



# Coinsult

## Advanced Manual Smart Contract Audit



**Project:** Fbox

**Website:** <https://www.fswaptools.vip>

**Low-Risk**

7 low-risk code  
issues found

**Medium-Risk**

0 medium-risk code  
issues found

**High-Risk**

0 high-risk code  
issues found

**Contract Address**

0xA874A23e4b0C9d1d726ec88ade07850A63432032

Disclaimer: Coinsult is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your own research.

# Disclaimer

Coinsult is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

Coinsult is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your own research.

The information provided in this audit is for informational purposes only and should not be considered investment advice. Coinsult does not endorse, recommend, support or suggest to invest in any project.

Coinsult can not be held responsible for when a project turns out to be a rug-pull, honeypot or scam.

# Tokenomics

Rank	Address	Quantity (Token)	Percentage
1	0x5bb8506fcc5f27307f88ad1d73cce1b94876334c	10,000,000	100.0000%

# Source Code

Coinsult was comissioned by Fbox to perform an audit based on the following smart contract:

<https://bscscan.com/address/0xA874A23e4b0C9d1d726ec88ade07850A63432032#code>

# Manual Code Review

In this audit report we will highlight all these issues:

## Low-Risk

7 low-risk code  
issues found

## Medium-Risk

0 medium-risk code  
issues found

## High-Risk

0 high-risk code  
issues found

The detailed report continues on the next page...

● **Low-Risk:** Could be fixed, will not bring problems.

## Contract contains Reentrancy vulnerabilities

Additional information: This combination increases risk of malicious intent. While it may be justified by some complex mechanics (e.g. rebase, reflections, buyback).

More information: Slither

```
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");

    uint256 senderBalance = _balances[sender];
    require(senderBalance >= amount, "ERC20: transfer amount exceeds balance");
    unchecked {
        _balances[sender] = senderBalance - amount;
    }

    if (!_bought[recipient] && !isContract(recipient) && recipient != _dead) {
        _bought[recipient] = true;
        buyUser.push(recipient);
    }

    if (_whites[sender] || _whites[recipient]) {
        _balances[recipient] += amount;
        emit Transfer(sender, recipient, amount);
        return;
    }
}
```

## Recommendation

Apply the check-effects-interactions pattern.

## Exploit scenario

```
function withdrawBalance(){
    // send userBalance[msg.sender] Ether to msg.sender
    // if msg.sender is a contract, it will call its fallback function
    if( ! (msg.sender.call.value(userBalance[msg.sender]))() ) ){
        throw;
    }
    userBalance[msg.sender] = 0;
}
```

Bob uses the re-entrancy bug to call withdrawBalance two times, and withdraw more than its initial deposit to the contract.

● **Low-Risk:** Could be fixed, will not bring problems.

## Avoid relying on `block.timestamp`

`block.timestamp` can be manipulated by miners.

```
function _swapFistForToken(address a2, uint256 tokenAmount) private {
    address[] memory path = new address[](2);
    path[0] = _fist; path[1] = a2;
    IPancakeRouter02(_router).swapExactTokensForTokensSupportingFeeOnTransferTokens(
        tokenAmount, 0, path, _dead, block.timestamp);
}
```

## Recommendation

Do not use `block.timestamp`, `now` or `blockhash` as a source of randomness

## Exploit scenario

```
contract Game {

    uint reward_determining_number;

    function guessing() external{
        reward_determining_number = uint256(block.blockhash(10000)) % 10;
    }
}
```

Eve is a miner. Eve calls `guessing` and re-orders the block containing the transaction. As a result, Eve wins the game.

● **Low-Risk:** Could be fixed, will not bring problems.

## Too many digits

Literals with many digits are difficult to read and review.

```
_mint(_main, 10000000 * 10 ** decimals());
```

## Recommendation

Use: Ether suffix, Time suffix, or The scientific notation

## Exploit scenario

```
contract MyContract{
    uint 1_ether = 1000000000000000000;
}
```

While 1\_ether looks like 1 ether, it is 10 ether. As a result, it's likely to be used incorrectly.

● **Low-Risk:** Could be fixed, will not bring problems.

## No zero address validation for some functions

Detect missing zero address validation.

```
function setAddrs(address mark, address flow, address wrap) public onlyOwner {
    _mark = mark;
    _flow = flow;
    _wrap = wrap;
}
```

## Recommendation

Check that the new address is not zero.

## Exploit scenario

```
contract C {

    modifier onlyAdmin {
        if (msg.sender != owner) throw;
        _;
    }

    function updateOwner(address newOwner) onlyAdmin external {
        owner = newOwner;
    }
}
```

Bob calls updateOwner without specifying the newOwner, so Bob loses ownership of the contract.

● **Low-Risk:** Could be fixed, will not bring problems.

## Unchecked transfer

The return value of an external transfer/transferFrom call is not checked.

```
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");

    uint256 senderBalance = _balances[sender];
    require(senderBalance >= amount, "ERC20: transfer amount exceeds balance");
    unchecked {
        _balances[sender] = senderBalance - amount;
    }

    if (!_bought[recipient] &&& !isContract(recipient) &&& recipient != _dead) {
        _bought[recipient] = true;
        buyUser.push(recipient);
    }

    if (_whites[sender] || _whites[recipient]) {
        _balances[recipient] += amount;
        emit Transfer(sender, recipient, amount);
        return;
    }
}
```

## Recommendation

Use SafeERC20, or ensure that the transfer/transferFrom return value is checked.

## Exploit scenario

```
contract Token {
    function transferFrom(address _from, address _to, uint256 _value) public returns (bool success);
}
contract MyBank{
    mapping(address => uint) balances;
    Token token;
    function deposit(uint amount) public{
        token.transferFrom(msg.sender, address(this), amount);
        balances[msg.sender] += amount;
    }
}
```

Several tokens do not revert in case of failure and return false. If one of these tokens is used in MyBank, deposit will not revert if the transfer fails, and an attacker can call deposit for free..



● **Low-Risk:** Could be fixed, will not bring problems.

## Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
function setMaxsell(uint256 val) public onlyOwner {  
    _maxsell = val;  
}
```

## Recommendation

Emit an event for critical parameter changes.

## Exploit scenario

```
contract C {  
  
    modifier onlyAdmin {  
        if (msg.sender != owner) throw;  
        _;  
    }  
  
    function updateOwner(address newOwner) onlyAdmin external {  
        owner = newOwner;  
    }  
}
```

updateOwner() has no event, so it is difficult to track off-chain changes in the buy price.

● **Low-Risk:** Could be fixed, will not bring problems.

## Conformance to Solidity naming conventions

Allow `_` at the beginning of the `mixed_case` match for private variables and unused parameters.

```
mapping(address => bool) public _buyed;
```

## Recommendation

Follow the Solidity naming convention.

## Rule exceptions

- Allow constant variable name/symbol/decimals to be lowercase (ERC20).
- Allow `_` at the beginning of the `mixed_case` match for private variables and unused parameters.

## Owner privileges

- Owner cannot set fees higher than 25%
- Owner cannot pause trading
- Owner can change max transaction amount
- Owner can exclude from fees
- ⚠ Owner can set max bonus value

## Extra notes by the team

No notes

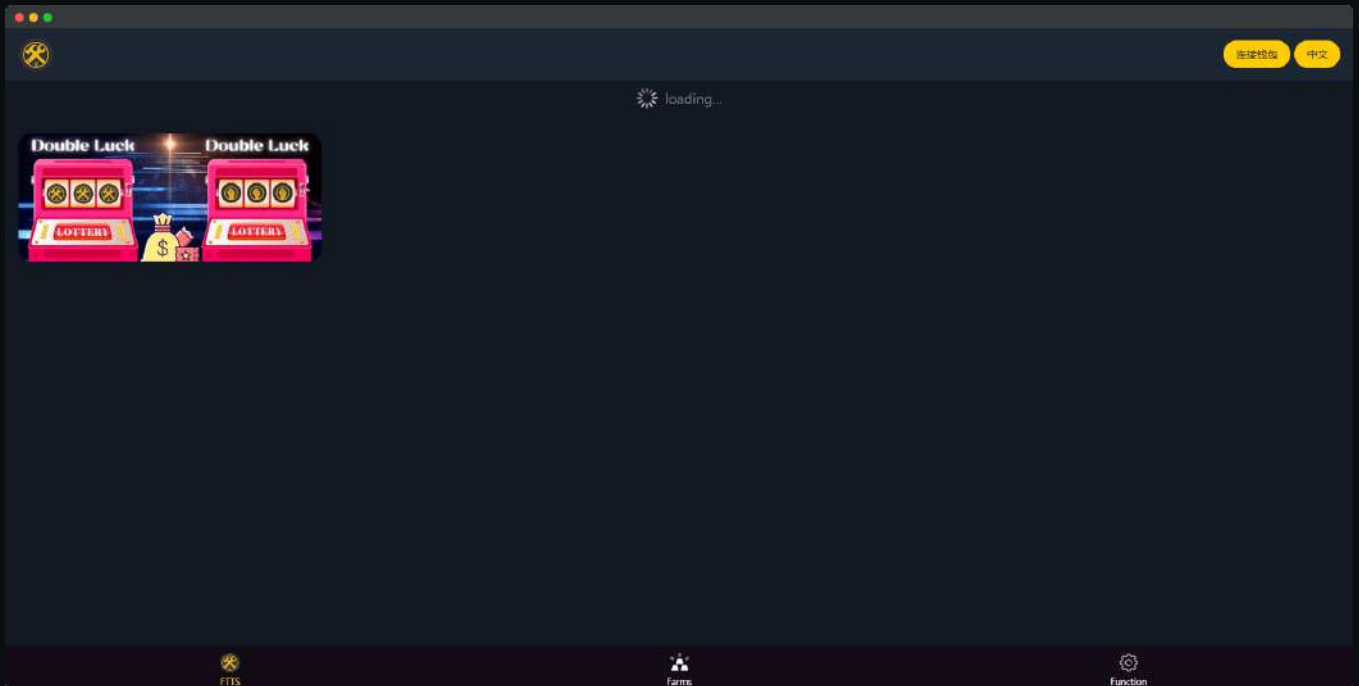
# Contract Snapshot

```
contract Token is Ownable, IERC20Metadata {
mapping(address => bool) public _buyed;
mapping(address => bool) public _whites;
mapping(address => uint256) private _balances;
mapping(address => mapping(address => uint256)) private _allowances;

string private _name;
string private _symbol;
uint256 private _totalSupply;
uint256 public _maxsell;
uint256 public _maxfist;
```

# Website Review

Coinsult checks the website completely manually and looks for visual, technical and textual errors. We also look at the security, speed and accessibility of the website. In short, a complete check to see if the website meets the current standard of the web development industry.



- Mobile Friendly
- Does not contain jQuery errors
- SSL Secured
- No major spelling errors

# Project Overview

● Not KYC verified by Coinsult

## Fbox

Audited by Coinsult.net



Date: 8 June 2022

✓ Advanced Manual Smart Contract Audit