BlockSAFU

ADVANCE MANUAL
SMART CONTRACT AUDIT

Project: BNOU Flexible Pool

Website: https://bitnou.com/

PASSED

BlockSAFU Score:

97

Contract Address:

0x64f88DEBdae30E8e4479C36cffB63f06e12c3154

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 with 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% (No fees).
- Sell 0% (No fees).

Tx Amount

- Owner cannot set a max tx amount.

Transfer Pausable

- Owner cannot pause.

Blacklist

- Owner cannot blacklist.

Ownership

- Owner cannot take back ownership.

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.

SMART CONTRACT REVIEW

Token Name	**BnouFlexiblePool**
Contract Address	0x64f88DEBdae30E8e4479C36cffB63f06e12c3154
Deployer Address	0x6b2a856A8954aa86eA66f9729597f4078D03e7a9
Owner Address	0x47a4ea43c6cf05e2541a76903f06d4b24fa4cc81
Gas Used for Buy	*will be updated after the DEX listing*
Gas Used for Sell	*will be updated after the DEX listing*
Contract Created	Sep-06-2022 03:40:16 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	v0.8.15+commit.e14f2714
Optimization	Yes with 200 runs
Sol License	MIT License
Top 5 Holders	*will be updated after the DEX listing*
Other	default evmVersion

Team Review

The Bnou team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 18 people in their telegram group (count in audit date).

Official Website And Social Media

Website: https://bitnou.com/

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

Discord: https://discord.com/invite/5Qb4bM7zYA

MANUAL CODE REVIEW

Minor-risk

0 minor-risk code issue found

Could be fixed, and will not bring problems.

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.

Critical-Risk

0 critical-risk code issues found

Must be fixed, and will bring problems.

EXTRA NOTES SMART CONTRACT

1. IERC20

```
interface IERC20 {
    /**
     * @dev Returns the amount of tokens in existence.
     */
    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
`to`.
     *
     * Returns a boolean value indicating whether the operation
succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transfer(address to, 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 `from` to `to` 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 from,
        address to,
        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. BNOUFlexiblePool Contract

```
contract BnouFlexiblePool is Ownable, Pausable {
    using SafeERC20 for IERC20;

    struct UserInfo {
        uint256 shares; // number of shares for a user
        uint256 lastDepositedTime; // keeps track of deposited
time for potential penalty
        uint256 bnouAtLastUserAction; // keeps track of bnou
deposited at the last user action
        uint256 lastUserActionTime; // keeps track of the last
user action time
    }

    IERC20 public immutable token; // Bnou token
    IBnouPool public immutable bnouPool; //Bnou pool

    mapping(address => UserInfo) public userInfo;

    uint256 public totalShares;
    address public admin;
    address public treasury;
    bool public staking = true;

    uint256 public constant MAX_PERFORMANCE_FEE = 2000; // 20%
    uint256 public constant MAX_WITHDRAW_FEE = 500; // 5%
    uint256 public constant MAX_WITHDRAW_FEE_PERIOD = 1 weeks; //
1 week
    uint256 public constant MAX_WITHDRAW_AMOUNT_BOOSTER = 10010;
// 1.001
    uint256 public constant MIN_DEPOSIT_AMOUNT = 0.00001 ether;
    uint256 public constant MIN_WITHDRAW_AMOUNT = 0.00001 ether;
    uint256 public constant MIN_WITHDRAW_AMOUNT_BOOSTER = 10000;
// 1

    //When call bnoupool.withdrawByAmount function,there will be a
loss of precision, so need to withdraw more.
    uint256 public withdrawAmountBooster = 10001; // 1.0001
    uint256 public performanceFee = 200; // 2%
    uint256 public withdrawFee = 10; // 0.1%
    uint256 public withdrawFeePeriod = 72 hours; // 3 days
```

```
    event DepositBnou(address indexed sender, uint256 amount,
uint256 shares, uint256 lastDepositedTime);
    event WithdrawShares(address indexed sender, uint256 amount,
uint256 shares);
    event ChargePerformanceFee(address indexed sender, uint256
amount, uint256 shares);
    event ChargeWithdrawFee(address indexed sender, uint256
amount);
    event Pause();
    event Unpause();
    event NewAdmin(address admin);
    event NewTreasury(address treasury);
    event NewPerformanceFee(uint256 performanceFee);
    event NewWithdrawFee(uint256 withdrawFee);
    event NewWithdrawFeePeriod(uint256 withdrawFeePeriod);
    event NewWithdrawAmountBooster(uint256 withdrawAmountBooster);

    constructor(
        IERC20 _token,
        IBnouPool _bnouPool,
        address _initializer
    ) {
        token = _token;
        bnouPool = _bnouPool;

        // Infinite approve
        IERC20(_token).safeApprove(address(_bnouPool),
type(uint256).max);
        _transferOwnership(_initializer);
    }

    /**
     * @notice Checks if the msg.sender is the admin address
     */
    modifier onlyAdmin() {
        require(msg.sender == admin, "admin: wut?");
        _;
    }

    /**
     * @notice Deposits funds into the Bnou Flexible Pool.
```

```
     * @dev Only possible when contract not paused.
     * @param _amount: number of tokens to deposit (in BNOU)
     */
    function deposit(uint256 _amount) external whenNotPaused {
        require(staking, "Not allowed to stake");
        require(_amount > MIN_DEPOSIT_AMOUNT, "Deposit amount must
be greater than MIN_DEPOSIT_AMOUNT");
        UserInfo storage user = userInfo[msg.sender];
        //charge performanceFee
        bool chargeFeeFromDeposite;
        uint256 currentPerformanceFee;
        uint256 performanceFeeShares;
        if (user.shares > 0) {
            uint256 totalAmount = (user.shares * balanceOf()) /
totalShares;
            uint256 earnAmount = totalAmount -
user.bnouAtLastUserAction;
            currentPerformanceFee = (earnAmount * performanceFee)
/ 10000;
            if (currentPerformanceFee > 0) {
                performanceFeeShares = (currentPerformanceFee *
totalShares) / balanceOf();
                user.shares -= performanceFeeShares;
                totalShares -= performanceFeeShares;
                if (_amount >= currentPerformanceFee) {
                    chargeFeeFromDeposite = true;
                } else {
                    // withdrawByAmount have a MIN_WITHDRAW_AMOUNT
limit ,so need to withdraw more than MIN_WITHDRAW_AMOUNT.
                    uint256 withdrawAmount = currentPerformanceFee
< MIN_WITHDRAW_AMOUNT
                        ? MIN_WITHDRAW_AMOUNT
                        : currentPerformanceFee;
                    //There will be a loss of precision when call
withdrawByAmount, so need to withdraw more.
                    withdrawAmount = (withdrawAmount *
withdrawAmountBooster) / 10000;
                    bnouPool.withdrawByAmount(withdrawAmount);

                    currentPerformanceFee = available() >=
currentPerformanceFee ? currentPerformanceFee : available();
                    token.safeTransfer(treasury,
```

```
currentPerformanceFee);
                    emit ChargePerformanceFee(msg.sender,
currentPerformanceFee, performanceFeeShares);
                }
            }
        }
        uint256 pool = balanceOf();
        token.safeTransferFrom(msg.sender, address(this),
_amount);
        if (chargeFeeFromDeposite) {
            token.safeTransfer(treasury, currentPerformanceFee);
            emit ChargePerformanceFee(msg.sender,
currentPerformanceFee, performanceFeeShares);
            pool -= currentPerformanceFee;
        }
        uint256 currentShares;
        if (totalShares != 0) {
            currentShares = (_amount * totalShares) / pool;
        } else {
            currentShares = _amount;
        }

        user.shares += currentShares;
        user.lastDepositedTime = block.timestamp;

        totalShares += currentShares;

        _earn();

        user.bnouAtLastUserAction = (user.shares * balanceOf()) /
totalShares;
        user.lastUserActionTime = block.timestamp;

        emit DepositBnou(msg.sender, _amount, currentShares,
block.timestamp);
    }

    /**
     * @notice Withdraws funds from the Bnou Flexible Pool
     * @param _shares: Number of shares to withdraw
     */
    function withdraw(uint256 _shares) public {
```

```
        UserInfo storage user = userInfo[msg.sender];
        require(_shares > 0, "Nothing to withdraw");
        require(_shares <= user.shares, "Withdraw amount exceeds
balance");
        //charge performanceFee
        uint256 totalAmount = (user.shares * balanceOf()) /
totalShares;
        uint256 earnAmount = totalAmount -
user.bnouAtLastUserAction;
        uint256 currentPerformanceFee;
        uint256 performanceFeeShares;
        if (earnAmount > 0) {
            currentPerformanceFee = (earnAmount * performanceFee)
/ 10000;
            performanceFeeShares = (currentPerformanceFee *
totalShares) / balanceOf();
            user.shares -= performanceFeeShares;
            totalShares -= performanceFeeShares;
        }
        //Update withdraw shares
        if (_shares > user.shares) {
            _shares = user.shares;
        }
        //The current pool balance should not include
currentPerformanceFee.
        uint256 currentAmount = (_shares * (balanceOf() -
currentPerformanceFee)) / totalShares;
        user.shares -= _shares;
        totalShares -= _shares;
        uint256 withdrawAmount = currentAmount +
currentPerformanceFee;
        if (staking) {
            // withdrawByAmount have a MIN_WITHDRAW_AMOUNT limit
,so need to withdraw more than MIN_WITHDRAW_AMOUNT.
            withdrawAmount = withdrawAmount < MIN_WITHDRAW_AMOUNT
? MIN_WITHDRAW_AMOUNT : withdrawAmount;
            //There will be a loss of precision when call
withdrawByAmount, so need to withdraw more.
            withdrawAmount = (withdrawAmount *
withdrawAmountBooster) / 10000;
            bnouPool.withdrawByAmount(withdrawAmount);
        }
```

```
        uint256 currentWithdrawFee;
        if (block.timestamp < user.lastDepositedTime +
withdrawFeePeriod) {
            currentWithdrawFee = (currentAmount * withdrawFee) /
10000;
            currentAmount -= currentWithdrawFee;
        }
        //Combine two fees to reduce gas
        uint256 totalFee = currentPerformanceFee +
currentWithdrawFee;
        if (totalFee > 0) {
            totalFee = available() >= totalFee ? totalFee :
available();
            token.safeTransfer(treasury, totalFee);
            if (currentPerformanceFee > 0) {
                emit ChargePerformanceFee(msg.sender,
currentPerformanceFee, performanceFeeShares);
            }
            if (currentWithdrawFee > 0) {
                emit ChargeWithdrawFee(msg.sender,
currentWithdrawFee);
            }
        }

        currentAmount = available() >= currentAmount ?
currentAmount : available();
        token.safeTransfer(msg.sender, currentAmount);

        if (user.shares > 0) {
            user.bnouAtLastUserAction = (user.shares *
balanceOf()) / totalShares;
        } else {
            user.bnouAtLastUserAction = 0;
        }

        user.lastUserActionTime = block.timestamp;

        emit WithdrawShares(msg.sender, currentAmount, _shares);
    }

    /**
```

```
     * @notice Withdraws all funds for a user
     */
    function withdrawAll() external {
        withdraw(userInfo[msg.sender].shares);
    }

    /**
     * @notice Sets admin address
     * @dev Only callable by the contract owner.
     */
    function setAdmin(address _admin) external onlyOwner {
        require(_admin != address(0), "Cannot be zero address");
        admin = _admin;
        emit NewAdmin(admin);
    }

    /**
     * @notice Sets treasury address
     * @dev Only callable by the contract owner.
     */
    function setTreasury(address _treasury) external onlyOwner {
        require(_treasury != address(0), "Cannot be zero
address");
        treasury = _treasury;
        emit NewTreasury(treasury);
    }

    /**
     * @notice Sets performance fee
     * @dev Only callable by the contract admin.
     */
    function setPerformanceFee(uint256 _performanceFee) external
onlyAdmin {
        require(_performanceFee <= MAX_PERFORMANCE_FEE,
"performanceFee cannot be more than MAX_PERFORMANCE_FEE");
        performanceFee = _performanceFee;
        emit NewPerformanceFee(performanceFee);
    }

    /**
     * @notice Sets withdraw fee
     * @dev Only callable by the contract admin.
```

```
     */
    function setWithdrawFee(uint256 _withdrawFee) external
onlyAdmin {
        require(_withdrawFee <= MAX_WITHDRAW_FEE, "withdrawFee
cannot be more than MAX_WITHDRAW_FEE");
        withdrawFee = _withdrawFee;
        emit NewWithdrawFee(withdrawFee);
    }

    /**
     * @notice Sets withdraw fee period
     * @dev Only callable by the contract admin.
     */
    function setWithdrawFeePeriod(uint256 _withdrawFeePeriod)
external onlyAdmin {
        require(
            _withdrawFeePeriod <= MAX_WITHDRAW_FEE_PERIOD,
            "withdrawFeePeriod cannot be more than
MAX_WITHDRAW_FEE_PERIOD"
        );
        withdrawFeePeriod = _withdrawFeePeriod;
        emit NewWithdrawFeePeriod(withdrawFeePeriod);
    }

    /**
     * @notice Sets withdraw amount booster
     * @dev Only callable by the contract admin.
     */
    function setWithdrawAmountBooster(uint256
_withdrawAmountBooster) external onlyAdmin {
        require(
            _withdrawAmountBooster >= MIN_WITHDRAW_AMOUNT_BOOSTER,
            "withdrawAmountBooster cannot be less than
MIN_WITHDRAW_AMOUNT_BOOSTER"
        );
        require(
            _withdrawAmountBooster <= MAX_WITHDRAW_AMOUNT_BOOSTER,
            "withdrawAmountBooster cannot be more than
MAX_WITHDRAW_AMOUNT_BOOSTER"
        );
        withdrawAmountBooster = _withdrawAmountBooster;
        emit NewWithdrawAmountBooster(withdrawAmountBooster);
```

```
    }

    /**
     * @notice Withdraws from Bnou Pool without caring about
rewards.
     * @dev EMERGENCY ONLY. Only callable by the contract admin.
     */
    function emergencyWithdraw() external onlyAdmin {
        require(staking, "No staking bnou");
        staking = false;
        bnouPool.withdrawAll();
    }

    /**
     * @notice Withdraw unexpected tokens sent to the Bnou
Flexible Pool
     */
    function inCaseTokensGetStuck(address _token) external
onlyAdmin {
        require(_token != address(token), "Token cannot be same as
deposit token");

        uint256 amount = IERC20(_token).balanceOf(address(this));
        IERC20(_token).safeTransfer(msg.sender, amount);
    }

    /**
     * @notice Triggers stopped state
     * @dev Only possible when contract not paused.
     */
    function pause() external onlyAdmin whenNotPaused {
        _pause();
        emit Pause();
    }

    /**
     * @notice Returns to normal state
     * @dev Only possible when contract is paused.
     */
    function unpause() external onlyAdmin whenPaused {
        _unpause();
        emit Unpause();
```

```
    }

    /**
     * @notice Calculates the price per share
     */
    function getPricePerFullShare() external view returns
(uint256) {
        return totalShares == 0 ? 1e18 : (balanceOf() * 1e18) /
totalShares;
    }

    /**
     * @notice Custom logic for how much the pool to be borrowed
     * @dev The contract puts 100% of the tokens to work.
     */
    function available() public view returns (uint256) {
        return token.balanceOf(address(this));
    }

    /**
     * @notice Calculates the total underlying tokens
     * @dev It includes tokens held by the contract and held in
BnouPool
     */
    function balanceOf() public view returns (uint256) {
        (uint256 shares, , , , , , , , ) =
bnouPool.userInfo(address(this));
        uint256 pricePerFullShare =
bnouPool.getPricePerFullShare();

        return token.balanceOf(address(this)) + (shares *
pricePerFullShare) / 1e18;
    }

    function _earn() internal {
        uint256 bal = available();
        if (bal > 0) {
            bnouPool.deposit(bal, 0);
        }
    }
}
```

READ CONTRACT (ONLY NEED TO KNOW)

1. MAX_PERFORMANCE_FEE
2000 uint256
(Function for read max performance fee)

2. MAX_WITHDRAW_AMOUNT_BOOSTER
10010 uint256
(Function for read read max withdraw amount booster)

3. MAX_WITHDRAW_FEE
500 uint256
(Function for read duration factor)

4. owner
0x47a4ea43c6cf05e2541a76903f06d4b24fa4cc81 address
(Function for read owner address)

4. token
0x221e4c3bbabc3e33a8be082c1f96037a9761a9f2 address
(Function for read bnou address)

4. admin
0x47a4ea43c6cf05e2541a76903f06d4b24fa4cc81 address
(Function for read admin address)

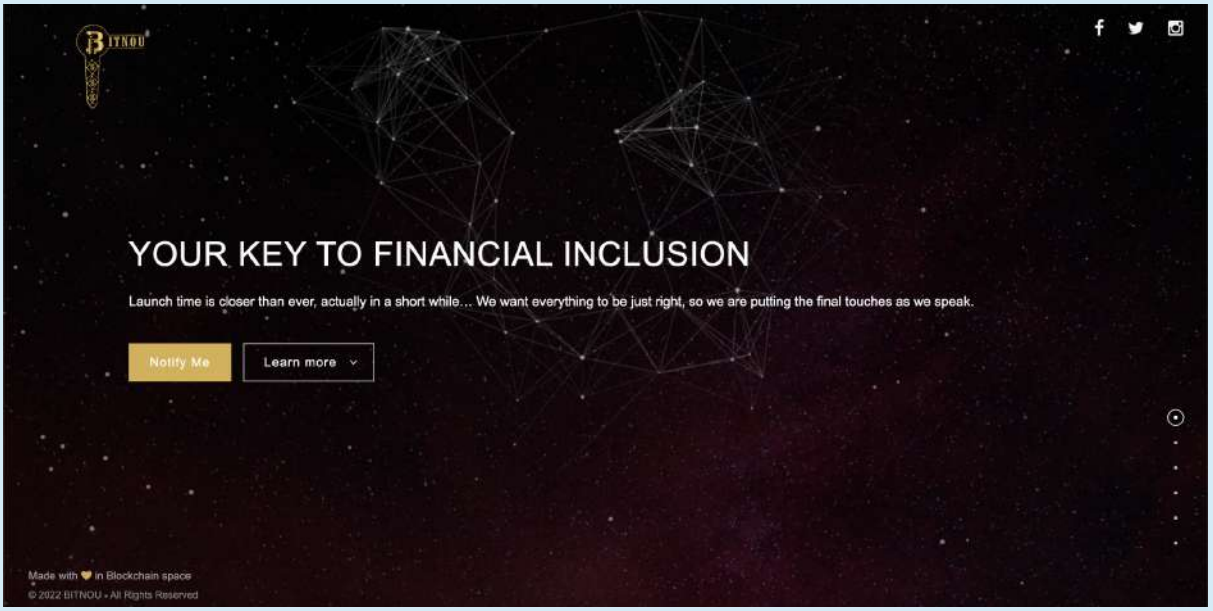
WRITE CONTRACT

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. setAdmin
_admin (address)
(Its function for set admin address)

WEBSITE REVIEW



- **Mobile Friendly**
- **Contains no code error**
- **SSL Secured (By Let's Encrypt SSL)**

Web-Tech stack: Apache, Bootstrap, Animate css

Domain .com (Hostgator) - Tracked by whois

First Contentful Paint:	1.6s
Fully Loaded Time	5.6s
Performance	46%
Accessibility	79%
Best Practices	58%
SEO	80%

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 KYC by Blocksafu

HONEYPOT REVIEW

- Ability to sell.
- The owner is not able to pause the contract.
- The owner can’t set fees

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.