RugFreeCoins Audit

Win Token

Smart Contract Security Audit

June 11, 2021

Contents

Audit details

Audited project

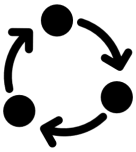
WIN Token

Contract Address

0xe4d353048993d6f79bf65bb5e4f7faffe3639eb3

Client contact

WIN Token Team



Blockchain

Binance smart chain

Project website

https://www.wintoken.online/

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 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 Rugfreecoins and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Rugfreecoins) owe no duty of care towards you or any other person, nor does Rugfreecoins 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 Rugfreecoins 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, Rugfreecoins hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Rugfreecoins, 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

Rugfreecoins was commissioned by WIN Token to perform an audit of the smart contract.

https://bscscan.com/address/0xe4d353048993d6f79bf65bb5e4f7faffe3639eb3

The focus of this audit is to verify that the smart contract is secure, resilient and working according to the specifications.

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

About the project

WIN ($TICKETS) aspires to become the most known and recognized lottery token in the crypto sphere. In order for a cryptocurrency to grow and gain traction, especially in the Altcoin market, it must have a ‘use-case’, which only usually comes with the promise of a better future. The WIN team is motivated by the idea that the coin will have a use-case from day one! The gambling industry has been around for centuries, and there will be an ever-growing crowd of ‘gamblers’ and players in the crypto sphere, as cryptocurrencies slowly become the staple in terms of money transactions around the world. In order to support this transition, WIN wishes to establish itself as the most competitive and well-known lottery token in the industry, all the while progressively growing a following.

With WIN, the chances of winning are relative to how many tokens you hold, which means that all holders are incentivized to buy more tokens in the long term if they wish to increase their chances of winning the lottery.

In addition, the requirement to be eligible at winning the lottery is relatively low, at the value of 777,777 tokens, or around $4 at launch, which means all holders are incentivized to HODL in the hopes of winning a huge prize.

Tokenomics

8% fee per every transaction.

- **1%** of every trade goes to Liquidity Pool.
- **1%** of every trade goes to holders pockets.
- **1%** of every trade token will get burnt.
- **5%** of every trade transfer to BNBs for daily prize.

Target market and the concept

Target market

- Anyone who is ready to hold and be eligible to win in the daily lottery.
- Anyone who is ready to hold a large portion of tokens and be eligible to get a high chance of winning in the daily lottery.
- Anyone who's interested in earning a passive income. (1% interest for the investment per each transaction)
- Traders to balance the market.

Core concept

WIN ($TICKETS) is with a strong use case specifically targeting the gambling industry aiming any long-term believers and holders to give a chance to be eligible for the daily lottery and win. The most unique core part of the WIN is that the chances of winning are relative to how many tokens investors hold, which means that all holders are incentivized to buy more tokens in the long term if they wish to increase their chances of winning the lottery.

The concept encourages,

- Investors hold the token for a long time, which makes them believe in the project and keep the hopes high of expecting to win a huge prize at once.
- Investors buy more and more since the chance of winning is higher.
- The concept is revolutionary and certainly can get the attraction of new investors as the project progresses along.
- Project market price and market cap can keep stable if everything goes according to the plan since keeping tokens will seem more profitable than selling.

How chances of winning are calculated

Chances of winning will be calculated in indirect proportion to how many tokens each holder has. This means that having more tokens does increase your chances of winning, but not in linear fashion. Instead, a logarithmic function will be used to convert the proportion of holdings that each investor has and calculate their chances of winning accordingly. This will lower the discrepancy in probability of winning between a whale and a small investor, while keeping our largest investors at an advantage.

Below table demonstrates the chance of winning the daily lottery:

No. of tokens	% chance of winning	Log transformation	% of winning (log transformation)
15	0.50	1.17609	0.36784
07	0.23	0.84510	0.26432
05	0.17	0.69897	0.21861
03	0.10	0.477120	0.14923
Total		**3.19728**	**1**

Lottery prize distribution

The token will be distributed manually among the winners. The converted BNB from transactions will be sent to one wallet address, which will be known to the public. The wallet address has been coded into WIN contract is the following:

0xdDFA5a69196e525cb1dEC148406D41A0fA340b34

Everyone will be able to monitor this wallet address and be reassured that tokens are distributed to the randomly chosen winners.

Winners will be chosen on a random draw, live every day on WIN's Discord Channel.

Prizes will not be distributed on a daily, but weekly basis and prizes will be distributed as follows:

- One winner daily
- Seven winners for a week (seven days)

The total amount for the seven days will be split into 7 portions,

- 6/7 portion will be equally distributed among the winners.
- 1/7 portion will be allocated for marketing

Potential to grow with score points

1.	Project efficiency	7/10
2.	Project uniqueness	8/10
3	Information quality	8/10
4	Service quality	7/10
5	System quality	7/10
6	Impact on the community	7/10
7	Impact on the business	8/10
8	Preparing for the future	7/10
Total Points		**7.38/10**

Contract details

Token contract details for 11th June 2021 (Day before launch)

Contract name	WIN Token
Contract address	0xe4d353048993d6f79bf65bb5e4f7faffe3639eb3
Token supply	7,777,777,777 $TICKETS
Token ticker	$TICKETS
Decimals	9
Token holders	4
Transaction count	7
Contract deployer address	0x364baa97738eB6C288a50D2205379C168b5AC5c7
Contract's current owner address	0x364baa97738eb6c288a50d2205379c168b5ac5c7

Token distribution

As at 11th June 2021 (Day before the launch)

Token will be distributed as follows:

Foundation Wallet	• 3%
Marketing Wallet	• 2%
Airdrops	• 1%
Private Sale	• 9%
Manual Burn	• 50%
Locked LP	• 15%
Pre-Sale	• 20%

Contract code function details

No	Category	Item	Result
1	Coding conventions	BRC20 Token standards	pass
		compile errors	pass
		Compiler version security	pass
		visibility specifiers	pass
		Gas consumption	low issue
		SafeMath features	pass
		Fallback usage	pass
		tx.origin usage	pass
		deprecated items	pass
		Redundant code	pass
		Overriding variables	pass
2	Function call audit	Authorization of function call	pass
		Low level function (call/delegate call) security	pass
		Returned value security	pass
		Selfdestruct function security	pass
3	Business security	Access control of owners	pass
		Business logics	pass
		Business implementations	pass
4	Integer overflow/underflow		pass
5	Reentrancy		pass
6	Exceptional reachable state		pass
7	Transaction ordering dependence		pass
8	Block properties dependence		pass
9	Pseudo random number generator (PRNG)		pass
10	DoS (Denial of Service)		pass
11	Token vesting implementation		pass
12	Fake deposit		pass
13	Event security		pass

Below table represents the summary of the contracts and methods in the token contract. We scanned the whole contract and listed down all the Interfaces, functions and implementations with its visibility and mutability.

Contract	**Type**	**Bases**		
L	Function Name	Visibility	Mutability	Modifiers
IERC20	**Interface**			
L	totalSupply	External ❗		NO❗
L	balanceOf	External ❗		NO❗
L	transfer	External ❗	🛑	NO❗
L	allowance	External ❗		NO❗
L	approve	External ❗	🛑	NO❗
L	transferFrom	External ❗	🛑	NO❗
SafeMath	**Library**			
L	add	Internal 🔒		
L	sub	Internal 🔒		
L	sub	Internal 🔒		
L	mul	Internal 🔒		
L	div	Internal 🔒		

└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
Context	**Implementation**			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Address	**Library**			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	_functionCallWithValue	Private 🔐	🛑	
Ownable	**Implementation**	**Context**		
└		Internal 🔒	🛑	

L	owner	Public ❗		NO❗
L	renounceOwnership	Public ❗	🛑	onlyOwner
L	transferOwnership	Public ❗	🛑	onlyOwner
L	geUnlockTime	Public ❗		NO❗
L	lock	Public ❗	🛑	onlyOwner
L	unlock	Public ❗	🛑	NO❗
IUniswapV2Factory	**Interface**			
L	feeTo	External ❗		NO❗
L	feeToSetter	External ❗		NO❗
L	getPair	External ❗		NO❗
L	allPairs	External ❗		NO❗
L	allPairsLength	External ❗		NO❗
L	createPair	External ❗	🛑	NO❗
L	setFeeTo	External ❗	🛑	NO❗
L	setFeeToSetter	External ❗	🛑	NO❗
IUniswapV2Pair	**Interface**			
L	name	External ❗		NO❗

└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transfer	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	DOMAIN_SEPARATOR	External ❗		NO❗
└	PERMIT_TYPEHASH	External ❗		NO❗
└	nonces	External ❗		NO❗
└	permit	External ❗	🛑	NO❗
└	MINIMUM_LIQUIDITY	External ❗		NO❗
└	factory	External ❗		NO❗
└	token0	External ❗		NO❗
└	token1	External ❗		NO❗
└	getReserves	External ❗		NO❗
└	price0CumulativeLast	External ❗		NO❗

L	price1CumulativeLast	External ❗		NO❗
L	kLast	External ❗		NO❗
L	mint	External ❗	🛑	NO❗
L	burn	External ❗	🛑	NO❗
L	swap	External ❗	🛑	NO❗
L	skim	External ❗	🛑	NO❗
L	sync	External ❗	🛑	NO❗
L	initialize	External ❗	🛑	NO❗
IUniswapV2Router01	**Interface**			
L	factory	External ❗		NO❗
L	WETH	External ❗		NO❗
L	addLiquidity	External ❗	🛑	NO❗
L	addLiquidityETH	External ❗	💵	NO❗
L	removeLiquidity	External ❗	🛑	NO❗
L	removeLiquidityETH	External ❗	🛑	NO❗
L	removeLiquidityWithPermit	External ❗	🛑	NO❗
L	removeLiquidityETHWithPermit	External ❗	🛑	NO❗

L	swapExactTokensForTokens	External ❗	🛑	NO❗
L	swapTokensForExactTokens	External ❗	🛑	NO❗
L	swapExactETHForTokens	External ❗	💵	NO❗
L	swapTokensForExactETH	External ❗	🛑	NO❗
L	swapExactTokensForETH	External ❗	🛑	NO❗
L	swapETHForExactTokens	External ❗	💵	NO❗
L	quote	External ❗		NO❗
L	getAmountOut	External ❗		NO❗
L	getAmountIn	External ❗		NO❗
L	getAmountsOut	External ❗		NO❗
L	getAmountsIn	External ❗		NO❗
IUniswapV2Router02	**Interface**	**IUniswapV2Router01**		
L	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
L	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗

L	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
ADMAMANT	**Implementation**	**Context, IERC20, Ownable**		
L		Public ❗	🛑	NO❗
L	name	Public ❗		NO❗
L	symbol	Public ❗		NO❗
L	decimals	Public ❗		NO❗
L	totalSupply	Public ❗		NO❗
L	balanceOf	Public ❗		NO❗
L	transfer	Public ❗	🛑	NO❗
L	allowance	Public ❗		NO❗
L	approve	Public ❗	🛑	NO❗
L	transferFrom	Public ❗	🛑	NO❗
L	increaseAllowance	Public ❗	🛑	NO❗
L	decreaseAllowance	Public ❗	🛑	NO❗
L	isExcludedFromReward	Public ❗		NO❗

L	totalFees	Public ❗		NO❗
L	totalDonationBNB	Public ❗		NO❗
L	deliver	Public ❗	🛑	NO❗
L	reflectionFromToken	Public ❗		NO❗
L	tokenFromReflection	Public ❗		NO❗
L	excludeFromReward	Public ❗	🛑	onlyOwner
L	includeInReward	External ❗	🛑	onlyOwner
L	_transferBothExcluded	Private 🔐	🛑	
L		External ❗	💵	NO❗
L	_reflectFee	Private 🔐	🛑	
L	_getValues	Private 🔐		
L	_getTValues	Private 🔐		
L	_getRValues	Private 🔐		
L	_getRate	Private 🔐		
L	_getCurrentSupply	Private 🔐		
L	_takeLiquidity	Private 🔐	🛑	
L	calculateTaxFee	Private 🔐		
L	calculateLiquidityFee	Private 🔐		

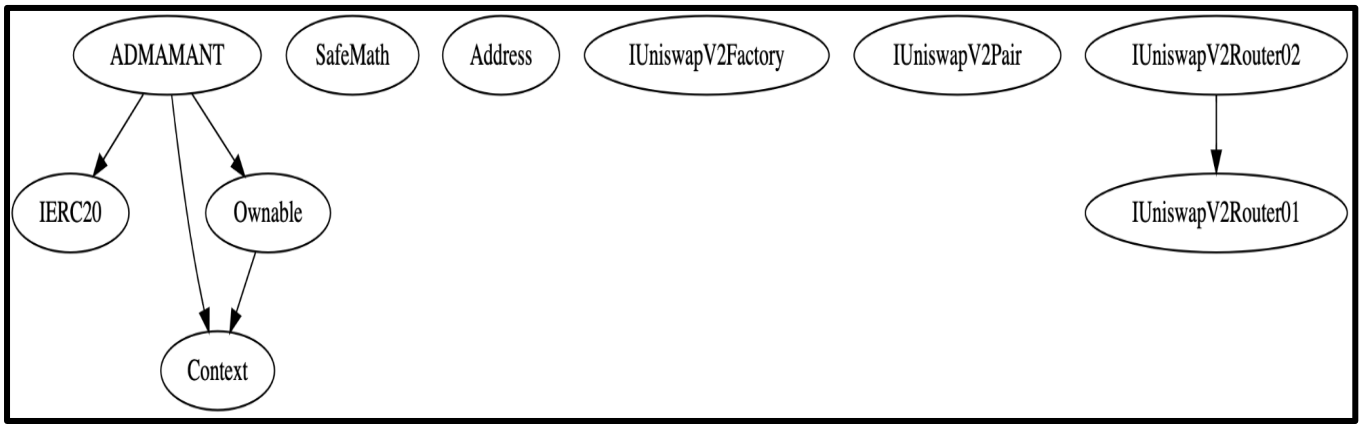
L	calculateDonationFee	Private 🔐		
L	removeAllFee	Private 🔐	🛑	
L	restoreAllFee	Private 🔐	🛑	
L	isExcludedFromFee	Public ❗		NO❗
L	_approve	Private 🔐	🛑	
L	_transfer	Private 🔐	🛑	
L	swapAndLiquify	Private 🔐	🛑	lockTheSwap
L	swapTokensForEth	Private 🔐	🛑	
L	addLiquidity	Private 🔐	🛑	
L	_tokenTransfer	Private 🔐	🛑	
L	_transferStandard	Private 🔐	🛑	
L	_transferToExcluded	Private 🔐	🛑	
L	_transferFromExcluded	Private 🔐	🛑	
L	excludeFromFee	Public ❗	🛑	onlyOwner
L	includeInFee	Public ❗	🛑	onlyOwner
L	enableAllFees	External ❗	🛑	onlyOwner
L	disableAllFees	External ❗	🛑	onlyOwner
L	setMaxTxPercent	External ❗	🛑	onlyOwner

L	setSwapAndLiquifyEnabled	Public ❗	🛑	onlyOwner
L	TransferCharityETH	Private 🔐	🛑	

Legend

Symbol	Meaning
🛑	**Function can modify state**
💵	**Function is payable**

Inheritance Hierarchy



Security issue checking status

❖ **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)

- Renouncing ownership will leave the contract without an owner

```
function renounceOwnership() public virtual onlyOwner {
    emit OwnershipTransferred(_owner, address(0));
    _owner = address(0);
}
```

- Owner can transfer the ownership

```
function transferOwnership(address newOwner) public virtual onlyOwner {
    require(newOwner != address(0), "Ownable: new owner is the zero address");
    emit OwnershipTransferred(_owner, newOwner);
    _owner = newOwner;
}
```

- The contract will lock for the owner after transferring the ownership by specifying a period, and the contract will get unlocked once the specified period is over.

```
//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;
}
```

- Owner can only transfer more than 10% of tokens and owner can change the maximum transaction amount.

```
function setMaxTxPercent(uint256 maxTxPercent) external onlyOwner() {
    require(maxTxPercent > 10, "Cannot set transaction amount less than 10 percent!");
    _maxTxAmount = _tTotal.mul(maxTxPercent).div(
        10**2
    );
}
```

❖ The owner can exclude or include any account from Reward. (Any transaction fee or liquidity fee won't be applicable when the owner excludes any wallets until the owner includes them back.)

```
function excludeFromReward(address account⬆) public onlyOwner() {
    require(account⬆ != 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D, 'We can not exclude Pancake router.');
    require(!_isExcluded[account⬆], "Account is already excluded");
    if(_rOwned[account⬆] > 0) {
        _tOwned[account⬆] = tokenFromReflection(_rOwned[account⬆]);
    }
    _isExcluded[account⬆] = true;
    _excluded.push(account⬆);
}

ftrace | funcSig
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 owner Can enable or disable all fees. (No fee will be there within any transaction if the owner disable the fee until the owner enables it back)

```
function removeAllFee() private {
    if(_taxFee == 0 && _liquidityFee == 0 && _burnFee == 0 && _donationFee == 0) return;

    _previousTaxFee = _taxFee;
    _previousLiquidityFee = _liquidityFee;
_previousBurnFee = _burnFee;
_previousDonationFee = _donationFee;
    _taxFee = 0;
    _liquidityFee = 0;
_burnFee = 0;
_donationFee = 0;
}

ftrace | funcSig
function restoreAllFee() private {
    _taxFee = _previousTaxFee;
    _liquidityFee = _previousLiquidityFee;
_burnFee = _previousBurnFee;
_donationFee = _previousDonationFee;
}
```

❖ The owner can enable swap and liquify.

```
ftrace | funcSig
function setSwapAndLiquifyEnabled(bool _enabled⬆) public onlyOwner {
    swapAndLiquifyEnabled = _enabled⬆;
    emit SwapAndLiquifyEnabledUpdated(_enabled⬆);
}
```

❖ Owner will exclude from the fee

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

Audit conclusion

While conducting the audit of the WIN Token smart contract, it was observed that there is nothing alarming with the code and the contract contains only low severity issues.