Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

June, 2021

Audit Details

Audited project

FairLife

Deployer address

0xcF2f935dF98e4bE1733Ce2BD86D0E0bcE5427E7d

Client contacts:

FairLife team

Blockchain

Binance Smart Chain

Project website:

Not provided by FairLife team

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 FairLife to perform an audit of smart contracts:
https://bscscan.com/address/0x8a646ec31EE33B12FF47E6C7DAaF4BC4df9ae54a#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

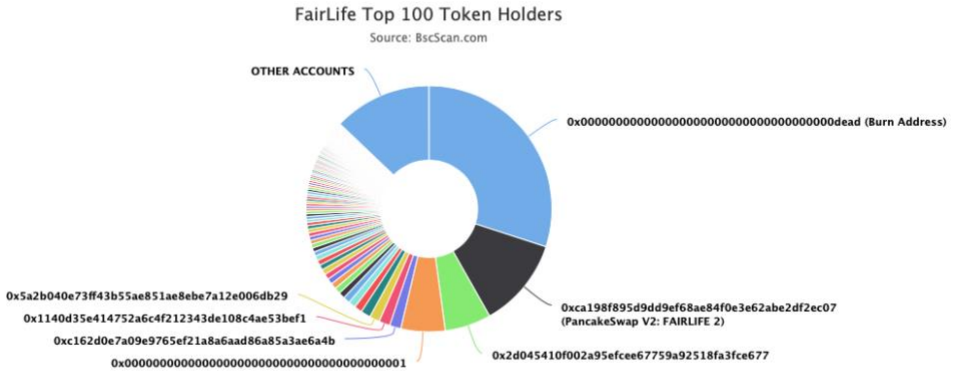
Token contract details for 14.06.2021

Contract name	FairLife
Contract address	0x8a646ec31EE33B12FF47E6C7DAaF4BC4df9ae54a
Total supply	1,000,000,000,000,000
Token ticker	FAIRLIFE
Decimals	8
Token holders	5,858
Transactions count	12,710
Top 100 holders dominance	87.12%
Liquidity fee	4
Tax fee	2
Total fees	5522670810805006191914
Uniswap V2 pair	0xca198f895d9dd9ef68ae84f0e3e62abe2df2ec07
Contract deployer address	0xcF2f935dF98e4bE1733Ce2BD86D0E0bcE5427E7d
Contract's current owner address	0xcf2f935df98e4be1733ce2bd86d0e0bce5427e7d

FairLife Token Distribution

The top 100 holders collectively own 87.12% (871,234,815,781,974.00 Tokens) of FairLife

Token Total Supply: 1,000,000,000,000,000.00 Token | Total Token Holders: 5,858

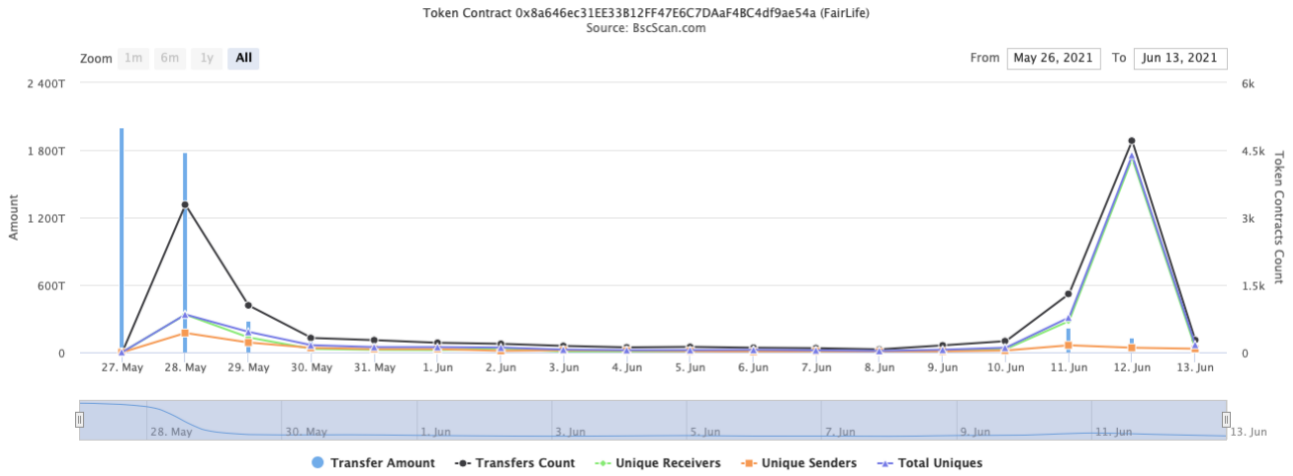


(A total of 871,234,815,781,974.00 tokens held by the top 100 accounts from the total supply of 1,000,000,000,000,000.00 token)

FairLife Contract Interaction Details

Time Series: Token Contract Overview

Thu 27, May 2021 - Sun 13, Jun 2021



FairLife Top 10 Token Holders

Rank	Address	Quantity (Token)	Percentage
1	Burn Address	300,000,000,000,000	30.0000%
2	PancakeSwap V2: FAIRLIFE 2	118,102,392,675,726.27537345	11.8102%
3	0x2d045410f002a95efcee67759a92518fa3fce677	60,800,000,000,000	6.0800%
4	0x0000000000000000000000000000000000000001	58,085,034,541,935.82520735	5.8085%
5	0xc162d0e7a09e9765ef21a8a6aad86a85a3ae6a4b	14,893,380,204,785.80550712	1.4893%
6	0x1140d35e414752a6c4f212343de108c4ae53bef1	14,742,030,918,954.43320299	1.4742%
7	0x5a2b040e73ff43b55ae851ae8ebe7a12e006db29	13,281,668,797,889.29905014	1.3282%
8	0x12b641deecde43ebd20194c1722929b07beeffcb	12,895,461,446,156.38328753	1.2895%
9	0xeedc9d63364c6815f5d3e5ed9fcdda0a6feb6be1	10,395,174,176,616.01190184	1.0395%
10	0x29dd4d42dea7aaa68820e4d3e43bbbbae1c73cc8	9,479,891,130,016.5982989	0.9480%

FairLife LP Token Holders

Rank	Address	Quantity	Percentage
1	0xd81fe52ae0f7b1684ad844da2def5609cfc8da79	2,065.17375579646134508	78.3642%
2	0xcf2f935df98e4be1733ce2bd86d0e0bce5427e7d	560.836364494240761598	21.2812%
3	0x07d80ae6f36a5e08dca74ce884a24d39db9934ed	9.34430921853784858	0.3546%
4	0x0000000000000000000000000000000000000000	0.000000000000001	0.0000%

Contract functions details

```
+ [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

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

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

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

+ FairLife (Context, IERC20, Ownable)
- [Pub] <Constructor> #
- [Pub] lockTimeOfWallet
- [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] totalFees
- [Pub] lockWallet #
- [Pub] deliver #
- [Pub] reflectionFromToken
- [Pub] tokenFromReflection
- [Pub] excludeFromReward #
  - modifiers: onlyOwner
- [Ext] includeInReward #
  - modifiers: onlyOwner
- [Prv] _transferBothExcluded #
- [Pub] excludeFromFee #
  - modifiers: onlyOwner
- [Pub] setCharityAddress #
  - modifiers: onlyOwner
- [Pub] setMarketingDevAddress #
  - modifiers: onlyOwner
- [Pub] showCharityaddress
- [Pub] showMarketingaddress
- [Pub] includeInFee #
  - modifiers: onlyOwner
- [Ext] setCharityFeePercent #
  - modifiers: onlyOwner
- [Ext] setTaxFeePercent #
  - modifiers: onlyOwner
- [Ext] setMarketingDevFeePercent #
  - modifiers: onlyOwner
- [Ext] setLiquidityFeePercent #
  - modifiers: onlyOwner
- [Ext] setMaxTxPercent #
  - modifiers: onlyOwner
- [Pub] setSwapAndLiquifyEnabled #
  - modifiers: onlyOwner
- [Ext] preparePresale #
  - modifiers: onlyOwner
- [Ext] afterPresale #

  - modifiers: onlyOwner
- [Ext] <Fallback> ($)
- [Prv] _reflectFee #
- [Prv] _getValues
- [Prv] _getTValues
- [Prv] _getRValues
- [Prv] _getRate
- [Prv] _getCurrentSupply
- [Prv] _takeLiquidity #
- [Prv] calculateTaxFee
- [Prv] calculateLiquidityPlusCharityFee
- [Prv] removeAllFee #
- [Prv] restoreAllFee #
- [Pub] isExcludedFromFee
- [Prv] _approve #
- [Prv] _transfer #
- [Prv] swapAndLiquify #
  - modifiers: lockTheSwap
- [Prv] swapTokensForEth #
- [Prv] addLiquidity #
- [Prv] _tokenTransfer #
- [Prv] _transferStandard #
- [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.	Medium issue
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

1. TakeFee always true

Issue:

- The function _transfer(address from, ...) checks for excluded from fee addresses and should disable fees if there are some. But takeFee parameter never turns to false value.

```
//indicates if fee should be deducted from transfer
bool takeFee = true;

//if any account belongs to _isExcludedFromFee account then remove the fee
if(_isExcludedFromFee[from] || _isExcludedFromFee[to]){
    takeFee = true;
}
```

Recommendation:

This code block is not needed if you don't want to remove fees. Otherwise, takeFee value should be false after excluded addresses checking if there are some.

Low Severity Issues

2. 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 change the tax, marketing, charity and liquidity fee.

```
ftrace | funcSig
function setCharityFeePercent(uint256 charityFee) external onlyOwner {
    _charityFee = 0;
    if(charityFee < 6) {
        _charityFee = charityFee;
    }
}

ftrace | funcSig
function setTaxFeePercent(uint256 taxFee) external onlyOwner {
    _taxFee = 0;
    if(taxFee < 6) {
        _taxFee = taxFee;
    }
}

ftrace | funcSig
function setMarketingDevFeePercent(uint256 marketingAndDevBudget) external onlyOwner {
    _marketingAndDevBudget = 0;
    if(marketingAndDevBudget < 6) {
        _marketingAndDevBudget = marketingAndDevBudget;
    }
}

ftrace | funcSig
function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner {
    _liquidityFee = 0;
    if(liquidityFee < 6) {
        _liquidityFee = liquidityFee;
    }
}
```

- Owner can change the maximum transaction amount.

```
ftrace | funcSig
function setMaxTxPercent(uint256 maxTxPercent) external onlyOwner {
    _maxTxAmount = _tTotal.mul(maxTxPercent).div(
        10**3
    );
}
```

- Owner can exclude from the fee.

```
function excludeFromFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = true;
}
```

- Owner can change charity and marketing addresses.

```
ftrace | funcSig
function setCharityAddress(address payable charity) public onlyOwner {
    _charityAddress = charity;
}

ftrace | funcSig
function setMarketingDevAddress(address payable marketing) public onlyOwner {
    _marketingDevAddress = marketing;
}
```

- Owner can enable after presale mode(transaction amount = 0,5%, all fees, swap to liquidity).

```
ftrace | funcSig
function afterPresale() external onlyOwner {
    _maxTxAmount = _tTotal.mul(5).div(
        10**3
    );
    restoreAllFee();
    swapAndLiquifyEnabled = true;
}
```

- Owner can enable presale mode(transaction amount = 100%, no fee, no swap to liquidity).

```
function preparePresale() external onlyOwner {
    _maxTxAmount = _tTotal.mul(100).div(
        10**2
    );
    removeAllFee();
    swapAndLiquifyEnabled = false;
}
```

- Owner can lock and unlock. By the way, using these functions the owner could leave as owner 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));
}

//Unlocks the contract for owner when _lockTime is exceeds
ftrace | funcSig
function unlock() public virtual {
    require(_previousOwner == msg.sender, "You don't have permission to unlock");
    require(block.timestamp > _lockTime , "Contract is locked until 7 days");
    emit OwnershipTransferred(_owner, _previousOwner);
    _owner = _previousOwner;
}
```

Conclusion

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

Liquidity locking details NOT provided by the team.

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.