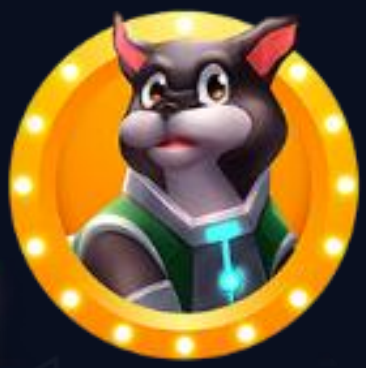
expelee

A Secure Place For Web3

SMART CONTRACT AUDIT OF

TECHDOGE Presale



Contract Address

0x45b6f6533919B573043F6bC8674BB8F9128bA31d

Audit Summary

Expelee team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analysed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

Audit Result: **PASSED**

Ownership: **NOT RENOUNCED**

KYC Verification: **DONE**

Audit Date: 24/06/2022

Audit Team: **EXPELEE**

Be aware that smart contracts deployed on the blockchain aren't resistant to internal exploit, external vulnerability, or hack. For a detailed understanding of risk severity, source code vulnerability, functional hack, and audit disclaimer, kindly refer to the audit.

DISCLAMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always Do your own research and protect yourselves from being scammed. The Expelee team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools.

Under no circumstances did Expelee receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams.

This document should not be presented as a reason to buy or not buy any particular token. The Expelee team disclaims any liability for the resulting losses.

Contract Review

Contract Name	**TECHDOGE**
Compiler Version	**v0.8.9+commit.e5eed63a**
Optimization	**Yes with 200 runs**
License	**MIT license**
Explorer	**https://bscscan.com/address/0x45b6f6533919B573043F6bC8674BB8F9128bA31d#code**
Symbol	**TECHDOGE**
Decimals	**18**
Total Supply	**1,000,000,000,000,000**
Domain	**https://www.techdoge.app/**

Project Review

Token Name: TECHDOGE

Web Site: https://www.techdoge.app/

Twitter: https://twitter.com/TECHDOGEAPP

Telegram: https://t.me/TECHDOGE_Global

Contract Address:
0x45b6f6533919B573043F6bC8674BB8F9128bA31d

Platform: Binance Smart Chain

Token Type: BEP 20

Language: SOLIDITY

Audit Methodology

The scope of this report is to audit the smart contract source code. We have scanned the contract and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Category

Smart Contract Vulnerabilities	- Unhandled Exceptions - Transaction Order Dependency - Integer Overflow - Unrestricted Action - Incorrect Inheritance Order - Typographical Errors - Requirement Violation
Source Code Review	- Gas Limit and Loops - Deployment Consistency - Repository Consistency - Data Consistency - Token Supply Manipulation
Functional Assessment	- Operations Trail & Event Generation - Assets Manipulation - Liquidity Access

Vulnerability Checklist

№	Description.	Result
1	Compiler warnings.	Passed
2	Race conditions and Re-entrancy. Cross-function raceconditions.	Passed
3	Possible delays in data delivery.	Passed
4	Oracle calls.	Passed
5	Front running.	Passed
6	Timestamp dependence.	Passed
7	Integer Overflow and Underflow.	Passed
8	DoS with Revert.	Passed
9	DoS with block gas limit.	Passed
10	Methods execution permissions.	Passed
11	Economy model.	Passed
12	The impact of the exchange rate on the logic.	Passed
13	Private user data leaks.	Passed
14	Malicious Event log.	Passed
15	Scoping and Declarations.	Passed
16	Uninitialized storage pointers.	Passed
17	Arithmetic accuracy.	Passed
18	Design Logic.	Passed
19	Cross-function race conditions.	Passed
20	Safe Zeppelin module.	Passed
21	Fallback function security.	Passed

Manual Audit

Low-Risk

3 low-risk code issues found

Medium-Risk

1 medium-risk code issues found

High-Risk

0 high-risk code issues found

Low-Risk

1) Contract contains Reentrancy vulnuerabilities

```
function swapBack() private {
    uint256 contractBalance = balanceOf(address(this));

    // prevent extremely large dumps.
    if(contractBalance > swapTokensAtAmount * 5){
        contractBalance = swapTokensAtAmount * 5;
    }

    uint256 totalTokensToSwap = tokensForLiquidity + tokensForMarketing + tokensForDev + tokensForBuyBack;
    bool success;

    if(contractBalance == 0 || totalTokensToSwap == 0) {return;}
```

Recommendation

Apply the check-effects-interaction pattern

2) Missing Zero Address Validation

Detect missing zero address validation.

```
function updateMarketingWallet(address newMarketingWallet) external onlyOwner {
        emit marketingWalletUpdated(newMarketingWallet, marketingWallet);
        marketingWallet = newMarketingWallet;
    }

function updateDevWallet(address newWallet) external onlyOwner {
        emit devWalletUpdated(newWallet, devWallet);
        devWallet = newWallet;
    }

function updateBuyBackWallet(address newWallet) external onlyOwner {
        emit buyBackWalletUpdated(newWallet, buyBackWallet);
        buyBackWallet = newWallet;
    }
```

Recommendation

Check that the address is not zero.

3) Divide before Multiply

```
function _transfer(
        address from,
        address to,
        uint256 amount
    ) internal override {
        require(from != address(0), "ERC20: transfer from the zero address");
        require(to != address(0), "ERC20: transfer to the zero address");

         if(amount == 0) {
            super._transfer(from, to, 0);
            return;
        }
        if(!tradingActive){
            require(_isExcludedFromFees[from] || _isExcludedFromFees[to], "Trading is not active.");
        }
        if(
            swapEnabled &&
            !swapping &&
            !automatedMarketMakerPairs[from] &&
            !_isExcludedFromFees[from] &&
            !_isExcludedFromFees[to]
        ) {
            swapping = true ;
            swapBack();
            swapping = false;
        }
        bool takeFee = !swapping;
```

Recommendation

Consider ordering multiplication before division

Medium-Risk

1) Functions that send ether to arbitary Destination

```
function swapBack() private {
        uint256 contractBalance = balanceOf(address(this));
        // prevent extremely large dumps.
        if(contractBalance > swapTokensAtAmount * 5){
            contractBalance = swapTokensAtAmount * 5;
        }
        uint256 totalTokensToSwap = tokensForLiquidity + tokensForMarketing + tokensForDev + tokensForBuyBack;
        bool success;
        if(contractBalance == 0 || totalTokensToSwap == 0) {return;}
        // Halve the amount of liquidity tokens
        uint256 liquidityTokens = contractBalance * tokensForLiquidity / totalTokensToSwap / 2;
        uint256 amountToSwapForETH = contractBalance.sub(liquidityTokens);
        uint256 initialETHBalance = address(this).balance;
        swapTokensForEth(amountToSwapForETH);
        uint256 ethBalance = address(this).balance.sub(initialETHBalance);
        uint256 ethForMarketing = ethBalance.mul(tokensForMarketing).div(totalTokensToSwap);
        uint256 ethForDev = ethBalance.mul(tokensForDev).div(totalTokensToSwap);
        uint256 ethForBuyBack = ethBalance.mul(tokensForBuyBack).div(totalTokensToSwap);
        uint256 ethForLiquidity = ethBalance - ethForMarketing - ethForDev - ethForBuyBack;
        tokensForLiquidity = 0;
        tokensForMarketing = 0;
        tokensForDev = 0;
        tokensForBuyBack = 0;
        (success,) = address(devWallet).call{value: ethForDev}("");
        (success,) = address(buyBackWallet).call{value: ethForBuyBack}("");
```

Recommendation

Ensure that an arbitary user cannot withdraw unauthorized funds.

Manual Audit (Contract Function)

```
contract TECHDOGE is ERC20, Ownable {
    using SafeMath for uint256;

    IUniswapV2Router02 public immutable uniswapV2Router;
    address public immutable uniswapV2Pair;
    address public constant deadAddress = address(0xdead);

    bool private swapping;

    address public marketingWallet;
    address public devWallet;
    address public buyBackWallet;

    uint256 public swapTokensAtAmount;

    bool public tradingActive = false;
    bool public swapEnabled = false;

    uint256 public buyTotalFees;
    uint256 public buyMarketingFee;
    uint256 public buyLiquidityFee;
    uint256 public buyDevFee;
    uint256 public buyBuyBackFee;

    uint256 public sellTotalFees;
    uint256 public sellMarketingFee;
    uint256 public sellLiquidityFee;
    uint256 public sellDevFee;
    uint256 public sellBuyBackFee;

    uint256 public tokensForMarketing;
    uint256 public tokensForLiquidity;
    uint256 public tokensForDev;
    uint256 public tokensForBuyBack;

    // exlcude from fees and max transaction amount
    mapping (address => bool) private _isExcludedFromFees;

    // store addresses that a automatic market maker pairs. Any transfer *to* these addresses
    // could be subject to a maximum transfer amount
    mapping (address => bool) public automatedMarketMakerPairs;

    event ExcludeFromFees(address indexed account, bool isExcluded);

    event SetAutomatedMarketMakerPair(address indexed pair, bool indexed value);

    event marketingWalletUpdated(address indexed newWallet, address indexed oldWallet);

    event devWalletUpdated(address indexed newWallet, address indexed oldWallet);
```

```
event buyBackWalletUpdated(address indexed newWallet, address indexed oldWallet);

   event SwapAndLiquify(
       uint256 tokensSwapped,
       uint256 ethReceived,
       uint256 tokensIntoLiquidity
   );
```

Important Points To Consider

✓Source does not contain blacklist capability

✓ The owner cannot stop Trading.

✓ Verified contract source

✗Ownership renounced or source does not contain an owner contract

✗ Source does not contain a fee modifier

✓ Buy fee can’t be set more than 15%

✓ Sell fee can’t be set more than 15%

About Expelee

Expelee is a community driven organisation dedicated to fostering an anti-rug movement. We're here to keep investment safe from fraudsters. We've encountered several rug pulls and know how it feels to be duped, which is why we don't want anybody else to go through the same experience. We are here to raise awareness through our
services so that the future of cryptocurrency can be rug-free.

The auditing process focuses to the following considerations with collaboration of an expert team:

- Functionality test of the Smart Contract to determine if proper logic has been followed throughout the whole process.
- Manually detailed examination of the code line by line by experts.
- Live test by multiple clients using Test net.
- Analysing failure preparations to check how the Smart
- Contract performs in case of any bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analysing the security of the on-chain data.

Social Media

/Expelee

/ExpeleeOfficial

/expelee-co