Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

November, 2021

Audit Details

Audited project

Miyazaki Inu

Deployer address

0xCFb9d1d62294551734ebBcFAbcE1F9F297DAF1DA

Client contacts:

Miyazaki Inu team

Blockchain

Ethereum

Project website:

https://www.miyazaki-inu.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 Miyazaki Inu to perform an audit of smart contracts:

https://etherscan.io/address/0xd5e7d22362bcc9881d06512d3189eae79dd98d70#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

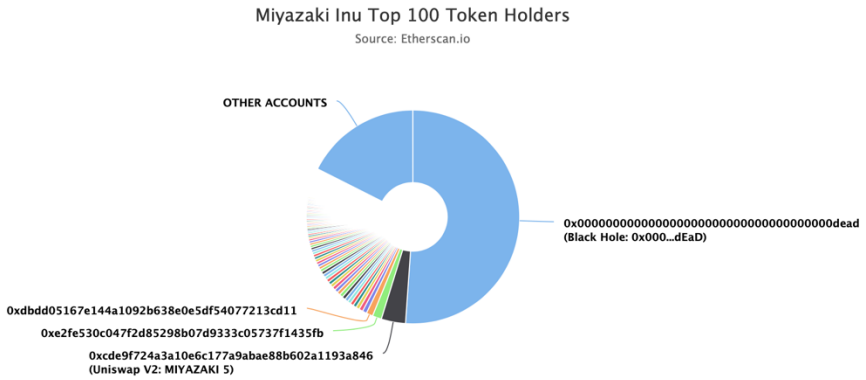
Token contract details for 0.11.2021

Contract name	Miyazaki Inu
Contract address	0xD5e7D22362BCC9881D06512d3189eAe79DD98d70
Total supply	1,000,000,000,000,000
Token ticker	SPKI
Decimals	9
Token holders	1,337
Transactions count	5,674
Top 100 holders dominance	82.43%
Reflect fee	100
Liquidity fee	100
Marketing fee	800
Uniswap V2 pair	0xcDe9F724A3a10E6C177a9abAe88b602A1193A846
Contract deployer address	0xCFb9d1d62294551734ebBcFAbcE1F9F297DAF1DA
Contract's current owner address	0xCFb9d1d62294551734ebBcFAbcE1F9F297DAF1DA

Miyazaki Inu Token Distribution

The top 100 holders collectively own 82.43% (824,284,631,556,805.00 Tokens) of Miyazaki Inu

Token Total Supply: 1,000,000,000,000,000.00 Token | Total Token Holders: 1,337

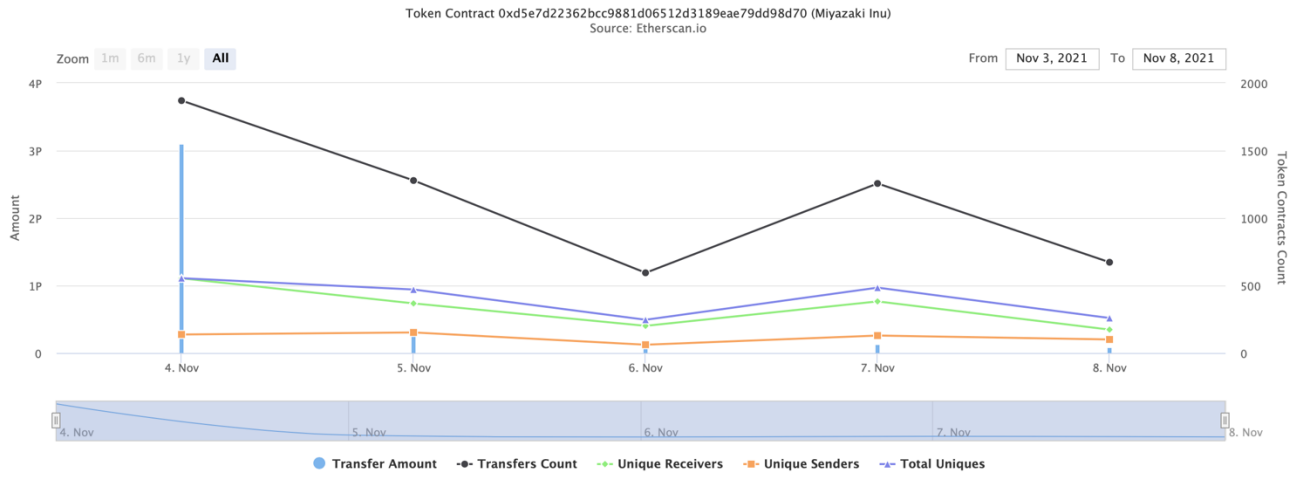


(A total of 824,284,631,556,805.00 tokens held by the top 100 accounts from the total supply of 1,000,000,000,000,000.00 token)

Miyazaki Inu Contract Interaction Details

Time Series: Token Contract Overview

Thu 4, Nov 2021 - Mon 8, Nov 2021



Miyazaki Inu Top 10 Token Holders

Rank	Address	Quantity	Percentage
1	Black Hole: 0x000...dEaD	511,229,938,532,527.51542937	51.1230%
2	Uniswap V2: MIYAZAKI 5	36,646,415,808,853.990539866	3.6646%
3	0xe2fe530c047f2d85298b07d9333c05737f1435fb	13,942,809,957,791.549784807	1.3943%
4	0xdbdd05167e144a1092b638e0e5df54077213cd11	10,067,250,659,034.180702789	1.0067%
5	0xbd92b01ce84b323c089783e6fe593c93235152a1	6,491,555,227,642.960330312	0.6492%
6	0xe0c3ba96be3783ae6048f0f72ace091c16d73eb0	5,834,822,882,906.307967628	0.5835%
7	0x999bcda82e88e19ff23b6e45a809f3fafcfb8f3c	5,389,738,325,128.399750574	0.5390%
8	0x60220548f753c5bade57a8c4a3d7dd9d491cd722	5,112,648,356,186.956906936	0.5113%
9	0x3467b971a92389be7fe77c556d7c8fa15dd55e1a	5,112,590,004,260.882564006	0.5113%
10	0x1e9bb90a49f0ea886c0cef905cb98bb8b933dd51	5,112,590,004,260.882564006	0.5113%

Contract functions details

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

+ [Int] IERC20Upgradeable
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] transfer #
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transferFrom #

+ [Int] IUniswapV2Factory
  - [Ext] getPair
  - [Ext] createPair #

+ [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] factory

+ [Int] IUniswapV2Router01
  - [Ext] factory
  - [Ext] WETH
  - [Ext] addLiquidityETH ($)

+ [Int] IUniswapV2Router02 (IUniswapV2Router01)
  - [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
  - [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens ($)
  - [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

+ [Int] AntiSnipe
  - [Ext] checkUser #
  - [Ext] setLaunch #
  - [Ext] setLpPair #
  - [Ext] setProtections #
  - [Ext] setGasPriceLimit #
  - [Ext] removeSniper #
  - [Ext] setBlacklistEnabled #

+ AaTokenContract (Context, IERC20Upgradeable)
  - [Pub] <Constructor> ($)
  - [Ext] intializeContract #
    - modifiers: onlyOwner
  - [Ext] <Fallback> ($)
  - [Pub] owner
  - [Ext] transferOwner #
    - modifiers: onlyOwner
  - [Pub] renounceOwnership #
    - modifiers: onlyOwner
  - [Ext] totalSupply
  - [Ext] decimals
  - [Ext] symbol
  - [Ext] name
  - [Ext] getOwner
  - [Ext] allowance
  - [Pub] balanceOf
  - [Pub] transfer #
  - [Pub] approve #
  - [Prv] _approve #
  - [Pub] approveContractContingency #
    - modifiers: onlyOwner
  - [Ext] transferFrom #
  - [Pub] increaseAllowance #
  - [Pub] decreaseAllowance #
  - [Pub] setNewRouter #
    - modifiers: onlyOwner
  - [Ext] setLpPair #
    - modifiers: onlyOwner
  - [Pub] isExcludedFromFees
  - [Pub] setExcludedFromFees #
    - modifiers: onlyOwner
  - [Pub] isExcludedFromReward
  - [Pub] setExcludedFromReward #
    - modifiers: onlyOwner
  - [Ext] setInitializer #
    - modifiers: onlyOwner
  - [Ext] removeSniper #
    - modifiers: onlyOwner
  - [Ext] setProtectionSettings #
    - modifiers: onlyOwner
  - [Ext] setGasPriceLimit #
    - modifiers: onlyOwner
  - [Ext] setTaxesBuy #
    - modifiers: onlyOwner
  - [Ext] setTaxesSell #
    - modifiers: onlyOwner
  - [Ext] setTaxesTransfer #
    - modifiers: onlyOwner
  - [Ext] setRatios #
    - modifiers: onlyOwner
  - [Ext] setMaxTxPercent #
    - modifiers: onlyOwner
  - [Ext] setMaxWalletSize #
    - modifiers: onlyOwner

- [Ext] setSwapSettings #
  - modifiers: onlyOwner
- [Ext] setWallets #
  - modifiers: onlyOwner
- [Pub] setContractSwapEnabled #
  - modifiers: onlyOwner
- [Prv] _hasLimits
- [Pub] tokenFromReflection
- [Int] _transfer #
- [Prv] contractSwap #
  - modifiers: lockTheSwap
- [Prv] _checkLiquidityAdd #
- [Pub] enableTrading #
  - modifiers: onlyOwner
- [Prv] _finalizeTransfer #
- [Prv] _getValues #
- [Prv] _getRate
- [Prv] _takeLiquidity #
- [Ext] sweepContingency #
  - modifiers: onlyOwner

($) = 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 `setExcludedFromReward()` 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 setExcludedFromReward(address account, bool enabled) public onlyOwner {
    if (enabled == true) {
        require(!_isExcluded[account], "Account is already excluded.");
        if(_rOwned[account] > 0) {
            _tOwned[account] = tokenFromReflection(_rOwned[account]);
        }
        _isExcluded[account] = true;
        _excluded.push(account);
    } else if (enabled == false) {
        require(_isExcluded[account], "Account is already included.");
        if(_excluded.length == 1){
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
        } else {
            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 `_getRate()` also uses the loop for evaluating reflect rate. It also could be aborted with `OUT_OF_GAS` exception if there will be a long excluded addresses list.

```
function _getRate() private view returns(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 - _rOwned[_excluded[i]];
        tSupply = tSupply - _tOwned[_excluded[i]];
    }
    if (rSupply < _rTotal / _tTotal) return _rTotal / _tTotal;
    return rSupply / tSupply;
}
```

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

Notes:

- contactSwap function triggers only on sell.

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

- Owner can approve contract contingency.

```
function approveContractContingency() public onlyOwner returns (bool) {
    _approve(address(this), address(dexRouter), type(uint256).max);
    return true;
}
```

- Owner can change Uniswap router address.

```
function setNewRouter(address newRouter) public onlyOwner() {
    IUniswapV2Router02 _newRouter = IUniswapV2Router02(newRouter);
    address get_pair = IUniswapV2Factory(_newRouter.factory()).getPair(address(this), _newRouter.WETH());
    if (get_pair == address(0)) {
        lpPair = IUniswapV2Factory(_newRouter.factory()).createPair(address(this), _newRouter.WETH());
    }
    else {
        lpPair = get_pair;
    }
    dexRouter = _newRouter;
    _approve(address(this), address(dexRouter), type(uint256).max);
}
```

- Owner can include in and exclude from fees.

```
function setExcludedFromFees(address account, bool enabled) public onlyOwner {
    _isExcludedFromFees[account] = enabled;
}
```

- Owner can change lp pair.

```
function setLpPair(address pair, bool enabled) external onlyOwner {
    if (enabled == false) {
        lpPairs[pair] = false;
        antiSnipe.setLpPair(pair, false);
    } else {
        if (timeSinceLastPair != 0) {
            require(block.timestamp - timeSinceLastPair > 1 weeks, "Cannot set a new pair this week!");
        }
        lpPairs[pair] = true;
        timeSinceLastPair = block.timestamp;
        antiSnipe.setLpPair(pair, true);
    }
}
```

- Owner can include in and exclude from reward.

```
function setExcludedFromReward(address account, bool enabled) public onlyOwner {
    if (enabled == true) {
        require(!_isExcluded[account], "Account is already excluded.");
        if(_rOwned[account] > 0) {
            _tOwned[account] = tokenFromReflection(_rOwned[account]);
        }
        _isExcluded[account] = true;
        _excluded.push(account);
    } else if (enabled == false) {
        require(_isExcluded[account], "Account is already included.");
        if(_excluded.length == 1){
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
        } else {
            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 remove sniper from antiSnipe.

```
function removeSniper(address account) external onlyOwner() {
    antiSnipe.removeSniper(account);
}
```

- Owner can change antiSnipe protection settings.

```
function setProtectionSettings(bool _antiSnipe, bool _antiGas, bool _antiBlock, bool _antiSpecial) external onlyOwner() {
    antiSnipe.setProtections(_antiSnipe, _antiGas, _antiBlock, _antiSpecial);
}
```

- Owner can change antiSnipe GAS limit.

```
function setGasPriceLimit(uint256 gas) external onlyOwner {
    require(gas >= 75, "Too low.");
    antiSnipe.setGasPriceLimit(gas);
}
```

- Owner can change liquidity, reflect and marketing fees.

```
function setTaxesBuy(uint16 reflectFee, uint16 liquidityFee, uint16 marketingFee) external onlyOwner {
    require(reflectFee <= staticVals.maxReflectFee
            && liquidityFee <= staticVals.maxLiquidityFee
            && marketingFee <= staticVals.maxMarketingFee);
    require(liquidityFee + reflectFee + marketingFee <= 3450);
    _buyTaxes.liquidityFee = liquidityFee;
    _buyTaxes.reflectFee = reflectFee;
    _buyTaxes.marketingFee = marketingFee;
}

function setTaxesSell(uint16 reflectFee, uint16 liquidityFee, uint16 marketingFee) external onlyOwner {
    require(reflectFee <= staticVals.maxReflectFee
            && liquidityFee <= staticVals.maxLiquidityFee
            && marketingFee <= staticVals.maxMarketingFee);
    require(liquidityFee + reflectFee + marketingFee <= 3450);
    _sellTaxes.liquidityFee = liquidityFee;
    _sellTaxes.reflectFee = reflectFee;
    _sellTaxes.marketingFee = marketingFee;
}

function setTaxesTransfer(uint16 reflectFee, uint16 liquidityFee, uint16 marketingFee) external onlyOwner {
    require(reflectFee <= staticVals.maxReflectFee
            && liquidityFee <= staticVals.maxLiquidityFee
            && marketingFee <= staticVals.maxMarketingFee);
    require(liquidityFee + reflectFee + marketingFee <= 3450);
    _transferTaxes.liquidityFee = liquidityFee;
    _transferTaxes.reflectFee = reflectFee;
    _transferTaxes.marketingFee = marketingFee;
}
```

- Owner can change liquidity, marketing and total ratios.

```
function setRatios(uint16 liquidity, uint16 marketing) external onlyOwner {
    require (liquidity + marketing == 100, "Must add up to 100%");
    _ratios.liquidityRatio = liquidity;
    _ratios.marketingRatio = marketing;
    _ratios.totalRatio = liquidity + marketing;
}
```

- Owner can change maximum transaction amount.

```
function setMaxTxPercent(uint256 percent, uint256 divisor) external onlyOwner {
    uint256 check = (_tTotal * percent) / divisor;
    require(check >= (_tTotal / 1000), "Max Transaction amt must be above 0.1% of total supply.");
    _maxTxAmount = check;
    maxTxAmountUI = (startingSupply * percent) / divisor;
}
```

- Owner can change maximum token amount per wallet.

```
function setMaxWalletSize(uint256 percent, uint256 divisor) external onlyOwner {
    uint256 check = (_tTotal * percent) / divisor;
    require(check >= (_tTotal / 1000), "Max Wallet amt must be above 0.1% of total supply.");
    _maxWalletSize = check;
    maxWalletSizeUI = (startingSupply * percent) / divisor;
}
```

- Owner can change swap settings.

```
function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor, uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
    swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
    swapAmount = (_tTotal * amountPercent) / amountDivisor;
}
```

- Owner can change marketing wallet address.

```
function setWallets(address payable marketingWallet) external onlyOwner {
    _marketingWallet = payable(marketingWallet);
}
```

- Owner can enable / disable contract swap.

```
function setContractSwapEnabled(bool _enabled) public onlyOwner {
    contractSwapEnabled = _enabled;
    emit ContractSwapEnabledUpdated(_enabled);
}
```

- Owner can enable trading.

```
function enableTrading() public onlyOwner {
    require(!tradingEnabled, "Trading already enabled!");
    require(_hasLiqBeenAdded, "Liquidity must be added.");
    setExcludedFromReward(address(this), true);
    setExcludedFromReward(lpPair, true);
    try antiSnipe.setLaunch(lpPair, uint32(block.number), uint64(block.timestamp)) {} catch {}
    tradingEnabled = true;
}
```

Conclusion

Smart contracts contain severity issues and owner privileges! Liquidity pair contract's security is not checked due to out of scope. 85% of liquidity goes to marketing address.

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.