expelee

A Secure Place For Web3

SMART CONTRACT AUDIT OF

Chili Ecosystem Fair Launch

Contract Address

0x71F15AD33B4DF6b727dF98aA212ae6699BECb84d

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: 16/07/2022

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	**ChiliEcosystem**
Compiler Version	**v0.8.13+commit.abaa5c0e**
Optimization	**Yes with 200 runs**
License	**MIT license**
Explorer	**https://bscscan.com/address/0x71F15AD33B4DF6b727dF98aA212ae6699BECb84d#code**
Symbol	**CHILI**
Decimals	**18**
Total Supply	**100,000,000**
Domain	**https://chiliecosystem.finance/**

Project Review

Token Name: Chili Ecosystem

Web Site: https://chiliecosystem.finance/

Twitter: https://twitter.com/ChiliEcosystem

Telegram: https://t.me/chiliecosystem

Contract Address:
0x71F15AD33B4DF6b727dF98aA212ae6699BECb84d

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

Audit Summary

```
Compiled with solc
Number of lines: 871 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 8 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 12
Number of informational issues: 14
Number of low issues: 5
Number of medium issues: 0
Number of high issues: 0
ERCs: ERC20
```

Name	# functions	ERCS	ERC20 info	Complex code	Features
Address	1			No	Send ETH
IFactory	1			No	
IRouter	5			No	Receive ETH
ChiliEcosystem	61	ERC20	∞ Minting	Yes	Receive ETH
			Approve Race Cond.		Send ETH
					Ecrecover
					Assembly

Low-Risk

1) Contract contains Reentrancy vulnuerabilities

```
function _transfer(address sender, address recipient, uint256 amount) internal override {
       require(amount > 0, "Transfer amount must be greater than zero");

       uint256 feesum;
       uint256 fee;

       //set fee to zero if fees in contract are handled or exempted
       if (_liquidityMutex || exemptFee[sender] || exemptFee[recipient]) fee = 0;

       //calculate fee
       else{
           feesum = taxes.marketing + taxes.liquidity;
       }
       fee = amount * feesum / 100;

       //send fees if threshold has been reached
       //don't do this on buys, breaks swap
       if (providingLiquidity && sender != pair && feesum > 0 && !_liquidityMutex) handle_fees();

       if(address(this).balance > 0 && taxes.marketing > 0){
            swapBNBForUSDT();
       }

       //rest to recipient
       super._transfer(sender, recipient, amount - fee);
       if(fee > 0){
           super._transfer(sender, address(this) ,fee);
       }
   }
```

Recommendation

Apply the check-effects-interaction pattern

2) Dangerous strict equalities

Use of strict equalities that can be easily manipulated by an attacker.

```
function _writeCheckpoint(
        address delegatee,
        uint32 nCheckpoints,
        uint256 oldVotes,
        uint256 newVotes
    )
        internal
    {
        uint32 blockNumber = safe32(block.number, "CHILI::_writeCheckpoint: block number exceeds 32 bits");

        if (nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber) {
            checkpoints[delegatee][nCheckpoints - 1].votes = newVotes;
        } else {
            checkpoints[delegatee][nCheckpoints] = Checkpoint(blockNumber, newVotes);
            numCheckpoints[delegatee] = nCheckpoints + 1;
        }

        emit DelegateVotesChanged(delegatee, oldVotes, newVotes);
    }
```

Recommendation

Don’t use strict equality to determine if an account has enough Ehter or tokens.

3) Missing zero address validation

```
function updateMarketingWallet(address newWallet) external onlyOwner{
        marketingWallet = newWallet;
    }
```

Recommendation

Check that the address is not zero.

4) Local variable shadowing.

Detection os shadowing using local variables.

```
function allowance(address owner, address spender) public override view returns (uint256) {
        return _allowances[owner][spender];
    }
```

Recommendation

Rename the local vaariable that shadow another component.

5) Unused return

```
function addLiquidity(uint256 tokenAmount, uint256 bnbAmount) private {
        // approve token transfer to cover all possible scenarios
        _approve(address(this), address(router), tokenAmount);

        // add the liquidity
        router.addLiquidityETH{value: bnbAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            lpRecipient,
            block.timestamp
        );
    }
```

Recommendation

Ensure that all the return values of the function calls are stored.

Manual Audit (Contract Function)

```
contract ChiliEcosystem is BEP20 {
    using Address for address payable;

    mapping (address => bool) public exemptFee;

    IRouter public router;
    address public pair;
    address constant USDT = 0x55d398326f99059fF775485246999027B3197955;

    bool private _liquidityMutex;
    bool public providingLiquidity = true;
    uint256 public tokenLiquidityThreshold = 1000 * 10e18;


    address public marketingWallet = 0x653B4f9CeEC0eFf784E52DBBdD95A5229cE76059;
    address public lpRecipient;

    struct Taxes {
        uint256 marketing;
        uint256 liquidity;
    }

    Taxes public taxes = Taxes(11,1);

    /// @notice A record of each accounts delegate
    mapping (address => address) internal _delegates;

    /// @notice A checkpoint for marking number of votes from a given block
    struct Checkpoint {
        uint32 fromBlock;
        uint256 votes;
    }

    /// @notice A record of votes checkpoints for each account, by index
    mapping (address => mapping (uint32 => Checkpoint)) public checkpoints;

    /// @notice The number of checkpoints for each account
    mapping (address => uint32) public numCheckpoints;

    /// @notice The EIP-712 typehash for the contract's domain
    bytes32 public constant DOMAIN_TYPEHASH = keccak256("EIP712Domain(string name,uint256 chainId,address
verifyingContract)");
```

Important Points To Consider

✓ Verified contract source

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

✗ Ownership renounced or source does not contain an owner contract

✗ Buy fee is less than 10% (12%)

✗ Sell fee is less than 10% (11%)

✗ Owner/creator wallet contains less than 10% of circulating token supply (24.01%)

— Tokens burned: 10%, circulating supply: 90,000,000

WARNING: This token was flagged due to evidence of a bug, hack, or scam: (By Token Sniffer)

- Exploit #012: Transfer block #3

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