Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

July, 2021

Audit Details

Audited project

WIZARD

Deployer address

0x500C34c97C2e97841C25090Fd16408975F73192b

Client contacts:

WIZARD team

Blockchain

Binance Smart Chain

Project website:

https://wizard.financial/

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 WIZARD to perform an audit of smart contracts:

https://bscscan.com/address/0x5066c68cae3b9bdacd6a1a37c90f2d1723559d18#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

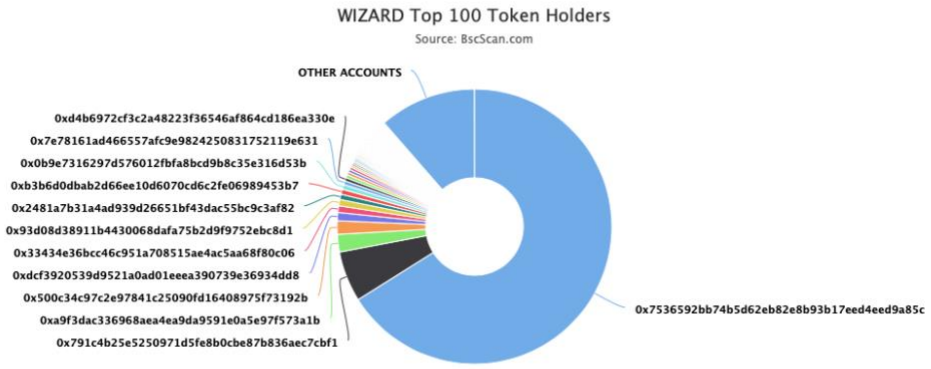
Token contract details for 20.07.2021

Contract name	WIZARD
Contract address	0x5066C68cAe3B9BdaCD6A1A37c90F2d1723559D18
Total supply	1,983,367.5436
Token ticker	WIZARD
Decimals	18
Token holders	3,862
Transactions count	10,137
Top 100 holders dominance	88.58%
Liquidity fee	1
Tax fee	1
Total fees	16632456399331222589231
Uniswap V2 pair	0x791c4b25e5250971d5fe8b0cbe87b836aec7cbf1
Contract deployer address	0x500C34c97C2e97841C25090Fd16408975F73192b
Contract's current owner address	0x500c34c97c2e97841c25090fd16408975f73192b

WIZARD Token Distribution

The top 100 holders collectively own 88.58% (1,756,800.59 Tokens) of WIZARD

Token Total Supply: 1,983,367.52 Token | Total Token Holders: 3,862

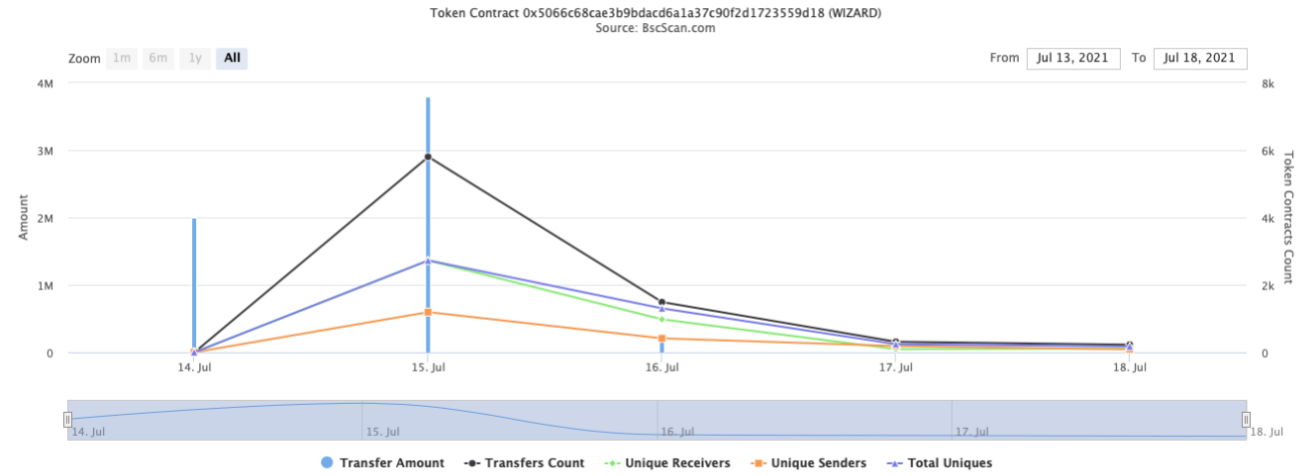


(A total of 1,756,800.59 tokens held by the top 100 accounts from the total supply of 1,983,367.52 token)

WIZARD Contract Interaction Details

Time Series: Token Contract Overview

Wed 14, Jul 2021 - Sun 18, Jul 2021



WIZARD Top 10 Token Holders

Rank	Address	Quantity (Token)	Percentage
1	0x7536592bb74b5d62eb82e8b93b17eed4eed9a85c	1,310,843.423180526566234525	66.0918%
2	0x791c4b25e5250971d5fe8b0cbe87b836aec7cbf1	117,936.77700725658240487	5.9463%
3	0xa9f3dac336968aea4ea9da9591e0a5e97f573a1b	40,816.601282105213864046	2.0579%
4	0x500c34c97c2e97841c25090fd16408975f73192b	31,300.170938588400689444	1.5781%
5	0xdcf3920539d9521a0ad01eeea390739e36934dd8	19,187.629920637742264072	0.9674%
6	0x33434e36bcc46c951a708515ae4ac5aa68f80c06	15,462.200438177323427419	0.7796%
7	0x93d08d38911b4430068dafa75b2d9f9752ebc8d1	15,005.881975391926888947	0.7566%
8	0x2481a7b31a4ad939d26651bf43dac55bc9c3af82	12,090.189011107219163465	0.6096%
9	0xb3b6d0dbab2d66ee10d6070cd6c2fe06989453b7	11,853.736731654231485748	0.5977%
10	0x0b9e7316297d576012fbfa8bcd9b8c35e316d53b	11,434.805266605445006403	0.5765%

WIZARD Top 10 LP Token Holders

Rank	Address	Quantity	Percentage
1	0x7536592bb74b5d62eb82e8b93b17eed4eed9a85c	7,700	84.0339%
2	0xd4b6972cf3c2a48223f36546af864cd186ea330e	513.798165104727317	5.6073%
3	0x7b3ca828e189739660310b47fc89b3a3e8a0e564	490.090635258284627509	5.3486%
4	0x500c34c97c2e97841c25090fd16408975f73192b	396.420664878043304714	4.3263%
5	0x07d80ae6f36a5e08dca74ce884a24d39db9934ed	43.691104249409179786	0.4768%
6	0xa71ed00c0e6aceec3401b2e8f2207edd96df7635	17.128868243488026972	0.1869%
7	0x0f80bc7dcc2af8e698d4c21ec16de1c441f7dbcf	1.380738700174829538	0.0151%
8	0x3f92fb7fc67dd27e2548bb3e5a57fe942f8fa7cb	0.359215110662005354	0.0039%
9	0x4b6067ffda38a4fa4647fdef8417aad7d9b413d0	0.085251734238139552	0.0009%
10	0x6233fcb52e6d9a401c1fb57a6e46805c6ebdbad7	0.009112866039799775	0.0001%

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] 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 #

+ WIZARD (Context, IERC20, Ownable)
  - [Pub] <Constructor> #
  - [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] totalBurn
  - [Pub] deliver #
  - [Pub] reflectionFromToken
  - [Pub] tokenFromReflection
  - [Pub] excludeFromReward #
    - modifiers: onlyOwner
  - [Ext] includeInReward #
    - modifiers: onlyOwner
  - [Prv] _transferBothExcluded #
  - [Pub] excludeFromFee #
    - modifiers: onlyOwner
  - [Pub] includeInFee #
    - modifiers: onlyOwner
  - [Ext] setTaxFeePercent #
    - modifiers: onlyOwner
  - [Ext] setLiquidityFeePercent #
    - modifiers: onlyOwner
  - [Ext] setBurnFeePercent #
    - modifiers: onlyOwner
  - [Ext] setCharityFeePercent #
    - modifiers: onlyOwner
  - [Ext] setMaxTxPercent #
    - modifiers: onlyOwner
  - [Pub] setSwapAndLiquifyEnabled #
    - modifiers: onlyOwner
  - [Ext] <Fallback> ($)
  - [Prv] _reflectFee #
  - [Prv] _getValues
  - [Prv] _getTValues
  - [Prv] _getRValues
  - [Prv] _getRate
  - [Prv] _getCurrentSupply
  - [Prv] _takeLiquidity #
  - [Prv] calculateTaxFee
  - [Prv] calculateLiquidityFee
  - [Prv] calculateBurnFee
```

- [Prv] calculateCharityFee
- [Pub] addBlacklist #
  - modifiers: onlyOwner
- [Pub] removeBlackList #
  - modifiers: onlyOwner
- [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 #
- [Pub] changecharitywallet #
  - modifiers: onlyOwner
- [Pub] changeminTokenNumberToSell #
  - modifiers: onlyOwner

($) = 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 change the tax, burn, charity and liquidity fee.

```
ftrace | funcSig
function setTaxFeePercent(uint256 taxFee↑) external onlyOwner() {

    uint256 newfees = taxFee↑ + _liquidityFee + _burnFee + _charityFee;
    require (newfees <= 5, "Slippage must be less than 5%");
    _taxFee = taxFee↑;
}

ftrace | funcSig
function setLiquidityFeePercent(uint256 liquidityFee↑) external onlyOwner() {

    uint256 newfees = _taxFee + liquidityFee↑ + _burnFee + _charityFee;
    require (newfees <= 5, "Slippage must be less than 5%");
    _liquidityFee = liquidityFee↑;
}

ftrace | funcSig
function setBurnFeePercent(uint256 burnfee↑) external onlyOwner() {

    uint256 newfees = _taxFee + _liquidityFee + burnfee↑ + _charityFee;
    require (newfees <= 5, "Slippage must be less than 5%");
    _burnFee = burnfee↑;
}

ftrace | funcSig
function setCharityFeePercent(uint256 charityfee↑) external onlyOwner() {

    uint256 newfees = _taxFee + _liquidityFee + _burnFee + charityfee↑;
    require (newfees <= 5, "Slippage must be less than 5%");
    _charityFee = charityfee↑;
}
```

- Owner can change the maximum transaction amount.

```
ftrace | funcSig
function setMaxTxPercent(uint256 maxTxPercent↑) external onlyOwner() {

    _maxTxAmount = _tTotal.mul(maxTxPercent↑).div(
        10**2
    );
}
```

- Owner can exclude from the fee.

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

- Owner can add and remove from blacklist.

```
ftrace | funcSig
function addBlacklist(address _defaulter↑) public onlyOwner {

    blacklist[_defaulter↑] = true;
}

ftrace | funcSig
function removeBlackList(address _defaulter↑) public onlyOwner {

    blacklist[_defaulter↑] = false;
}
```

- **Owner can change charity wallet.**

```
ftrace | funcSig
function changecharitywallet(address _newaddress) public onlyOwner {

    charityaddress = _newaddress;
      // exclude charity from fee
    _isExcludedFromFee[charityaddress] = true;

    // exclude charity from standard 4% reflection distribution reward
    _isExcluded[charityaddress] = true;
    _excluded.push(charityaddress);

}
```

- **Owner can change minimum number of tokens to sell to add to liquidity.**

```
ftrace | funcSig
function changeminTokenNumberToSell(uint256 _newvalue)public onlyOwner {

      numTokensSellToAddToLiquidity = _newvalue;
}
```

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

```
//Locks the contract for owner for the amount of time provided
function lock(uint256 time) public virtual onlyOwner {
    _previousOwner = _owner;
    _owner = address(0);
    _lockTime = now + time;
    emit OwnershipTransferred(_owner, address(0));
}

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

Conclusion

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

Liquidity locking details provided by the team:
https://bscscan.com/tx/0xa5dd1dcff8b5535c329b78b0c8dcf33fa7a9e5ee2e3400636432e8878dfc48df

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.