Coinsult

Advanced Manual Smart Contract Audit

Project: BullDogeChain

Website: http://bulldogechain.io/

🟢 **Low-Risk**	🟡 **Medium-Risk**	🔴 **High-Risk**
5 low-risk code issues found	0 medium-risk code issues found	0 high-risk code issues found

Contract Address

0xf462a351cff44716B0d31DF87976467FDF83CB9d

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

Not available

Source Code

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

https://bscscan.com/address/0xf462a351cff44716B0d31DF87976467FDF83CB9d#code

Safu contract by ContractChecker

Manual Code Review

In this audit report we will highlight all these issues:

🟢 Low-Risk

5 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.

Avoid relying on block.timestamp

block.timestamp can be manipulated by miners.

```
if(antibotEnabled){
    if (launchedAt == 0 &amp;&amp; to == (uniswapV2Pair)) {
        launchedAt = block.timestamp;
    }else if(launchedAt + 30 &gt; block.timestamp) {
        if(from == uniswapV2Pair) {
            lastTransactionTime[to] = block.timestamp;
        } else {
            require(amount = 15, "You need to wait 15 seconds before selling at launch");
            lastTransactionTime[from] = block.timestamp;
        }
    }else if( launchedAt &gt; 0 ) {
        antibotEnabled = false;
    }
}
```

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.

No zero address validation for some functions

Detect missing zero address validation.

```
function changeMarketingWallet(address _treasuryWallet) external onlyOwner {
    require(_treasuryWallet != treasuryWallet, "Marketing wallet is already that address");
    require(!isContract(_treasuryWallet), "Marketing wallet cannot be a contract");
    treasuryWallet = _treasuryWallet;
    emit TreasuryWalletChanged(treasuryWallet);
}
```

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`, soBob loses ownership of the contract.

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

Functions that send Ether to arbitrary destinations

Unprotected call to a function sending Ether to an arbitrary address.

```
function sendBNB(address payable recipient, uint256 amount) internal {
    require(address(this).balance &gt;= amount, "Address: insufficient balance");

    (bool success, ) = recipient.call{value: amount}("");
    require(success, "Address: unable to send value, recipient may have reverted");
}
```

Recommendation

Ensure that an arbitrary user cannot withdraw unauthorized funds.

Exploit scenario

```
contract ArbitrarySend{
    address destination;
    function setDestination(){
        destination = msg.sender;
    }

    function withdraw() public{
        destination.transfer(this.balance);
    }
}
```

Bob calls `setDestination` and `withdraw`. As a result he withdraws the contract's balance.

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

Unchecked transfer

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

```
function claimStuckTokens(address token) external onlyOwner {
    require(token != address(this), "Owner cannot claim native tokens");
    if (token == address(0x0)) {
        payable(msg.sender).transfer(address(this).balance);
        return;
    }
    IERC20 ERC20token = IERC20(token);
    uint256 balance = ERC20token.balanceOf(address(this));
    ERC20token.transfer(msg.sender, balance);
}
```

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 setSwapTokensAtAmount(uint256 newAmount) external onlyOwner {
    require(
        newAmount &gt; totalSupply() / 1_000_000,
        "New Amount must more than 0.0001% of total supply"
    );
    swapTokensAtAmount = newAmount;
}

function setSwapWithLimit(bool _swapWithLimit) external onlyOwner {
    require(
        swapWithLimit != _swapWithLimit,
        "Swap with limit is already set to that state"
    );
    swapWithLimit = _swapWithLimit;
}
```

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.

Owner privileges

- 🟢 Owner cannot set fees higher than 25%
- 🟢 Owner cannot pause trading
- 🟢 Owner cannot change max transaction amount
- 🟡 Owner can exclude from fees

⚠️ Owner can enable antibot

Extra notes by the team

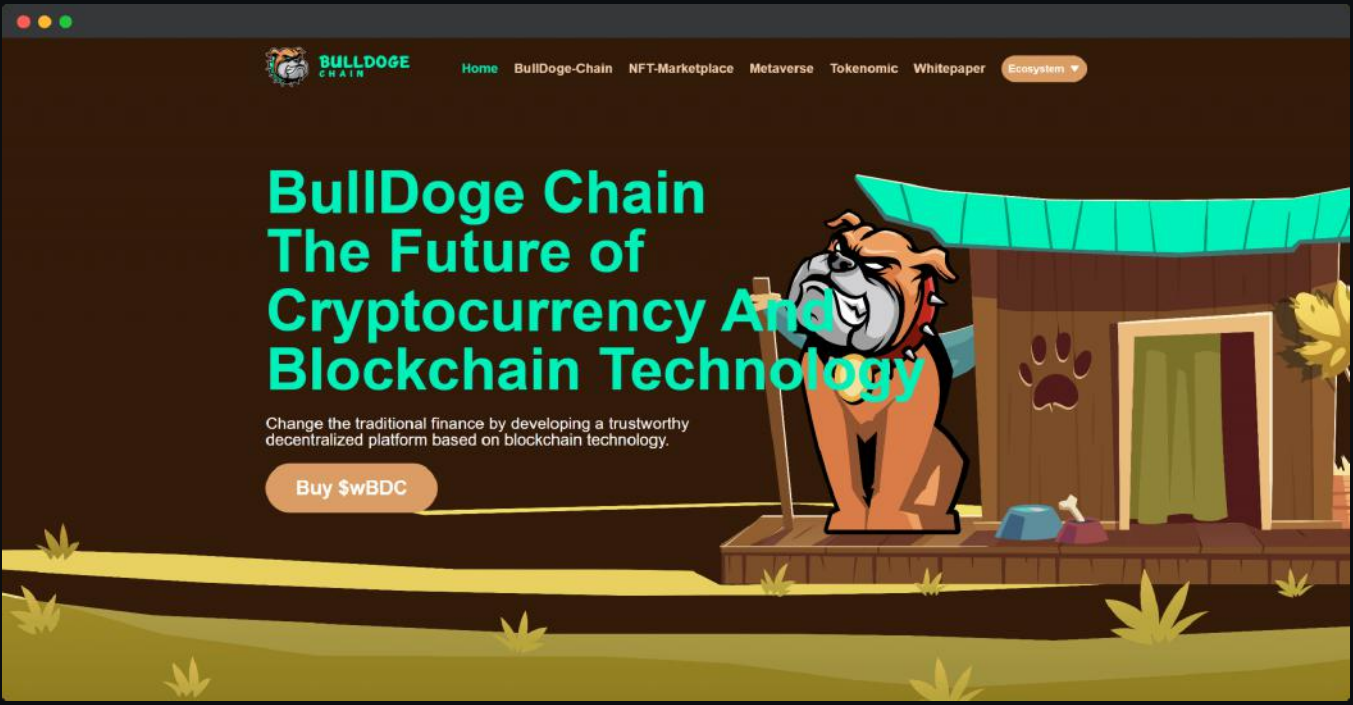
No notes

Contract Snapshot

```
contract BullDogeChain is ERC20, Ownable {
uint256 public  liquidityFee  = 1;
uint256 public  treasuryFee   = 1;

address public treasuryWallet = 0x8aF83D3703B05d73C31fB435B40F73967Cb29028;

IUniswapV2Router02 public uniswapV2Router;
address public  uniswapV2Pair;

address private DEAD = 0x000000000000000000000000000000000000dEaD;

uint256 public swapTokensAtAmount;
bool    public swapEnabled = true;
bool    public swapWithLimit;
bool    private swapping;

uint256 public launchedAt;
bool    public antibotEnabled = true;
mapping (address =&gt; uint) private lastTransactionTime;

mapping (address =&gt; bool) private _isExcludedFromFees;
mapping (address =&gt; bool) private automatedMarketMakerPairs;

event ExcludeFromFees(address indexed account, bool isExcluded);
event TreasuryWalletChanged(address marketingWallet);
event SetAutomatedMarketMakerPair(address indexed pair, bool indexed value);
event SwapAndLiquify(uint256 tokensSwapped, uint256 bnbReceived, uint256 tokensIntoLiqudity);
event SwapAndSendTreasury(uint256 tokensSwapped, uint256 bnbSend);

constructor () ERC20("BullDoge Chain", "wBDC")
{
    transferOwnership(0xD6bC008D73232Bb765054dFd500D4169B25e6Fd8);

    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E2560
    address _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
        .createPair(address(this), _uniswapV2Router.WETH());
```

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.



BullDoge Chain The Future of Cryptocurrency And Blockchain Technology

Change the traditional finance by developing a trustworthy decentralized platform based on blockchain technology.

● Mobile Friendly

● Does not contain jQuery errors

● SSL Secured

● No major spelling errors

Project Overview

● Not KYC verified by Coinsult

BullDogeChain

Audited by Coinsult.net

Date: 25 August 2022

✓ Advanced Manual Smart Contract Audit