



TechRate
AUDIT COMPANY

Smart Contract Security Audit

TechRate

November, 2021

Audit Details



Audited project
META SHIBA



Deployer address
0x45d9089109041b3f80859204287ee47734741826



Client contacts:
META SHIBA team



Blockchain
Ethereum



Project website:
mshiba.finance

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 META SHIBA to perform an audit of smart contracts:

<https://etherscan.io/address/0x9cf77be84214beb066f26a4ea1c38ddcc2afbcf7#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 26.11.2021

Contract name	META SHIBA
Contract address	0x45d9089109041b3f80859204287ee47734741826
Total supply	2,000,000,000,000,000
Token ticker	MSHIBA
Decimals	18
Token holders	658
Transactions count	4,736
Top 100 holders dominance	92.35%
Burn fee	5
Tax fee	5
Total fees	19753759877995758498007135471118
Uniswap V2 pair	0x847e0b52589c9e6fa2dcc42b8ffb34ec924d4cf8
Contract deployer address	0x9cF77be84214beb066F26a4ea1c38ddcc2AFbcf7
Contract's current owner address	0x00

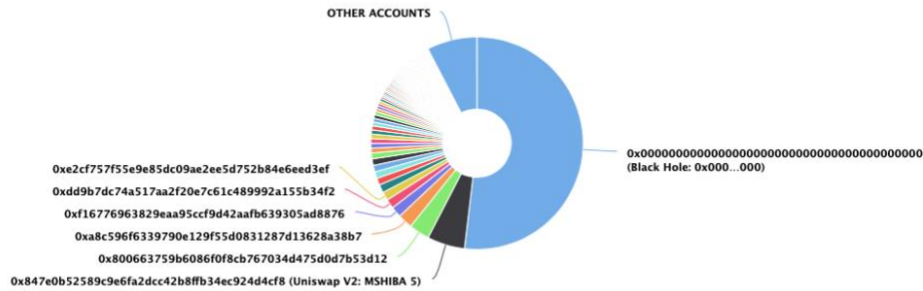
META SHIBA Token Distribution

The top 100 holders collectively own 92.35% (1,846,999,876,726,350.00 Tokens) of Meta Shiba

Token Total Supply: 2,000,000,000,000,000.00 Token | Total Token Holders: 658

Meta Shiba Top 100 Token Holders

Source: Etherscan.io



(A total of 1,846,999,876,726,350.00 tokens held by the top 100 accounts from the total supply of 2,000,000,000,000,000.00 token)

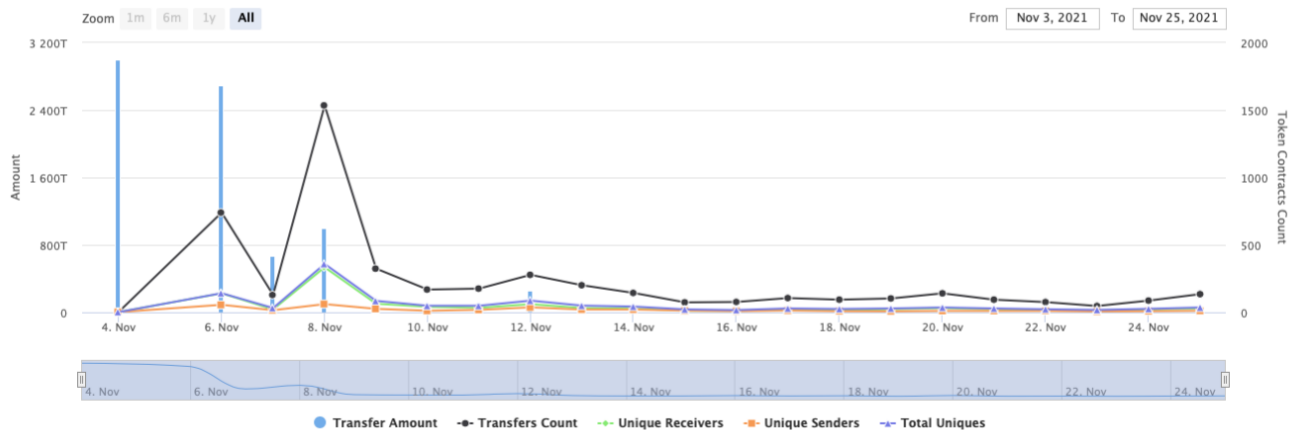
META SHIBA Contract Interaction Details

Time Series: Token Contract Overview


Thu 4, Nov 2021 - Thu 25, Nov 2021

Token Contract 0x9cf77be84214beb066f26a4ea1c38ddcc2afbcf7 (Meta Shiba)

Source: Etherscan.io



META SHIBA Top 10 Token Holders

Rank	Address	Quantity (Token)	Percentage
1	Black Hole: 0x000...000	1,038,022,951,927,210.467182855301897547	51.9011%
2	 Uniswap V2: MSHIBA 5	114,446,667,213,484.039923490066938138	5.7223%
3	0x800663759b6086f0f8cb767034d475d0d7b53d12	62,559,432,479,536.9482582558788162	3.1280%
4	0xa8c596f6339790e129f55d0831287d13628a38b7	44,684,834,277,069.092645020271995038	2.2342%
5	0xf16776963829eaa95ccf9d42aafb639305ad8876	32,961,347,302,362.810257026062077159	1.6481%
6	0xdd9b7dc74a517aa2f20e7c61c489992a155b34f2	28,148,354,208,188.05921889632261855	1.4074%
7	0xe2cf757f55e9e85dc09ae2ee5d752b84e6eed3ef	25,917,073,997,964.481614205447967052	1.2959%
8	0x6dc699130f2fac9eb0d9081871cc869bdf9affc	24,829,030,202,240.788760662096233424	1.2415%
9	0xfcf6a3d7eb8c62a5256a020e48f153c6d5dd6909	20,968,475,715,043.519915317909487709	1.0484%
10	0x7315c8fc87363dadad1708bcc5f98727a06ea110	20,519,096,086,253.15708294136011412	1.0260%



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)
 - [Pub] <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 #
- + MSHIBA (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] excludeFromReward #
 - modifiers: onlyOwner
 - [Ext] includeInReward #
 - modifiers: onlyOwner
 - [Pub] excludeFromLimit #
 - modifiers: onlyOwner
 - [Ext] includeInLimit #
 - modifiers: onlyOwner
 - [Prv] _transferBothExcluded #
 - [Pub] excludeFromFee #
 - modifiers: onlyOwner
 - [Pub] includeInFee #
 - modifiers: onlyOwner
 - [Ext] setTaxFeePercent #
 - modifiers: onlyOwner
 - [Ext] setSellFeePercent #
 - modifiers: onlyOwner
 - [Ext] setBuyFeePercent #
 - modifiers: onlyOwner
 - [Ext] setMaxBalPercent #
 - modifiers: onlyOwner
 - [Ext] setSwapAmount #
 - modifiers: onlyOwner
 - [Pub] setSwapAndLiquifyEnabled #
 - modifiers: onlyOwner
 - [Pub] setTaxEnable #
 - modifiers: onlyOwner
 - [Ext] <Fallback> (\$)
 - [Prv] _reflectFee #
 - [Prv] _getValues
 - [Prv] _getTValues
 - [Prv] _getRValues
 - [Prv] _getRate
 - [Prv] _getCurrentSupply

- [Prv] _takeLiquidity #
- [Prv] _burn #
- [Prv] calculateTaxFee
- [Prv] calculateBurnFee
- [Prv] calculateLiquidityFee
- [Prv] removeAllFee #
- [Prv] restoreAllFee #
- [Pub] isExcludedFromFee
- [Prv] _approve #
- [Prv] _transfer #
- [Prv] swapAndLiquify #
 - modifiers: lockTheSwap
- [Prv] swapTokensForEth #
- [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:

Check that the excluded array length is not too big.

Notes:

- Fees are taken only on buy and sell.

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

- Owner can exclude/include in limit.

```
function excludeFromLimit(address account↑) public onlyOwner() {
    require(!_isExcludedBal[account↑], "Account is already excluded");
    _isExcludedBal[account↑] = true;
}

ftrace | funcSig
function includeInLimit(address account↑) external onlyOwner() {
    require(!_isExcludedBal[account↑], "Account is already excluded");
    _isExcludedBal[account↑] = false;
}
```

- Owner can change the tax and buy/sell fees.

```
function setTaxFeePercent(uint256 taxFee↑) external onlyOwner() {
    _taxFee = taxFee↑;
    emit SetTaxFeePercent(taxFee↑);
}

ftrace | funcSig
function setSellFeePercent(uint256 sellFee↑) external onlyOwner() {
    _sellFees = sellFee↑;
    emit SetSellFeePercent(sellFee↑);
}

ftrace | funcSig
function setBuyFeePercent(uint256 buyFee↑) external onlyOwner() {
    _buyFees = buyFee↑;
    emit SetBuyFeePercent(buyFee↑);
}
```

- Owner can change the maximum balance amount.

```
function setMaxBalPercent(uint256 maxBalPercent↑) external onlyOwner() {
    _maxBalAmount = _tTotal.mul(maxBalPercent↑).div(
        10**2
    );
    emit SetMaxBalPercent(maxBalPercent↑);
}
```

- Owner can change the number of tokens to add to liquidity.

```
function setSwapAmount(uint256 amount↑) external onlyOwner() {
    numTokensSellToAddToLiquidity = amount↑;
}
```

- Owner can enable/disable fees.

```
function setTaxEnable (bool _enable↑) public onlyOwner {
    _taxEnabled = _enable↑;
    emit SetTaxEnable(_enable↑);
}
```

- Owner can exclude from the fee.

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

- Owner can lock and unlock. By the way, using these functions the owner could retake privileges even after the ownership was renounced.

```
function lock(uint256 time↑) public virtual onlyOwner {
    _previousOwner = _owner;
    _owner = address(0);
    _lockTime = block.timestamp + 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(block.timestamp > _lockTime, "Contract is locked until 7 days");
    emit OwnershipTransferred(_owner, _previousOwner);
    _owner = _previousOwner;
}
```


Conclusion

Smart contracts contain low severity issues! Liquidity pair contract's security is not checked due to out of scope.

Ownership renounce details are provided by the team:

<https://etherscan.io/tx/0xca97e58a02a78faf318ae67a0a9c0adf256ad4a24ed1db52d9f91a41feaf4bac>

Liquidity locking details are provided by the team:

<https://etherscan.io/tx/0x8d794b853ddeb309f8219f352a175b1100c3abc32706838b09420b44f3c9056a>

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.