freshcoins

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

ANCESTRY LEGION

## LEGION

$LEGION

01/03/2022

# TABLE OF CONTENTS


1 DISCLAIMER

2 INTRODUCTION

3-4 AUDIT OVERVIEW

5-9 OWNER PRIVILEGES

10 CONCLUSION AND ANALYSIS

11 TOKEN DETAILS

12 LEGION TOKEN ANALYTICS &
TOP 10 TOKEN HOLDERS

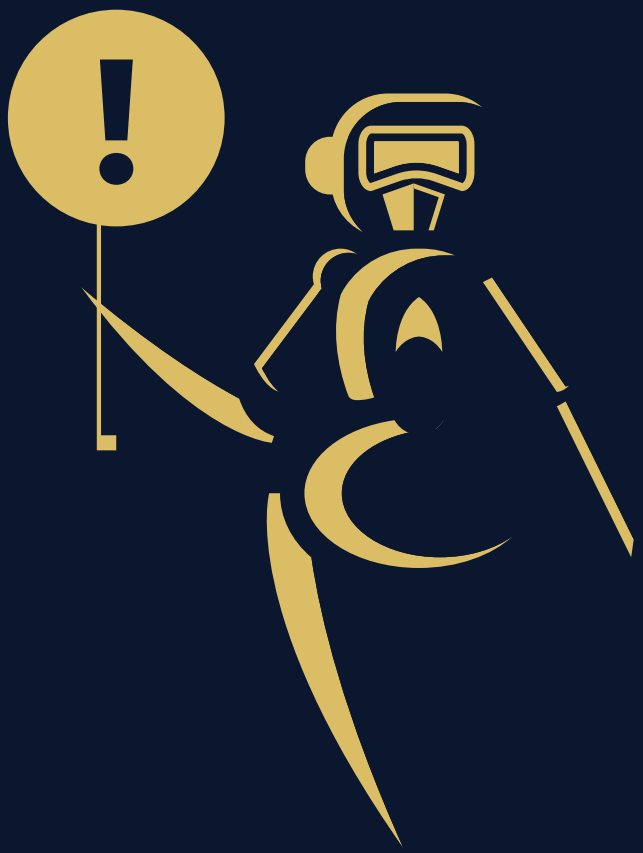
13 TECHNICAL DISCLAIMER


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

0xe91200492B1Ac813663ABc45d2456dfB7617dE89

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 01/03/2022

# AUDIT OVERVIEW

92

Security Score

90

**Static Scan**
Automatic scanning for common vulnerabilities

94

**ERC Scan**
Automatic checks for ERC's conformance

0 High

0 Medium

1 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 | Low |
| 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 exclude/include wallet from tax

```
function F21__Exclude_Account_From_Paying_Fees(address account) external onlySecurityManager {
        ieff[account] = true;
}

function F22__Enable_Account_Must_Pay_Fees(address account) external onlySecurityManager {
        ieff[account] = false;
}
```

## Contract owner can exclude/include wallet from rewards

```
function F23__Exclude_Account_from_Receiving_Reflections(address account) external onlySecurityManager {
        // Account will not receive reflections
        require(!ier[account], "Account is already excluded");
        if(_rOwned[account] > 0) {
          _tOwned[account] = tokenFromReflection(_rOwned[account]);
        }
        ier[account] = true;
          _excluded.push(account);
}

function F24__Enable_Account_will_Receive_Reflections(address account) external onlySecurityManager {
        require(ier[account]);
        for (uint256 i = 0; i < _excluded.length; i++) {
          if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            ier[account] = false;
            _excluded.pop();
            break;
          }
        }
}
```

## Contract owner can change max tx amount

```
function F10_Set_Buy_Max_Tx_Amount(uint256 percent_of_total_supply_or_tokens_amount)
          external onlySecurityManager() {

          uint256 old_bmta = bmta.div(10**_decimals);
          if (percent_of_total_supply_or_tokens_amount <= 100) {
            bmta = _tTotal.mul(percent_of_total_supply_or_tokens_amount).div(100);
          } else {
            bmta = percent_of_total_supply_or_tokens_amount * 10**_decimals;
          }
          emit Buy_Max_Tx_Amount_Updated(old_bmta, bmta);
}
```

```
function F11_Set_Sell_Max_Tx_Amount(uint256 percent_of_total_supply_or_tokens_amount)
        external onlySecurityManager() {

        uint256 old_smta = smta.div(10**_decimals);
        if (percent_of_total_supply_or_tokens_amount <= 100) {
          smta = _tTotal.mul(percent_of_total_supply_or_tokens_amount).div(100);
        } else {
          smta = percent_of_total_supply_or_tokens_amount * 10**_decimals;
        }
        emit Sell_Max_Tx_Amount_Updated(old_smta, smta);
}
```

## Contract owner can enable/disable trading

```
function F03__Enable_Public_Trading() external onlySecurityManager {
        Public_Trading_Enabled = true;
}

function F04__Disable_Public_Trading() external onlySecurityManager {
        Public_Trading_Enabled = false;
}
```

## Contract owner can include/exclude address from transactions

```
function F01_Blacklist_Malicious_Account(address account) external ExceptAccounts(account) {
        require(CEO == msg.sender || Security_Manager[msg.sender] || Chief_Security_Officer == msg.sender);
        require(!isBlacklisted[account], "Address is already blacklisted");
        isBlacklisted[account] = true;
}

function F02_Whitelist_Account(address account) external onlySecurityManager {
        require(isBlacklisted[account], "Address is already whitelisted");
        isBlacklisted[account] = false;
}
```

## Contract owner can withdraw BEP20 tokens from smart contract

```
function F34__Rescue_Other_Tokens_Sent_To_This_Contract(IERC20 token, address receiver, uint256
amount) external {
        // This feature is very appreciated:
        // To be able to send back to a user other BEP20 tokens
        // that the user have sent to this contract by mistake.
        require(CEO == msg.sender || Security_Manager[msg.sender] || Chief_Security_Officer == msg.sender);
        require(token != IERC20(address(this)), "Only other tokens can be rescued");
        require(token.balanceOf(address(this)) >= amount, "Insufficient balance");
        require(receiver != address(this));
        require(receiver != address(0));
        token.transfer(receiver, amount);
}
```

## Contract owner can change swap settings

```
function F33__Set_Min_Amount_Tokens_For_ProjectFundingSwap(uint256 amount) external onlySecurity-
Manager {
        // Example: 10 tokens --> amount = 100000000 (i.e. 10 * 10**7 decimals) = 0.0002%
        matpfs = amount * 10 **_decimals;
}
```

## Contract owner can change gap between trades

```
function F05__Update_When_Account_Can_Sell_Again(address account, uint256 unix_time) external onlySecurityManager {
        // Tips:
        // To allow selling immediately:  unix_time = 0
        SAT[account] = unix_time;
}

function F06__Set_Normal_Waiting_Time_Between_Sells(uint256 wait_seconds) external onlySecurityManager {
        // Example:
        // 60 seconds waiting time:  wait_seconds = 60
        //
        // To disable the waiting time: wait_seconds = 0
        require (wait_seconds <= wttsai2 ||
            wttsai2 == 0,
            "The normal waiting time cannot be larger than waiting time after price impact2");
        nwtbs = wait_seconds;
        if (pi2 != 0 && wttsai2 == 0) {
           wttsai2 = wait_seconds;
        }
}

function F07__Set_Waiting_Time_For_Next_Sell_After_Impact2(uint256 wait_seconds) external onlySecurityManager {
        //Examples:  Must wait 3 days:  wait_seconds = 259200
        //                7 days:  wait_seconds = 604800
        //
        // Requires pi2 to be enabled (to be more than zero)
        require (pi2 != 0,
            "The waiting time after impact2 cannot be set when pi2 is 0");
        // Must be at least same but usually longer
        // waiting time than the normal waiting time
        require (wait_seconds >= nwtbs,
            "The waiting time after impact2 cannot be less than the normal waiting time");
        wttsai2 = wait_seconds;
}
```

## Contract owner can change tax distribution percentage

```
function F16__Set_Product_Development_Fee_Portion(uint256 fee_percent) external onlyCEO {
        // Example: 50% of total Project Fee --> fee_percent = 50
        // IMPORTANT: pdf + mf = 100
        uint256 Total_All_Portions =  fee_percent + mf;
        require(Total_All_Portions <= 100,
        "The sum of all fees portions must be less or equal 100");
        pdf = fee_percent;
}

function F17__Set_Marketing_Fee_Portion(uint256 fee_percent) external onlyMarketingManager {
        // Example: 50% of total Project Fee --> fee_percent = 50
        // IMPORTANT: pdf + mf = 100
        uint256 Total_All_Portions =  pdf + fee_percent;
        require(Total_All_Portions <= 100,
        "The sum of all fees portions must be less or equal 100");
        mf = fee_percent;
}
```

## Contract owner can change tax up to 100% especially for sell transactions

```
function F11__Set_Fees_For_Transfers(
        uint256 project_fee_percent, uint256 reflections_fee_percent) external onlySecurityManager {
        // Project fee for (normal) transfers between wallets.
        tpf = project_fee_percent;
        trf = reflections_fee_percent;
}

function F12__Set_Fees_For_Buys(
        uint256 project_fee_percent, uint256 reflections_fee_percent) external onlySecurityManager {
        bpf = project_fee_percent;
        brf = reflections_fee_percent;
}

function F13__Set_Fees_For_Sells_With_Price_Impact1(
        uint256 project_fee_under_impact1,
        uint256 project_fee_above_impact1,
        uint256 reflections_fee_under_impact1,
        uint256 reflections_fee_above_impact1
        ) external onlySecurityManager {
        //The fees are a percentage number
        require(pi1 != 0, "Cannot set price impact1 fees when impact1 is 0");
        require(project_fee_under_impact1 <= project_fee_above_impact1,
        "The project fee under price impact1 cannot be larger than the fee above price impact1");
        require(reflections_fee_under_impact1 <= reflections_fee_above_impact1,
        "The reflections fee under price impact1 cannot be larger than the fee above price impact1");
        if (pi2 != 0) {
          if (spfai2 == 0) {
            spfai2 = project_fee_above_impact1;
          } else {
            require(project_fee_above_impact1 <= spfai2);
          }
          if (srfai2 == 0) {
            srfai2 = reflections_fee_above_impact1;
          } else {
            require(reflections_fee_above_impact1 <= srfai2);
          }
        }
        spfui1 = project_fee_under_impact1;
        spfai1 = project_fee_above_impact1;
        srfui1 = reflections_fee_under_impact1;
        srfai1 = reflections_fee_above_impact1;
}
```

## Contract owner can transfer ownership

```
function F25__CEO_Change__By_Chief_Security_Officer(address New_CEO) public virtual onlyCSO {
        //CEO can be changed only by CSO
        require(New_CEO != address(0));
        require(New_CEO != address(this));
        require(New_CEO != Chief_Security_Officer);
        require(!Security_Manager[New_CEO]);
        require(!Marketing_Manager[New_CEO]);
        address Previous_CEO = CEO;
        CEO = New_CEO;
        ieff[New_CEO] = true;
        ieff[Previous_CEO] = false;
        emit CEO_Changed_By_Chief_Security_Officer(Previous_CEO, New_CEO);
}
```

## Contract owner can change pdw and mw addresses

### Current values:

pdw ( Product_Development_Wallet ) : 0x683e1dbad64d719f3a4c33c3b2870fe0fc891f1f

mw ( Marketing_Wallet ) : 0x6ab51b667d231e2b97fd3ec7796063a33c24c149

```
function F18__Set_Product_Development_Wallet(address account) external onlyCEO {
        pdw = account;
}

function F19__Set_Marketing_Wallet(address account) external onlyMarketingManager {
        mw = account;
}
```

# 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 1 LOW issue during the first review.

# TOKEN DETAILS

## Details

| | |
|---|---|
| Buy fees: | 2% |
| Sell fees: | 6% |
| Max TX: | 2 |
| Max Sell: | 2 |

## Honeypot Risk

| | |
|---|---|
| Ownership: | Owned |
| Blacklist: | Detected |
| Modify Max TX: | Detected |
| Modify Max Sell: | Detected |
| Disable Trading: | Detected |

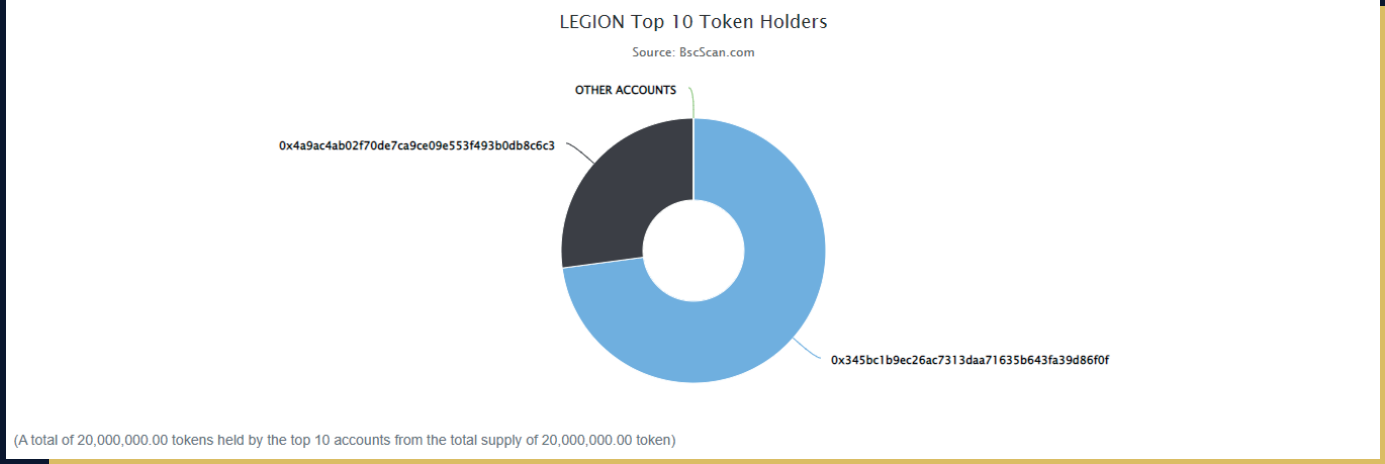
## Rug Pull Risk

| | |
|---|---|
| Liquidity: | N/A |
| Holders: | Clean |

# LEGION TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (20,000,000.00 Tokens) of LEGION

Token Total Supply: 20,000,000.00 Token | Total Token Holders: 2




(A total of 20,000,000.00 tokens held by the top 10 accounts from the total supply of 20,000,000.00 token)

| Rank | Address | Quantity (Token) | Percentage |
|---|---|---|---|
| 1 | 0x345bc1b9ec26ac7313daa71635b643fa39d86f0f | 14,581,450 | 72.9073% |
| 2 | 0x4a9ac4ab02f70de7ca9ce09e553f493b0db8c6c3 | 5,418,550 | 27.0928% |

# 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.