Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

November, 2021

Audit Details

Audited project

FarmerDoge (Launch Day)

Deployer address

0x967D716F672CeAC898647A3f27dBd224F3238520

Client contacts:

FarmerDoge (Launch Day)team

Blockchain

Binance Smart Chain

Project website:

https://www.farmerdoge.net/

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 FarmerDoge (Launch Day) to perform an audit of smart contracts:
https://bscscan.com/address/0x54d625b45bcb1326f64ce79fac313f4d8f47ae24#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

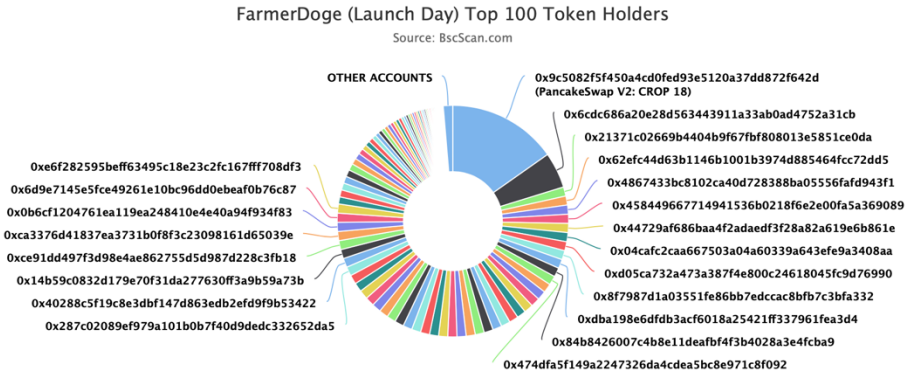
Token contract details for 02.11.2021

Contract name	FarmerDoge (Launch Day)
Contract address	0x54D625B45BCb1326F64cE79FAc313f4D8f47ae24
Total supply	10,000,000,000
Token ticker	CROP
Decimals	9
Token holders	174
Transactions count	2,326
Top 100 holders dominance	98.67%
Liquidity fee	1000
Buyback fee	0
Reflection fee	500
Marketing fee	500
Uniswap V2 pair	0x9C5082F5f450A4CD0FED93E5120A37DD872F642D
Contract deployer address	0x967D716F672CeAC898647A3f27dBd224F3238520
Contract's current owner address	0x967D716F672CeAC898647A3f27dBd224F3238520

FarmerDoge (Launch Day) Token Distribution

The top 100 holders collectively own 98.67% (9,866,811,686.60 Tokens) of FarmerDoge (Launch Day)

Token Total Supply: 10,000,000,000.00 Token | Total Token Holders: 174

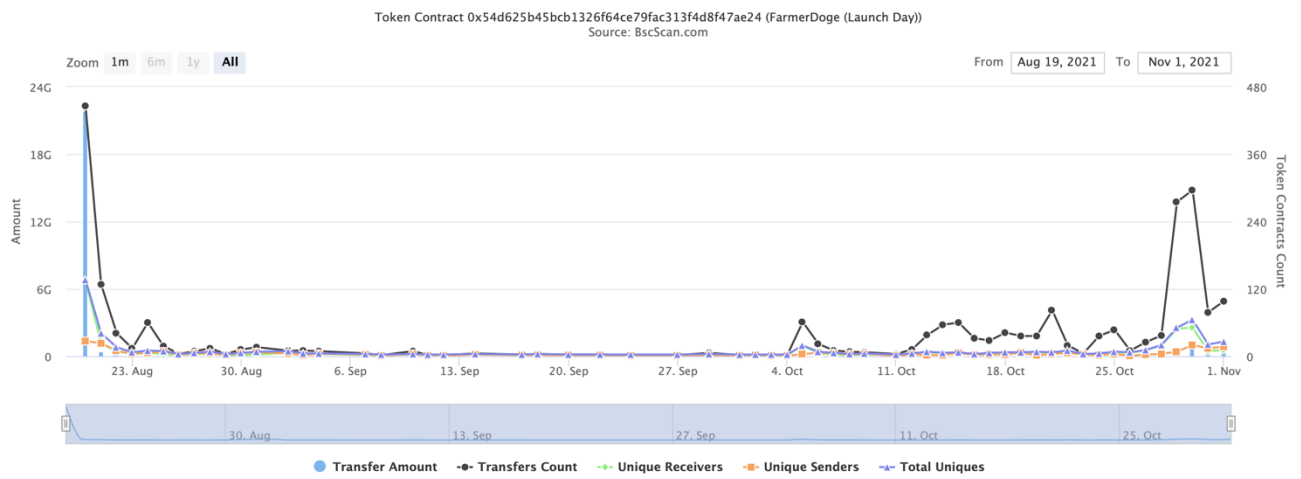


(A total of 9,866,811,686.60 tokens held by the top 100 accounts from the total supply of 10,000,000,000.00 token)

FarmerDoge (Launch Day) Contract Interaction Details

Time Series: Token Contract Overview

Fri 20, Aug 2021 - Mon 1, Nov 2021



FarmerDoge (Launch Day) Top 10 Token Holders

Rank	Address	Quantity	Percentage
1	PancakeSwap V2: CROP 18	1,523,728,250.188498503	15.2373%
2	0x6cdc686a20e28d563443911a33ab0ad4752a31cb	500,000,000	5.0000%
3	0x21371c02669b4404b9f67fbf808013e5851ce0da	127,000,000	1.2700%
4	0x62efc44d63b1146b1001b3974d885464fcc72dd5	127,000,000	1.2700%
5	0x4867433bc8102ca40d728388ba05556fafd943f1	127,000,000	1.2700%
6	0x458449667714941536b0218f6e2e00fa5a369089	127,000,000	1.2700%
7	0x44729af686baa4f2adaedf3f28a82a619e6b861e	127,000,000	1.2700%
8	0x04cafc2caa667503a04a60339a643efe9a3408aa	127,000,000	1.2700%
9	0xd05ca732a473a387f4e800c24618045fc9d76990	127,000,000	1.2700%
10	0x8f7987d1a03551fe86bb7edccac8bfb7c3bfa332	127,000,000	1.2700%

Contract functions details

+ [Lib] Address
  - [Int] isContract
  - [Int] sendValue #
  - [Int] functionCall #
  - [Int] functionCall #
  - [Int] functionCallWithValue #
  - [Int] functionCallWithValue #
  - [Prv] _functionCallWithValue #

+ Context
  - [Int] _msgSender
  - [Int] _msgData

+ [Int] IERC20
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] transfer #
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transferFrom #

+ Ownable (Context)
  - [Pub] <Constructor> #
  - [Pub] owner
  - [Pub] renounceOwnership #
    - modifiers: onlyOwner
  - [Pub] transferOwnership #
    - modifiers: onlyOwner

+ [Int] IDEXFactory
  - [Ext] createPair #

+ [Int] IDEXRouter
  - [Ext] factory
  - [Ext] WETH
  - [Ext] addLiquidity #
  - [Ext] addLiquidityETH ($)
  - [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
  - [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens ($)
  - [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

+ [Int] IDividendDistributor
  - [Ext] changeToken #
  - [Ext] setDistributionCriteria #
  - [Ext] setShare #
  - [Ext] deposit ($)
  - [Ext] process #
  - [Ext] claimDividend #
  - [Ext] checkUnpaidDividends
  - [Ext] checkTokenChangeProgress

+ DividendDistributor (IDividendDistributor)
  - [Pub] <Constructor> #
  - [Ext] changeToken #
    - modifiers: onlyToken
  - [Ext] checkTokenChangeProgress
  - [Int] processTokenChange #
  - [Ext] setDistributionCriteria #
    - modifiers: onlyToken
  - [Ext] setShare #
    - modifiers: onlyToken
  - [Ext] deposit ($)
    - modifiers: onlyToken
  - [Ext] process #
    - modifiers: onlyToken
  - [Int] shouldDistribute
  - [Int] distributeDividend #
  - [Ext] claimDividend #
  - [Pub] getUnpaidEarnings
  - [Ext] checkUnpaidDividends
  - [Int] getCumulativeDividends
  - [Int] addShareholder #
  - [Int] removeShareholder #

+ FarmerDoge (IERC20, Ownable)
  - [Pub] <Constructor> #
  - [Ext] <Fallback> ($)
  - [Ext] totalSupply
  - [Ext] decimals
  - [Ext] symbol
  - [Ext] name
  - [Ext] getOwner
  - [Pub] balanceOf
  - [Ext] allowance
  - [Ext] updateTokenDetails #
    - modifiers: onlyOwner
  - [Ext] airdrop #
    - modifiers: onlyOwner
  - [Pub] approve #
  - [Ext] approveMax #
  - [Ext] transfer #
  - [Ext] transferFrom #
  - [Int] _transferFrom #
  - [Int] _basicTransfer #
  - [Int] checkWalletLimit
  - [Int] checkTxLimit
  - [Ext] setup #
    - modifiers: onlyOwner
  - [Ext] setDiscountToken #
    - modifiers: onlyOwner
  - [Int] shouldTakeFee
  - [Int] getDiscountRate
  - [Ext] checkDiscountRate
  - [Pub] getTotalFee
  - [Int] takeFee #
  - [Int] shouldSwapBack

- [Int] swapBack #
  - modifiers: swapping
- [Int] shouldAutoBuyback
- [Ext] triggerManualBuyback #
  - modifiers: onlyOwner
- [Ext] manualTokenPurchase #
  - modifiers: onlyOwner
- [Ext] clearBuybackMultiplier #
  - modifiers: onlyOwner
- [Int] triggerAutoBuyback #
- [Int] buyTokens #
  - modifiers: swapping
- [Ext] setAutoBuybackSettings #
  - modifiers: onlyOwner
- [Int] launched
- [Int] launch #
- [Ext] setTxLimit #
  - modifiers: onlyOwner
- [Ext] setReflectToken #
  - modifiers: onlyOwner
- [Ext] checkReflectTokenUpdate
  - modifiers: onlyOwner
- [Ext] setMaxWallet #
  - modifiers: onlyOwner
- [Ext] setSellMultiplier #
  - modifiers: onlyOwner
- [Ext] setDumpMultiplier #
  - modifiers: onlyOwner
- [Ext] setDiscountMultiplier #
  - modifiers: onlyOwner
- [Ext] setIsDividendExempt #
  - modifiers: onlyOwner
- [Ext] setIsFeeExempt #
  - modifiers: onlyOwner
- [Ext] setIsTxLimitExempt #
  - modifiers: onlyOwner
- [Ext] setFees #
  - modifiers: onlyOwner
- [Ext] setFeeReceivers #
  - modifiers: onlyOwner
- [Ext] setSwapBackSettings #
  - modifiers: onlyOwner
- [Ext] setTargetLiquidity #
  - modifiers: onlyOwner
- [Ext] setDistributionCriteria #
  - modifiers: onlyOwner
- [Ext] setDistributorSettings #
  - modifiers: onlyOwner
- [Pub] getCirculatingSupply
- [Pub] getLiquidityBacking
- [Pub] isOverLiquified
- [Ext] availableDividends
- [Ext] claimDividends #
- [Ext] processDividends #

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 function `airdrop()` uses the loop to distribute amounts to multiple accounts. Function will be aborted with `OUT_OF_GAS` exception if there will be a long addresses list.

```
function airdrop(address[] memory addresses, uint256[] memory amounts, uint256 vestingPeriod, bool fromContract) external onlyOwner {
    require(addresses.length > 0 && amounts.length > 0 && addresses.length == amounts.length);
    address from = fromContract ? address(this) : msg.sender;
    for (uint i = 0; i < addresses.length; i++) {
        if(balanceOf(addresses[i]) == 0) {
            _allowances[from][addresses[i]] = amounts[i];
            _transferFrom(from, addresses[i], amounts[i]);
            airDropVestingPeriod[addresses[i]] = block.timestamp + vestingPeriod;
            airDropBalance[addresses[i]] = amounts[i];
            airDropped[addresses[i]] = true;
        }
    }
}
```

Recommendation:

Be careful about addresses array length.

Owner privileges (In the period when the owner is not renounced)

- Owner can change name and symbol.

```
function updateTokenDetails(string memory newName, string memory newSymbol) external onlyOwner {
    _name = newName;
    _symbol = newSymbol;
}
```

- Owner can airdrop.

```
function airdrop(address[] memory addresses, uint256[] memory amounts, uint256 vestingPeriod, bool fromContract) external onlyOwner {
    require(addresses.length > 0 && amounts.length > 0 && addresses.length == amounts.length);
    address from = fromContract ? address(this) : msg.sender;
    for (uint i = 0; i < addresses.length; i++) {
        if(balanceOf(addresses[i]) == 0) {
            _allowances[from][addresses[i]] = amounts[i];
            _transferFrom(from, addresses[i], amounts[i]);
            airDropVestingPeriod[addresses[i]] = block.timestamp + vestingPeriod;
            airDropBalance[addresses[i]] = amounts[i];
            airDropped[addresses[i]] = true;
        }
    }
}
```

- Owner can setup reflect token.

```
function setup(address reflectToken) external onlyOwner {
    require(!launched());
    currentlyServing = reflectToken;
    distributor = new DividendDistributor(routerAddress, currentlyServing, WBNB);
    discountTokenMaxWallet = 1;
}
```

- Owner can change discount token.

```
function setDiscountToken(address _discountToken, uint256 _discountMaxWallet) external onlyOwner {
    require(_discountToken.isContract());
    discountToken = IERC20(_discountToken);
    _discountDenominator = (_discountDenominator * _discountMaxWallet) / discountTokenMaxWallet;
    discountTokenMaxWallet = _discountMaxWallet;
}
```

- Owner can trigger manual buyBack.

```
function triggerManualBuyback(uint256 amount, bool triggerBuybackMultiplier) external onlyOwner {
    buyTokens(amount, DEAD);
    if(triggerBuybackMultiplier){
        buybackMultiplierTriggeredAt = block.timestamp;
        emit BuybackMultiplierActive(buybackMultiplierLength);
    }
}
```

- Owner can manual purchase token.

```
function clearBuybackMultiplier() external onlyOwner {
    buybackMultiplierTriggeredAt = 0;
}
```

- Owner can clear buyBack multiplier.

```
function clearBuybackMultiplier() external onlyOwner {
    buybackMultiplierTriggeredAt = 0;
}
```

- Owner can change autoBuyBack settings.

```
function setAutoBuybackSettings(bool _enabled, uint256 _cap, uint256 _amount, uint256 _period) external onlyOwner {
    autoBuybackEnabled = _enabled;
    autoBuybackCap = _cap;
    autoBuybackAccumulator = 0;
    autoBuybackAmount = _amount;
    autoBuybackBlockPeriod = _period;
    autoBuybackBlockLast = block.number;
}
```

- Owner can change transaction token amount limit.

```
function setTxLimit(uint256 numerator, uint256 divisor) external onlyOwner {
    require(numerator > 0 && divisor > 0 && divisor <= 10000);
    _maxTxAmount = (_totalSupply * numerator) / divisor;
}
```

- Owner can change reflect token.

```
function setReflectToken(address newToken, bool forceChange) external onlyOwner {
    require(newToken.isContract(), "Enter valid contract address");
    distributor.changeToken(newToken, forceChange);
    currentlyServing = newToken;
}
```

- Owner can change maximum token amount per wallet.

```
function setMaxWallet(uint256 numerator, uint256 divisor) external onlyOwner() {
    require(numerator > 0 && divisor > 0 && divisor <= 10000);
    _maxWalletSize = (_totalSupply * numerator) / divisor;
}
```

- Owner can change sell, dump and discount multipliers.

```
function setMaxWallet(uint256 numerator, uint256 divisor) external onlyOwner() {
    require(numerator > 0 && divisor > 0 && divisor <= 10000);
    _maxWalletSize = (_totalSupply * numerator) / divisor;
}

function setSellMultiplier(uint256 numerator, uint256 divisor) external onlyOwner() {
    require(divisor > 0 && numerator / divisor <= 3, "Taxes too high");
    _sellMultiplierNumerator = numerator;
    _sellMultiplierDenominator = divisor;
}

function setDumpMultiplier(uint256 numerator, uint256 divisor, uint256 dumpThreshold, uint256 dumpTimer) external onlyOwner() {
    require(divisor > 0 && numerator / divisor <= 2 , "Taxes too high");
    _dumpProtectionNumerator = numerator;
    _dumpProtectionDenominator = divisor * _maxTxAmount;
    _dumpProtectionThreshold = dumpThreshold;
    _dumpProtectionTimer = dumpTimer;
}
```

- Owner can include in and exclude from fees.

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

- Owner can include in and exclude from dividends.

```
function setIsDividendExempt(address holder, bool exempt) external onlyOwner {
    require(holder != address(this) && holder != pair);
    isDividendExempt[holder] = exempt;
    if(exempt){
        distributor.setShare(holder, 0);
    }else{
        distributor.setShare(holder, _balances[holder]);
    }
}
```

- Owner can include in and exclude from transaction limit.

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

- Owner can change liquidity, buyBack, reflection and marketing fees.

```
function setFees(uint256 _liquidityFee, uint256 _buybackFee, uint256 _reflectionFee, uint256 _marketingFee, uint256 _feeDenominator) external onlyOwner {
    liquidityFee = _liquidityFee;
    buybackFee = _buybackFee;
    reflectionFee = _reflectionFee;
    marketingFee = _marketingFee;
    totalFee = _liquidityFee + _buybackFee + _reflectionFee + _marketingFee;
    feeDenominator = _feeDenominator;
    require(totalFee < feeDenominator / 4);
}
```

- Owner can change fee receivers.

```
function setFeeReceivers(address _autoLiquidityReceiver, address _marketingFeeReceiver) external onlyOwner {
    autoLiquidityReceiver = _autoLiquidityReceiver;
    marketingFeeReceiver = payable(_marketingFeeReceiver);
}
```

- Owner can change swapBack settings.

```
function setSwapBackSettings(bool _enabled, uint256 _denominator) external onlyOwner {
    require(_denominator > 0);
    swapEnabled = _enabled;
    swapThreshold = _totalSupply / _denominator;
}
```

- Owner can change target liquidity.

```
function setTargetLiquidity(uint256 _target, uint256 _denominator) external onlyOwner {
    targetLiquidity = _target;
    targetLiquidityDenominator = _denominator;
}
```

- Owner can change distribution criteria.

```
function setDistributionCriteria(uint256 _minPeriod, uint256 _minDistribution) external onlyOwner {
    distributor.setDistributionCriteria(_minPeriod, _minDistribution);
}
```

- Owner can change distribution GAS limit.

```
function setDistributorSettings(uint256 gas) external onlyOwner {
    require(gas < 750000);
    distributorGas = gas;
}
```

Conclusion

Smart contracts contain low severity issues and owner privileges! Liquidity pair contract's security is not checked due to out of scope.

Liquidity locking details provided by the team:
https://dxsale.app/app/v2_9/dxlockview?id=0&add=0x967D716F672CeAC898647A3f27dBd224F3238520&type=lplock&chain=BSC

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.