



BlockSAFU

ADVANCE MANUAL SMART CONTRACT AUDIT



Project: LinkDao Staking

Website: <https://linkdao.network/>



BlockSAFU Score:

98

Contract Address:

0x68B7Ee70Ab8c692605F0f2345008398c0Ec1A20B

0xd4769eB0cca4d29af46361052b55767e016Df956

Disclaimer: BlockSAFU is not responsible for any financial losses.
Nothing in this contract audit is financial advice, please do your own reasearch.

DISCLAIMER

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. BlockSAFU's 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.

SMART CONTRACT REVIEW

Token Name	LinkDaoRecurring_V24
Type	Staking Contract
Contract Address	0x68B7Ee70Ab8c692605F0f2345008398c0Ec1A20B 0xd4769eB0cca4d29af46361052b55767e016Df956
Deployer Address	0x7ab8B73FCd3AbB89F99d96694fa7701D2a83Ce34
Owner Address	0x7ab8B73FCd3AbB89F99d96694fa7701D2a83Ce34
Stake	0%
Unstake	0%
Gas Used for Buy	<i>will be updated after the DEX listing</i>
Gas Used for Sell	<i>will be updated after the DEX listing</i>
Contract Created	Aug-23-2022 08:07:58 AM +UTC
Initial Liquidity	<i>will be updated after the DEX listing</i>
Liquidity Status	Locked
Unlocked Date	<i>will be updated after the DEX listing</i>
Verified CA	Yes
Compiler	v0.8.15+commit.e14f2714
Optimization	No with 200 runs
Sol License	MIT License
Top 5 Holders	<i>will be updated after the DEX listing</i>
Other	default evmVersion

TAX

Stake Fee	0%	Unstake Fee	0%
-----------	----	-------------	----



OVERVIEW


Fees

- Stake 0%.
- Unstake 0%.

Unstake

- Investor can unstake anytime

Rewards

- **Owners cannot take the allocation of staking rewards tokens, so investors are ensured to get rewards according to what is allocated. Make sure the amount is allocated accordingly.**
- 

Team Review

The LinkDao team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 36,304 people in their telegram group (count in audit date).

Official Website And Social Media

Website: <https://linkdao.network/>

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

Twitter: <https://twitter.com/LinkdaoN>

MANUAL CODE REVIEW

Minor-risk

0 minor-risk code issues 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 problem.

Critical-Risk

0 critical-risk code issues found

Must be fixed, and will bring problem.

EXTRA NOTES SMART CONTRACT

1. IERC20

```
interface IERC20 {  
    /**  
     * @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);  
    ...  
}
```

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. LinkDao Contract

```
contract LinkDaoRecurring_V24 is Ownable {
    using SafeMath for uint256;
    using SafeERC20 for IERC20;

    address public LINKDAO_TOKEN_ADDRESS;
    uint256 public MAX_STAKE = 100000 ether;
    uint256 public MAX_STAKE_PER_USER = 5000 ether;
    uint256 public MIN_STAKE = 1 ether;

    uint256 public REWARD_PERIOD = 30 days;

    uint256[] public RELEASE_PERIODS = [
0,0,0,0,0,0,5000,5000,5000,5000,5000,5000,5000,5000,5000,5000,5000,5000,
5000,5000,5000,5000,5000,10000,10000
];
    uint256 public TOTAL_RELEASE_DAYS = 720 days;

    uint256 public ROI_PERCENTAGE = 3000; // 3%
    uint256 public TOTAL_PERCENTAGE = 100000;
    uint256 public MAX_REWARD_PERCENTAGE = 48900;

    uint256 public totalInvestments;
    uint256 public totalInvestors;
    uint256 public totalReward;

    uint256 public currentID;
    uint256 public deposited_LKDReward;

    struct Investor{
        address investor;
        uint256 totalInvestment;
        uint256 activeAmount;
        uint256 totalReward;
        uint256 startDate;
        uint256[] userInvestments;
    }

    struct Investment{
        address investor;
```

```

    uint256 totalAmount;
    uint256 totalReward;
    uint256 activeAmount;
    uint256 claimedAmount;
    uint256 startDate;
    uint256 lastCheckpoint;
    uint256 endDate;
    uint256 maxReward;
}

mapping(address=>Investor) public investors;
mapping(uint256=>Investment) public investments;

constructor(address _LinkDaoToken) {
    require(_LinkDaoToken!=address(0), "Invalid LinkDao token
address");

    LINKDAO_TOKEN_ADDRESS = _LinkDaoToken;
}

function investAmount(uint256 _amount) external{
    require(_amount>=MIN_STAKE, "Cannot stake less than 1
LKD");
    require(totalInvestments.add(_amount)<=MAX_STAKE, "Cannot
stake more than 100,000 LKD");

    require(investors[msg.sender].totalInvestment.add(_amount)<=MAX_ST
AKE_PER_USER, "Cannot stake more than 100,000 LKD");

    currentID = currentID.add(1);

    investments[currentID] = Investment({
        investor:msg.sender,
        totalAmount:_amount,
        totalReward:0,
        activeAmount:_amount,
        claimedAmount:0,
        startDate:block.timestamp,
        lastCheckpoint:block.timestamp,
        endDate:block.timestamp.add(TOTAL_RELEASE_DAYS),

        maxReward:_amount.mul(MAX_REWARD_PERCENTAGE).div(TOTAL_PERCENTAGE)

```

```
});
```

```
IERC20(LINKDAO_TOKEN_ADDRESS).safeTransferFrom(msg.sender, address(  
this),_amount);
```

```
    if(investors[msg.sender].investor==address(0)){  
        investors[msg.sender].investor = msg.sender;  
        investors[msg.sender].startDate = block.timestamp;  
        totalInvestors = totalInvestors.add(1);  
    }
```

```
        investors[msg.sender].totalInvestment =  
investors[msg.sender].totalInvestment.add(_amount);  
        investors[msg.sender].activeAmount =  
investors[msg.sender].activeAmount.add(_amount);  
        investors[msg.sender].userInvestments.push(currentID);  
  
        totalInvestments = totalInvestments.add(_amount);  
    }
```

```
    function getTotalProfit(address _investorAddress) public view  
returns(uint256 totalProfit) {  
        for(uint256  
i=0;i<investors[_investorAddress].userInvestments.length;i++){  
            totalProfit =  
totalProfit.add(getTotalProfitForInvestment(  
                investors[_investorAddress].userInvestments[i]  
            ));  
        }  
    }
```

```
    function getWithdrawableTotalProfit(address _investorAddress)  
public view returns(uint256 totalProfit) {  
        for(uint256  
i=0;i<investors[_investorAddress].userInvestments.length;i++){  
            (uint256 currentProfit,) =  
getWithdrawableTotalProfitForInvestment(  
                investors[_investorAddress].userInvestments[i]  
            );  
            totalProfit = totalProfit.add(currentProfit);  
        }  
    }
```

```

    }

    function getTotalProfitForInvestment(uint256 _investmentID)
    public view returns(uint256 totalProfit){
        uint256 activeAmount =
        investments[_investmentID].activeAmount;

        if(activeAmount==0){
            return totalProfit;
        }

        uint256 currentTime =
            block.timestamp < investments[_investmentID].endDate ?
            block.timestamp : investments[_investmentID].endDate;

        uint256 timePeriod = currentTime -
            investments[_investmentID].lastCheckpoint;

        totalProfit =
            activeAmount.mul(ROI_PERCENTAGE).mul(timePeriod).div(REWARD_PERIOD
            .mul(TOTAL_PERCENTAGE));

        if(investments[_investmentID].totalReward.add(totalProfit)>investm
            ents[_investmentID].maxReward){
            totalProfit =
            investments[_investmentID].maxReward.sub(investments[_investmentID
            ].totalReward);
        }

        uint256 releaseStartIndex =
            (investments[_investmentID].lastCheckpoint.sub(investments[_invest
            mentID].startDate)).div(REWARD_PERIOD);
        uint256 releaseEndIndex =
            (currentTime.sub(investments[_investmentID].startDate)).div(REWARD
            _PERIOD);

        for(uint256 i = releaseStartIndex; i<releaseEndIndex;i++){
            if(RELEASE_PERIODS[i]>0){
                uint256 releaseAmount =

```

```
investments[_investmentID].totalAmount.mul(RELEASE_PERIODS[i]).div  
(TOTAL_PERCENTAGE);
```

```
        totalProfit = totalProfit.add(releaseAmount);  
    }  
}  
}
```

```
function getWithdrawableTotalProfitForInvestment(uint256  
_investmentID) public view returns(uint256 totalProfit,uint256  
amountToRelease){
```

```
    uint256 activeAmount =  
investments[_investmentID].activeAmount;
```

```
    if(activeAmount==0){  
        return (totalProfit,amountToRelease);  
    }
```

```
    uint256 currentTime =  
        block.timestamp < investments[_investmentID].endDate ?  
block.timestamp : investments[_investmentID].endDate;
```

```
    uint256 timePeriod = currentTime -  
investments[_investmentID].lastCheckpoint;
```

```
    totalProfit =  
activeAmount.mul(ROI_PERCENTAGE).mul(timePeriod.div(REWARD_PERIOD)  
)div(TOTAL_PERCENTAGE);
```

```
if(investments[_investmentID].totalReward.add(totalProfit)>investm  
ents[_investmentID].maxReward){  
    totalProfit =  
investments[_investmentID].maxReward.sub(investments[_investmentID  
].totalReward);  
}
```

```
    uint256 releaseStartIndex =  
(investments[_investmentID].lastCheckpoint.sub(investments[_invest  
mentID].startDate)).div(REWARD_PERIOD);  
    uint256 releaseEndIndex =
```

```
(currentTime.sub(investments[_investmentID].startDate)).div(REWARD_PERIOD);
```

```
    for(uint256 i = releaseStartIndex; i<releaseEndIndex;i++){  
        if(RELEASE_PERIODS[i]>0){  
            uint256 releaseAmount =  
investments[_investmentID].totalAmount.mul(RELEASE_PERIODS[i]).div  
(TOTAL_PERCENTAGE);
```

```
                totalProfit = totalProfit.add(releaseAmount);  
                amountToRelease =  
amountToRelease.add(releaseAmount);  
            }  
        }  
    }
```

```
function getLkdBalance() public view returns(uint256 balance){  
    balance =  
IERC20(LINKDAO_TOKEN_ADDRESS).balanceOf(address(this));  
}
```

```
function withdrawReward() external {  
    uint256 totalRewardToRelease;  
    for(uint256 i =  
0;i<investors[msg.sender].userInvestments.length;i++){  
        uint256 _investmentID =  
investors[msg.sender].userInvestments[i];
```

```
        (uint256 currentInvestmentProfit,uint256  
releaseAmount) =  
getWithdrawableTotalProfitForInvestment(_investmentID);
```

```
        totalRewardToRelease =  
totalRewardToRelease.add(currentInvestmentProfit);
```

```
        investments[_investmentID].totalReward =  
investments[_investmentID].totalReward.add(currentInvestmentProfit  
.sub(releaseAmount));  
        investments[_investmentID].claimedAmount =  
investments[_investmentID].claimedAmount.add(currentInvestmentProf  
it);
```

```

        investments[_investmentID].activeAmount =
investments[_investmentID].activeAmount.sub(releaseAmount);

        investors[msg.sender].activeAmount =
investors[msg.sender].activeAmount.sub(releaseAmount);

        uint256 numberOfMonths =
(block.timestamp.sub(investments[_investmentID].lastCheckpoint)).div(
REWARD_PERIOD);
        investments[_investmentID].lastCheckpoint =
investments[_investmentID].lastCheckpoint.add(numberOfMonths.mul(
REWARD_PERIOD));
    }
    require(totalRewardToRelease>0, "No withdrawable reward
yet");

IERC20(LINKDAO_TOKEN_ADDRESS).safeTransfer(msg.sender, totalRewardToRelease);

    investors[msg.sender].totalReward =
investors[msg.sender].totalReward.add(totalRewardToRelease);

    totalReward = totalReward.add(totalRewardToRelease);
}

function withdrawSingleReward(uint256 _investmentID) external
{
    require(investments[_investmentID].investor==msg.sender, "Only
investor can claim the reward");

    uint256 totalRewardToRelease;

    (uint256 currentInvestmentProfit, uint256 releaseAmount) =
getWithdrawableTotalProfitForInvestment(_investmentID);

    totalRewardToRelease =
totalRewardToRelease.add(currentInvestmentProfit);

    investments[_investmentID].totalReward =
investments[_investmentID].totalReward.add(currentInvestmentProfit

```



```

.sub(releaseAmount));
    investments[_investmentID].claimedAmount =
investments[_investmentID].claimedAmount.add(currentInvestmentProf
it);

    investments[_investmentID].activeAmount =
investments[_investmentID].activeAmount.sub(releaseAmount);
    investors[msg.sender].activeAmount =
investors[msg.sender].activeAmount.sub(releaseAmount);

    uint256 numberOfMonths =
(block.timestamp.sub(investments[_investmentID].lastCheckpoint)).div(
REWARD_PERIOD);
    investments[_investmentID].lastCheckpoint =
investments[_investmentID].lastCheckpoint.add(numberOfMonths.mul(R
EWARD_PERIOD));

IERC20(LINKDAO_TOKEN_ADDRESS).safeTransfer(msg.sender, totalRewardToRelease);

    investors[msg.sender].totalReward =
investors[msg.sender].totalReward.add(totalRewardToRelease);

    totalReward = totalReward.add(totalRewardToRelease);
}

function depositLKD(uint256 _amount) external onlyOwner{
    deposited_LKDReward = deposited_LKDReward.add(_amount);

IERC20(LINKDAO_TOKEN_ADDRESS).safeTransferFrom(msg.sender, address(
this), _amount);
}

}

```

Constructor LinkDaoRecurring_V24

Contract Address: 0x68B7Ee70Ab8c692605F0f2345008398c0Ec1A20B

```
uint256 public REWARD_PERIOD = 30 days;

uint256[] public RELEASE_PERIODS = [
    0,0,0,0,0,0,5000,5000,5000,5000,5000,5000,5000,5000,5000,5000,
    5000,5000,5000,5000,5000,10000,10000
];

uint256 public TOTAL_RELEASE_DAYS = 720 days;

uint256 public ROI_PERCENTAGE = 3000; // 3%
uint256 public TOTAL_PERCENTAGE = 100000;
uint256 public MAX_REWARD_PERCENTAGE = 48900;
```

Constructor LinkDaoRecurring_V18

Contract Address: 0xd4769eB0cca4d29af46361052b55767e016Df956

[illegible]

READ CONTRACT (ONLY NEED TO KNOW)

1. LINKDAO_TOKEN_ADDRESS

0xaf027427dc6d31a3e7e162a710a5fe27e63e275f uint256

(Shows Contract Linkdao)

2. MAX_REWARD_PERCENTAGE

48900 uint256

3. MAX_STAKE

```
100000000000000000000000000000000 uint256
```

4. MAX_STAKE_PER_USER

50000000000000000000000000000000 uint256

5. MIN_STAKE

1000000000000000000000000 uint256

WRITE CONTRACT

1. depositLKD

_amount (uint256)

(The form is filled with the amount for deposit LinkDao)

2. investAmount

_amount (uint256)

(The form is filled with the amount for invest)

3. renounceOwnership

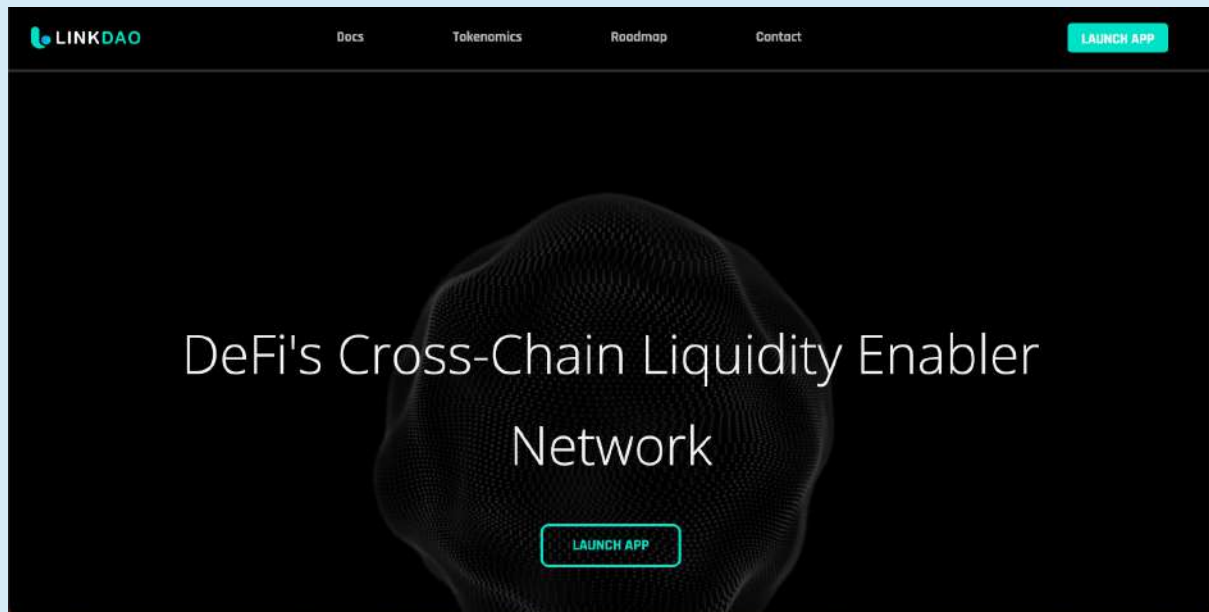
(Call function for renounce ownership)

4. transferOwnership

newOwner (address)

(Its function is to change the owner)

WEBSITE REVIEW



- **Mobile Friendly**
- **Contains no code error**
- **SSL Secured (By R3 SSL)**

Web-Tech stack: jQuery, Bootstrap, Wordpress

Domain .com (hostinger) - Tracked by whois

First Contentful Paint:	492ms
Fully Loaded Time	1.6s
Performance	100%
Accessibility	94%
Best Practices	92%
SEO	91%

SECURITY REVIEW

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

- Owner cannot set tax/fee at stake or unstake
- Owner can't prevent investors from unstake
- Owners cannot take the allocation of staking rewards tokens, so investors are ensured to get rewards according to what is allocated. Make sure the amount is allocated accordingly.

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.