DexBerry BEP20 Token Smart Contract Code Review and Security Analysis Report

4th Oct,2021

Contents

Commission

Audited Project	**DexBerry Token**
Contract Owner	**0x0184aa703297589f25326c5b69e5d497cdfad1b1**
Smart Contract	**0x1d01d0b1a7b86f0dc2162b374ea52663e43e1750**
Blockchain	**Binance Main Smart Chain**

Block Solutions was commissioned by DEXBERRY Token owners to perform an audit of their main smart contract. 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.

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

DEXBERRY Properties

Contract name	DEXBERRY Token
Contract address	0x1d01d0b1a7b86f0dc2162b374ea52663e43e1750
Total supply	2,500,000,000
Token ticker	DEXBY
Decimals	18
Token holders	1
Transaction’s count	4
Top 100 holder’s dominance	100%
Liquidity fee	3
Tax fee	4
Total fees	10
Mintable	Yes
Burnable	Yes
Uniswap V2 pair	0x844cdf0e98c824ef01ada9cb8f8a2fdb5ae4acdc
Contract deployer address	0x5459C593fbF201513D2c4d488368c75E1c65A61d
Contract’s current owner address	0x0184aa703297589f25326c5b69e5d497cdfad1b1
Charity Address	0x3fb15566fe521aa4125656b0c7582b7de986573b

Contract Functions

View

i. function allowance(address owner, address spender)
ii. function balanceOf(address account)
iii. function decimals()
iv. function geUnlockTime()
v. function isExcludedFromFee(address account)
vi. function isExcludedFromReward(address account)
vii. function name()
viii. function owner()
ix. function reflectionFromToken(uint256 tAmount, bool deductTransferFee)
x. function symbol()
xi. function totalSupply()
xii. function tokenFromReflection(uint256 rAmount)
xiii. function totalFees() public

Virtual

i. increaseAllowance()
ii. decreaseAllowance()
iii. function renounceOwnership()
iv. function transferOwnership(address newOwner)
v. function _msgData()
vi. function _msgSender()
vii. function lock(uint256 time)
viii. function unlock()

Executables

i. function approve(address spender, uint value)
ii. function decreaseAllowance(address spender, uint256 subtractedValue)
iii. function deliver(uint256 tAmount)
iv. function increaseAllowance(address spender, uint256 addedValue)
v. function transfer(address recipient, uint256 amount)
vi. function transferFrom(address sender, address recipient, uint256 amount)
vii. function unlock()

Owner Executables

i. function excludeFromFee(address account)
ii. function excludeFromReward(address account)
iii. function includeInFee(address account)
iv. function includeInReward(address account)
v. function lock(uint256 time)
vi. function renounceOwnership()
vii. function setBurnFee (uint burnFee)
viii. function setCharityFee (uint charityFee)
ix. function setLiquidityFee (uint liquidityFee)
x. function setMaxTxPercent(uint256 maxTxPercent)
xi. function setSwapAndLiquifyEnabled(bool _enabled)
xii. function setTaxFee (uint taxFee)
xiii. function setcharityWallet(address newWallet)
xiv. unction transferOwnership(address newOwner)

Checklist

Compiler errors.	Passed
Possible delays in data delivery.	Passed
Timestamp dependence.	Low Severity
Integer Overflow and Underflow.	Passed
Race Conditions and Reentrancy.	Passed
DoS with Revert.	Passed
DoS with block gas limit.	Passed
Methods execution permissions.	Mid Severity
Economy model of the contract.	Passed
Private user data leaks.	Passed
Malicious Events Log.	Passed
Scoping and Declarations.	Passed
Uninitialized storage pointers.	Passed
Arithmetic accuracy.	Passed
Design Logic.	Passed
Impact of the exchange rate.	Not Checked
Oracle Calls.	Passed
Cross-function race conditions.	Passed
Fallback function security.	Passed
Front Running.	Not Checked
Safe Open Zeppelin contracts and implementation usage.	Passed
Whitepaper-Website-Contract correlation.	Not Checked

Owner privileges

DEXBERRY Contract

The Owner can limit the Max transaction percentage.

```
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 set charity wallet.

```
function setcharityWallet(address newWallet) external onlyOwner() {
    charityWallet = newWallet;
}
```

The owner can disable or enable swap and Liquify at any time.

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

The owner can exclude accounts from reward at any time.

```
function excludeFromReward(address account) public onlyOwner() {
    require(account != 0x05fF2B0DB69458A0750badebc4f9e13aDd608C7F, '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);
}
```

The owner can exclude accounts from fees at any time.

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

The owner can include accounts in fees at any time.

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

The owner can include accounts in reward at any time.

```
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;
        }
    }
}
```

This function can 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));
}
```

Renouncing ownership will leave the contract without an owner, thereby removing any functionality that is only available to the owner.

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

The Owner can set burn fee.

```
function setBurnFee (uint burnFee) external onlyOwner() {
    _burnFee = burnFee;
}
```

The Owner can set charity fee.

```
function setCharityFee (uint charityFee) external onlyOwner() {
    _charityFee = charityFee;
}
```

The Owner can set tax fee.

```
function setTaxFee (uint taxFee) external onlyOwner() {
    _taxFee = taxFee;
}
```

The Owner can set Liquidity fee.

```
function setLiquidityFee (uint liquidityFee) external onlyOwner() {
    _liquidityFee = liquidityFee;
}
```

The Owner can set charity Wallet.

```
function setcharityWallet(address newWallet) external onlyOwner() {
    charityWallet = newWallet;
}
```

Transfers ownership of the contract to a new account (`new Owner`).Can only be called by the current owner.

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

Conclusions

The Smart Contract code passed the audit successfully on the Binance Mainnet with some considerations to take. There were three low severity warnings raised meaning that they should be taken into consideration but if the confidence in the owner is good, they can be dismissed.
The last change is advisable in order to provide more security to new holders. Nonetheless this is not necessary if the holders and/or investors feel confident with the contract owners.