Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

October, 2021

Audit Details

Audited project

Doge Superbowl

Deployer address

0x6a5d39e4a049a6c6cc77012d6b10649f964524e9

Client contacts:

Doge Superbowl team

Blockchain

Binance Smart Chain

Project website:

https://dogesbowl.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 Doge Superbowl to perform an audit of smart contracts:
https://bscscan.com/address/0x6a43f8f4b12fcd3b3eb86b319f92eb17c955dda3#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.

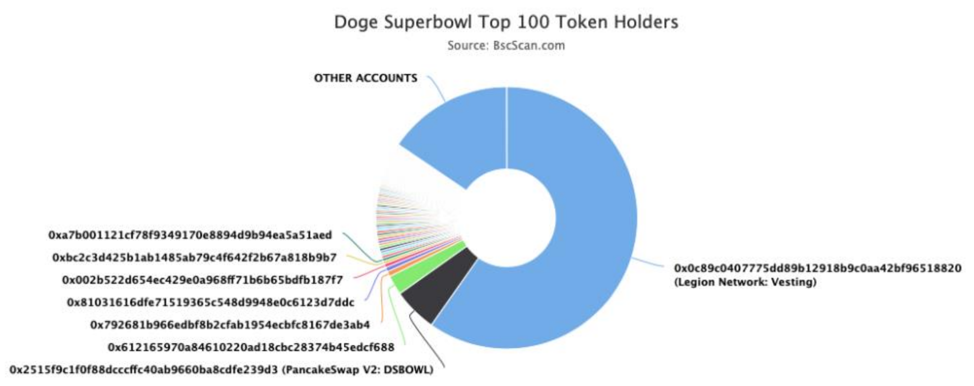
Contracts Details

Token contract details for 01.10.2021

Contract name	Doge Superbowl
Contract address	0x6a43f8F4b12FcD3B3EB86b319F92eb17c955DDA3
Total supply	96,361,942.737685
Token ticker	DSBOWL
Decimals	18
Token holders	3,589
Transactions count	13,674
Top 100 holders dominance	84.46%
WBNB_BUSD pair	0x58f876857a02d6762e0101bb5c46a8c1ed44dc16
WBNB_IGT pair	0x2515f9c1f0f88dcccffc40ab9660ba8cdfe239d3
Total burned	3638057262314815178331769
Fee total	1100
Contract deployer address	0x6a5d39e4a049a6c6cc77012d6b10649f964524e9
Contract's current owner address	0x6a5d39e4a049a6c6cc77012d6b10649f964524e9

Doge Superbowl Token Distribution

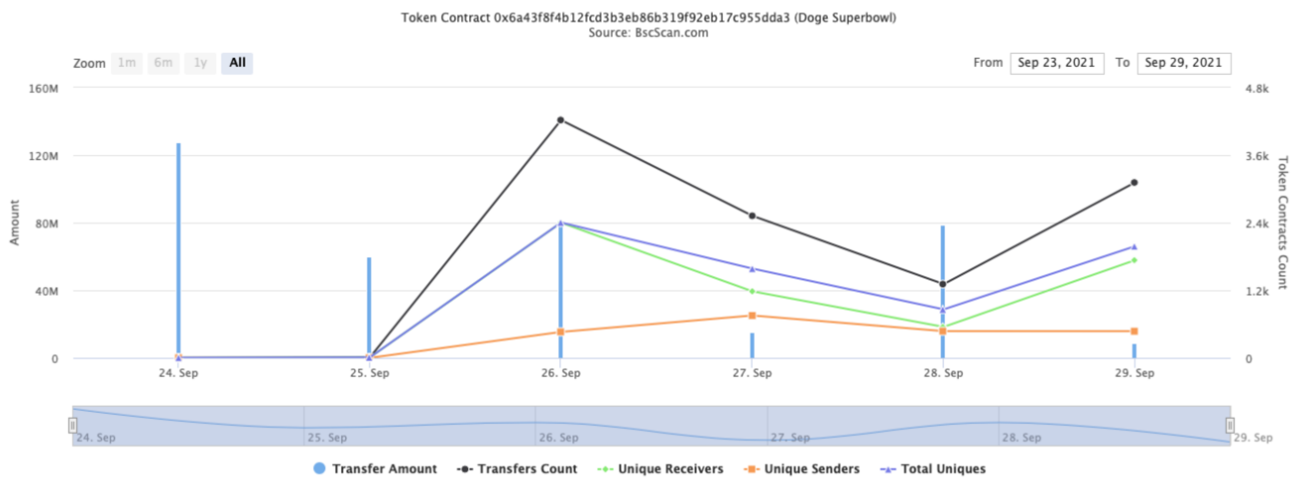
The top 100 holders collectively own 84.46% (81,386,174.13 Tokens) of Doge Superbowl | Token Total Supply: 96,361,942.74 Token | Total Token Holders: 3,589



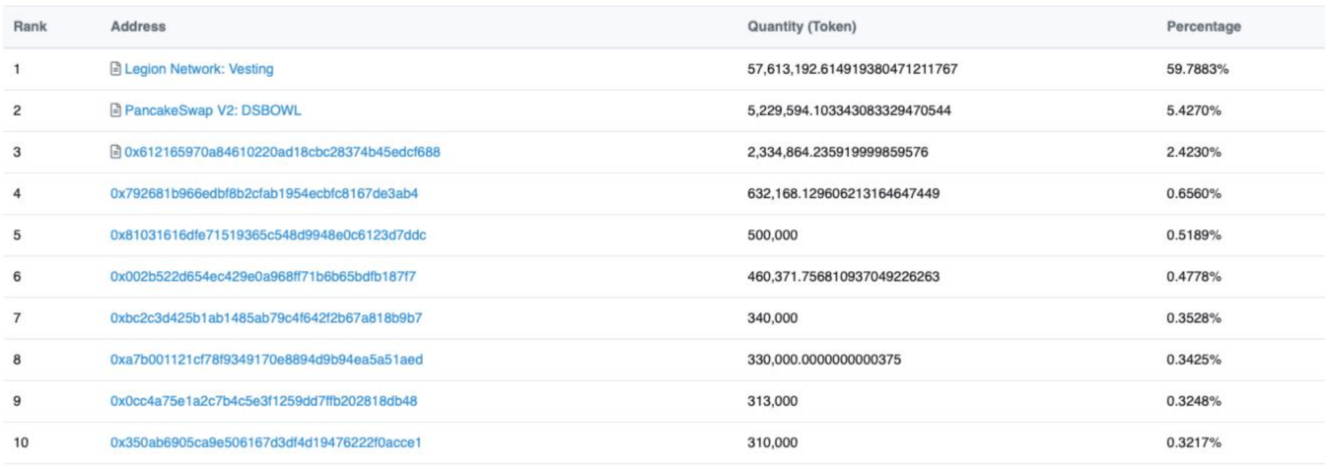
(A total of 81,386,174.13 tokens held by the top 100 accounts from the total supply of 96,361,942.74 token)

Doge Superbowl Contract Interaction Details

Time Series: Token Contract Overview | Fri 24, Sept 2021 - Wed 29, Sept 2021



Doge Superbowl Top 10 Token Holders



Rank	Address	Quantity (Token)	Percentage
1	Legion Network: Vesting	57,613,192.614919380471211767	59.7883%
2	PancakeSwap V2: DSBOWL	5,229,594.103343083329470544	5.4270%
3	0x612165970a84610220ad18cbc28374b45edcf688	2,334,864.235919999859576	2.4230%
4	0x792681b966edbf8b2cfab1954ecbfc8167de3ab4	632,168.129606213164647449	0.6560%
5	0x81031616dfe71519365c548d9948e0c6123d7ddc	500,000	0.5189%
6	0x002b522d654ec429e0a968ff71b6b65bdfb187f7	460,371.756810937049226263	0.4778%
7	0xbc2c3d425b1ab1485ab79c4f642f2b67a818b9b7	340,000	0.3528%
8	0xa7b001121cf78f9349170e8894d9b94ea5a51aed	330,000.0000000000375	0.3425%
9	0x0cc4a75e1a2c7b4c5e3f1259dd7ffb202818db48	313,000	0.3248%
10	0x350ab6905ca9e506167d3df4d19476222f0acce1	310,000	0.3217%

Contract functions details

```
+ DogeSuperBowl (Authorized, ERC20)
  - [Ext] getOwner
  - [Pub] getFeeTotal
  - [Ext] togglePauseToken #
    - modifiers: isAuthorized
  - [Ext] togglePauseStake #
    - modifiers: isAuthorized
  - [Ext] getSwapHelperAddress
  - [Ext] setFees #
    - modifiers: isAuthorized
  - [Ext] setFeesDirectWallet #
    - modifiers: isAuthorized
  - [Pub] setMaxTxAmountWithDecimals #
    - modifiers: isAuthorized
  - [Ext] setMaxTxAmount #
    - modifiers: isAuthorized
  - [Pub] setMaxAccountAmountWithDecimals #
    - modifiers: isAuthorized
  - [Ext] setMaxAccountAmount #
    - modifiers: isAuthorized
  - [Pub] setExemptOperatePausedToken #
    - modifiers: isAuthorized
  - [Pub] setExemptFee #
    - modifiers: isAuthorized
  - [Pub] setExemptTxLimit #
    - modifiers: isAuthorized
  - [Pub] setExemptAmountLimit #
    - modifiers: isAuthorized
  - [Pub] setExemptStaker #
    - modifiers: isAuthorized
  - [Pub] setAdministrationWallet #
    - modifiers: isAuthorized
  - [Ext] <Fallback> ($)
  - [Pub] <Constructor> #
    - modifiers: ERC20
  - [Pub] decimals
  - [Int] _mint #
  - [Int] _beforeTokenTransfer
  - [Int] _afterTokenTransfer
  - [Int] _transfer #
  - [Prv] exchangeFeeParts #
  - [Ext] buyBackAndHold #
    - modifiers: isAuthorized
  - [Pub] buyBackAndHoldWithDecimals #
    - modifiers: isAuthorized
  - [Ext] buyBackAndBurn #
    - modifiers: isAuthorized
  - [Pub] buyBackAndBurnWithDecimals #
    - modifiers: isAuthorized
  - [Prv] buyBackWithDecimals #
  - [Int] getAmountOut
```

- [Int] isReversed
- [Int] tokenTransfer #
- [Int] tokenTransferFrom #
- [Int] swapToken #
- [Int] getTokenBalanceOf
- [Int] getTokenReserves
- [Prv] walletHolder
- [Ext] setWBNB_IGT_PAIR #
  - modifiers: isAuthorized
- [Ext] setWBNB_BUSD_Pair #
  - modifiers: isAuthorized
- [Ext] getWBNB_IGT_PAIR
- [Ext] getWBNB_BUSD_Pair

($) = 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.	Passed
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

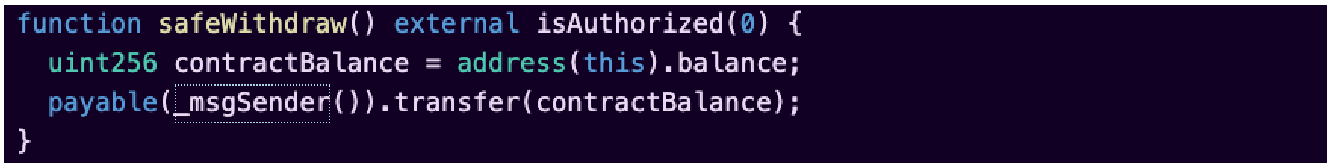
No medium severity issues found.

Low Severity Issues

No low severity issues found.

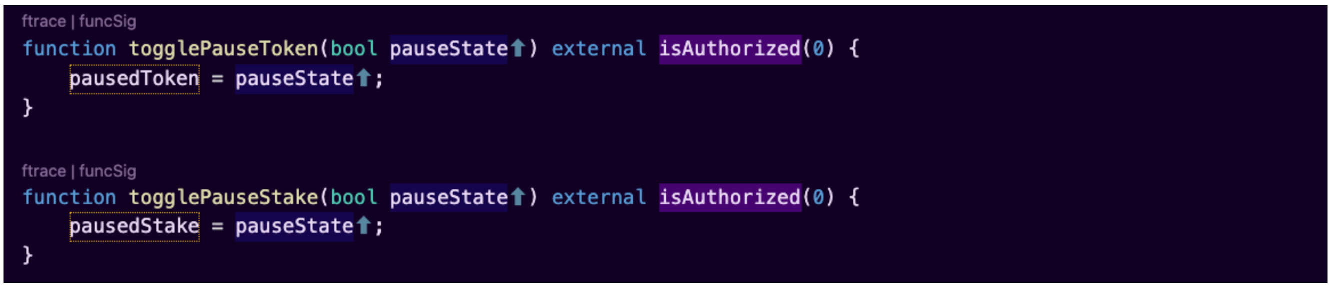
Owner privileges (In the period when the owner is not renounced)

- Authorized_0 addresses can withdraw contract balance.



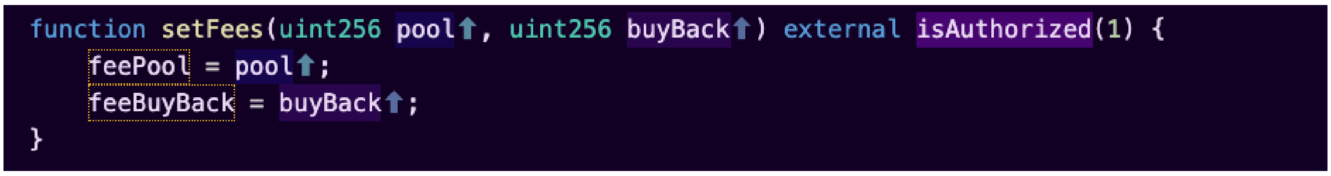
```
function safeWithdraw() external isAuthorized(0) {
  uint256 contractBalance = address(this).balance;
  payable(_msgSender()).transfer(contractBalance);
}
```

- Authorized_0 addresses can enable/disable token and stake pause.



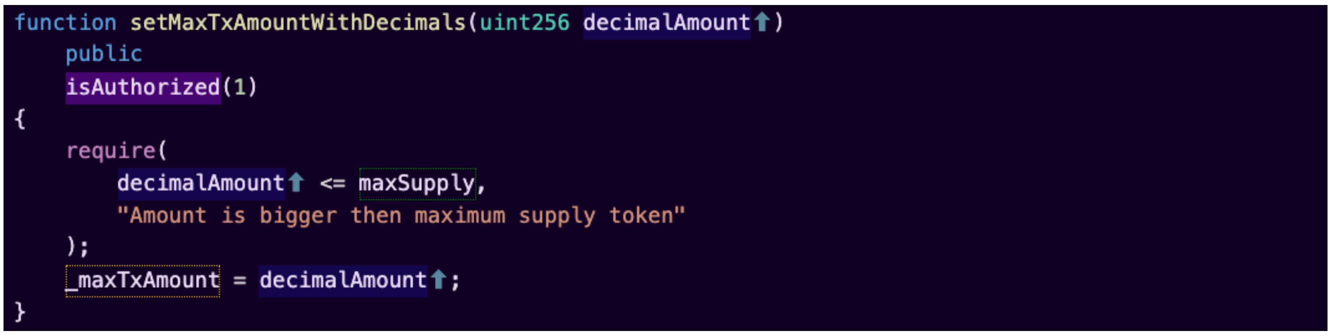
```
ftrace | funcSig
function togglePauseToken(bool pauseState) external isAuthorized(0) {
    pausedToken = pauseState;
}

ftrace | funcSig
function togglePauseStake(bool pauseState) external isAuthorized(0) {
    pausedStake = pauseState;
}
```

- Authorized_1 addresses can change fees.



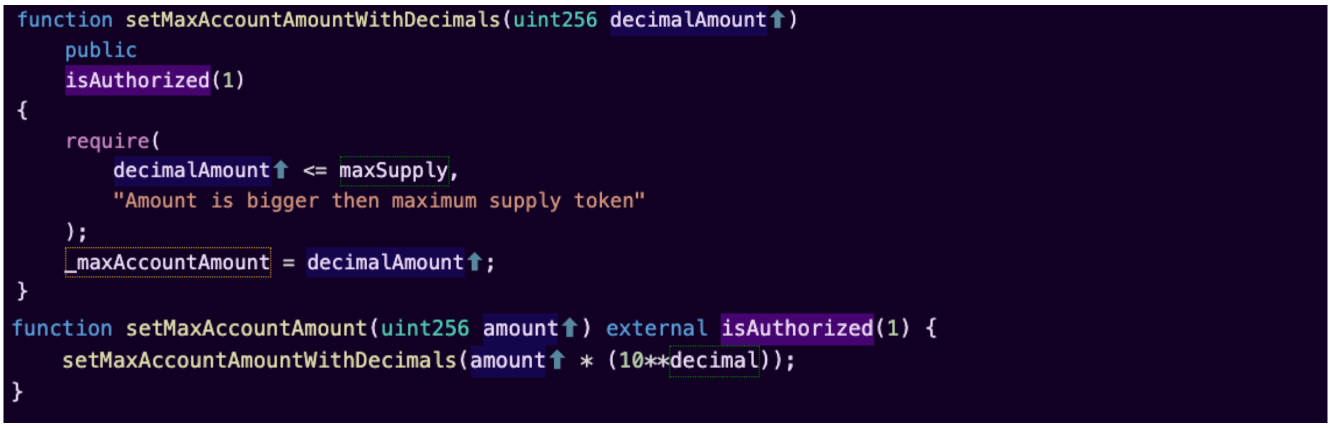
```
function setFees(uint256 pool, uint256 buyBack) external isAuthorized(1) {
    feePool = pool;
    feeBuyBack = buyBack;
}
```

- Authorized_1 addresses can change max transaction amount.



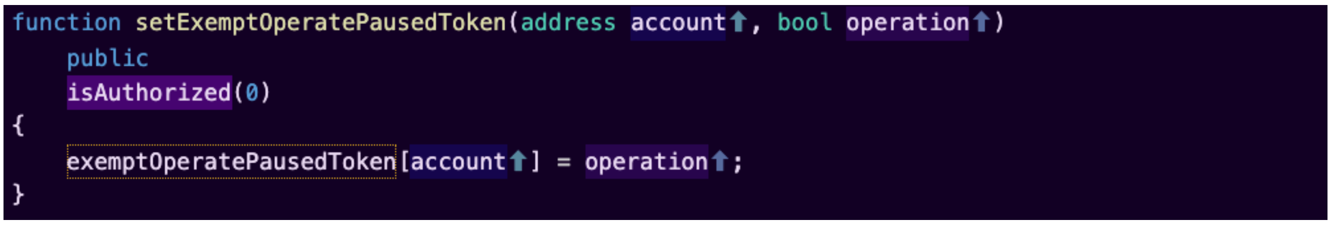
```
function setMaxTxAmountWithDecimals(uint256 decimalAmount)
    public
    isAuthorized(1)
{
    require(
        decimalAmount <= maxSupply,
        "Amount is bigger then maximum supply token"
    );
    _maxTxAmount = decimalAmount;
}
```

- Authorized_1 addresses can change max account amount.



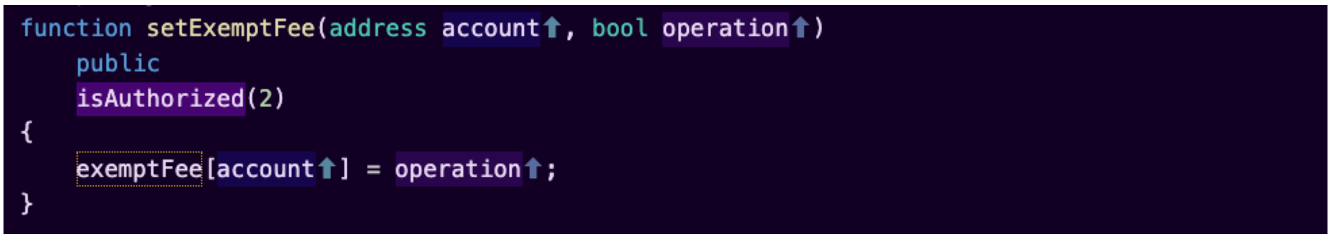
```
function setMaxAccountAmountWithDecimals(uint256 decimalAmount)
    public
    isAuthorized(1)
{
    require(
        decimalAmount <= maxSupply,
        "Amount is bigger then maximum supply token"
    );
    _maxAccountAmount = decimalAmount;
}
function setMaxAccountAmount(uint256 amount) external isAuthorized(1) {
    setMaxAccountAmountWithDecimals(amount * (10**decimal));
}
```

- Authorized_0 addresses can exclude from token pause.



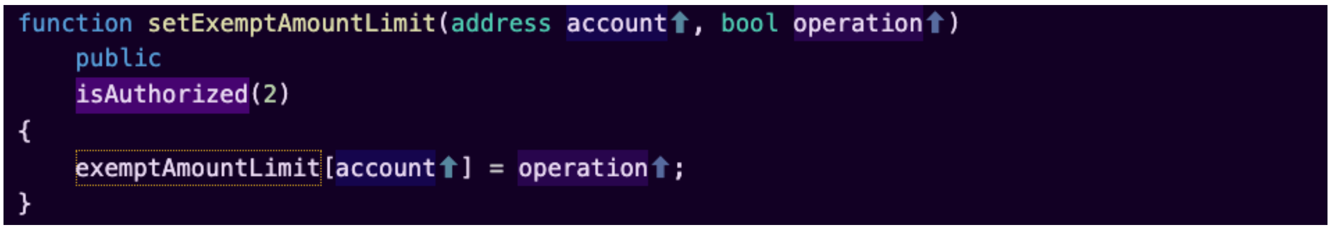
```
function setExemptOperatePausedToken(address account, bool operation)
    public
    isAuthorized(0)
{
    exemptOperatePausedToken[account] = operation;
}
```

- Authorized_2 addresses can exclude from fees.



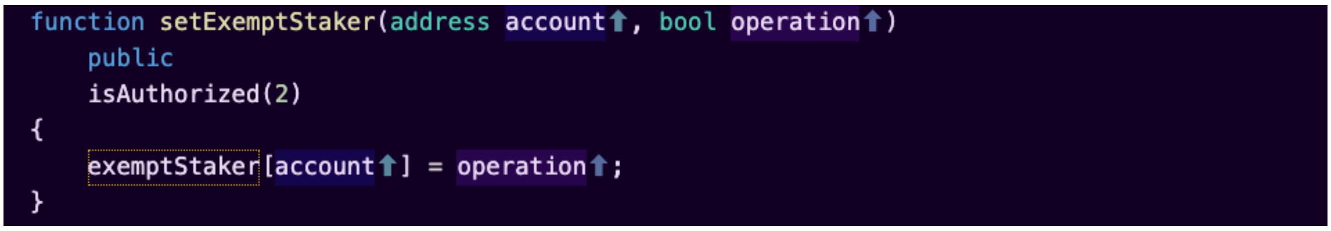
```
function setExemptFee(address account, bool operation)
    public
    isAuthorized(2)
{
    exemptFee[account] = operation;
}
```

- Authorized_2 addresses can exclude from amount limit.



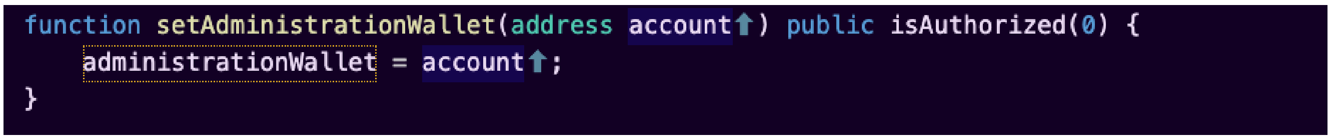
```
function setExemptAmountLimit(address account, bool operation)
    public
    isAuthorized(2)
{
    exemptAmountLimit[account] = operation;
}
```

- **Authorized_2 addresses can include account in exemptStaker array.**



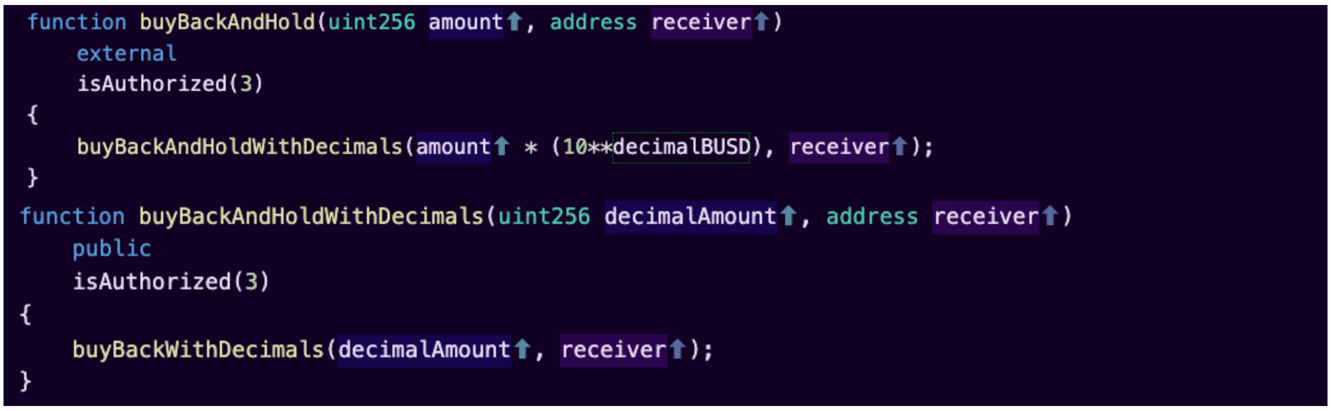
```
function setExemptStaker(address account, bool operation)
    public
    isAuthorized(2)
{
    exemptStaker[account] = operation;
}
```

- **Authorized_0 addresses can change administration wallet.**



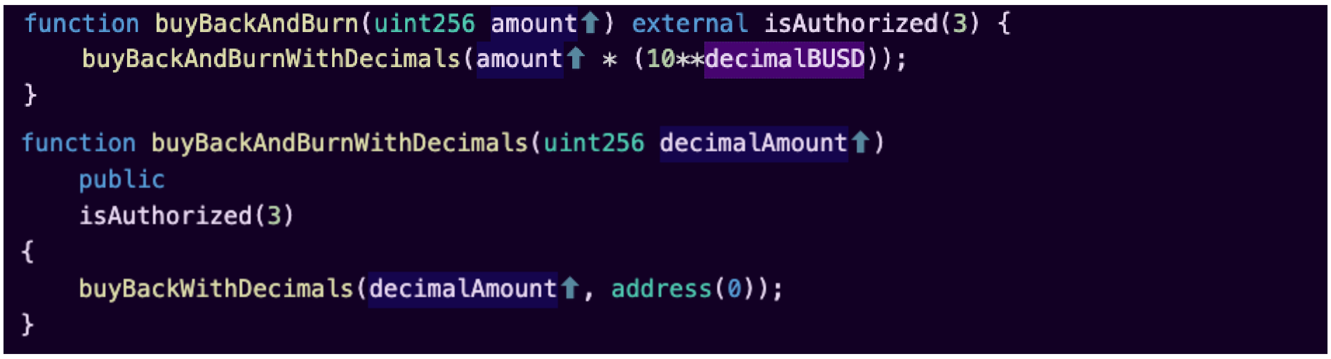
```
function setAdministrationWallet(address account) public isAuthorized(0) {
    administrationWallet = account;
}
```

- **Authorized_3 addresses can manually buyback.**



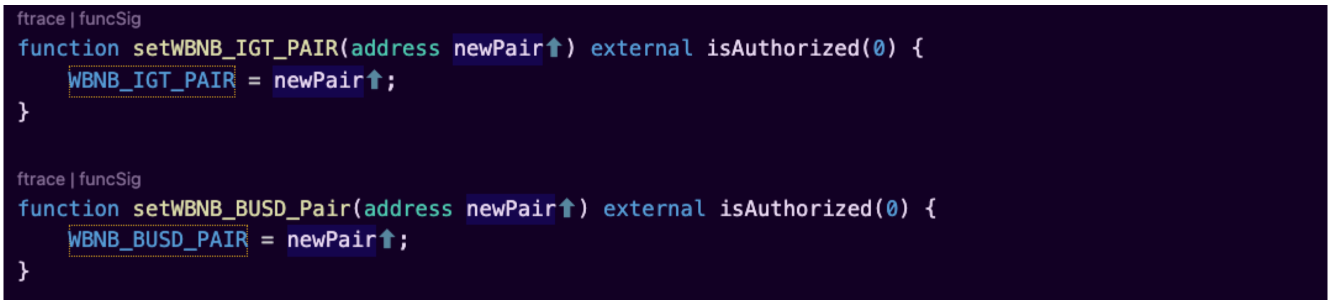
```
function buyBackAndHold(uint256 amount, address receiver)
    external
    isAuthorized(3)
{
    buyBackAndHoldWithDecimals(amount * (10**decimalBUSD), receiver);
}
function buyBackAndHoldWithDecimals(uint256 decimalAmount, address receiver)
    public
    isAuthorized(3)
{
    buyBackWithDecimals(decimalAmount, receiver);
}
```

- **Authorized_3 addresses can manually buyback and burn.**



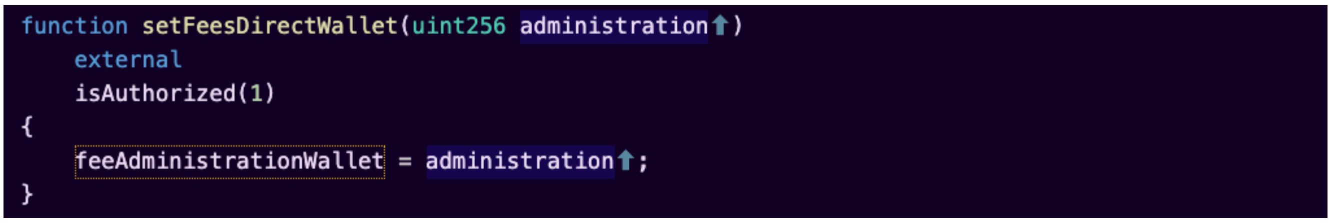
```
function buyBackAndBurn(uint256 amount) external isAuthorized(3) {
    buyBackAndBurnWithDecimals(amount * (10**decimalBUSD));
}
function buyBackAndBurnWithDecimals(uint256 decimalAmount)
    public
    isAuthorized(3)
{
    buyBackWithDecimals(decimalAmount, address(0));
}
```

- **Authorized_0 addresses can change WBNB_IGT_PAIR and WBNB_BUSD_PAIR.**



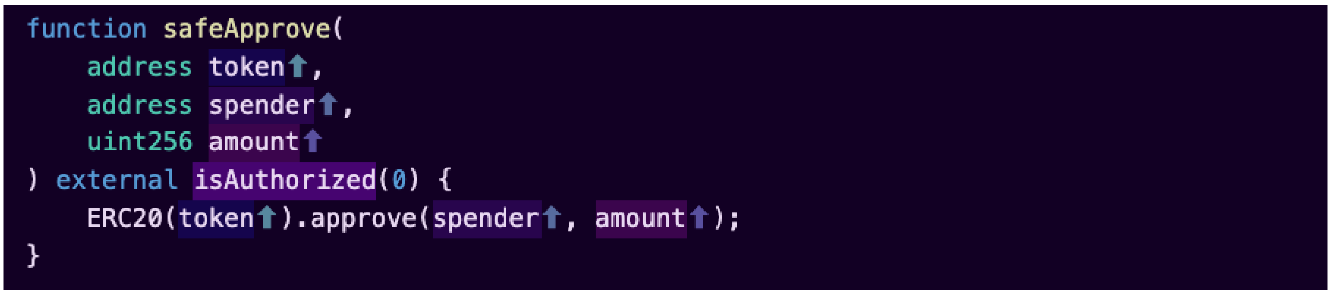
```
ftrace | funcSig
function setWBNB_IGT_PAIR(address newPair) external isAuthorized(0) {
    WBNB_IGT_PAIR = newPair;
}

ftrace | funcSig
function setWBNB_BUSD_Pair(address newPair) external isAuthorized(0) {
    WBNB_BUSD_PAIR = newPair;
}
```

- **Authorized_1 addresses can change fee administration wallet.**



```
function setFeesDirectWallet(uint256 administration)
    external
    isAuthorized(1)
{
    feeAdministrationWallet = administration;
}
```

- **Authorized_0 addresses can approve any ERC20 token amount for spender.**



```
function safeApprove(
    address token,
    address spender,
    uint256 amount
) external isAuthorized(0) {
    ERC20(token).approve(spender, amount);
}
```

Conclusion

Smart contracts contain owner privileges! Liquidity pairs 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.