Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

October, 2021

Audit Details

Audited project

StarDust

Deployer address

0x01834c6dA9A1eBB0b653574e3DfDd5E21C901b5e

Client contacts:

StarDust team

Blockchain

Binance Smart Chain

Project website:

Not provided

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 StarDust to perform an audit of smart contracts:

https://bscscan.com/address/0x277819bf69667b48af57abc52dddcb92ab6a2c45#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

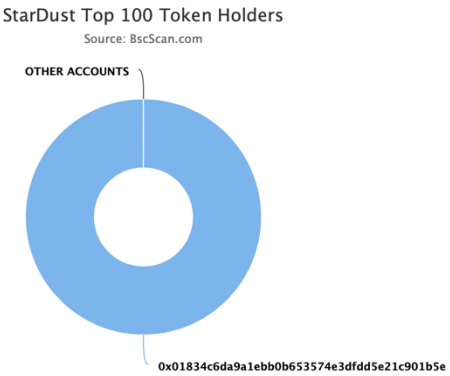
Token contract details for 13.10.2021

Contract name	StarDust
Contract address	0x277819bF69667B48Af57aBC52DddCb92Ab6A2c45
Total supply	100,000,000
Token ticker	SD
Decimals	9
Token holders	1
Transactions count	1
Top 100 holders dominance	100%
Contract deployer address	0x01834c6dA9A1eBB0b653574e3DfDd5E21C901b5e
Contract’s current owner address	0x01834c6dA9A1eBB0b653574e3DfDd5E21C901b5e

StarDust Token Distribution

The top 100 holders collectively own 100.00% (100,000,000.00 Tokens) of StarDust

Token Total Supply: 100,000,000.00 Token | Total Token Holders: 1



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

Rank	Address	Quantity (Token)	Percentage
1	0x01834c6da9a1ebb0b653574e3dfdd5e21c901b5e	100,000,000	100.0000%

StarDust Contract Interaction Details

Time Series: Token Contract Overview

Sun 10, Oct 2021 - Sun 10, Oct 2021

StarDust Top 10 Token Holders

Rank	Address	Quantity	Percentage
1	0x01834c6da9a1ebb0b653574e3dfdd5e21c901b5e	100,000,000	100.0000%

Contract functions details

+ [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 #

+ [Int] IUniswapV2Factory
  - [Ext] feeTo
  - [Ext] feeToSetter
  - [Ext] getPair
  - [Ext] allPairs
  - [Ext] allPairsLength
  - [Ext] createPair #
  - [Ext] setFeeTo #
  - [Ext] setFeeToSetter #

+ [Lib] SignedSafeMath
  - [Int] mul
  - [Int] div
  - [Int] sub
  - [Int] add

+ [Lib] SafeMath
  - [Int] tryAdd
  - [Int] trySub
  - [Int] tryMul
  - [Int] tryDiv
  - [Int] tryMod
  - [Int] add

  - [Int] sub
  - [Int] mul
  - [Int] div
  - [Int] mod
  - [Int] sub
  - [Int] div
  - [Int] mod

+ [Lib] SafeCast
  - [Int] toUint224
  - [Int] toUint128
  - [Int] toUint96
  - [Int] toUint64
  - [Int] toUint32
  - [Int] toUint16
  - [Int] toUint8
  - [Int] toUint256
  - [Int] toInt128
  - [Int] toInt64
  - [Int] toInt32
  - [Int] toInt16
  - [Int] toInt8
  - [Int] toInt256

+ Context
  - [Int] _msgSender
  - [Int] _msgData

+ [Int] IERC20
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] transfer #
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transferFrom #

+ [Int] IERC20Metadata (IERC20)
  - [Ext] name
  - [Ext] symbol
  - [Ext] decimals

+ ERC20 (Context, IERC20, IERC20Metadata)
  - [Pub] <Constructor> #
  - [Pub] name
  - [Pub] symbol
  - [Pub] decimals
  - [Pub] totalSupply
  - [Pub] balanceOf
  - [Pub] transfer #
  - [Pub] allowance
  - [Pub] approve #
  - [Pub] transferFrom #
  - [Pub] increaseAllowance #
  - [Pub] decreaseAllowance #
  - [Int] _transfer #

  - [Int] _mint #
  - [Int] _burn #
  - [Int] _approve #
  - [Int] _beforeTokenTransfer #
  - [Int] _afterTokenTransfer #

+ Ownable (Context)
  - [Pub] <Constructor> #
  - [Pub] owner
  - [Pub] renounceOwnership #
    - modifiers: onlyOwner
  - [Pub] transferOwnership #
    - modifiers: onlyOwner
  - [Prv] _setOwner #

+ [Lib] IterableMapping
  - [Pub] get
  - [Pub] getIndexOfKey
  - [Pub] getKeyAtIndex
  - [Pub] size
  - [Pub] set #
  - [Pub] remove #

+ [Int] DividendPayingTokenOptionalInterface
  - [Ext] withdrawableDividendOf
  - [Ext] withdrawnDividendOf
  - [Ext] accumulativeDividendOf

+ [Int] DividendPayingTokenInterface
  - [Ext] dividendOf
  - [Ext] distributeDividends ($)
  - [Ext] withdrawDividend #

+ DividendPayingToken (ERC20, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface)
  - [Pub] <Constructor> #
    - modifiers: ERC20
  - [Ext] <Fallback> ($)
  - [Pub] distributeDividends ($)
  - [Pub] withdrawDividend #
  - [Int] _withdrawDividendOfUser #
  - [Pub] dividendOf
  - [Pub] withdrawableDividendOf
  - [Pub] withdrawnDividendOf
  - [Pub] accumulativeDividendOf
  - [Int] _transfer #
  - [Int] _mint #
  - [Int] _burn #
  - [Int] _setBalance #

+ StarDustDividendTracker (DividendPayingToken, Ownable)
  - [Pub] <Constructor> #
    - modifiers: DividendPayingToken
  - [Int] _transfer
  - [Pub] _minimumTokenBalanceForReward #

    - modifiers: onlyOwner
  - [Pub] withdrawDividend
  - [Ext] excludeFromDividends #
    - modifiers: onlyOwner
  - [Ext] updateClaimWait #
    - modifiers: onlyOwner
  - [Ext] getLastProcessedIndex
  - [Ext] getNumberOfTokenHolders
  - [Pub] getAccount
  - [Pub] getAccountAtIndex
  - [Prv] canAutoClaim
  - [Ext] setBalance #
    - modifiers: onlyOwner
  - [Pub] process #
  - [Pub] processAccount #
    - modifiers: onlyOwner

+ SafeToken (Ownable)
  - [Pub] <Constructor> #
  - [Pub] setSafeManager #
    - modifiers: onlyOwner
  - [Ext] withdraw #
  - [Ext] withdrawBNB #

+ LockToken (Ownable)
  - [Pub] <Constructor> #
  - [Ext] openTrade #
    - modifiers: onlyOwner
  - [Ext] closeTrade #
    - modifiers: onlyOwner
  - [Ext] includeToWhiteList #
    - modifiers: onlyOwner

+ StarDust (ERC20, Ownable, SafeToken, LockToken)
  - [Pub] setFee #
    - modifiers: onlyOwner
  - [Pub] setExtraFeeOnSell #
    - modifiers: onlyOwner
  - [Pub] setMaxSelltx #
    - modifiers: onlyOwner
  - [Pub] setMarketingWallet #
    - modifiers: onlyOwner
  - [Pub] <Constructor> #
    - modifiers: ERC20
  - [Ext] <Fallback> ($)
  - [Pub] updateUniswapV2Router #
    - modifiers: onlyOwner
  - [Pub] excludeFromFees #
    - modifiers: onlyOwner
  - [Pub] setExcludeFromMaxTx #
    - modifiers: onlyOwner
  - [Pub] setExcludeFromAll #
    - modifiers: onlyOwner
  - [Pub] excludeMultipleAccountsFromFees #
    - modifiers: onlyOwner

- [Pub] setAutomatedMarketMakerPair #
  - modifiers: onlyOwner
- [Pub] setSWapToensAtAmount #
  - modifiers: onlyOwner
- [Prv] _setAutomatedMarketMakerPair #
- [Pub] updateGasForProcessing #
  - modifiers: onlyOwner
- [Ext] updateClaimWait #
  - modifiers: onlyOwner
- [Ext] getClaimWait
- [Ext] getTotalDividendsDistributed
- [Pub] isExcludedFromFees
- [Pub] isExcludedFromMaxTx
- [Pub] withdrawableDividendOf
- [Pub] dividendTokenBalanceOf
- [Ext] getAccountDividendsInfo
- [Ext] getAccountDividendsInfoAtIndex
- [Ext] processDividendTracker #
- [Ext] claim #
- [Ext] getLastProcessedIndex
- [Ext] getNumberOfDividendTokenHolders
- [Ext] excludeFromDividends #
  - modifiers: onlyOwner
- [Pub] setSwapAndLiquifyEnabled #
  - modifiers: onlyOwner
- [Int] _transfer #
  - modifiers: open
- [Prv] swapAndLiquify #
  - modifiers: lockTheSwap
- [Prv] swapTokensForBnb #
- [Prv] swapAndSendBNBToMarketing #
- [Prv] addLiquidity #

($) = 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.	Low issues
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. Rounding error

Issue:

- At each calculation with division, it is goes first. In Solidity we don't have floating points, but instead we get rounding errors.

```
function swapAndLiquify(uint256 contractTokenBalance) private lockTheSwap {
    // take liquidity fee, keep a half token
    // halfLiquidityToken = totalAmount * (liquidityFee/2totalFee)
    uint256 tokensToAddLiquidityWith = contractTokenBalance.div(totalFees.mul(2)).mul(liquidityFee);
    // swap the remaining to BNB
    uint256 toSwap = contractTokenBalance-tokensToAddLiquidityWith;
    // capture the contract's current ETH balance.
    // this is so that we can capture exactly the amount of ETH that the
    // swap creates, and not make the liquidity event include any ETH that
    // has been manually sent to the contract
    uint256 initialBalance = address(this).balance;

    // swap tokens for ETH
    swapTokensForBnb(toSwap, address(this)); // <- this breaks the ETH -> HATE swap when swap+liquify is triggered

    uint256 deltaBalance = address(this).balance-initialBalance;

    // take worthy amount bnb to add liquidity
    // worthyBNB = deltaBalance * liquidity/(2totalFees - liquidityFee)
    uint256 bnbToAddLiquidityWith = deltaBalance.mul(liquidityFee).div(totalFees.mul(2).sub(liquidityFee));

    // add liquidity to uniswap
    addLiquidity(tokensToAddLiquidityWith, bnbToAddLiquidityWith);
    // worthy Marketing fee
    uint256 MarketingAmount = deltaBalance.sub(bnbToAddLiquidityWith).div(totalFees.sub(liquidityFee)).mul(MarketingFee);
    MarketingWallet.transfer(MarketingAmount);
```

Recommendation:
Do division after multiplication.

2. Out of gas

Issue:

- The function `excludeMultipleAccountsFromFees()` uses the loop to exclude multiple accounts from fees. Function will be aborted with `OUT_OF_GAS` exception if there will be a long addresses list.

```
function excludeMultipleAccountsFromFees(address[] calldata accounts, bool excluded) public onlyOwner {
    for(uint256 i = 0; i < accounts.length; i++) {
        _isExcludedFromFees[accounts[i]] = excluded;
    }

    emit ExcludeMultipleAccountsFromFees(accounts, excluded);
}
```

- The function includeToWhiteList() also uses the loop to whitelist addresses. Function will be aborted with OUT_OF_GAS exception if there will be a long addresses list.

```
function includeToWhiteList(address[] memory _users) external onlyOwner {
    for(uint8 i = 0; i < _users.length; i++) {
        _whiteList[_users[i]] = true;
    }
}
```

Recommendation:
Check that the addresses array length is not too big.

Notes:

- Liquidity adding not in 1 to 1 proportion.

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

- Owner can change minimumTokenBalanceForDividends value.

```
function _minimumTokenBalanceForReward(uint256 amount) public onlyOwner {
    minimumTokenBalanceForDividends = amount;
}
```

- Owner can exclude from dividends.

```
function excludeFromDividends(address account) external onlyOwner {
    dividendTracker.excludeFromDividends(account);
}
```

- Owner can change claimWait value.

```
function updateClaimWait(uint256 claimWait) external onlyOwner {
    dividendTracker.updateClaimWait(claimWait);
}
```

- Owner can change safeManager address.

```
function setSafeManager(address payable _safeManager) public onlyOwner {
    safeManager = _safeManager;
}
```

- SafeManager address can withdraw ERC20 tokens.

```
function withdraw(address _token, uint256 _amount) external {
    require(msg.sender == safeManager);
    IERC20(_token).transfer(safeManager, _amount);
}
```

- SafeManager address can withdraw contract BNBs.

```
function withdrawBNB(uint256 _amount) external {
    require(msg.sender == safeManager);
    safeManager.transfer(_amount);
}
```

- Owner can open and close trading.

```
function openTrade() external onlyOwner {
    isOpen = true;
}

function closeTrade() external onlyOwner {
    isOpen = false;
}
```

- Owner can whitelist addresses for trade when trading is closed.

```
function includeToWhiteList(address[] memory _users) external onlyOwner {
    for(uint8 i = 0; i < _users.length; i++) {
        _whiteList[_users[i]] = true;
    }
}
```

- Owner can change BNB rewards, liquidity and marketing fees.

```
function setFee(uint256 _bnbRewardFee, uint256 _liquidityFee, uint256 _MarketingFee) public onlyOwner {
    BNBRewardsFee = _bnbRewardFee;
    liquidityFee = _liquidityFee;
    MarketingFee = _MarketingFee;

    totalFees = BNBRewardsFee.add(liquidityFee).add(MarketingFee); // total fee transfer and buy
}
```

- Owner can change extraFeeOnSell value.

```
function setExtraFeeOnSell(uint256 _extraFeeOnSell) public onlyOwner {
    extraFeeOnSell = _extraFeeOnSell; // extra fee on sell
}
```

- Owner can change maxSellTransactionAmount value.

```
function setMaxSelltx(uint256 _maxSellTxAmount) public onlyOwner {
    maxSellTransactionAmount = _maxSellTxAmount;
}
```

- Owner can change marketing address.

```
function setMarketingWallet(address payable _newMarketingWallet) public onlyOwner {
    MarketingWallet = _newMarketingWallet;
}
```

- Owner can change Uniswap router.

```
function updateUniswapV2Router(address newAddress) public onlyOwner {
    require(newAddress != address(uniswapV2Router), "StarDust: The router already has that address");
    emit UpdateUniswapV2Router(newAddress, address(uniswapV2Router));
    uniswapV2Router = IUniswapV2Router02(newAddress);
}
```

- Owner can exclude from fees.

```
function excludeFromFees(address account, bool excluded) public onlyOwner {
    require(_isExcludedFromFees[account] != excluded, "StarDust: Account is already the value of 'excluded'");
    _isExcludedFromFees[account] = excluded;

    emit ExcludeFromFees(account, excluded);
}
function excludeMultipleAccountsFromFees(address[] calldata accounts, bool excluded) public onlyOwner {
    for(uint256 i = 0; i < accounts.length; i++) {
        _isExcludedFromFees[accounts[i]] = excluded;
    }

    emit ExcludeMultipleAccountsFromFees(accounts, excluded);
}
```

- Owner can exclude from max buy / sell transaction amount.

```
function setExcludeFromMaxTx(address _address, bool value) public onlyOwner {
    _isExcludedFromMaxTx[_address] = value;
}
```

- Owner can exclude from and include in addresses to automatedMarketMakerPairs array.

```
function setAutomatedMarketMakerPair(address pair, bool value) public onlyOwner {
    require(pair != uniswapV2Pair, "StarDust: The PancakeSwap pair cannot be removed from automatedMarketMakerPairs");

    _setAutomatedMarketMakerPair(pair, value);
}
```

- Owner can change swapTokensAtAmount value.

```
function setSWapToensAtAmount(uint256 _newAmount) public onlyOwner {
    swapTokensAtAmount = _newAmount;
}
```

- Owner can change gas for processing.

```
function updateGasForProcessing(uint256 newValue) public onlyOwner {
    require(newValue >= 200000 && newValue <= 500000, "StarDust: gasForProcessing must be between 200,000 and 500,000");
    require(newValue != gasForProcessing, "StarDust: Cannot update gasForProcessing to same value");
    emit GasForProcessingUpdated(newValue, gasForProcessing);
    gasForProcessing = newValue;
}
```

- Owner can enable / disable swap and liquify.

```
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. The further transfers and operations with the funds raise are not related to this particular contract.

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.