BlockSAFU

ADVANCE MANUAL
SMART CONTRACT AUDIT

Project: ApeWar

Website: https://apewartoken.com/

BlockSAFU Score:

0

Contract Address:

0xC2b54Fac4F2504Df6b55DE474892910f47BCd754

DISCLAMER

BlockSAFU has completed this report to provide a summary of the Smart Contract functions, and any security, dependency, or cybersecurity vulnerabilities. This is often a constrained report on our discoveries based on our investigation and understanding of the current programming versions as of this report's date. To understand the full scope of our analysis, it is vital for you to at the date of this report. To understand the full scope of our analysis, you need to review the complete report. Although we have done our best in conducting our investigation and creating this report, it is vital to note that you should not depend on this report and cannot make any claim against BlockSAFU or its Subsidiaries and Team members on the premise of what has or has not been included in the report. Please remember to conduct your independent examinations before making any investment choices. We do not provide investment advice or in any way claim to determine if the project will be successful or not.

By perusing this report or any portion of it, you concur to the terms of this disclaimer. In the unlikely situation where you do not concur to the terms, you should immediately terminate reading this report, and erase and discard any duplicates of this report downloaded and/or printed by you. This report is given for data purposes as it were and on a non-reliance premise and does not constitute speculation counsel. No one should have any right to depend on the report or its substance, and BlockSAFU and its members (including holding companies, shareholders, backups, representatives, chiefs, officers, and other agents) BlockSAFU and its subsidiaries owe no obligation of care towards you or any other person, nor does BlockSAFU make any guarantee or representation to any individual on the precision or completeness of the report.

ABOUT THE AUDITOR:

BlockSAFU (BSAFU) is an Anti-Scam Token Utility that reviews Smart Contracts and Token information to Identify Rug Pull and Honey Pot scamming activity. BlockSAFUs Development Team consists of several Smart Contract creators, Auditors Developers, and Blockchain experts. BlockSAFU provides solutions, prevents, and hunts down scammers. BSAFU is a utility token with features Audit, KYC, Token Generators, and Bounty Scammers. It will enrich the crypto ecosystem.

OVERVIEW

Mint Function

- No mint functions.

Fees

- Buy 0% (owner can’t set fees).
- Sell 0% (owner can’t set fees).

Tx Amount

- Owner can’t set max tx amount.

Transfer Pausable

- Owner cannot pause.

Blacklist

- Owner can’t set blacklist.

Ownership

- Owner cannot take back ownership.

Proxy

- This contract has no proxy.

Anti Whale

- Owner can’t limit the number of wallet holdings.

Trading Cooldown

- Owner can’t set the selling time interval.

Owner can take balance from holder

SMART CONTRACT REVIEW

Token Name	**APEWAR**
Token Symbol	**APW**
Token Decimal	18
Total Supply	1,000,000,000 **APW**
Contract Address	0xC2b54Fac4F2504Df6b55DE474892910f47BCd754
Deployer Address	0x33713B87baB352C46BBa4953ab6Cb11aFe895d93
Owner Address	0x33713B87baB352C46BBa4953ab6Cb11aFe895d93
Tax Fees Buy	0%
Tax Fees Sell	0%
Gas Used for Buy	*will be update after dex listing*
Gas Used for Sell	*will be update after dex listing*
Contract Created	Sep-28-2022 03:19:31 PM +UTC
Initial Liquidity	*will be update after dex listing*
Liquidity Status	Locked
Unlocked Date	*will be update after dex listing*
Verified CA	Yes
Compiler	v0.8.7+commit.e28d00a7
Optimization	Yes with 200 runs
Sol License	MIT License
Other	default evmVersion

TAX

BUY	0%	address	**SELL**	0%
Buy Fee	0%		Sell Fee	0%

Token Holder

Rank	Address	Quantity	Percentage	Analytics
1	Pinksale: PinkLock V2	690,000,000	69.0000%	
2	0x5f9c70e767c7a39081c5990a93b5e6269ac930bd	300,000,960	30.0001%	
3	0x33713b87bab352c46bba4953ab6cb11afe895d93	9,999,040	0.9999%	

[Download CSV Export]

Team Review

The APEWAR team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 3,641 people in their telegram group (count in audit date).

Official Website And Social Media

Website: https://apewartoken.com/

Telegram Group: https://t.me/Apewarofficial

Twitter: https://twitter.com/apewartoken

MANUAL CODE REVIEW

🟢 Minor-risk

1 minor-risk code issue found

Could be fixed, and will not bring problems.

1. The return value of an external transfer/transferFrom return value is checked. Recommendation: use SafeERC20, or ensure that the transfer/transferFrom return value is checked

```
function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) external returns (bool);
```

🟡 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 problem.

Critical-Risk

1 critical-risk code issues found

Must be fixed, and will bring problem.

1. **Owner can take balance from wallet address holder**

   Recommendation: Remove this access

   Risk Scenario: Owner can take balance holder

```
contract HasForeignAsset is Ownable {

    function assetBalance(IBEP20 asset) external view
returns(uint256) {
        return asset.balanceOf(address(this));
    }

    function getAsset(IBEP20 asset) external onlyOwner {
        asset.transfer(owner(), this.assetBalance(asset));
    }
}
```

EXTRA NOTES SMART CONTRACT

1. IBEP20

```
interface IBEP20 {
  /**
   * @dev Returns the number of tokens in existence.
   */
  function totalSupply() external view returns (uint256);
…
  function balanceOf(address account) external view returns (uint256);
…
  function transfer(address recipient, uint256 amount) external returns (bool);
…
  function allowance(address owner, address spender) external view returns (uint256);
…
  function approve(address spender, uint256 amount) external returns (bool);
…
  function transferFrom(
    address sender,
    address recipient,
    uint256 amount
  ) external returns (bool);

  /**
   * @dev Emitted when `value` tokens are moved from one account (`from`) to
   * another (`to`).
   *
   * Note that `value` may be zero.
   */
  event Transfer(address indexed from, address indexed to, uint256 value);
…
}
```

IBEP20 Normal Base Template

2. SafeMath Contract

```
library SafeMath {
...
    function add(uint256 a, uint256 b) internal pure returns
(uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");
        return c;
    }
...
    function sub(uint256 a, uint256 b, string memory errorMessage)
internal pure returns (uint256) {
        require(b <= a, errorMessage);
        uint256 c = a - b;

        return c;
    }
    /**
     * @dev Returns the multiplication of two unsigned integers,
reverting on
     * overflow.
     *
     * Counterpart to Solidity's `*` operator.
     *
     * Requirements:
     *
     * - Multiplication cannot overflow.
     */
  ...
    function mod(
        uint256 a,
        uint256 b,
        string memory errorMessage
    ) internal pure returns (uint256) {
        unchecked {
            require(b > 0, errorMessage);
            return a % b;
        }
    }
}
```

Standard Safemath contract

3. APEWAR Contract

```
contract COB is IBEP20, HasForeignAsset {
    using SafeMath for uint256;
    using Address for address;

    mapping (address => uint256) private _balances;

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

    uint256 private _totalSupply;

    string private _name;
    string private _symbol;
    uint8 private _decimals;

    /**
     * @dev Sets the values for {name} and {symbol}, initializes
{decimals} with
     * a default value of 18.
     */
    constructor () {
        _name = "APEWAR";
        _symbol = "APW";
        _decimals = 18;
        uint256 _maxSupply = 1000000000;
        _mintOnce(msg.sender, _maxSupply.mul(10 ** _decimals));
    }

    receive() external payable {
        revert();
    }

    /**
     * @dev Returns the name of the token.
     */
    function name() public view returns (string memory) {
        return _name;
    }

    /**
     * @dev Returns the symbol of the token, usually a shorter
```

```
version of the
     * name.
     */
    function symbol() public view returns (string memory) {
        return _symbol;
    }

    /**
     * @dev Returns the number of decimals used to get its user
representation.
     * For example, if `decimals` equals `2`, a balance of `505`
tokens should
     * be displayed to a user as `5,05` (`505 / 10 ** 2`).
     *
     * Tokens usually opt for a value of 18, imitating the
relationship between
     * Ether and Wei. This is the value {BEP20} uses, unless
{_setupDecimals} is
     * called.
     *
     * NOTE: This information is only used for _display_ purposes:
it in
     * no way affects any of the arithmetic of the contract,
including
     * {IBEP20-balanceOf} and {IBEP20-transfer}.
     */
    function decimals() public view returns (uint8) {
        return _decimals;
    }

    /**
     * @dev See {IBEP20-totalSupply}.
     */
    function totalSupply() public view override returns (uint256)
{
        return _totalSupply;
    }

    /**
     * @dev See {IBEP20-balanceOf}.
     */
    function balanceOf(address account) public view override
```

```
returns (uint256) {
        return _balances[account];
    }

    /**
     * @dev See {IBEP20-transfer}.
     *
     * Requirements:
     *
     * - `recipient` cannot be the zero address.
     * - the caller must have a balance of at least `amount`.
     */
    function transfer(address recipient, uint256 amount) public
virtual override returns (bool) {
        _transfer(_msgSender(), recipient, amount);
        return true;
    }

    /**
     * @dev See {IBEP20-allowance}.
     */
    function allowance(address owner, address spender) public view
virtual override returns (uint256) {
        return _allowances[owner][spender];
    }

    /**
     * @dev See {IBEP20-approve}.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     */
    function approve(address spender, uint256 amount) public
virtual override returns (bool) {
        _approve(_msgSender(), spender, amount);
        return true;
    }

    /**
     * @dev See {IBEP20-transferFrom}.
     *
```

```
     * Emits an {Approval} event indicating the updated allowance.
This is not
     * required by the EIP. See the note at the beginning of
{BEP20};
     *
     * Requirements:
     * - `sender` and `recipient` cannot be the zero address.
     * - `sender` must have a balance of at least `amount`.
     * - the caller must have allowance for ``sender``'s tokens of
at least
     * `amount`.
     */
    function transferFrom(address sender, address recipient,
uint256 amount) public virtual override returns (bool) {
        _transfer(sender, recipient, amount);
        _approve(sender, _msgSender(),
_allowances[sender][_msgSender()].sub(amount, "BEP20: transfer
amount exceeds allowance"));
        return true;
    }

    /**
     * @dev Atomically increases the allowance granted to
`spender` by the caller.
     *
     * This is an alternative to {approve} that can be used as a
mitigation for
     * problems described in {IBEP20-approve}.
     *
     * Emits an {Approval} event indicating the updated allowance.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     */
    function increaseAllowance(address spender, uint256
addedValue) public virtual returns (bool) {
        _approve(_msgSender(), spender,
_allowances[_msgSender()][spender].add(addedValue));
        return true;
    }
```

```
    /**
     * @dev Atomically decreases the allowance granted to
`spender` by the caller.
     *
     * This is an alternative to {approve} that can be used as a
mitigation for
     * problems described in {IBEP20-approve}.
     *
     * Emits an {Approval} event indicating the updated allowance.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     * - `spender` must have allowance for the caller of at least
     * `subtractedValue`.
     */
    function decreaseAllowance(address spender, uint256
subtractedValue) public virtual returns (bool) {
        _approve(_msgSender(), spender,
_allowances[_msgSender()][spender].sub(subtractedValue, "BEP20:
decreased allowance below zero"));
        return true;
    }

    /**
     * @dev Moves tokens `amount` from `sender` to `recipient`.
     *
     * This is internal function is equivalent to {transfer}, and
can be used to
     * e.g. implement automatic token fees, slashing mechanisms,
etc.
     *
     * Emits a {Transfer} event.
     *
     * Requirements:
     *
     * - `sender` cannot be the zero address.
     * - `recipient` cannot be the zero address.
     * - `sender` must have a balance of at least `amount`.
     */
    function _transfer(address sender, address recipient, uint256
amount) internal virtual {
```

```
        require(sender != address(0), "BEP20: transfer from the
zero address");
        require(recipient != address(0), "BEP20: transfer to the
zero address");

        _beforeTokenTransfer(sender, recipient, amount);

        _balances[sender] = _balances[sender].sub(amount, "BEP20:
transfer amount exceeds balance");
        _balances[recipient] = _balances[recipient].add(amount);
        emit Transfer(sender, recipient, amount);
    }

    /** @dev Creates `amount` tokens and assigns them to
`account`, increasing
     * the total supply.
     *
     * Emits a {Transfer} event with `from` set to the zero
address.
     *
     * Requirements
     *
     * - `to` cannot be the zero address.
     */
    function _mintOnce(address account, uint256 amount) internal
virtual {
        require(account != address(0), "BEP20: mint to the zero
address");

        _beforeTokenTransfer(address(0), account, amount);

        _totalSupply = _totalSupply.add(amount);
        _balances[account] = _balances[account].add(amount);
        emit Transfer(address(0), account, amount);
    }

    /**
     * @dev Destroys `amount` tokens from `account`, reducing the
     * total supply.
     *
     * Emits a {Transfer} event with `to` set to the zero address.
     *
```

```
     * Requirements
     *
     * - `account` cannot be the zero address.
     * - `account` must have at least `amount` tokens.
     */
    function _burn(address account, uint256 amount) internal
virtual {
        require(account != address(0), "BEP20: burn from the zero
address");

        _beforeTokenTransfer(account, address(0), amount);

        _balances[account] = _balances[account].sub(amount,
"BEP20: burn amount exceeds balance");
        _totalSupply = _totalSupply.sub(amount);
        emit Transfer(account, address(0), amount);
    }

    function burn(uint256 amount) public {
        _burn(_msgSender(), amount);
    }
    /**
     * @dev Sets `amount` as the allowance of `spender` over the
`owner` s tokens.
     *
     * This internal function is equivalent to `approve`, and can
be used to
     * e.g. set automatic allowances for certain subsystems, etc.
     *
     * Emits an {Approval} event.
     *
     * Requirements:
     *
     * - `owner` cannot be the zero address.
     * - `spender` cannot be the zero address.
     */
    function _approve(address owner, address spender, uint256
amount) internal virtual {
        require(owner != address(0), "BEP20: approve from the zero
address");
        require(spender != address(0), "BEP20: approve to the zero
address");
```

```
        _allowances[owner][spender] = amount;
        emit Approval(owner, spender, amount);
    }

    /**
     * @dev Sets {decimals} to a value other than the default one
of 18.
     *
     * WARNING: This function should only be called from the
constructor. Most
     * applications that interact with token contracts will not
expect
     * {decimals} to ever change, and may work incorrectly if it
does.
     */
    function _setupDecimals(uint8 decimals_) internal {
        _decimals = decimals_;
    }
    /**
     * @dev Hook that is called before any transfer of tokens.
This includes
     * minting and burning.
     *
     * Calling conditions:
     *
     * - when `from` and `to` are both non-zero, `amount` of
``from``'s tokens
     * will be to transferred to `to`.
     * - when `from` is zero, `amount` tokens will be minted for
`to`.
     * - when `to` is zero, `amount` of ``from``'s tokens will be
burned.
     * - `from` and `to` are never both zero.
     *
     * To learn more about hooks, head to
xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
     */
    function _beforeTokenTransfer(address from, address to,
uint256 amount) internal virtual { }
}t) internal virtual { }
}
```

4. Foreign Asset

```
contract HasForeignAsset is Ownable {

    function assetBalance(IBEP20 asset) external view
returns(uint256) {
        return asset.balanceOf(address(this));
    }

    function getAsset(IBEP20 asset) external onlyOwner {
        asset.transfer(owner(), this.assetBalance(asset));
    }
}
```

Owner can take balance from wallet address holder - Critical Risk

READ CONTRACT (ONLY NEED TO KNOW)

1. decimals
18 uint8
(Shows Contract buy cooldown status)

2. name
RYS string
(Function for read Token name)

3. owner
0x33713b87bab352c46bba4953ab6cb11afe895d93 address
(function for read owner address)

WRITE CONTRACT

1. renounceOwnership
(Renouncing ownership will leave the contract without an owner, thereby removing any functionality that is only available to the owner)

2. transferOwnership
newOwner (address)
(Its function is to change the owner)

3. burn
amount (uint256)
(its function for burn)

WEBSITE REVIEW

https://apewartoken.com/

- **Mobile Friendly**
- **Contains no code error**
- **SSL Secured (By n/a)**

Web-Tech stack: n/a

Domain .com (n/a) - Tracked by whois

First Contentful Paint:	n/as
Fully Loaded Time	n/as
Performance	n/a%
Accessibility	n/a%
Best Practices	n/a%
SEO	n/a%

RUG-PULL REVIEW

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

- Locked Liquidity (Locked by pinksale)

  will be updated after listing dex

- TOP 5 Holder.

  will be updated after listing dex

- The Team KYC by PinkSale

HONEYPOT REVIEW

- Ability to sell.
- The owner can't set fees over 25%
- The owner can't set max tx amount below 0.1% of total supply
- Owner can take balance from wallet address holder

Note: Please check the disclaimer above and note, that the audit makes no statements or warranties on the 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.