Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

June, 2021

Audit Details

Audited project

Lepracoin

Deployer address

0xcA71DdB30ddC906048bcec5dF305683F289C8C40

Client contacts:

Lepracoin team

Blockchain

Binance Smart Chain

Project website:

https://officiallepracoin.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 Lepracoin to perform an audit of smart contracts:

https://bscscan.com/address/0xcA71DdB30ddC906048bcec5dF305683F289C8C40#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.

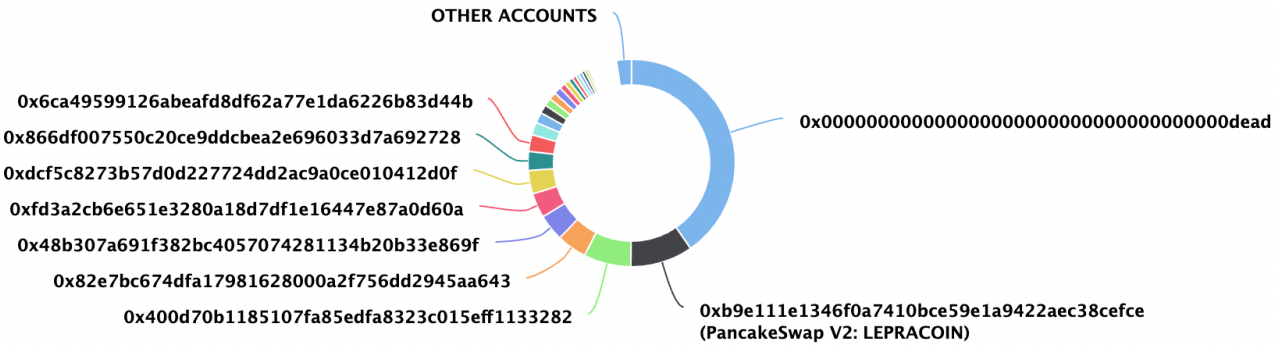
The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Contracts Details

Token contract details for 02.06.2021

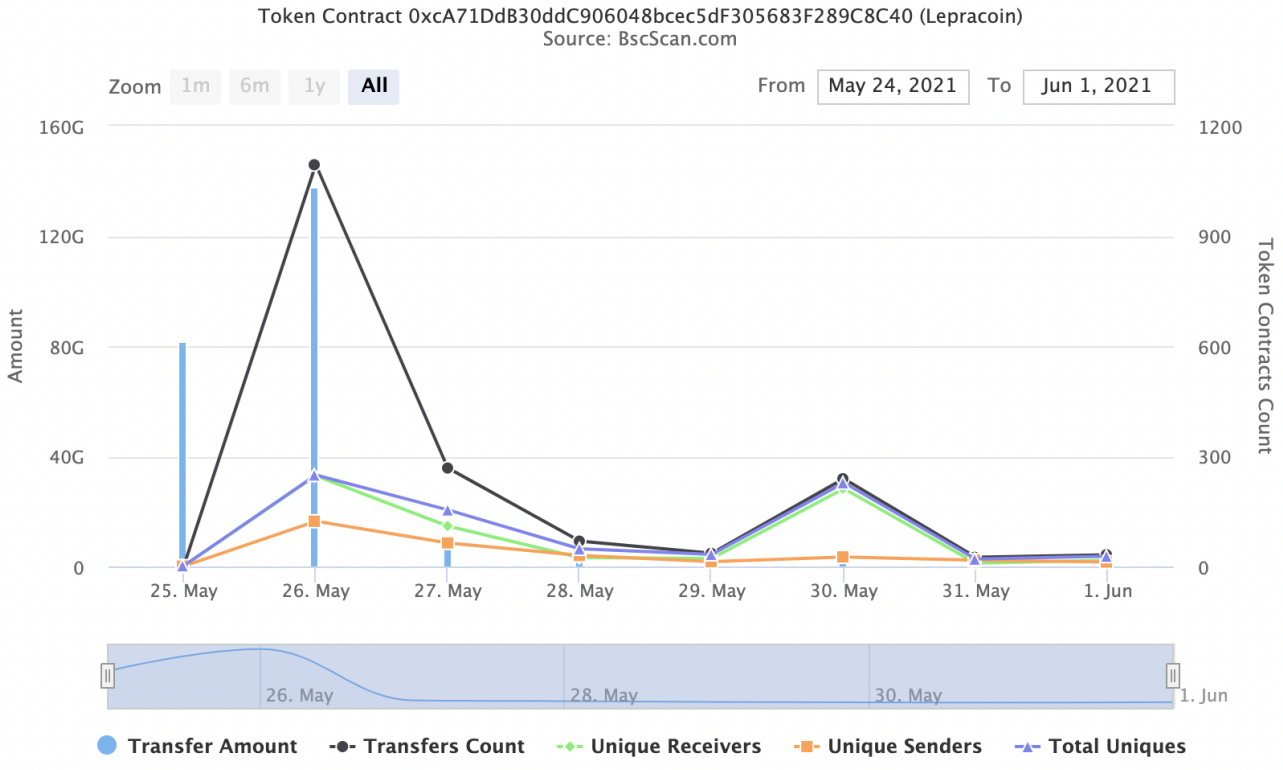
Contract name	Lepracoin
Contract address	0xcA71DdB30ddC906048bcec5dF305683F289C8C40
Total supply	81_221_031_943
Token ticker	LEPRACOIN
Decimals	9
Token holders	468
Transactions count	1790
Top 100 holders dominance	97.67%
Liquidity fee	4
Tax fee	3
CharityFee	0
Burn fee	1
Total fees	2336904169331423180
Pancake V2 pair	0xb9e111e1346f0a7410bce59e1a9422aec38cefce
Contract deployer address	0x3191a321cddd7aa804cef342582d5e7ec2801dc3
Contract’s current owner address	0x3191a321cddd7aa804cef342582d5e7ec2801dc3

Lepracoin Token Distribution



(A total of 79,319,514,775.61 tokens held by the top 100 accounts from the total supply of 81,220,317,746.56 token)

Lepracoin Contract Interaction Details



Lepracoin Top 10 Token Holders

Rank	Address	Quantity	Percentage
1	0x000000000000000000000000000000000000dead	32,800,000,000	40.3840%
2	PancakeSwap V2: LEPRACOIN	7,914,799,194.41044688	9.7449%
3	0x400d70b1185107fa85edfa8323c015eff1133282	6,005,021,555.647644357	7.3935%
4	0x82e7bc674dfa17981628000a2f756dd2945aa643	3,793,108,968.683188192	4.6701%
5	0x48b307a691f382bc4057074281134b20b33e869f	3,376,670,901.150313388	4.1574%
6	0xfd3a2cb6e651e3280a18d7df1e16447e87a0d60a	3,112,969,285.937605976	3.8327%
7	0xdcf5c8273b57d0d227724dd2ac9a0ce010412d0f	2,968,169,818.796507398	3.6545%
8	0x866df007550c20ce9ddcbea2e696033d7a692728	2,406,685,573.740756984	2.9632%
9	0x6ca49599126abeafd8df62a77e1da6226b83d44b	2,120,318,900.785211341	2.6106%
10	0x43e113733a430565c4d29082e9e0c23b7fb6306f	1,613,141,952.622699369	1.9861%

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 #

+ Lepracoin (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] reflectionFromToken
- [Pub] tokenFromReflection
- [Pub] setRouterAddress #
  - modifiers: onlyOwner
- [Ext] withdrawEth #
  - modifiers: onlyOwner
- [Ext] buybackBurn #
  - modifiers: onlyOwner
- [Pub] setCharityWallet #
  - modifiers: onlyOwner
- [Pub] excludeFromReward #
  - modifiers: onlyOwner
- [Ext] includeInReward #
  - modifiers: onlyOwner
- [Prv] _transferBothExcluded #
- [Pub] excludeFromFee #
  - modifiers: onlyOwner
- [Pub] includeInFee #
  - modifiers: onlyOwner
- [Ext] setTaxFeePercent #
  - modifiers: onlyOwner
- [Ext] setCharityFeePercent #
  - 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] _takeDonation #
- [Prv] calculateTaxFee
- [Prv] calculateCharityFee
- [Prv] calculateBurnFee
- [Prv] calculateLiquidityFee
- [Prv] removeAllFee #
- [Prv] restoreAllFee #
- [Pub] isExcludedFromFee
- [Prv] _approve #
- [Prv] _transfer #
- [Prv] swapAndLiquify #
  - modifiers: lockTheSwap
- [Prv] swapTokensForEth #
- [Prv] swapEthForTokens #
- [Prv] addLiquidity #
- [Prv] _tokenTransfer #
- [Prv] _transferStandard #
- [Prv] _transferToExcluded #
- [Prv] _transferFromExcluded #

($) = 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:

- For `includeInReward` function contract can save index from `_excluded` array in map and there is possibility to use that map in `require` checking. For ex.:

  `require(_indexOfExcluded[account] == 0)`

  For `getCurrentSupply` we recommend to check that the excluded array length is not too big.

Information

Same calculation method:

- For `includeInReward` function contract can save index from The function `_takeDonation()` increase the `totalDonated` variable by adding reflect charity value. First of all it's different counting method from total fee or total burn. And secondary it doesn't recalculate donated value when owner changes `_isExluded` status of `CHARITY_WALLET`.

```
function _takeDonation(uint256 tCharity⬆) private {
    uint256 currentRate =  _getRate();
    uint256 rCharity = tCharity⬆.mul(currentRate);
    _rOwned[CHARITY_WALLET] = _rOwned[CHARITY_WALLET].add(rCharity);
    totalDonated = totalDonated.add(rCharity);
    if(_isExcluded[CHARITY_WALLET])
        _tOwned[CHARITY_WALLET] = _tOwned[CHARITY_WALLET].add(tCharity⬆);
}
```

Owner privileges (In the period when the owner is not renounced)

- Owner can renounce, transfer, lock and unlock

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

/**
 * @dev Transfers ownership of the contract to a new account (`newOwner`).
 * Can only be called by the current owner.
 */
ftrace | funcSig
function transferOwnership(address newOwner⬆) public virtual onlyOwner {
    require(newOwner⬆ != address(0), "Ownable: new owner is the zero address");
    emit OwnershipTransferred(_owner, newOwner⬆);
    _owner = newOwner⬆;
}

ftrace | funcSig
function geUnlockTime() public view returns (uint256) {
    return _lockTime;
}

//Locks the contract for owner for the amount of time provided
ftrace | funcSig
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
ftrace | funcSig
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 change the tax, donation and liquidity fee

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

ftrace | funcSig
function setCharityFeePercent(uint256 charityFee⬆) external onlyOwner() {
    _charityFee = charityFee⬆;
}

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

- 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 and include in the reward array.

```
ftrace | funcSig
function excludeFromReward(address account⬆) public onlyOwner() {
    // require(account != 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D, 'We can not exclude Uniswap 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 not 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;
        }
    }
}
```

- Owner can exclude from and include in the fee.

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

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

- Owner can set router address.

```
ftrace | funcSig
function setRouterAddress(address newRouter⬆) public onlyOwner() {
    IUniswapV2Router02 _newPancakeRouter = IUniswapV2Router02(newRouter⬆);
    uniswapV2Pair = IUniswapV2Factory(_newPancakeRouter.factory()).createPair(address(this), _newPancakeRouter.WETH());
    uniswapV2Router = _newPancakeRouter;
}
```

- **Owner can call withdrawEth and buybackBurn functions.**

```
ftrace | funcSig
function withdrawEth(uint amount⬆) external onlyOwner {
    msg.sender.transfer(amount⬆);
}

ftrace | funcSig
function buybackBurn(uint256 amount⬆) external onlyOwner {
    swapEthForTokens(amount⬆);
}

ftrace | funcSig
function swapEthForTokens(uint256 ethAmount⬆) private {
    // generate the uniswap pair path of weth -> token
    address[] memory path = new address[](2);
    path[0] = uniswapV2Router.WETH();
    path[1] = address(this);

    // make the swap
    uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount⬆}(
        0, // accept any amount of token
        path,
        0x000000000000000000000000000000000000dEaD,
        block.timestamp
    );
}
```

- **Owner can change donation address.**

```
ftrace | funcSig
function setCharityWallet(address _charityWallet⬆) public onlyOwner {
    CHARITY_WALLET = _charityWallet⬆;
}
```

- **Owner can enable/disable swap and liquify.**

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

Conclusion

Smart contracts contain low 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.