freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Yieldify

$YIFY

05/03/2023

TOKEN OVERVIEW

Fees

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

Fees privileges

- Can change fees up to 10%

Ownership

- Owned

Minting

- Mint function detected (isn't a security issue)

Max Tx Amount / Max Wallet Amount

- Can’t change max tx amount or wallet amount

Blacklist

- No blacklist function

Other privileges

- Can exclude/include from fees
- Can exclude/include from staking

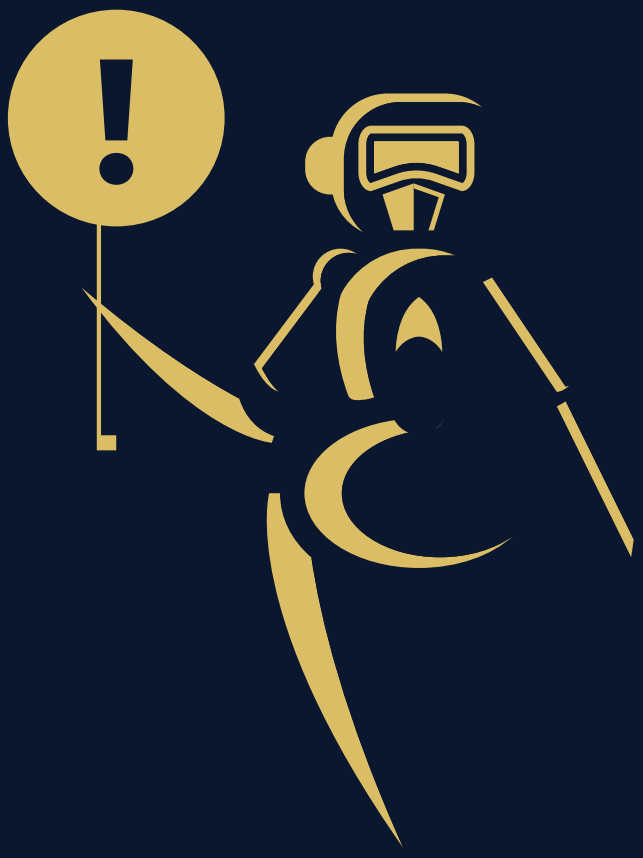
TABLE OF CONTENTS

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 Yieldify (Customer) to conduct a Smart Contract Code Review and Security Analysis.

0x090FE3A2328e0aD40e5CA399eEa883d307c4174C

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 05/03/2023

WEBSITE DIAGNOSTIC

https://yieldify.app/


0-49


50-89


90-100

Performance

Accessibility

Best Practices

SEO

Progressive Web App

Socials


Twitter

https://twitter.com/YieldifyBSC


Telegram

https://t.me/YieldifyPortal

AUDIT OVERVIEW

84

Security Score

98

Static Scan
Automatic scanning for common vulnerabilities

80

ERC Scan
Automatic checks for ERC's conformance

- 0 High
- 5 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	Passed
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 exclude an address from transactions

- Mint function detected

```
function _addToken(address addr, uint256 amount) private {
    totalSupply+=amount;
    uint256 newAmount = balanceOf[addr] + amount;
    if (excludedFromStaking[addr]) {
      balanceOf[addr] = newAmount;
      return;
    }
    totalShares += amount;
    uint256 payment = _newDividentsOf(addr);
    alreadyPaidShares[addr] = profitPerShare * newAmount;
    toBePaid[addr] += payment;
    balanceOf[addr] = newAmount;
}

function _removeToken(address addr, uint256 amount) private {
    totalSupply-=amount;
    uint256 newAmount = balanceOf[addr] - amount;
    if (excludedFromStaking[addr]) {
      balanceOf[addr] = newAmount;
      return;
    }

    uint256 payment = _newDividentsOf(addr);
    balanceOf[addr] = newAmount;
    alreadyPaidShares[addr] = profitPerShare * getShares(addr);
    toBePaid[addr] += payment;
    totalShares -= amount;
}
```

The transfer function in this smart contract works in the following way: it removes a specified amount of tokens from the sender's account (using the _removeToken function) and burns them, then it adds new tokens to the recipient's account (using the _addToken function), without fees. Additionally, it mints a specific amount of fee tokens to the contract address (using the _addToken function).

```
...

uint256 totalTaxedToken = (amount * tax) / 1000;
    uint256 taxedAmount = amount - totalTaxedToken;


    _removeToken(sender, amount);
    _addToken(address(this), totalTaxedToken);
    emit Transfer(sender, address(this),totalTaxedToken);
    _addToken(recipient, taxedAmount);
    emit Transfer(sender, recipient, taxedAmount);
...
```

Only generates as much yield that is defined by the APY. It doesn't allow to change the amount or the recipient wallet, mint is part of the tokenomics.

Minting is done only based on defined rules and they are:

X% yearly yield to the contract wallet based on time passed

2% bonus tokens for compounders

Contract owner can exclude/include wallet from tax

```
function SetExcludedStatus(address account, bool flag) external onlyOwner {
    require(
      account != address(this) && account!=address(treasury)&& account != address(0xdead),
      "can't Include"
    );
    excluded[account] = flag;
    emit OnExclude(account, flag);
}
```

Contract owner can exclude/include wallet from staking

```
function SetStakingExcluded(address addr, bool exclude) public onlyOwner {
    uint256 shares;
    if (exclude) {
      require(!excludedFromStaking[addr]);
      uint256 newDividents = _newDividentsOf(addr);
      shares = getShares(addr);
      excludedFromStaking[addr] = true;
      totalShares -= shares;
      alreadyPaidShares[addr] = shares * profitPerShare;
      toBePaid[addr] += newDividents;
    } else _includeToStaking(addr);
    emit OnExcludeFromStaking(addr, exclude);
}
```

Contract owner can change swap settings

```
function SwitchSwapAndLiquify(bool disabled) external onlyOwner {
    swapAndLiquifyDisabled = disabled;
    emit OnSwitchSwapAndLiquify(disabled);
}
```

Contract owner can change lock time for treasury wallet

treasury wallet : 0xe121749d61b62d15eaee1985895a5d5382c0d661

uint TreasuryLockTime= 26 weeks; (Mar-03-2023 03:03:52 PM +UTC Yieldify contract deployed)

```
function lockTreasury(uint totalTime) external onlyOwner{
    require(totalTime>TreasuryLockTime);
    TreasuryLockTime=totalTime;
}
```

Staking referral system implemented in smart contract (dApp utility)

```
function addReferer(address account, address referrer_) external{
        address msgSender=msg.sender;
        require(msgSender==account||authorized[msgSender]);
        require(account!=referrer_,"can't refer yourself");
        require(referrer[account]==address(0),"already referred someone");
        referrer[account]=referrer_;
        emit OnAddReferer(account,referrer_);
    }

    function compound() external Lock{
        address currentAccount=msg.sender;
        uint dividents=_handleDividents(currentAccount);
        uint tokenBefore=balanceOf[currentAccount];
        treasury.swapForToken(currentAccount,dividents);
        uint newToken=balanceOf[currentAccount]-tokenBefore;
        uint bonusAmount=newToken*2/100;
        _addToken(address(this),bonusAmount);
        emit Transfer(address(0),currentAccount,bonusAmount);
    }

    function buyWithReferral(uint amountIn, uint minAmountOut, address referrer_) external{
        address currentAccount=msg.sender;
        Stablecoin.transferFrom(currentAccount,address(treasury),amountIn);
        treasury.swapForTokenDirect(currentAccount,amountIn,minAmountOut);
        require(currentAccount!=referrer_,"can't refer yourself");
        require(referrer[currentAccount]==address(0),"already referred someone");
        referrer[currentAccount]=referrer_;
        emit OnAddReferer(currentAccount,referrer_);

    }
```

Public function to calculate staking yield

To sustain the yield value as desired, _addToken (mint function) will mint new tokens to the contract address

```
function generateYield() public{
        uint timestamp=block.timestamp;
        uint timePassed=timestamp-LastYieldTimestamp;
        if(timePassed==0) return;
        LastYieldTimestamp=timestamp;
        uint yield;
        uint timeSinceLaunch=timestamp-LaunchTimestamp;
        if(timeSinceLaunch<365 days)
            yield=333;
        ...
        else yield=55;

        uint yieldToken=totalShares*yield*timePassed/(365 days*100);
        _addToken(address(this),yieldToken);
        accumulatedYieldToken+=yieldToken;
    }
```

Contract owner can set/change launch time

```
function Launch() external {
    SetupLaunchTimestamp(block.timestamp);
}

function SetupLaunchTimestamp(uint256 timestamp) public onlyOwner {
    require(block.timestamp < LaunchTimestamp);

    LaunchTimestamp = timestamp;
    LastYieldTimestamp=timestamp;
    emit OnSetLaunchTimestamp(timestamp);
}
```

Contract owner can change FeeDistributor address

Current value:

FeeDistributor : 0xdb76d2ce5e6babe4460ad8276c1ede68ed918000

```
function ChangeFeeDistributor(address newDistributor)
    external
    onlyOwner
 {
    FeeDistributor = newDistributor;
    emit OnChangeFeeDistributor(newDistributor);
}
```

Contract owner can change tax up to 10%

```
function setTaxes(uint newTaxes) external onlyOwner{
    require (newTaxes <= 100,"Taxes can be max 10%");
    taxes=newTaxes;
    emit OnSetTaxes(taxes);
}
```

Contract owner can claim tokens from treasury wallet (with limitations)

```
function claimTreasury(uint amount) external onlyOwner{
    require(block.timestamp>LaunchTimestamp+TreasuryLockTime,"Locked");
    treasury.transferStablecoin(msg.sender,amount);
}
```

Contract owner can withdraw stuck tokens from smart contract

YFIY tokens and _pancakePairAddress excluded

```
function WithdrawStrandedToken(address strandedToken) external onlyOwner {
    require(
      (strandedToken != _pancakePairAddress) &&
        strandedToken != address(this)
    );
    IBEP20 token = IBEP20(strandedToken);
    token.transfer(FeeDistributor, token.balanceOf(address(this)));
}
```

- Contract owner transfer ownership

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

- Contract owner can transfer ownership

```
function renounceOwnership() public 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 no HIGH issues during the first review.

TOKEN DETAILS

Details

Buy fees:	5%
Sell fees:	5%
Max TX:	N/A
Max Sell:	N/A

Honeypot Risk

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

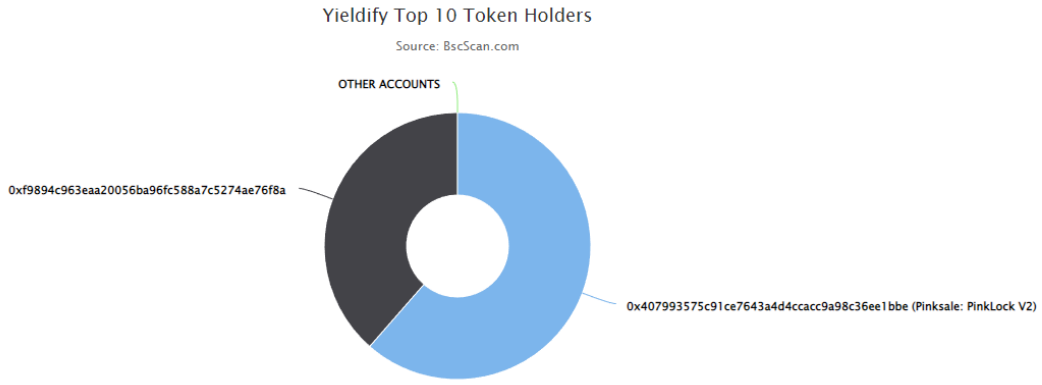
Others

Liquidity:	N/A
Holders:	Clean

YIFY TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (1,000,000.00 Tokens) of Yieldify

Token Total Supply: 1,000,000.00 Token | Total Token Holders: 2



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

Rank	Address	Quantity (Token)	Percentage
1	Pinksale: PinkLock V2	614,440	61.4440%
2	0xf9894c963eaa20056ba96fc588a7c5274ae76f8a	385,560	38.5560%

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.