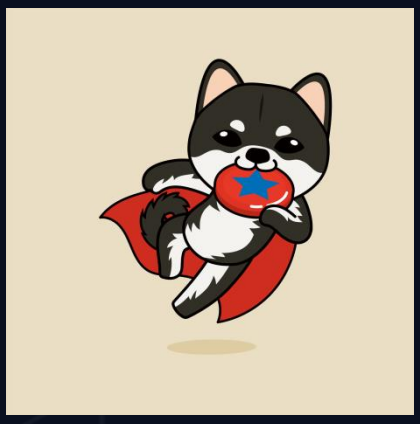
expelee

A Secure Place For Web3

SMART CONTRACT AUDIT OF

DiscDoge Presale



Contract Address

0x55bD484ba92cb52c9ea393e3Ff55d3E3A4Fb1D87

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: Not done till date of audit

Audit Date: 20/06/2022

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	**DiscDoge**
Compiler Version	**v0.6.12+commit.27d51765**
Optimization	**No with 200 runs**
License	**MIT license**
Explorer	**https://bscscan.com/address/0x55bD484ba92cb52c9ea393e3Ff55d3E3A4Fb1D87#code**
Symbol	**DDoge**
Decimals	**18**
Total Supply	**10,000,000,000**
Domain	**https://discdoge.com/**

Project Review

Token Name: DiscDoge

Web Site: https://discdoge.com/

Twitter: https://twitter.com/discdoge1

Telegram: https://t.me/discdoge

Contract Address:

0x55bD484ba92cb52c9ea393e3Ff55d3E3A4Fb1D87

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

5 low-risk code issues found

Medium-Risk

0 medium-risk code issues found

High-Risk

0 high-risk code issues found

Low-Risk

1) Contract contains Reentrancy vulnuerabilities

```
function _transfer(address
    sender,

    address recipient,uint256
    amount

) internal virtual {

    require(sender != address(0), "ERC20: transfer from the zero address");
    require(recipient != address(0), "ERC20: transfer to the zero address");


    _beforeTokenTransfer(sender, recipient, amount);


    _balances[sender] = _balances[sender].sub(amount, "ERC20: transfer amount exceeds balance");

    _balances[recipient] = _balances[recipient].add(amount);emit
    Transfer(sender, recipient, amount);

}
```

Recommendation

Apply the check-effects-interaction pattern

2) No zero address validation for some functions

Detect missing zero address validation.

```
function updateUniswapV2Router(address newAddress) public onlyOwner {

    require(newAddress != address(uniswapV2Router), "TOKEN: The router already has that address");emit
    UpdateUniswapV2Router(newAddress, address(uniswapV2Router));

    uniswapV2Router = IUniswapV2Router02(newAddress);

    address _uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory())

        .createPair(address(this), uniswapV2Router.WETH());
    uniswapV2Pair = _uniswapV2Pair;

}
```

Recommendation

Check that the new address is not zero.

3) Missing events arithmethics

Detect missing events for critical arithmetic parameters .

```
function setFee(
    uint256    _MktFee,   uint256
    _DevFee,uint256 _LpFee,
    uint256 _BurnFee,
    uint256    _UsefulShare,   uint256
    _OtherShare
) public onlyOwner { MktFee =
    _MktFee;DevFee = _DevFee;
    LpFee = _LpFee;
    BurnFee = _BurnFee;

    UsefulShare = _UsefulShare;OtherShare =
    _OtherShare;

    AllFee = MktFee.add(DevFee).add(LpFee).add(BurnFee);AllShare =
    UsefulShare.add(OtherShare);
}
```

Recommendation

Check that the new address is not zero.

4) Functions that send Ether to arbitary destinations

Unprotected call to a function sending Ether to arbitary address.

```
uint256 BNBAmount = address(this).balance;

uint256 BNBAmountForDev = BNBAmount.div(2);
uint256 BNBAmountForMkt = BNBAmount.sub(BNBAmountForDev);

payable(_AppWalletAddress).transfer(BNBAmountForDev);
payable(_MktWalletAddress).transfer(BNBAmountForMkt);

swapping = false;
```

Recommendation

Ensure that an arbitary user cannot withdraw unauthorized funds

5) Reductant Statements

Detect the usage of reductant usage statements that have no effect.

```
function _msgData() internal view virtual returns (bytes calldata) {
    this; // silence state mutability warning without generating bytecode
    return msg.data;
}
```

Recommendation

Remove redundant statements if they congest code but offer no value.

Important Points To Consider

✓ The owner cannot mint tokens after Initial commit

✗ Source does not contain blacklist capability

✓ Verified contract source

✓ Token is sellable (not a honeypot) at this time

✗ Ownership renounced or source does not contain an owner contract

✗ Source does not contain a fee modifier

✓ Buy fee is less than 10% (6%)

✗ Sell fee is less than 10% (17%)

✓ Owner/creator wallet contains less than 10% of circulating token supply (0%)

✓ All other holders possess less than 10% of circulating token supply

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