BlockSAFU

ADVANCE MANUAL
SMART CONTRACT AUDIT

Project: ShiroGo

Website: shirogo.org

BlockSAFU Score: 47

Contract Address:

0x4736ECC20c8ccec146e2084B58A2e5614302cbFF

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

BlockSAFU was commissioned by ShiroGo Finance Token to complete a Smart Contract audit. The objective of the Audit is to achieve the following:

- Review the Project and experience and Development team
- Ensure that the Smart Contract functions are necessary and operate as intended.
- Identify any vulnerabilities in the Smart Contract code.

DISCLAIMER: This Audit is intended to inform about token Contract Risks, the result does not imply an endorsement or provide financial advice in any way, all investments are made at your own risk. (https://blocksafu.com/)

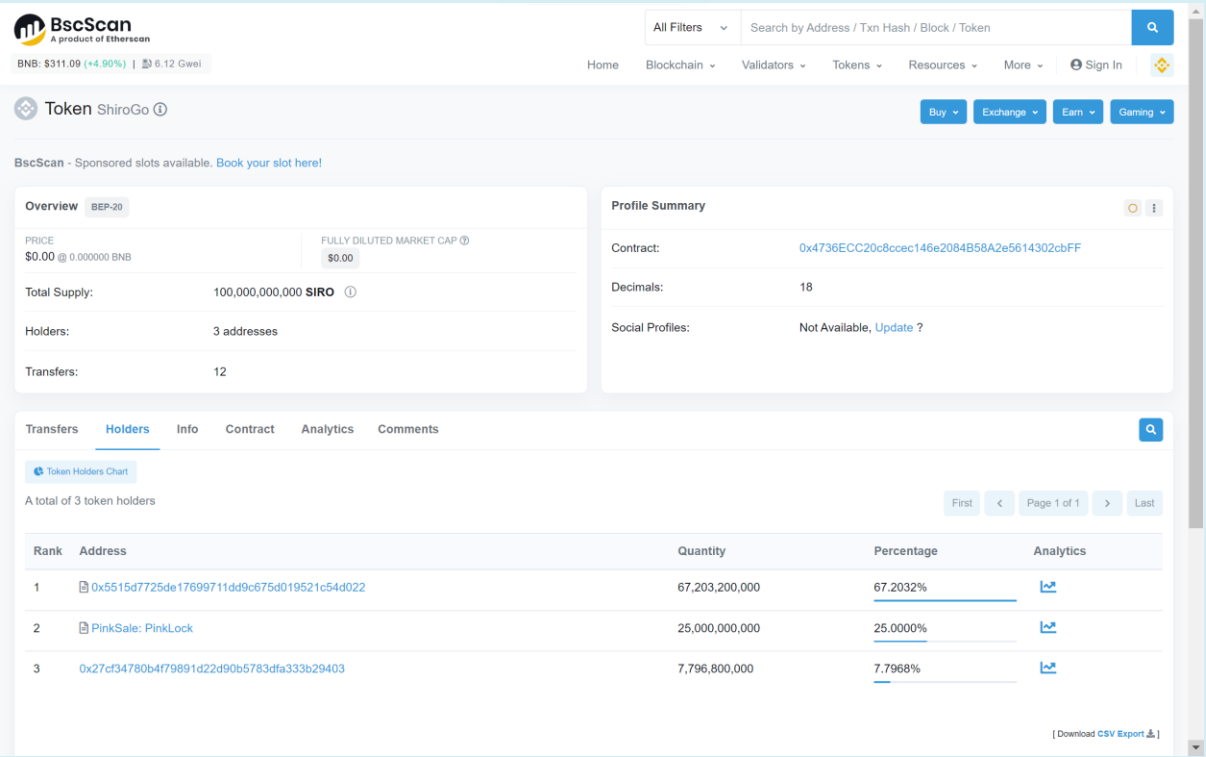
SMART CONTRACT REVIEW

Token Name	ShiroGo
Token Symbol	SHIRO
Token Decimal	18
Total Supply	100,000,000,000 **CTY**
Contract Address	0x4736ECC20c8ccec146e2084B58A2e5614302cbFF
Deployer Address	0x27cf34780b4f79891d22d90b5783dfa333b29403
Owner Address	0x27cf34780b4f79891d22d90b5783dfa333b29403
Tax Fees Buy	5%
Tax Fees Sell	5%
Gas Used for Buy	*will be updated after the DEX listing*
Gas Used for Sell	*will be updated after the DEX listing*
Contract Created	May-14-2022 10:13:48 AM +UTC
Initial Liquidity	*will be updated after the DEX listing*
Liquidity Status	Locked
Unlocked Date	*will be updated after the DEX listing*
Verified CA	Yes
Compiler	v0.6.12+commit.27d51765
Optimization	Disable with 200 runs
Sol License	MIT License
Top 5 Holders	*will be updated after the DEX listing*
Other	default evmVersion

TAX

BUY	5%	**SELL**	5%
Fees	2%	Fees	2%
LP Liquidity	3%	LP Liquidity	3%

TOP HOLDER



Team Review

The ShiroGo team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 6,468 people in their telegram group (count in audit date).

OFFICIAL WEBSITE AND SOCIAL MEDIA

Website: http://shirogo.org/

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

Whitepaper: https://drive.google.com/file/d/19v-amyrZS-e31F2AqUskbpwCmZwLrILw/view

Twitter: https://twitter.com/ShiroGoApp

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/transfer from the return value is checked.
   Recommendation: use SafeERC20, or ensure that the transfer/transfers from return value are checked

```
function transferFrom(address sender, address recipient, uint256 amount) public override
returns (bool) {
        _transfer(sender, recipient, amount);
        _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount,
"ERC20: transfer amount exceeds allowance"));
        return true;
    }
```

🟡 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

2 critical-risk code issues found

Must be fixed, and will bring problem.

1. The owner can set tax fees and liquidity fees up to 100%.

```
function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
        _taxFee = taxFee;
    }

function setLiquidityFeePercent(uint256 liquidityFee) external
onlyOwner() {
        _liquidityFee = liquidityFee;
    }
```

2. The owner can set maximum transaction without minimum limit.

```
function setMaxTxPercent(uint256 maxTxPercent) external
onlyOwner() {
        _maxTxAmount = _tTotal.mul(maxTxPercent).div(
            10**2
        );
    }
```

EXTRA NOTES SMART CONTRACT

1. IERC20

```
interface IERC20 {

    function totalSupply() external view returns (uint256);

    /**
     * @dev Returns the amount of tokens owned by `account`.
     */
    function balanceOf(address account) external view returns (uint256);

    /**
     * @dev Moves `amount` tokens from the caller's account to `recipient`.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transfer(address recipient, uint256 amount) external returns
(bool);

    /**
     * @dev Returns the remaining number of tokens that `spender` will be
     * allowed to spend on behalf of `owner` through {transferFrom}. This is
     * zero by default.
     *
     * This value changes when {approve} or {transferFrom} are called.
     */
    function allowance(address owner, address spender) external view returns
(uint256);

    /**
     * @dev Sets `amount` as the allowance of `spender` over the caller's
tokens.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * IMPORTANT: Beware that changing an allowance with this method brings the
risk
     * that someone may use both the old and the new allowance by unfortunate
     * transaction ordering. One possible solution to mitigate this race
     * condition is to first reduce the spender's allowance to 0 and set the
     * desired value afterwards:
     * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
     *
     * Emits an {Approval} event.
     */
    function approve(address spender, uint256 amount) external returns (bool);

    /**
     * @dev Moves `amount` tokens from `sender` to `recipient` using the
     * allowance mechanism. `amount` is then deducted from the caller's
     * allowance.
     *
```

```
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * Emits a {Transfer} event.
     */
    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);

    /**
     * @dev Emitted when the allowance of a `spender` for an `owner` is set by
     * a call to {approve}. `value` is the new allowance.
     */
    event Approval(address indexed owner, address indexed spender, uint256
value);
}
```

IERC20 Normal Base Template

2. SafeMath Library

```
library SafeMath {
    /**
     * @dev Returns the addition of two unsigned integers,
reverting on
     * overflow.
     *
     * Counterpart to Solidity's `+` operator.
     *
     * Requirements:
     *
     * - Addition cannot overflow.
     */
    function add(uint256 a, uint256 b) internal pure returns
(uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");

        return c;
    }

    /**
     * @dev Returns the subtraction of two unsigned integers,
reverting on
     * overflow (when the result is negative).
     *
     * Counterpart to Solidity's `-` operator.
```

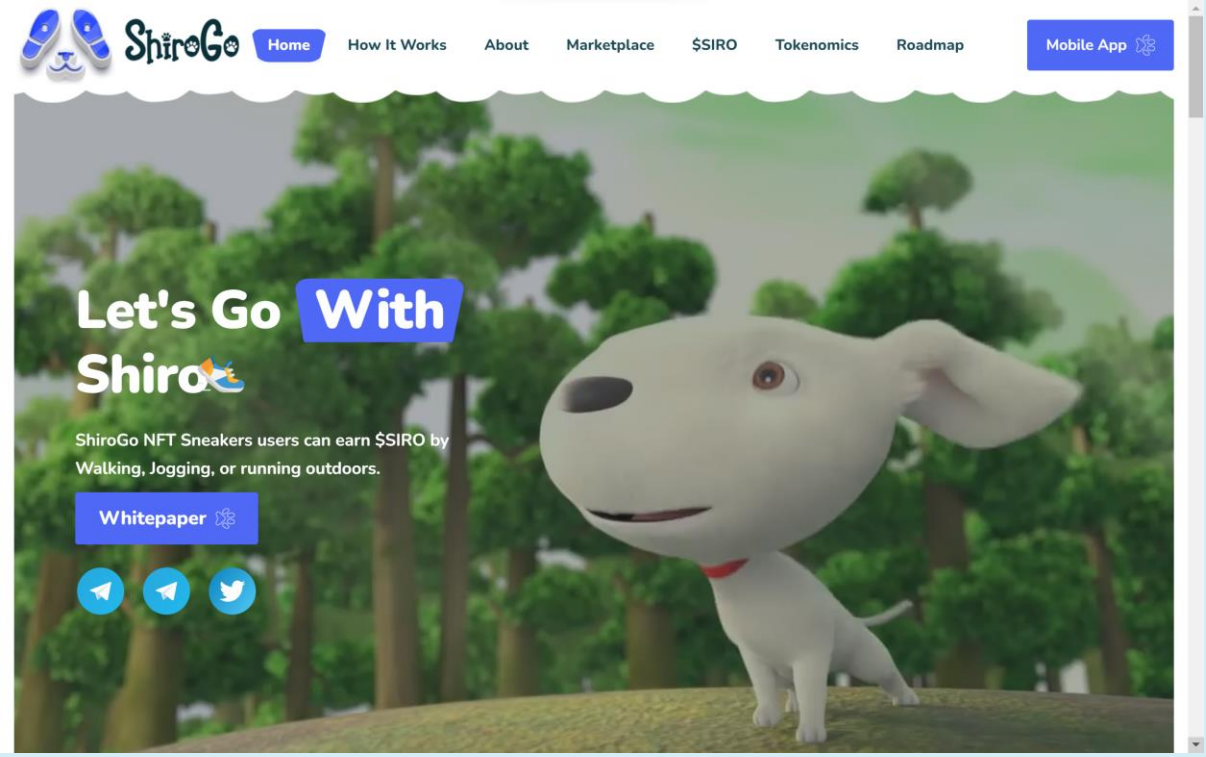
```
     *
     * Requirements:
     *
     * - Subtraction cannot overflow.
     */
    function sub(uint256 a, uint256 b) internal pure returns
(uint256) {
        return sub(a, b, "SafeMath: subtraction overflow");
    }

    /**
     * @dev Returns the subtraction of two unsigned integers,
reverting with custom message on
     * overflow (when the result is negative).
     *
     * Counterpart to Solidity's `-` operator.
     *
     * Requirements:
     *
     * - Subtraction cannot overflow.
     */
    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 mul(uint256 a, uint256 b) internal pure returns
(uint256) {
        // Gas optimization: this is cheaper than requiring 'a'
not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-
contracts/pull/522
        if (a == 0) {
            return 0;
```

```
        }

        uint256 c = a * b;
        require(c / a == b, "SafeMath: multiplication overflow");

        return c;
    }

    /**
     * @dev Returns the integer division of two unsigned integers.
Reverts on
     * division by zero. The result is rounded towards zero.
     *
     * Counterpart to Solidity's `/` operator. Note: this function
uses a
     * `revert` opcode (which leaves remaining gas untouched)
while Solidity
     * uses an invalid opcode to revert (consuming all remaining
gas).
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function div(uint256 a, uint256 b) internal pure returns
(uint256) {
        return div(a, b, "SafeMath: division by zero");
    }

    /**
     * @dev Returns the integer division of two unsigned integers.
Reverts with custom message on
     * division by zero. The result is rounded towards zero.
     *
     * Counterpart to Solidity's `/` operator. Note: this function
uses a
     * `revert` opcode (which leaves remaining gas untouched)
while Solidity
     * uses an invalid opcode to revert (consuming all remaining
gas).
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function div(uint256 a, uint256 b, string memory errorMessage)
internal pure returns (uint256) {
        require(b > 0, errorMessage);
        uint256 c = a / b;
```

```
        // assert(a == b * c + a % b); // There is no case in
which this doesn't hold

        return c;
    }

    /**
     * @dev Returns the remainder of dividing two unsigned
integers. (unsigned integer modulo),
     * Reverts when dividing by zero.
     *
     * Counterpart to Solidity's `%` operator. This function uses
a `revert`
     * opcode (which leaves remaining gas untouched) while
Solidity uses an
     * invalid opcode to revert (consuming all remaining gas).
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function mod(uint256 a, uint256 b) internal pure returns
(uint256) {
        return mod(a, b, "SafeMath: modulo by zero");
    }

    /**
     * @dev Returns the remainder of dividing two unsigned
integers. (unsigned integer modulo),
     * Reverts with custom message when dividing by zero.
     *
     * Counterpart to Solidity's `%` operator. This function uses
a `revert`
     * opcode (which leaves remaining gas untouched) while
Solidity uses an
     * invalid opcode to revert (consuming all remaining gas).
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function mod(uint256 a, uint256 b, string memory errorMessage)
internal pure returns (uint256) {
        require(b != 0, errorMessage);
        return a % b;
    }
}
```

Normal SafeMath Library

WEBSITE REVIEW



- **Mobile Friendly**
- **Contains no code error**
- **SSL Secured (By zero SSL)**

Web-Tec stack: jQuery, Web Flow, LiteSpeed

Domain .com - (register.it) - Tracked by whois

First Contentful Paint:	1.4s
Fully Loaded Time	6.5s
Performance	87%
Accessibility	91%
Best Practices	92%
SEO	100%

RUG-PULL REVIEW

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

- Locked Liquidity by PinkLock

  (*Will be updated after DEX listing*)

- TOP 5 Holder

  (*Will be updated after DEX listing*)

- KYC by PinkSale

HONEYPOT REVIEW

- Ability to sell
- The owner is not able to pause the contract
- The owner can set buy fees up to 100%
- The owner can set the maximum transaction amount to 0

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.