freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Basset Hound

$BASH

17/02/2023

TOKEN OVERVIEW

Fees

• **Buy fees:**   **9%**

• **Sell fees:**   **9%**

Fees privileges

• **Can change fees up to 10%**

Ownership

• **Owned**

Minting

• **No mint function**

Max Tx Amount / Max Wallet Amount

• **Can't change / set max tx amount or max wallet amount**

Blacklist

• **Blacklist function not detected**

Other privileges

• **Can exclude / include from fees**

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

0x5A953e4fb797F5250D79E560286ECbC6fB472092

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 17/02/2023

AUDIT OVERVIEW

94

Security Score

96 — Static Scan
Automatic scanning for common vulnerabilities

92 — ERC Scan
Automatic checks for ERC's conformance

0 High

2 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 mint tokens after initial contract deploy**

● **Contract owner can't exclude an address from transactions**

● **Contract owner can exclude/include wallet(s) from tax**

```solidity
function excludeFromFees(address account, bool excluded) external onlyOwner {
    require(_isExcludedFromFees[account] != excluded, "Account is already set to that state");
    _isExcludedFromFees[account] = excluded;

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

● **Contract owner can exclude/include wallet from dividends**

```solidity
function excludeFromDividends(address account) external onlyOwner {
    require(!excludedFromDividends[account]);
    excludedFromDividends[account] = true;

    _setBalance(account, 0);
    tokenHoldersMap.remove(account);

    emit ExcludeFromDividends(account);
}
```

● **Contract owner can change swap settings**

```solidity
function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner{
    require(newAmount > totalSupply() / 100_000, "SwapTokensAtAmount must be greater than 0.001% of total supply");
    swapTokensAtAmount = newAmount;
}
```

● **Contract owner can change treasuryWallet address**

Current value:

treasuryWallet: 0xc3c5a04c5d835212a77b3d2cee2c0b0cf3a735a1

```solidity
function changeTreasuryWallet(address _treasuryWallet) external onlyOwner {
    require(_treasuryWallet != treasuryWallet, "Marketing wallet is already that address");
    require(!isContract(_treasuryWallet), "Marketing wallet cannot be a contract");
    require(_treasuryWallet != DEAD, "Marketing wallet cannot be the zero address");
    treasuryWallet = _treasuryWallet;
    emit TreasuryWalletChanged(treasuryWallet);
}
```

● Contract owner can change fees up to 10%

```
function updateBuyFees(uint256 _liquidityFeeOnBuy, uint256 _treasuryFeeOnBuy, uint256 _rewardsFeeOn-
Buy, uint256 _buybackFeeOnBuy, uint256 _marketingFeeOnBuy) external onlyOwner {
    liquidityFeeOnBuy   = _liquidityFeeOnBuy;
    treasuryFeeOnBuy    = _treasuryFeeOnBuy;
    rewardsFeeOnBuy     = _rewardsFeeOnBuy;
    buybackFeeOnBuy     = _buybackFeeOnBuy;
    marketingFeeOnBuy   = _marketingFeeOnBuy;

    totalBuyFee = _liquidityFeeOnBuy + _treasuryFeeOnBuy + _rewardsFeeOnBuy + _buybackFeeOnBuy +
_marketingFeeOnBuy;

    require(totalBuyFee <= 10, "Buy fee cannot be more than 10%");

    emit BuyFeesUpdated(totalBuyFee);
}

function updateSellFees(uint256 _liquidityFeeOnSell, uint256 _treasuryFeeOnSell, uint256 _rewardsFeeOn-
Sell, uint256 _buybackFeeOnSell, uint256 _marketingFeeOnSell) external onlyOwner {
    liquidityFeeOnSell   = _liquidityFeeOnSell;
    treasuryFeeOnSell    = _treasuryFeeOnSell;
    rewardsFeeOnSell     = _rewardsFeeOnSell;
    buybackFeeOnSell     = _buybackFeeOnSell;
    marketingFeeOnSell   = _marketingFeeOnSell;

    totalSellFee = _liquidityFeeOnSell + _treasuryFeeOnSell + _rewardsFeeOnSell + _buybackFeeOnSell +
_marketingFeeOnSell;

    require(totalSellFee <= 10, "Sell fee cannot be more than 10%");

    emit SellFeesUpdated(totalSellFee);
}
```

● Contract owner can transfer ownership

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

function _transferOwnership(address newOwner) internal virtual {
    address oldOwner = _owner;
    _owner = newOwner;
    emit OwnershipTransferred(oldOwner, newOwner);
}
```

● Contract owner can renounce ownership

```
function renounceOwnership() public virtual onlyOwner {
    _transferOwnership(address(0));
}
```

● Contract owner can withdraw stuck tokens from smart contract

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

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:	9%
Sell fees:	9%
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

Rug Pull Risk

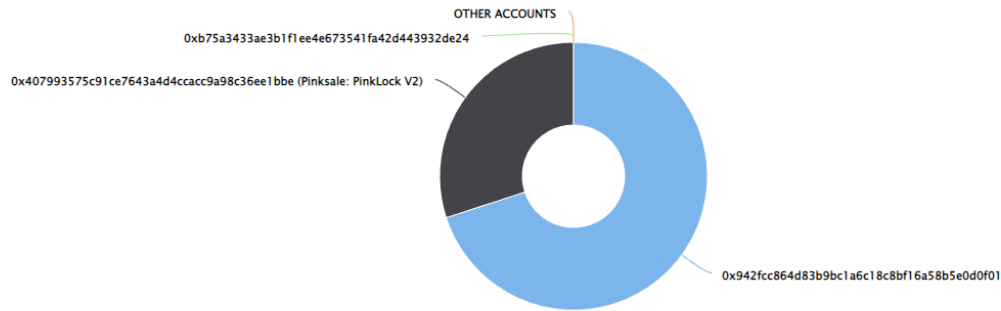|	
Liquidity:	N/A
Holders:	Clean

BASH TOKEN ANALYTICS
& TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (100,000,000,000,000,000.00 Tokens) of Basset Hound   Token Total Supply: 100,000,000,000,000,000.00 Token | Total Token Holders: 3

Basset Hound Top 10 Token Holders

Source: BscScan.com

OTHER ACCOUNTS

0xb75a3433ae3b1f1ee4e673541fa42d443932de24

0x407993575c91ce7643a4d4ccacc9a98c36ee1bbe (Pinksale: PinkLock V2)

0x942fcc864d83b9bc1a6c18c8bf16a58b5e0d0f01

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

Rank	Address	Quantity (Token)	Percentage
1	0x942fcc864d83b9bc1a6c18c8bf16a58b5e0d0f01	69,995,610,000,000,000	69.9956%
2	Pinksale: PinkLock V2	30,000,000,000,000,000	30.0000%
3	0xb75a3433ae3b1f1ee4e673541fa42d443932de24	4,390,000,000,000	0.0044%

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.