Coinsult

Advanced Manual Smart Contract Audit

Project: SAFE ZONE V2
Website: http://safetokens.net/

Low-risk	Medium-risk	High-risk
4 low-risk code issues found	0 medium-risk code issues found	0 high-risk code issues found

Contract address
0xa701EC6B9C1883fcF727FED7e41FE925A1b1E91C

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

Total Supply: 10,000,000,000
Total Holders: 133
Top 10 holders:

Rank	Address	Quantity (Token)	Percentage
1	PancakeSwap V2: SAFU 195	2,102,082,989.654134609	21.0208%
2	0x8937ddfc029b050ae9b1d2c74f3fd97d8cb7bf57	550,759,585.790594458	5.5076%
3	0x6b0e2d0cb04a06d23492a8156b4bddf76c1f00fa	405,223,135.84	4.0522%
4	0x1cab64b4a2eac248b5f2aef332425aba80246d9d	331,507,582.682917437	3.3151%
5	0x495f0b1eed52f8dc168eb857e0c66e252a3285ff	294,623,981.67390647	2.9462%
6	0x84cd9753137a05f2c490103ed88c5f975d347dd3	288,901,407.146077535	2.8890%
7	0x98fd650f36765df04635be4e6a2a81236e959e06	287,502,066.499612909	2.8750%
8	0x1b79135b1bca24cd6646bf14feb44b63b5778624	258,274,001.042105091	2.5827%
9	0xc68169ef439390527dea8da36e0ee7cc442341e7	249,879,932.076731425	2.4988%
10	0x5a9ae8ba3bd10f1875a32c62fb56bd88463428cc	240,434,350.366096975	2.4043%

The top 100 holders collectively own 99.69% (9,968,704,497.18 Tokens) of SAFE ZONE V2

Note: This is a snapshot of when the audit was performed.

Source code

Coinsult was commissioned by SAFE ZONE V2 to perform an audit based on the following smart contract:

https://bscscan.com/address/0xa701EC6B9C1883fcF727FED7e41FE925A1b1E91C#code

Manual Code Review

🟢 Low-risk

4 low-risk code issues found.
Could be fixed, will not bring problems.

- Sol frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks.
  - Recommended: Deploy with any of the following Solidity versions:
    - 0.5.16 - 0.5.17
    - 0.6.11 - 0.6.12
    - 0.7.5 - 0.7.6 Use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing.

```
pragma solidity ^0.7.4;
```

- The return value of `RWRD.transfer(shareholder, amount);` is not checked
  - Recommended: Use SafeERC20, or ensure that the transfer/transferFrom return value is checked.

```
    function distributeDividend(address shareholder) internal {
        if(shares[shareholder].amount == 0){ return; }

        uint256 amount = getUnpaidEarnings(shareholder);
        if(amount > 0){
            totalDistributed = totalDistributed.add(amount);
            RWRD.transfer(shareholder, amount);
            shareholderClaims[shareholder] = block.timestamp;
```

```
            shares[shareholder].totalRealised =
shares[shareholder].totalRealised.add(amount);
            shares[shareholder].totalExcluded =
getCumulativeDividends(shares[shareholder].amount);
        }
    }
```

- Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision.
    - Recommended: Consider ordering multiplication before division.

```
    function takeFee(address sender, uint256 amount, bool isSell)
internal returns (uint256) {

        uint256 multiplier = isSell ? sellMultiplier : 100;
        uint256 feeAmount =
amount.mul(totalFee).mul(multiplier).div(feeDenominator * 100);

        uint256 burnTokens = feeAmount.mul(burnFee).div(totalFee);
        uint256 contractTokens = feeAmount.sub(burnTokens);

        _balances[address(this)] =
_balances[address(this)].add(contractTokens);
        _balances[burnFeeReceiver] =
_balances[burnFeeReceiver].add(burnTokens);
        emit Transfer(sender, address(this), contractTokens);

        if(burnTokens > 0){
            emit Transfer(sender, burnFeeReceiver, burnTokens);
        }

        return amount.sub(feeAmount);
    }
```

- block.timestamp can be manipulated by miners.
    - Recommended: Avoid relying on block.timestamp.

```
      function shouldDistribute(address shareholder) internal view
returns (bool) {
        return shareholderClaims[shareholder] + minPeriod <
block.timestamp
                && getUnpaidEarnings(shareholder) > minDistribution;
    }
```

🟡 Medium-risk

0 medium-risk code issues found.
Should be fixed, could bring problems.

🔴 High-risk

0 high-risk code issues found
Must be fixed, and will bring problems.

Extra notes by the team

🟢 Contract has taxes: 12% buy tax and 11% sell tax

🟡 Functions that are not used could be removed.

🟡 The owner of this smart-contract can modify the trading fees of the token.

🟡 The ownership is not renounced.

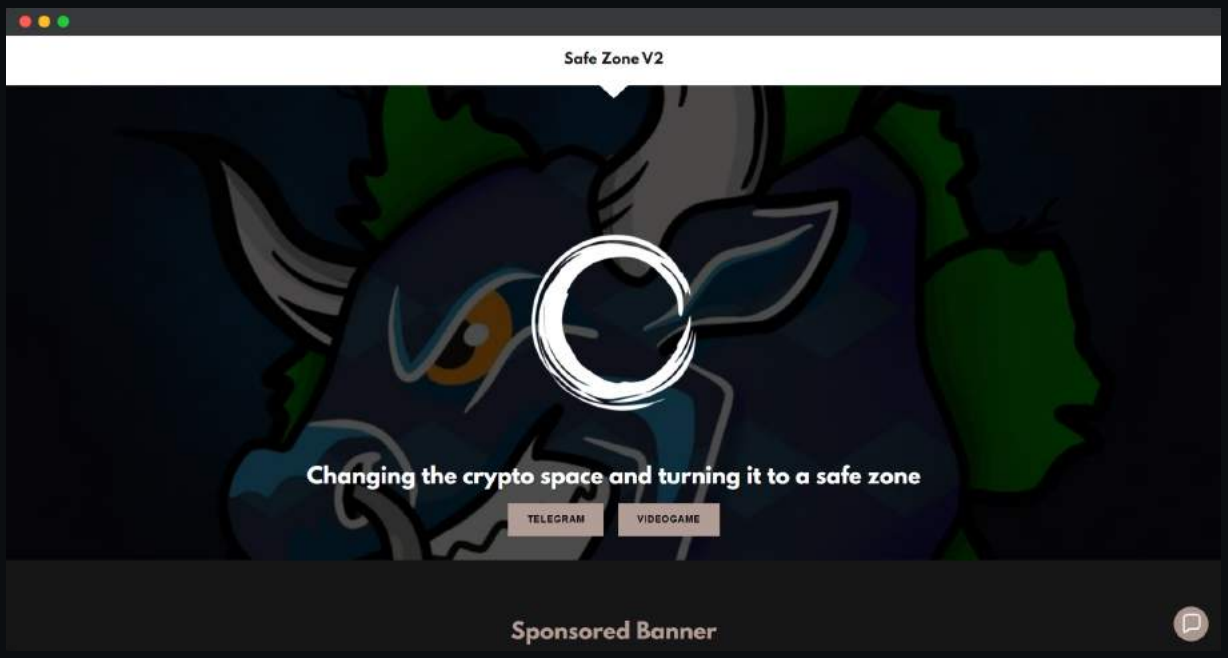
🟡 The owner can exclude from fees.

🔴 Fees can be set up to 50%

Contract Snapshot

```
contract safezonev2 is IBEP20, Auth {
    using SafeMath for uint256;

    address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c;
    address DEAD = 0x000000000000000000000000000000000000dEaD;
    address ZERO = 0x0000000000000000000000000000000000000000;

    string constant _name = "SAFE ZONE V2";
    string constant _symbol = "SAFU";
    uint8 constant _decimals = 9;

    uint256 _totalSupply = 10 * 10**9 * 10**_decimals;

    uint256 public _maxTxAmount = _totalSupply.mul(1).div(100);
    uint256 public _maxWalletToken = _totalSupply.mul(1).div(50);

    mapping (address => uint256) _balances;
    mapping (address => mapping (address => uint256)) _allowances;

    bool public blacklistMode = true;
    mapping (address => bool) public isBlacklisted;


    mapping (address => bool) isFeeExempt;
    mapping (address => bool) isTxLimitExempt;
    mapping (address => bool) isTimelockExempt;
    mapping (address => bool) isDividendExempt;

    uint256 private liquidityFee    = 99;
    uint256 private reflectionFee   = 0;
    uint256 private marketingFee    = 0;
    uint256 private ecosystemfee    = 0;
    uint256 private burnFee         = 0;
    uint256 private totalFee        = marketingFee + reflectionFee +
liquidityFee + ecosystemfee + burnFee;
    uint256 private feeDenominator  = 100;

    uint256 private sellMultiplier  = 100;

    address private autoLiquidityReceiver;
```

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
- 🟢 Contains no jQuery errors
- 🔴 Not SSL Secured
- 🟢 No major spelling errors

Note: No SSL certificate, do not connect wallet or enter your details

Loading speed: 85%

Rug-pull Review

Based on the available information analyzed by us, we come to the following conclusions:

- 🟢 Locked Liquidity
- 🟢 No large unlocked wallets
- 🟢 Doxxed Team (KYC)

Honeypot Review

Based on the available information analyzed by us, we come to the following conclusions:

- 🟢 Ability to sell
- 🟢 Owner not able to prevent selling
- 🟢 Accurate liquidity pair

Note: Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by the project owner.