet and treasuryWallet addresses

Current values:

marketingWallet: 0x1a097410D772E5cBd0FD7822831456C83dE100E9

treasuryWallet: 0xE1B0e79d55850CC4905cCCc0a26E7b86592E1E26

```
function changeMarketingWallet(address _marketingWallet) external onlyOwner{
    require(_marketingWallet != marketingWallet,"Marketing wallet is already that address");
    require(_marketingWallet != address(0),"Marketing wallet cannot be the zero address");
    marketingWallet = _marketingWallet;

    emit MarketingWalletChanged(marketingWallet);
}

function changeTreasuryWallet(address _treasuryWallet) external onlyOwner{
    require(_treasuryWallet != treasuryWallet,"Treasury wallet is already that address");
    require(_treasuryWallet != address(0),"Treasury wallet cannot be the zero address");
    treasuryWallet = _treasuryWallet;

    emit TreasuryWalletChanged(treasuryWallet);
}
```

Contract owner can change buy fees up to 5%, sell fees up to 5% and transfer fees up to 5%

```
function updateBuyFees(uint256 _liquidityFeeOnBuy, uint256 _marketingFeeOnBuy, uint256 _treasury-
FeeOnBuy) external onlyOwner {
    liquidityFeeOnBuy = _liquidityFeeOnBuy;
    marketingFeeOnBuy = _marketingFeeOnBuy;
    treasuryFeeOnBuy = _treasuryFeeOnBuy;

    _totalFeesOnBuy   = liquidityFeeOnBuy + marketingFeeOnBuy + treasuryFeeOnBuy;

    require(_totalFeesOnBuy <= 5, "Total Fees cannot exceed the maximum");

    emit UpdateBuyFees(liquidityFeeOnBuy, marketingFeeOnBuy);
}

function updateSellFees(uint256 _liquidityFeeOnSell, uint256 _marketingFeeOnSell, uint256 _treasuryFeeOn-
Sell) external onlyOwner {
    liquidityFeeOnSell = _liquidityFeeOnSell;
    marketingFeeOnSell = _marketingFeeOnSell;
    treasuryFeeOnSell = _treasuryFeeOnSell;

    _totalFeesOnSell   = liquidityFeeOnSell + marketingFeeOnSell + treasuryFeeOnSell;

    require(_totalFeesOnSell <= 5, "Total Fees cannot exceed the maximum");

    emit UpdateSellFees(liquidityFeeOnSell, marketingFeeOnSell);
}

function updateWalletToWalletTransferFee(uint256 _walletToWalletTransferFee) external onlyOwner {
    require(_walletToWalletTransferFee <= 5, "Wallet to Wallet Transfer Fee cannot exceed the maximum");
    walletToWalletTransferFee = _walletToWalletTransferFee;

    emit UpdateWalletToWalletTransferFee(walletToWalletTransferFee);
}
```

- Contract owner can transfer ownership

```
function transferOwnership(address newOwner) public virtual onlyOwner {
    require(newOwner != address(0), "Ownable: new owner is the zero address");
    emit OwnershipTransferred(_owner, newOwner);
    _owner = newOwner;
}
```

- Contract owner can renounce ownership

```
function renounceOwnership() public virtual onlyOwner {
    emit OwnershipTransferred(_owner, address(0));
    _owner = address(0);
}
```

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:	5%
Sell fees:	5%
Transfer fees:	5%
Max TX:	50,000,000
Max Sell:	50,000,000

Honeypot Risk

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

Rug Pull Risk

Liquidity:	N/A
Holders:	100% unlocked tokens

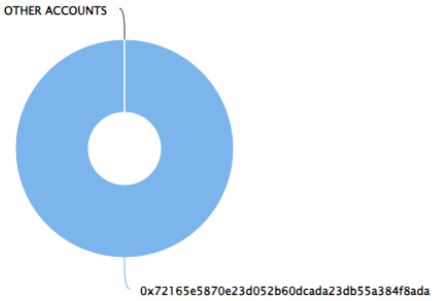
FUCR TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00%
(10,000,000,000.00 Tokens) of V_1 FUTURE CREATIVES

Token Total Supply: 10,000,000,000.00 Token | Total Token Holders: 1

V_1 FUTURE CREATIVES Top 10 Token Holders

Source: BscScan.com



(A total of 10,000,000,000.00 tokens held by the top 10 accounts from the total supply of 10,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0x72165E...384F8aDa	10,000,000,000	100.0000%

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.