Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

June, 2021

Audit Details

Audited project

Orange Grove Token

Deployer address

0xE36Fb4C39F684820cf911f143452dA0e3F7002ae

Client contacts:

Orange Grove Token team

Blockchain

Binance Smart Chain

Project website:

Not provided by Orange Grove Token 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 Orange Grove Token to perform an audit of smart contracts:

https://bscscan.com/address/0xa05e98a122bb33cc2c8ecf3a098fd52053493f80

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

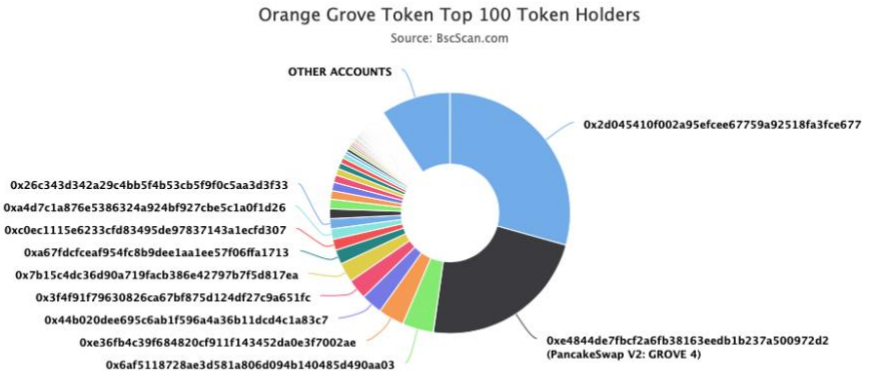
Token contract details for 21.06.2021

Contract name	Orange Grove Token
Contract address	0xa05e98a122bB33cC2c8ECf3A098FD52053493f80
Total supply	10,000,000
Token ticker	GROVE
Decimals	9
Token holders	483
Transactions count	4,621
Top 100 holders dominance	90.64%
Liquidity fee	3
Tax fee	2
Total fees	646092856193853
Uniswap V2 pair	0xe4844de7fbcf2a6fb38163eedb1b237a500972d2
Contract deployer address	0xE36Fb4C39F684820cf911f143452dA0e3F7002ae
Contract's current owner address	0xe36fb4c39f684820cf911f143452da0e3f7002ae

Orange Grove Token Token Distribution

The top 100 holders collectively own 90.64% (9,063,606.11 Tokens) of Orange Grove Token

Token Total Supply: 10,000,000.00 Token | Total Token Holders: 483

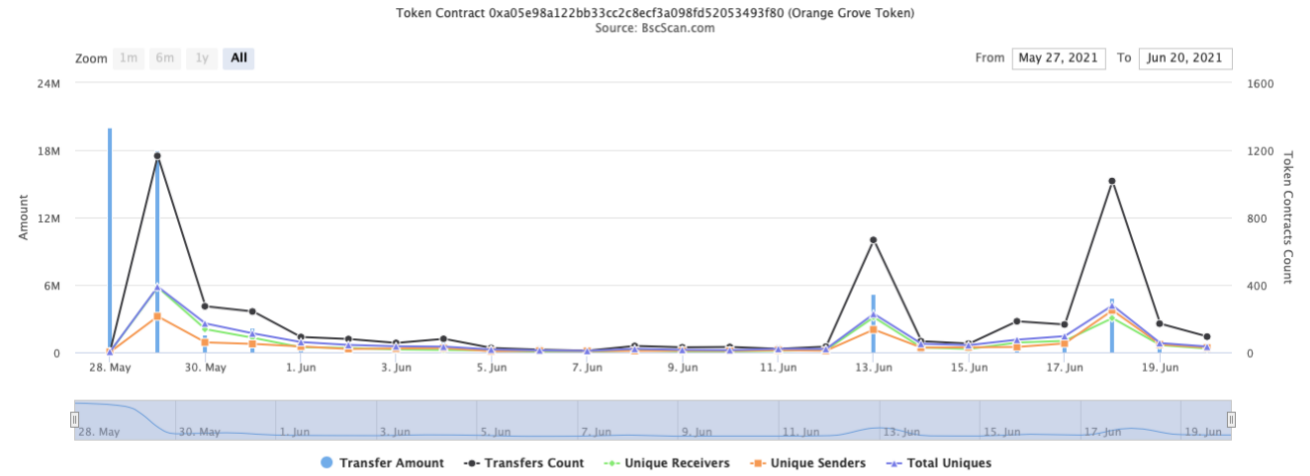


(A total of 9,063,606.11 tokens held by the top 100 accounts from the total supply of 10,000,000.00 token)

Orange Grove Token Contract Interaction Details

Time Series: Token Contract Overview

Fri 28, May 2021 - Sun 20, Jun 2021



Orange Grove Token Top 10 Token Holders

Rank	Address	Quantity (Token)	Percentage
1	0x2d045410f002a95efcee67759a92518fa3fce677	2,931,847.821654734	29.3185%
2	PancakeSwap V2: GROVE 4	2,291,889.293317434	22.9189%
3	0x6af5118728ae3d581a806d094b140485d490aa03	419,658.70211304	4.1966%
4	0xe36fb4c39f684820cf911f143452da0e3f7002ae	347,447.337566354	3.4745%
5	0x44b020dee695c6ab1f596a4a36b11dcd4c1a83c7	284,428.100343279	2.8443%
6	0x3f4f91f79630826ca67bf875d124df27c9a651fc	271,000	2.7100%
7	0x7b15c4dc36d90a719facb386e42797b7f5d817ea	260,341.975326248	2.6034%
8	0xa67fdcfceaf954fc8b9dee1aa1ee57f06ffa1713	192,343.105742827	1.9234%
9	0xc0ec1115e6233cfd83495de97837143a1ecfd307	148,000.83189203	1.4800%
10	0xa4d7c1a876e5386324a924bf927cbe5c1a0f1d26	140,450.574850669	1.4045%

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 #

+ GroveToken (Context, IERC20, Ownable)
  - [Pub] <Constructor> #
  - [Ext] _burn #
    - modifiers: onlyOwner
  - [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
  - [Prv] _transferBothExcluded #
  - [Pub] excludeFromFee #
    - modifiers: onlyOwner
  - [Pub] includeInFee #
    - modifiers: onlyOwner
  - [Ext] setTaxFeePercent #
    - modifiers: onlyOwner
  - [Ext] setReflecFeePercent #
    - modifiers: onlyOwner
  - [Ext] setLiquidityFeePercent #
    - 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] calculateReflecFee
  - [Prv] calculateLiquidityFee
  - [Prv] removeAllFee #
```

- [Prv] restoreAllFee #
- [Pub] isExcludedFromFee
- [Prv] sendBNBToMarketing #
- [Ext] _setMarketingWallet #
  - modifiers: onlyOwner
- [Ext] _setStakingWallet #
  - modifiers: onlyOwner
- [Ext] _setTeamWallet #
  - modifiers: onlyOwner
- [Ext] setLiquidityDivisor #
  - modifiers: onlyOwner
- [Prv] _approve #
- [Prv] _transfer #
- [Prv] swapAndLiquify #
  - modifiers: lockTheSwap
- [Prv] swapTokensForEth #
- [Prv] addLiquidity #
- [Prv] _tokenTransfer #
- [Prv] _transferStandard #
- [Prv] _transferToExcluded #
- [Prv] _transferFromExcluded #
- [Pub] setGroveRouter #
  - modifiers: onlyOwner
- [Pub] setAddressTimeLock #
  - modifiers: onlyOwner
- [Pub] isTimeLocked

($) = 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.	High 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

1. Wrong burning

Issue:

- Burn function subtract same amount value from _rOwned collection and _tTotal value. Them represents data in different rates, so the amount must be converted.

```
function _burn(address _who, uint256 _value) external onlyOwner {
    require(_value <= _rOwned[_who]);
    _rOwned[_who] = _rOwned[_who].sub(_value);
    _tTotal = _tTotal.sub(_value);
    emit Transfer(_who, address(0), _value);
}
```

Recommendation:

Please check if the addresses are included in reward or not and subtract the values correctly, by multiplying with the current rate.

Medium Severity Issues

No medium severity issues found.

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, reflect and liquidity fee.**

```
ftrace | funcSig
function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
    _taxFee = taxFee;
}

ftrace | funcSig
function setReflecFeePercent(uint256 reflecFee) external onlyOwner() {
    _reflecFee = reflecFee;
}

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

- **Owner can change the maximum transaction amount.**

```
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 change team, marketing and staking addresses.**

```
ftrace | funcSig
function _setMarketingWallet(address payable marketingWalletAddress) external onlyOwner() {
  _marketingWalletAddress = marketingWalletAddress;
}

ftrace | funcSig
function _setStakingWallet(address stakingWalletAddress) external onlyOwner() {
  _stakingWalletAddress = stakingWalletAddress;
  emit UpdateCoreAddress(stakingWalletAddress);
}

ftrace | funcSig
function _setTeamWallet(address teamWalletAddress) external onlyOwner() {
  _teamWalletAddress = teamWalletAddress;
  emit UpdateCoreAddress(teamWalletAddress);
}
```

- **Owner change _liquidityDivisor value.**

```
function setLiquidityDivisor(uint256 liquidityDivisor) external onlyOwner() {
  _liquidityDivisor = liquidityDivisor;
}
```

- **Owner change Uniswap router value.**

```
function setGroveRouter(address account) public onlyOwner {

    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(account);
    // Create a uniswap pair for this new token
    uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
    .createPair(address(this), _uniswapV2Router.WETH());

    // set the rest of the contract variables
    uniswapV2Router = _uniswapV2Router;

    emit SetGroveRouter(account);
}
```

- **Owner lock addresses.**

```
function setAddressTimeLock(address _account, uint256 noOfDays) public onlyOwner {
    uint256 _lockTime = _lockedAddresses[_account];
    require(block.timestamp > _lockTime , "The address is already locked");

    _lockTime = block.timestamp + 1000*60*60*24*noOfDays; // noOfDays days to milliseconds
    _lockedAddresses[_account] = _lockTime;
}
```

- **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 high 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.