



TechRate
AUDIT COMPANY

Smart Contract Security Audit

TechRate

November, 2021

Audit Details



Audited project

Shiba Robinhood



Deployer address

0x99d389D5441beD01b40DEc2208F671C650a54988



Client contacts:

Shiba Robinhood team



Blockchain

Binance Smart Chain



Project website:

<https://shibarobinhood.com>

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and TechRate and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (TechRate) owe no duty of care towards you or any other person, nor does TechRate make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and TechRate hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, TechRate hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against TechRate, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Background

TechRate was commissioned by Shiba Robinhood to perform an audit of smart contracts:

<https://bscscan.com/address/0xbdf3f3a7bb6e62d582553ca9a0a77deeffbcf5ea#code>

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Contracts Details

Token contract details for 23.11.2021

Contract name	Shiba Robinhood
Contract address	0xbDF3f3A7bb6e62d582553ca9a0a77DEefFBCF5Ea
Total supply	1,000,000,000,000,000,000
Token ticker	RSHIB
Decimals	18
Token holders	667
Transactions count	1,619
Top 100 holders dominance	99.99%
Antibot fee	1
Holder distributor fee	2
Liquidity pool fee	4
Marketing fee	3
Contract deployer address	0x99d389D5441beD01b40DEc2208F671C650a54988
Contract's current owner address	0x99d389D5441beD01b40DEc2208F671C650a54988

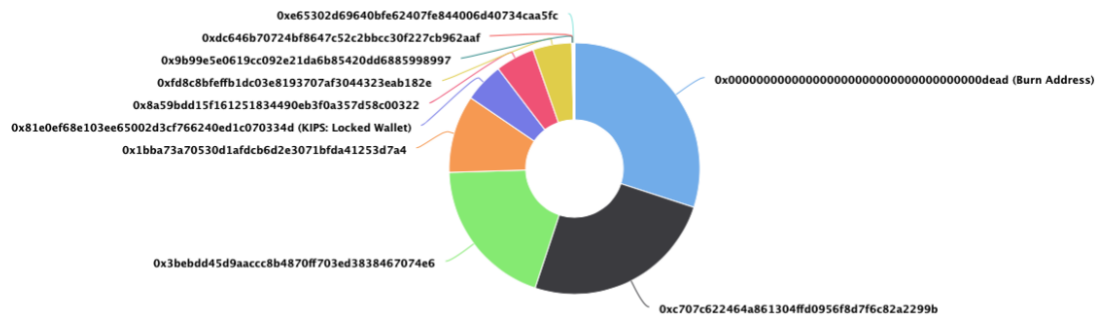
Shiba Robinhood Token Distribution

The top 100 holders collectively own 99.99% (999,910,717,038,434,000.00 Tokens) of Shiba Robinhood

Token Total Supply: 1,000,000,000,000,000.00 Token | Total Token Holders: 667

Shiba Robinhood Top 100 Token Holders

Source: BscScan.com



(A total of 999,910,717,038,434,000.00 tokens held by the top 100 accounts from the total supply of 1,000,000,000,000,000.00 token)

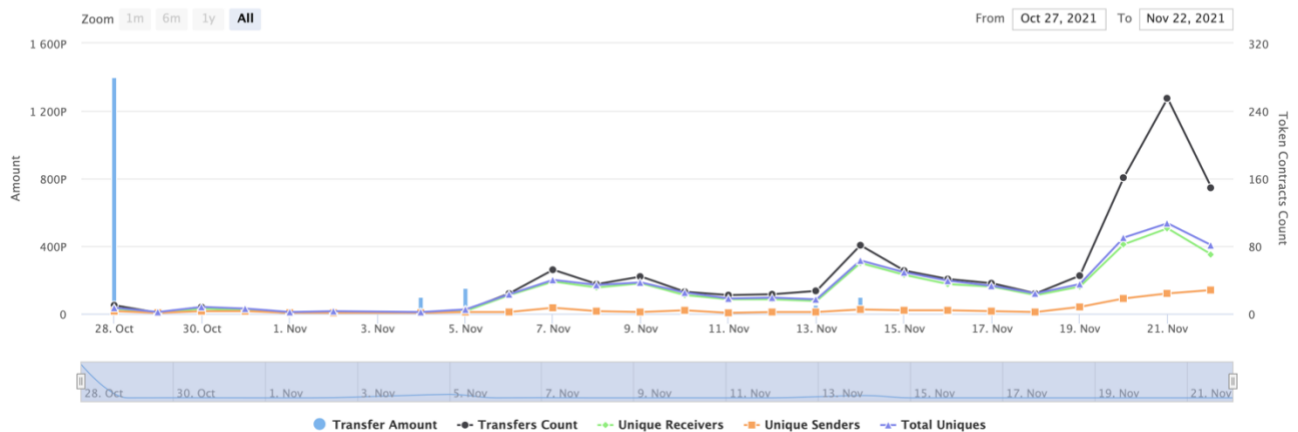
Shiba Robinhood Contract Interaction Details

Time Series: Token Contract Overview




Thu 28, Oct 2021 - Mon 22, Nov 2021

Token Contract 0xbdf3f3a7bb6e62d582553ca9a0a77deeffbcf5ea (Shiba Robinhood)

Source: BscScan.com



Shiba Robinhood Top 10 Token Holders

Rank	Address	Quantity (Token)	Percentage
1	Burn Address	300,000,000,000,000,000	30.0000%
2	0xc707c622464a861304ffd0956f8d7f6c82a2299b	250,999,999,999,999,000	25.1000%
3	0x3bebdd45d9aacc8b4870ff703ed3838467074e6	194,088,000,000,000,000	19.4088%
4	0x1bba73a70530d1afdc6d2e3071bfda41253d7a4	100,000,000,000,000,000	10.0000%
5	 KIPS: Locked Wallet	51,000,000,000,000,000	5.1000%
6	0x8a59bdd15f161251834490eb3f0a357d58c00322	50,428,843,468,430,900	5.0429%
7	0xfd8c8bfeffb1dc03e8193707af3044323eab182e	50,000,479,484,447,500	5.0000%
8	 0x9b99e5e0619cc092e21da6b85420dd6885998997	1,145,269,277,873,280.99361	0.1145%
9	0xdc646b70724bf8647c52c2bbcc30f227cb962aaf	650,000,000,000,000	0.0650%
10	 0xe65302d69640bfe62407fe844006d40734caa5fc	533,520,000,000,000	0.0534%



Contract functions details

- + Owned
 - [Pub] changeOwnership #
 - modifiers: onlyOwner
- + ERC20 (Owned)
 - [Pub] balanceOf
 - [Pub] transfer #
 - [Pub] transferFrom #
 - [Pub] approve #
 - [Pub] allowance
 - [Int] _mint #
 - [Int] _burn #
 - [Pub] setChangeStatus #
 - modifiers: onlyOwner
 - [Pub] setAntibotStatus #
 - modifiers: onlyOwner
 - [Pub] setHolderDistributorPercent #
 - modifiers: onlyOwner
 - [Pub] setLiquidityPoolPercent #
 - modifiers: onlyOwner
 - [Pub] setMarketingPercent #
 - modifiers: onlyOwner
 - [Pub] setAntibotPercent #
 - modifiers: onlyOwner
 - [Pub] setHoldersDistributorAddress #
 - modifiers: onlyOwner
 - [Pub] setLiquidityPoolAddress #
 - modifiers: onlyOwner
 - [Pub] setMarketingAddress #
 - modifiers: onlyOwner
 - [Pub] setAntibotAddress #
 - modifiers: onlyOwner
 - [Pub] tokenHolderList
 - [Pub] tokenHolderListIndex
 - [Pub] tokenHolderSinglebalance
 - [Pub] tokenHolderSingle
 - [Pub] tokenHolderAll
 - [Pub] distributeTokenToHolders #
 - modifiers: onlyOwner
 - [Int] _beforeTokenTransfer #
- + RSHIB (ERC20)
 - [Pub] <Constructor> #
 - [Ext] mint #
 - modifiers: onlyOwner
 - [Ext] burn #

(\$) = payable function

= non-constant function

Issues Checking Status

Issue description		Checking status
1.	Compiler errors.	Passed
2.	Race conditions and Reentrancy. Cross-function race conditions.	Passed
3.	Possible delays in data delivery.	Passed
4.	Oracle calls.	Passed
5.	Front running.	Passed
6.	Timestamp dependence.	Passed
7.	Integer Overflow and Underflow.	Passed
8.	DoS with Revert.	Passed
9.	DoS with block gas limit.	Low issues
10.	Methods execution permissions.	Passed
11.	Economy model of the contract.	Passed
12.	The impact of the exchange rate on the logic.	Passed
13.	Private user data leaks.	Passed
14.	Malicious Event log.	Passed
15.	Scoping and Declarations.	Passed
16.	Uninitialized storage pointers.	Passed
17.	Arithmetic accuracy.	Passed
18.	Design Logic.	Passed
19.	Cross-function race conditions.	Passed
20.	Safe Open Zeppelin contracts implementation and usage.	Passed
21.	Fallback function security.	Passed

Security Issues

✓ High Severity Issues

No high severity issues found.

✓ Medium Severity Issues

No medium severity issues found.

✓ Low Severity Issues

1. Out of gas

Issue:

- The functions `tokenHolderList()` and `tokenHolderListIndex()` use the loop to find addresses from the `tokenHoldersArr` list with not empty balances. Functions will be aborted with `OUT_OF_GAS` exception if there will be a long addresses list.

```
function tokenHolderList() public view returns(address[] memory userAddrList){
    uint balanceLength = 0 ;
    uint balanceLengthInner = 0 ;
    for(uint i=0; i < tokenHoldersArr.length; i++){
        if(balances[tokenHoldersArr[i]] > 0){
            balanceLength ++;
        }
    }

    userAddrList = new address[](balanceLength);
    for(uint i=0; i < tokenHoldersArr.length; i++){
        if(balances[tokenHoldersArr[i]] > 0){
            userAddrList[balanceLengthInner]=tokenHoldersArr[i];
            balanceLengthInner++;
        }
    }
}

function tokenHolderListIndex(uint index) public view returns(address[] memory userAddrList){
    uint balanceLength = 0 ;
    uint balanceLengthInner = 0 ;
    for(uint i=0; i < tokenHoldersArr.length; i++){
        if(balances[tokenHoldersArr[i]] > 0){
            balanceLength ++;
        }
    }

    userAddrList = new address[](balanceLength);
    for(uint i=0; i < tokenHoldersArr.length; i++){
        if(balances[tokenHoldersArr[i]] > 0){
            userAddrList[balanceLengthInner]=tokenHoldersArr[i];
            balanceLengthInner++;
        }
    }
}
```

- The function `distributeTokenToHolders()` also uses the loop for distribute tokens between addresses with not empty balances. It also could be aborted with `OUT_OF_GAS` exception if there will be a long addresses list.

```
function distributeTokenToHolders(uint amount) public onlyOwner returns(bool){
    require(balances[msg.sender]>amount,"Insufficient balance");
    require(tokenHolderList().length>0,"No Token Holder Found");
    uint sendAmt = amount/tokenHolderList().length;
    balances[msg.sender]-=amount;
    for(uint i=0; i < tokenHolderList().length; i++){

        address sendTo = tokenHolderList()[i];

        balances[sendTo]+=sendAmt;
        emit Transfer(msg.sender,sendTo,sendAmt);

    }
    return true;
}
```

Recommendation:

Check that the addresses array length is not too big.

Owner privileges

- Owner can enable / disable fees.

```
// Set change status
function setChangeStatus(bool val) public onlyOwner {
    require(change != val, "Already in this state");
    require(
        marketingAddress != address(0) &&
        liquidityPoolAddress != address(0) &&
        holdersDistributorAddress != address(0),
        "Change addresses cannot be zero"
    );
    change = val;
}

// Set change status
function setAntibotStatus(bool val) public onlyOwner {
    require(antibotStaus != val, "Already in this state");
    require(
        antibotAddress != address(0),
        "Change addresses cannot be zero"
    );
    antibotStaus = val;
}
```

- Owner can change holder distributor, liquidity pool, marketing and antibot fees.

```
// Set percent amount
function setHolderDistributorPercent(uint256 _percent) public onlyOwner {
    holderDistributorPercent = _percent;
}

function setLiquidityPoolPercent(uint256 _percent) public onlyOwner {
    liquidityPoolPercent = _percent;
}

function setMarketingPercent(uint256 _percent) public onlyOwner {
    marketingPercent = _percent;
}

function setAntibotPercent(uint256 _percent) public onlyOwner {
    antibotPercent = _percent;
}
```

- Owner can change holder distributor, liquidity pool, marketing and antibot wallet addresses.

```
function setHoldersDistributorAddress(address addr) public onlyOwner {
    holdersDistributorAddress = addr;
}

function setLiquidityPoolAddress(address addr) public onlyOwner {
    liquidityPoolAddress = addr;
}

function setMarketingAddress(address addr) public onlyOwner {
    marketingAddress = addr;
}

function setAntibotAddress(address addr) public onlyOwner {
    antibotAddress = addr;
}
```

- Owner can distribute token to other addresses.

```
function distributeTokenToHolders(uint256 amount)
    public
    onlyOwner
    returns (bool)
{
    require(balances[msg.sender] > amount, "Insufficient balance");
    require(tokenHolderList().length > 0, "No Token Holder Found");
    uint256 sendAmt = amount / tokenHolderList().length;
    balances[msg.sender] -= amount;
    for (uint256 i = 0; i < tokenHolderList().length; i++) {
        address sendTo = tokenHolderList()[i];

        balances[sendTo] += sendAmt;
        emit Transfer(msg.sender, sendTo, sendAmt);
    }
    return true;
}
```

- Owner can mint any amount of tokens.

```
function mint(address to, uint256 amount) external onlyOwner {
    require(to != address(0), "No mint to zero address");
    _mint(to, amount);
}
```

Conclusion

Smart contracts contain low severity issues and owner privileges!

TechRate note:

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