BlockSAFU

ADVANCE MANUAL
SMART CONTRACT AUDIT

Project: Starfleet

Website: https://project21.app/

BlockSAFU Score:

49

Contract Address:

0xb52d7FA82B718a833395caE180Dd0B8501fBe895

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 Starfleet 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	**Starfleet**
Token Symbol	**STARFLEET**
Token Decimal	9
Total Supply	21,000,000 **STARFLEET**
Contract Address	0xb52d7FA82B718a833395caE180Dd0B8501fBe895
Deployer Address	0xd5Fd602838E26d67176ED78cA15b5e3793212121
Owner Address	0xd5fd602838e26d67176ed78ca15b5e3793212121
Tax Fees Buy	2%
Tax Fees Sell	10%
Gas Used for Buy	*will be updated after the DEX listing*
Gas Used for Sell	*will be updated after the DEX listing*
Contract Created	Aug-23-2022 12:44:55 PM +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	v0.8.13+commit.abaa5c0e
Optimization	No with 200 runs
Sol License	MIT License
Top 5 Holders	*will be updated after the DEX listing*
Other	default evmVersion

TAX

BUY	2%	**SELL**	10%
buyFeeRate	2%	sellFeeRate	10%

OVERVIEW

Mint Function

- No mint functions.

Fees

- Buy 2% (owner can set fees up to 30%).
- Sell 10% (owner can set fees up to 14%).

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.
- Owner can set hidden owner

Proxy

- This contract has no proxy.

Anti Whale

- Owner cannot limit the number of wallet holdings.

Trading Cooldown

- Owner cannot set the selling time interval.

Token Metric

Team Review

The team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 11,234 people in their telegram group (count in audit date).

Official Website And Social Media

Website: https://project21.app/

Telegram Group: https://t.me/Project21_chat

Twitter: https://twitter.com/Project21_bsc

MANUAL CODE REVIEW

Minor-risk

2 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);
```

2. Does not include hidden owners, but this method is difficult for investors to see who is authorized.

```
function isAuthorized(address adr) public view returns (bool) {
        return authorizations[adr];
    }
```

Medium-risk

1 medium-risk code issues found

Should be fixed, could bring problems.

1. The owner can't set fees buy over 30% -medium risk, and can't set fees sell over 14%

```
function setFee (uint256 _sellFeeRate, uint256 _buyFeeRate) external onlyOwner {
        require (_sellFeeRate <= 14, "Fee can't exceed 14%");
        require (_buyFeeRate <= 30, "Fee can't exceed 30%");
        sellFeeRate = _sellFeeRate;
        buyFeeRate = _buyFeeRate;
    }
```

High-Risk

1 high-risk code issues found

Must be fixed, and will bring problem.

Owner can set hidden owner

```
function authorize(address adr) public onlyOwner {
        authorizations[adr] = true;
    }
```

Critical-Risk

0 critical-risk code issues found

Must be fixed, and will bring problem.

EXTRA NOTES SMART CONTRACT

1. IERC20

```
interface IERC20 {
    function totalSupply() external view returns (uint256);
    function decimals() external view returns (uint8);
    function symbol() external view returns (string memory);
    function name() external view returns (string memory);
    function getOwner() external view returns (address);
    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);
    event Transfer(address indexed from, address indexed to,
uint256 value);
    event Approval(address indexed owner, address indexed spender,
uint256 value);
}
```

IERC20 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. Starfleet Contract

```
contract STARFLEET is IERC20, Auth {
    using SafeMath for uint256;

    address private WETH;
    address private DEAD =
0x000000000000000000000000000000000000dEaD;
    address private ZERO =
0x0000000000000000000000000000000000000000;

    string private constant  _name = "STARFLEET";
    string private constant _symbol = "STARFLEET";
    uint8 private constant _decimals = 9;

    uint256 private _totalSupply = 21000000 * (10 ** _decimals);

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

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

    uint256 public buyFeeRate = 2;
    uint256 public sellFeeRate = 10;

    uint256 private feeDenominator = 100;

    address payable public marketingWallet =
payable(0x3d1fdF0B65425ED8C126617F2EA3d23502686d8d);

    IDEXRouter public router;
    address public pair;

    uint256 public launchedAt;
    bool private tradingOpen;

    //max buy of 0.5%
    uint256 private maxBuyTransaction = ( _totalSupply * 5 ) /
1000;
    uint256 public numTokensSellToAddToLiquidity = 100000 * 10**9;
```

```
    bool private inSwap;

    modifier swapping() { inSwap = true; _; inSwap = false; }

    constructor () Auth(msg.sender) {
        router =
IDEXRouter(0x10ED43C718714eb63d5aA57B78B54704E256024E);

        WETH = router.WETH();

        pair = IDEXFactory(router.factory()).createPair(WETH,
address(this));

        _allowances[address(this)][address(router)] =
type(uint256).max;

        isTxLimitExempt[msg.sender] = true;

        isFeeExempt[msg.sender] = true;
        isFeeExempt[marketingWallet] = true;

        isTimelockExempt[msg.sender] = true;
        isTimelockExempt[DEAD] = true;
        isTimelockExempt[address(this)] = true;

        _balances[msg.sender] = _totalSupply;

        emit Transfer(address(0), msg.sender, _totalSupply);
    }

    receive() external payable { }

    function totalSupply() external view override returns
(uint256) { return _totalSupply; }
    function decimals() external pure override returns (uint8) {
return _decimals; }
    function symbol() external pure override returns (string
memory) { return _symbol; }
    function name() external pure override returns (string memory)
{ return _name; }
    function getOwner() external view override returns (address) {
```

```
return owner; }
    function balanceOf(address account) public view override
returns (uint256) { return _balances[account]; }
    function allowance(address holder, address spender) external
view override returns (uint256) { return
_allowances[holder][spender]; }

    function approve(address spender, uint256 amount) public
override returns (bool) {
        _allowances[msg.sender][spender] = amount;
        emit Approval(msg.sender, spender, amount);
        return true;
    }

    function approveMax(address spender) external returns (bool) {
        return approve(spender, type(uint256).max);
    }

    function transfer(address recipient, uint256 amount) external
override returns (bool) {
        return _transferFrom(msg.sender, recipient, amount);
    }

    function transferFrom(address sender, address recipient,
uint256 amount) external override returns (bool) {
        if(_allowances[sender][msg.sender] != type(uint256).max){
            _allowances[sender][msg.sender] =
_allowances[sender][msg.sender].sub(amount, "Insufficient
Allowance");
        }

        return _transferFrom(sender, recipient, amount);
    }

    function _transferFrom(address sender, address recipient,
uint256 amount) internal returns (bool) {
        if(!authorizations[sender] && !authorizations[recipient]){
            require(tradingOpen, "Trading not yet enabled.");
        }

        // Checks max transaction limit
        checkTxLimit(sender, amount);
```

```
        if(inSwap){ return _basicTransfer(sender, recipient,
amount); }

        uint256 contractTokenBalance = balanceOf(address(this));

        bool overMinTokenBalance = contractTokenBalance >=
numTokensSellToAddToLiquidity;

        bool shouldSwapBack = (overMinTokenBalance &&
recipient==pair && balanceOf(address(this)) > 0);
        if(shouldSwapBack){
swapBack(numTokensSellToAddToLiquidity); }

        _balances[sender] = _balances[sender].sub(amount,
"Insufficient Balance");

        uint256 amountReceived = shouldTakeFee(sender, recipient)
? takeFee(sender, recipient, amount) : amount;

        _balances[recipient] =
_balances[recipient].add(amountReceived);

        emit Transfer(sender, recipient, amountReceived);
        return true;
    }

    function _basicTransfer(address sender, address recipient,
uint256 amount) internal returns (bool) {
        _balances[sender] = _balances[sender].sub(amount,
"Insufficient Balance");
        _balances[recipient] = _balances[recipient].add(amount);
        emit Transfer(sender, recipient, amount);
        return true;
    }

    function checkTxLimit(address sender, uint256 amount) internal
view {
        require(amount <= maxBuyTransaction ||
isTxLimitExempt[sender], "TX Limit Exceeded");
    }
```

```
    function shouldTakeFee(address sender, address recipient)
internal view returns (bool) {
        return ( !(isFeeExempt[sender] || isFeeExempt[recipient])
&&  (sender == pair || recipient == pair) );
   }

    function takeFee(address sender, address recipient, uint256
amount) internal returns (uint256) {
        uint256 transferFeeRate = recipient == pair ? sellFeeRate
: buyFeeRate;
        uint256 feeAmount;
        feeAmount =
amount.mul(transferFeeRate).div(feeDenominator);
        _balances[address(this)] =
_balances[address(this)].add(feeAmount);
        emit Transfer(sender, address(this), feeAmount);

        return amount.sub(feeAmount);
    }


    function swapBack(uint256 amount) internal swapping {
        swapTokensForEth(amount);
    }


    function swapTokensForEth(uint256 tokenAmount) private {

        // generate the uniswap pair path of token -> weth
        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = WETH;

        // make the swap
        router.swapExactTokensForETHSupportingFeeOnTransferTokens(
            tokenAmount,
            0, // accept any amount of ETH
            path,
            marketingWallet,
            block.timestamp
        );
    }
```

```
    function swapToken() public onlyOwner {

        uint256 contractTokenBalance = balanceOf(address(this));

        bool overMinTokenBalance = contractTokenBalance >=
numTokensSellToAddToLiquidity;

        bool shouldSwapBack = (overMinTokenBalance &&
balanceOf(address(this)) > 0);
        if(shouldSwapBack){
            swapTokensForEth(numTokensSellToAddToLiquidity);
         }
    }

    function openTrade() external onlyOwner {
        launchedAt = block.number;
        tradingOpen = true;
    }

    function setIsFeeExempt(address holder, bool exempt) external
onlyOwner {
        isFeeExempt[holder] = exempt;
    }

    function setFee (uint256 _sellFeeRate, uint256 _buyFeeRate)
external onlyOwner {
        require (_sellFeeRate <= 14, "Fee can't exceed 14%");
        require (_buyFeeRate <= 30, "Fee can't exceed 30%");
        sellFeeRate = _sellFeeRate;
        buyFeeRate = _buyFeeRate;
    }

    function manualBurn(uint256 amount) external onlyOwner returns
(bool) {
        return _basicTransfer(address(this), DEAD, amount);
    }

    function getCirculatingSupply() public view returns (uint256)
{
        return
_totalSupply.sub(balanceOf(DEAD)).sub(balanceOf(ZERO));
```

```
    }

    function setMarketingWallet(address _marketingWallet) external
onlyOwner {
        marketingWallet = payable(_marketingWallet);
    }

    function setSwapThresholdAmount (uint256 amount) external
onlyOwner {
        require (amount <= _totalSupply.div(100), "can't exceed
1%");
        numTokensSellToAddToLiquidity = amount * 10 ** 9;
    }

    function setIsTxLimitExempt(address holder, bool exempt)
external onlyOwner {
        isTxLimitExempt[holder] = exempt;
    }

    function setIsTimelockExempt(address holder, bool exempt)
external onlyOwner {
        isTimelockExempt[holder] = exempt;
    }

    function clearStuckBalance(uint256 amountPercentage, address
adr) external onlyOwner {
        uint256 amountETH = address(this).balance;
        payable(adr).transfer(
            (amountETH * amountPercentage) / 100
        );
    }

    function rescueToken(address tokenAddress, uint256 tokens)
        public
        onlyOwner
        returns (bool success)
    {
        return IERC20(tokenAddress).transfer(msg.sender, tokens);
    }

}
```

4. Tax Fee contract

```
function setFee (uint256 _sellFeeRate, uint256 _buyFeeRate)
external onlyOwner {
        require (_sellFeeRate <= 14, "Fee can't exceed 14%");
        require (_buyFeeRate <= 30, "Fee can't exceed 30%");
        sellFeeRate = _sellFeeRate;
        buyFeeRate = _buyFeeRate;
    }
```

The owner can't set fees buy over 30% -medium risk, and can't set fees sell over 14%

7. Hidden Owner

```
function authorize(address adr) public onlyOwner {
        authorizations[adr] = true;
    }
```

Owner and authorized can swap before open trade. rugpull potential **-warning**

READ CONTRACT (ONLY NEED TO KNOW)

1. buyFeeRate
2 uint256
(Shows Contract buy fee rate)

2. decimals
9 uint8
(Shows Contract decimals)

3. getCirculatingSupply
21000000000000000 uint256
(Shows blacklistMode)

4. getOwner
0xd5fd602838e26d67176ed78ca15b5e3793212121 address
(Function for read owner)

5. name
STARFLEET string
(Function for read token name)

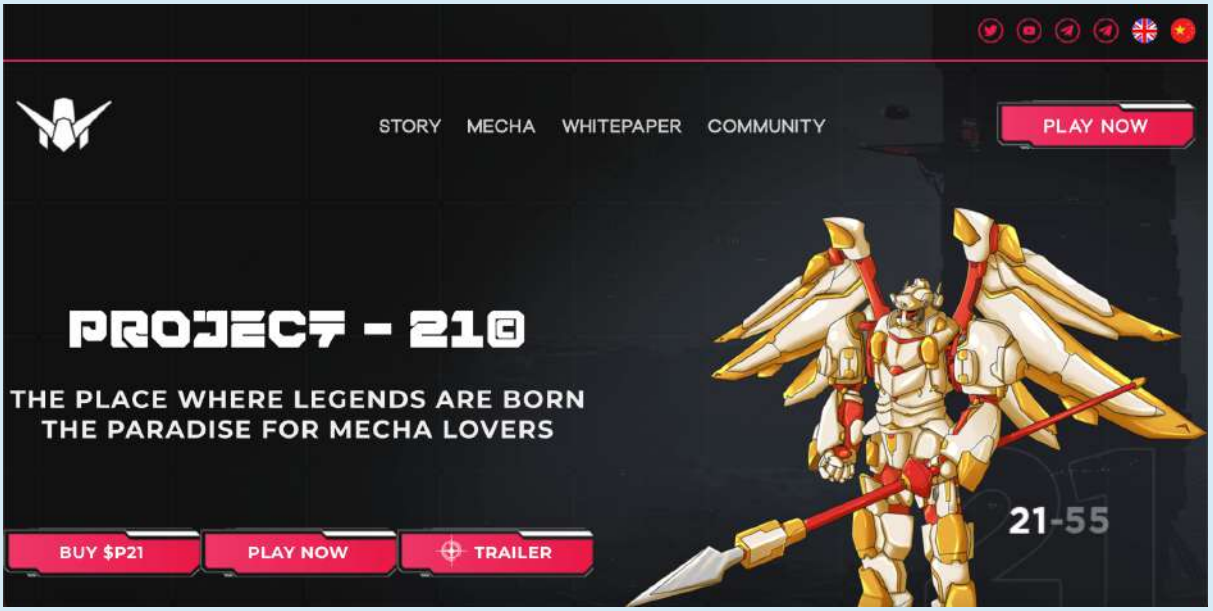
WRITE CONTRACT

1. authorize
adr address
(The form is filled with the address for set authorize to address) – warning

2. manualBurn
amount (uint256)
(The form is filled with the amount for manual burn)

3. transferOwnership
adr (address)
(Its function is to change the owner)

WEBSITE REVIEW



- **Mobile Friendly**
- **Contains no code error**
- **SSL Secured (By E1 SSL)**

Web-Tech stack: jQuery, Wordpress, Core-js

Domain .com (IONOS SE) - Tracked by whois

First Contentful Paint:	837ms
Fully Loaded Time	3.7s
Performance	41%
Accessibility	89%
Best Practices	67%
SEO	82%

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 DEX listing*)

- TOP 5 Holder.

  (*Will be updated after DEX listing*)

- The Team No KYC On Blocksafu

- Owner and authorized can swap before open trade. rugpull potential

HONEYPOT REVIEW

- Ability to sell.
- The owner is not able to pause the contract.
- The owner can set fees buy up to 30%

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.