Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

December, 2021

Audit Details

Audited project

FUTAINU

Deployer address

0x3887354E6E37C8ca3d7a3ba09D2f326A1Cb6c5Bc

Client contacts:

FUTAINU team

Blockchain

Binance Smart Chain

Project website:

Not provided

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 FUTAINU to perform an audit of smart contracts:
https://bscscan.com/address/0xed2cd9e19550899a062b3Ec1A0F64463F401a41a#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

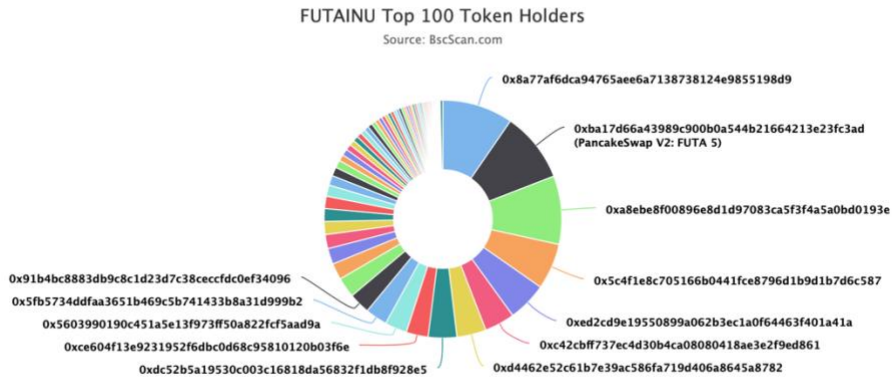
Token contract details for 04.12.2021

Contract name	FUTAINU
Contract address	0xed2cd9e19550899a062b3Ec1A0F64463F401a41a
Total supply	1,000,000,000,000
Token ticker	FUTA
Decimals	18
Token holders	97
Transactions count	318
Top 100 holders dominance	99.63%
Liquidity fee	2
Tax fee	1
Treasure fee	2
Total fees	13,254,917,422.826300088877914450
Uniswap V2 pair	0xba17D66A43989c900b0a544B21664213E23Fc3AD
Contract deployer address	0x3887354E6E37C8ca3d7a3ba09D2f326A1Cb6c5Bc
Contract's current owner address	0xf76C733ec90E6d32755e2052c3075231A8A78075

FUTAINU Token Distribution

The top 100 holders collectively own 99.63% (996,291,448,330.27 Tokens) of FUTAINU

Token Total Supply: 1,000,000,000,000.00 Token | Total Token Holders: 97

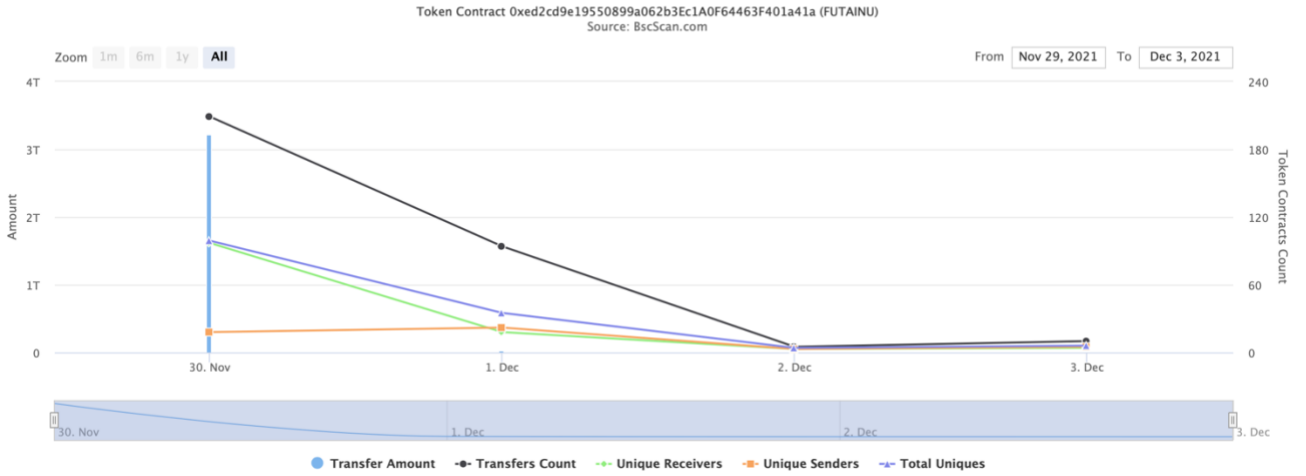


(A total of 996,291,448,330.27 tokens held by the top 100 accounts from the total supply of 1,000,000,000,000.00 token)

FUTAINU Interaction details

Time Series: Token Contract Overview

Tue 30, Nov 2021 - Fri 3, Dec 2021



FUTAINU Top 10 Token Holders

Rank	Address	Quantity	Percentage
1	0x8a77af6dca94765aee6a7138738124e9855198d9	96,162,570,635.940434941749458226	9.6163%
2	PancakeSwap V2: FUTA 5	95,061,814,463.850594834163369449	9.5062%
3	0xa8ebe8f00896e8d1d97083ca5f3f4a5a0bd0193e	93,231,049,764.081045333642333646	9.3231%
4	0x5c4f1e8c705166b0441fce8796d1b9d1b7d6c587	62,519,711,831.007643834822775173	6.2520%
5	0xed2cd9e19550899a062b3ec1a0f64463f401a41a	53,211,705,269.146152091804646079	5.3212%
6	0xc42cbff737ec4d30b4ca08080418ae3e2f9ed861	41,176,329,953.280408080608694675	4.1176%
7	0xd4462e52c61b7e39ac586fa719d406a8645a8782	39,886,815,571.070137029730759925	3.9887%
8	0xdc52b5a19530c003c16818da56832f1db8f928e5	38,786,035,606.885358869103354004	3.8786%
9	0xce604f13e9231952f6dbc0d68c95810120b03f6e	30,021,796,893.652640571206217861	3.0022%
10	0x5603990190c451a5e13f973ff50a822fcf5aad9a	30,009,085,346.528488501967558786	3.0009%

Contract functions details

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

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

+ [Lib] SafeMath
  - [Int] add
  - [Int] sub
  - [Int] sub
  - [Int] mul
  - [Int] div
  - [Int] div
  - [Int] mod
  - [Int] mod

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

+ [Int] IUniswapV2Factory
  - [Ext] feeTo
  - [Ext] feeToSetter
  - [Ext] getPair
  - [Ext] allPairs
  - [Ext] allPairsLength
  - [Ext] createPair #
  - [Ext] setFeeTo #
  - [Ext] setFeeToSetter #

+ [Int] IUniswapV2Pair
  - [Ext] name
  - [Ext] symbol
  - [Ext] decimals
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] allowance

  - [Ext] approve #
  - [Ext] transfer #
  - [Ext] transferFrom #
  - [Ext] DOMAIN_SEPARATOR
  - [Ext] PERMIT_TYPEHASH
  - [Ext] nonces
  - [Ext] permit #
  - [Ext] MINIMUM_LIQUIDITY
  - [Ext] factory
  - [Ext] token0
  - [Ext] token1
  - [Ext] getReserves
  - [Ext] price0CumulativeLast
  - [Ext] price1CumulativeLast
  - [Ext] kLast
  - [Ext] mint #
  - [Ext] burn #
  - [Ext] swap #
  - [Ext] skim #
  - [Ext] sync #
  - [Ext] initialize #

+ [Int] IUniswapV2Router01
  - [Ext] factory
  - [Ext] WETH
  - [Ext] addLiquidity #
  - [Ext] addLiquidityETH ($)
  - [Ext] removeLiquidity #
  - [Ext] removeLiquidityETH #
  - [Ext] removeLiquidityWithPermit #
  - [Ext] removeLiquidityETHWithPermit #
  - [Ext] swapExactTokensForTokens #
  - [Ext] swapTokensForExactTokens #
  - [Ext] swapExactETHForTokens ($)
  - [Ext] swapTokensForExactETH #
  - [Ext] swapExactTokensForETH #
  - [Ext] swapETHForExactTokens ($)
  - [Ext] quote
  - [Ext] getAmountOut
  - [Ext] getAmountIn
  - [Ext] getAmountsOut
  - [Ext] getAmountsIn

+ [Int] IUniswapV2Router02 (IUniswapV2Router01)
  - [Ext] removeLiquidityETHSupportingFeeOnTransferTokens #
  - [Ext] removeLiquidityETHWithPermitSupportingFeeOnTransferTokens #
  - [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
  - [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens ($)
  - [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

+ FUTAINU (Context, IERC20, Ownable)
  - [Pub] <Constructor> #
    - modifiers: Ownable
  - [Pub] name
  - [Pub] symbol

- [Pub] decimals
- [Pub] totalSupply
- [Pub] balanceOf
- [Pub] transfer #
- [Pub] allowance
- [Pub] approve #
- [Pub] transferFrom #
- [Pub] increaseAllowance #
- [Pub] decreaseAllowance #
- [Pub] isExcludedFromReward
- [Pub] isExcludedTxLimit
- [Pub] excludeTxLimit #
  - modifiers: onlyOwner
- [Pub] includeTxLimit #
  - modifiers: onlyOwner
- [Pub] isExcludedMxWalletSize
- [Pub] excludeMxWalletSize #
  - modifiers: onlyOwner
- [Pub] includeMxWalletSize #
  - modifiers: onlyOwner
- [Pub] totalFees
- [Pub] deliver #
- [Pub] reflectionFromToken
- [Pub] tokenFromReflection
- [Pub] excludeFromReward #
  - modifiers: onlyOwner
- [Ext] includeInReward #
  - modifiers: onlyOwner
- [Ext] settreasuryWallet #
  - modifiers: onlyOwner
- [Ext] setExcludedFromFee #
  - modifiers: onlyOwner
- [Ext] updateFees #
  - modifiers: onlyOwner
- [Ext] setMaxTxAmount #
  - modifiers: onlyOwner
- [Ext] setMaxWalletSize #
  - modifiers: onlyOwner
- [Pub] setSwapAndLiquifyEnabled #
  - modifiers: onlyOwner
- [Ext] <Fallback> ($)
- [Ext] setUniswapRouter #
  - modifiers: onlyOwner
- [Ext] setUniswapPair #
  - modifiers: onlyOwner
- [Ext] setExcludedFromAutoLiquidity #
  - modifiers: onlyOwner
- [Prv] _reflectFee #
- [Prv] _getTValues
- [Prv] _getRValues
- [Prv] _getRate
- [Prv] _getCurrentSupply
- [Prv] takeTransactionFee #
- [Prv] calculateFee
- [Pub] isExcludedFromFee

- [Prv] _approve #
- [Prv] _transfer #
- [Prv] swapAndLiquify #
  - modifiers: lockTheSwap
- [Prv] swapTokensForEth #
- [Prv] addLiquidity #
- [Prv] _tokenTransfer #
- [Prv] _transferStandard #
- [Prv] _transferBothExcluded #
- [Prv] _transferToExcluded #
- [Prv] _transferFromExcluded #

($) = 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 function `includeInReward()` uses the loop to find and remove addresses from the `_excluded` list. Function will be aborted with `OUT_OF_GAS` exception if there will be a long excluded addresses list.

```
function includeInReward(address account) external onlyOwner {
    require(_isExcluded[account], "Account is already excluded");

    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

- The function `_getCurrentSupply()` also uses the loop for evaluating total supply. It also could be aborted with `OUT_OF_GAS` exception if there will be a long excluded addresses list.

```
function _getCurrentSupply() private view returns(uint256, uint256) {
    uint256 rSupply = _rTotal;
    uint256 tSupply = _tTotal;
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
        rSupply = rSupply.sub(_rOwned[_excluded[i]]);
        tSupply = tSupply.sub(_tOwned[_excluded[i]]);
    }
    if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
    return (rSupply, tSupply);
}
```

Recommendation:
Check that the excluded array length is not too big.

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

- Owner can lock and unlock. By the way, using these functions the owner could retake privileges even after the ownership was renounced.

```
function lock(uint256 time) public virtual onlyOwner {
    _previousOwner = _owner;
    _owner = address(0);
    _lockTime = block.timestamp + time;
    emit OwnershipTransferred(_owner, address(0));
}

function unlock() public virtual {
    require(_previousOwner == msg.sender, "You don't have permission to unlock");
    require(block.timestamp > _lockTime , "Contract is still locked");
    emit OwnershipTransferred(_owner, _previousOwner);
    _owner = _previousOwner;
}
```

- Owner can include in and exclude from maximum transaction limit.

```
function excludeTxLimit(address account) public onlyOwner {
    _isExcludedFromTxLimit[account] = true;
}

function includeTxLimit(address account) public onlyOwner {
    _isExcludedFromTxLimit[account] = false;
}
```

- Owner can include in and exclude from maximum token per wallet limit.

```
function excludeMxWalletSize(address account) public onlyOwner {
    _isExcludedFromMxWalletSize[account] = true;
}

function includeMxWalletSize(address account) public onlyOwner {
    _isExcludedFromMxWalletSize[account] = false;
}
```

- Owner can change treasure wallet address.

```
function settreasuryWallet(address treasuryWallet) external onlyOwner {
    _treasuryWallet = treasuryWallet;
}
```

- Owner can include in and exclude from fees.

```
function setExcludedFromFee(address account, bool e) external onlyOwner {
    _isExcludedFromFee[account] = e;
}
```

- Owner can include in and exclude from reward.

```
function excludeFromReward(address account) public onlyOwner {
    require(!_isExcluded[account], "Account is already excluded");

    if (_rOwned[account] > 0) {
        _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
}

function includeInReward(address account) external onlyOwner {
    require(_isExcluded[account], "Account is already excluded");

    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

- Owner can change tax, liquidity and treasure fees.

```
function updateFees(uint256 taxFee, uint256 liquidityFee, uint256 treasuryFee) external onlyOwner {
    uint256 _totalFees = taxFee + liquidityFee + treasuryFee;
    require(_totalFees <=15, "Must Keep Fees at 15% or less" );

    _taxFee = taxFee;
    _liquidityFee = liquidityFee;
    _treasuryFee = treasuryFee;
    _swapFee = _liquidityFee + _treasuryFee;

}
```

- Owner can change the maximum transaction limit.

```
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner {
    _maxTxAmount = maxTxAmount;
}
```

- Owner can change maximum token per wallet limit.

```
function setMaxWalletSize(uint256 maxWalletSize) external onlyOwner {
    _maxWalletSize = maxWalletSize;
}
```

- Owner can enable / disable swap and liquify.

```
function setSwapAndLiquifyEnabled(bool e) public onlyOwner {
    _swapAndLiquifyEnabled = e;
    emit SwapAndLiquifyEnabledUpdated(e);
}
```

- Owner can change Uniswap router and pair.

```
function setUniswapRouter(address r) external onlyOwner {
    IUniswapV2Router02 uniswapV2Router = IUniswapV2Router02(r);
    _uniswapV2Router = uniswapV2Router;
}

function setUniswapPair(address p) external onlyOwner {
    _uniswapV2Pair = p;
}
```

- Owner can exclude from and include to autoliquidity.

```
function setExcludedFromAutoLiquidity(address a, bool b) external onlyOwner {
    _isExcludedFromAutoLiquidity[a] = b;
}
```

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/v3/dxlockview?id=0&add=0xf76C733ec90E6d32755e2052c3075231A8A78075&type=tokenlock&chain=BSC

https://dxsale.app/app/v3/dxlplocksearch?id=3&add=0xed2cd9e19550899a062b3Ec1A0F64463F401a41a&type=lplock1&chain=BSC

https://dxsale.app/app/v3/dxlplocksearch?id=2&add=0xed2cd9e19550899a062b3Ec1A0F64463F401a41a&type=lplock1&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.