



TechRate
AUDIT COMPANY

Smart Contract Security Audit

TechRate

November, 2021

Audit Details



Audited project

VeldoraBSC



Deployer address

0xda58947125dd67f0097cD290d25f8131bCD6bCF9



Client contacts:

VeldoraBSC team



Blockchain

Binance Smart Chain



Project website:

<http://VeldoraBSC.com>

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 VeldoraBSC to perform an audit of smart contracts:

<https://bscscan.com/address/0x3a5FcCBdcc2684be588575f063acbA78e09AAD4a#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

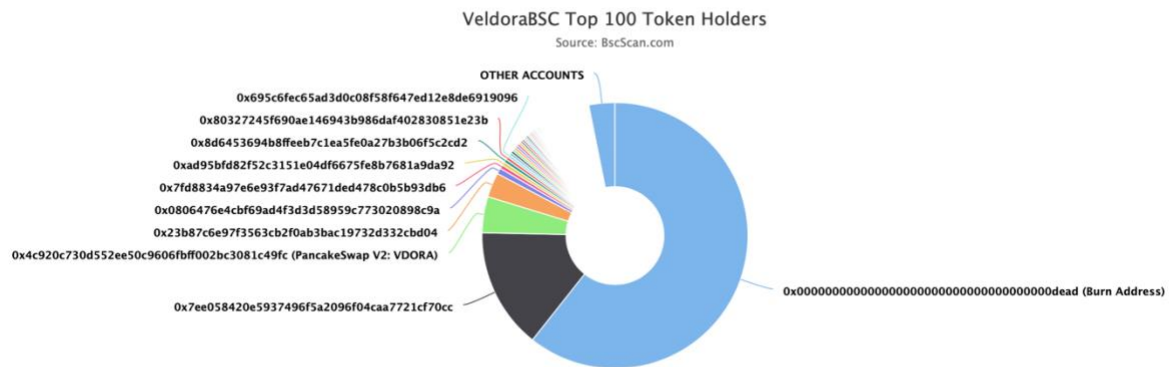
Token contract details for 07.11.2021

Contract name	VeldoraBSC
Contract address	0x3a5FcCBdcc2684be588575f063acbA78e09AAD4a
Total supply	1,000,000,000,000,000
Token ticker	VDORA
Decimals	9
Token holders	392
Transactions count	5,065
Top 100 holders dominance	96.84%
Burn fee	1
Liquidity fee	1
Marketing fee	5
Tax fee	4
Uniswap V2 pair	0x4c920C730D552EE50c9606FBff002BC3081c49fc
Contract deployer address	0xda58947125dd67f0097cD290d25f8131bCD6bCF9
Contract's current owner address	0xda58947125dd67f0097cD290d25f8131bCD6bCF9

VeldoraBSC Token Distribution

💡 The top 100 holders collectively own 96.84% (968,371,738,526,089.00 Tokens) of VeldoraBSC

💡 Token Total Supply: 1,000,000,000,000.00 Token | Total Token Holders: 392

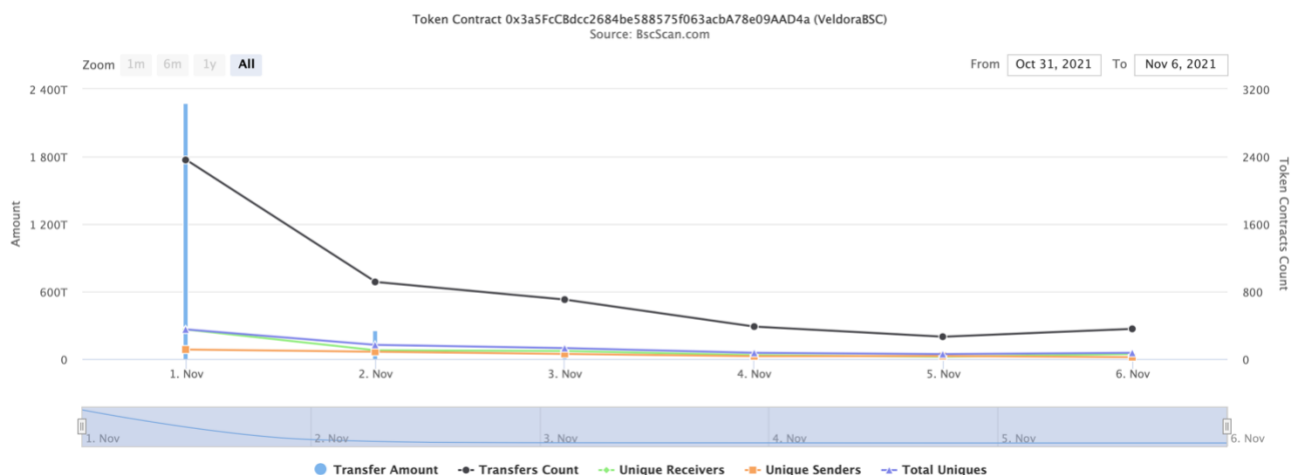


(A total of 968,371,738,526,089.00 tokens held by the top 100 accounts from the total supply of 1,000,000,000,000,000.00 token)



VeldoraBSC Contract Interaction Details

Time Series: Token Contract Overview

Mon 1, Nov 2021 - Sat 6, Nov 2021



VeldoraBSC Top 10 Token Holders

Rank	Address	Quantity	Percentage
1	Burn Address	606,586,861,057,798.601149629	60.6587%
2	 0x7ee058420e5937496f5a2096f04caa7721cf70cc	146,775,003,527,626	14.6775%
3	 PancakeSwap V2: VDORA	43,962,135,391,761.345333028	4.3962%
4	0x23b87c6e97f3563cb2f0ab3bac19732d332cbd04	29,980,000,000,000.341433597	2.9980%
5	0x0806476e4cbf69ad4f3d3d58959c773020898c9a	7,124,938,348,361.741399523	0.7125%
6	0x7fd8834a97e6e93f7ad47671ded478c0b5b93db6	4,613,310,040,581.464359134	0.4613%
7	0xad95bfd82f52c3151e04df6675fe8b7681a9da92	4,280,273,046,301.044984964	0.4280%
8	0x8d6453694b8ffeb7c1ea5fe0a27b3b06f5c2cd2	3,945,841,780,900.649055246	0.3946%
9	0x80327245f690ae146943b986daf402830851e23b	3,635,872,958,707.377056292	0.3636%
10	0x695c6fec65ad3d0c08f58f647ed12e8de6919096	3,359,064,249,104.820311882	0.3359%



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 #
- + VeldoraBSC (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] deliver #
 - [Pub] reflectionFromToken
 - [Pub] tokenFromReflection
 - [Pub] excludeFromReward #
 - modifiers: onlyOwner
 - [Ext] includeInReward #
 - modifiers: onlyOwner
 - [Pub] excludeFromFee #
 - modifiers: onlyOwner
 - [Pub] includeInFee #
 - modifiers: onlyOwner
 - [Pub] setMarketingAddress #
 - modifiers: onlyOwner
 - [Ext] setTaxFeePercent #
 - modifiers: onlyOwner
 - [Ext] setLiquidityFeePercent #
 - modifiers: onlyOwner
 - [Ext] setMarketingFeePercent #
 - modifiers: onlyOwner
 - [Ext] setBurnFeePercent #
 - modifiers: onlyOwner
 - [Ext] setMaxTxPercent #
 - modifiers: onlyOwner
 - [Ext] setNumTokensSellToAddToLiquidity #
 - modifiers: onlyOwner
 - [Ext] <Fallback> (\$)
 - [Pub] setSwapAndLiquifyEnabled #
 - modifiers: onlyOwner
 - [Prv] _reflectFee #
 - [Prv] _getValues
 - [Prv] _getTValues
 - [Prv] _getRValues
 - [Prv] _getRate
 - [Prv] _getCurrentSupply
 - [Prv] _takeContractTax #
 - [Prv] _transferBurn #

- [Prv] calculateTaxFee
- [Prv] calculateLiquidityFee
- [Prv] calculateMarketingFee
- [Prv] calculateBurnFee
- [Pub] isExcludedFromFee
- [Prv] _approve #
- [Prv] _transfer #
- [Prv] swapAndLiquify #
 - modifiers: lockTheSwap
- [Prv] swapTokensForEth #
- [Prv] addLiquidity #
- [Prv] _checkFirstLiquidityAdd #
- [Ext] addLiquidityProvider #
 - modifiers: onlyOwner
- [Ext] removeLiquidityProvider #
 - modifiers: onlyOwner
- [Prv] _hasLimits
- [Ext] setPurchaseLimit #
 - modifiers: onlyOwner
- [Prv] _exceedsPurchaseLimit
- [Prv] _checkSniper
- [Ext] isSniper
 - modifiers: onlyOwner
- [Ext] addSniper #
 - modifiers: onlyOwner
- [Ext] removeSniper #
 - modifiers: onlyOwner
- [Ext] setSnipeBlocks #
 - modifiers: onlyOwner
- [Prv] _tokenTransfer #
- [Prv] _transferStandard #
- [Prv] _transferToExcluded #
- [Prv] _transferFromExcluded #
- [Prv] _transferBothExcluded #
- [Prv] _completeTransfer #

(\$) = 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(!_isExcludedFromRewards[account], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcludedFromRewards[account] = false;
            _excluded.pop();
            break;
        }
    }

    emit AddressExcludedFromRewardsChanged(account, false);
}
```

- 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 leave as owner 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;
}
```

- Owner can include in and exclude from reward.

```
function excludeFromReward(address account) public onlyOwner {
    require(!_isExcludedFromRewards[account], "Account is already excluded");
    if(_rOwned[account] > 0) {
        _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcludedFromRewards[account] = true;
    _excluded.push(account);
    emit AddressExcludedFromRewardsChanged(account, true);
}

function includeInReward(address account) external onlyOwner {
    require(_isExcludedFromRewards[account], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcludedFromRewards[account] = false;
            _excluded.pop();
            break;
        }
    }
    emit AddressExcludedFromRewardsChanged(account, false);
}
```

- Owner can include in and exclude from fee.

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

function includeInFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = false;
    emit AddressExcludedFromFeesSet(account, false);
}
```


- Owner can change marketing wallet address.

```
function setMarketingAddress(address payable marketingAddress) public onlyOwner {
    _marketingAddress = marketingAddress;
    emit MarketingAddressChanged(marketingAddress);
}
```

- Owner can change the tax, liquidity, marketing and burn fees.

```
function setTaxFeePercent(uint256 taxFee) external onlyOwner {
    _taxFee = taxFee;
    emit RewardTaxChanged(taxFee);
}

function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner {
    _liquidityFee = liquidityFee;
    emit LiquidityTaxChanged(liquidityFee);
}

function setMarketingFeePercent(uint256 marketingFee) external onlyOwner {
    _marketingFee = marketingFee;
    emit MarketingTaxChanged(marketingFee);
}

function setBurnFeePercent(uint256 burnFee) external onlyOwner {
    _burnFee = burnFee;
    emit BurnTaxChanged(burnFee);
}
```

- Owner can change the maximum transaction amount.

```
function setMaxTxPercent(uint256 maxTxPercent) external onlyOwner {
    _maxTxAmount = _tTotal.mul(maxTxPercent).div(
        10**2
    );
    emit MaxTransactionAmountChanged(_maxTxAmount);
}
```

- Owner can change numTokensSellToAddToLiquidity value.

```
function setNumTokensSellToAddToLiquidity(uint256 newNumTokens) external onlyOwner {
    numTokensSellToAddToLiquidity = newNumTokens;
}
```

- Owner can enable / disable swap and liquify.

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

- Owner can add / remove liquidity provider addresses.

```
function addLiquidityProvider(address provider) external onlyOwner {
    require(!_liquidityHolders[provider], "Liquidity provider is already registered");
    _liquidityHolders[provider] = true;
    emit LiquidityProviderAdded(provider);
}

function removeLiquidityProvider(address provider) external onlyOwner {
    require(_liquidityHolders[provider], "Liquidity provider is not registered");
    _liquidityHolders[provider] = false;
    emit LiquidityProviderRemoved(provider);
}
```

- Owner can change purchase limits.

```
function setPurchaseLimit(uint256 blocks, uint256 limit) external onlyOwner {
    _purchaseLimitBlocks = blocks;
    _purchaseLimit = limit;
    emit PurchaseLimitChanged(blocks, limit);
}
```

- Owner can add / remove sniper addresses.

```
function addSniper(address sniperAddress) external onlyOwner {
    require(!_isSniper[sniperAddress], "Address already registered as sniper");
    _isSniper[sniperAddress] = true;
}

function removeSniper(address sniperAddress) external onlyOwner {
    require(_isSniper[sniperAddress], "Address not registered as sniper");
    _isSniper[sniperAddress] = false;
}
```

- Owner can change snipeBlockAmount value.

```
function setSnipeBlocks(uint256 blocks) onlyOwner external {
    snipeBlockAmount = blocks;
}
```

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://www.pinksale.finance/#/pinklock/record/1014?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.