Cyberscope

Audit Report

Trump Tweets

November 2022

Type	BEP20
Network	BSC
Address	0xad7B8e6486d04B84f66Cc9750fd1a4fc214Bd8FC
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	Trump_Tweets
Compiler Version	v0.8.7+commit.e28d00a7
Optimization	200 runs
Licence	MIT
Explorer	https://bscscan.com/token/0xad7B8e6486d04B84f66Cc9750fd1a4fc214Bd8FC
Symbol	TrumpTweets
Decimals	9
Total Supply	150,000,000
Domain	trumptweetsbsc.com

Source Files

Filename	**SHA256**
contract.sol	dc784411961283597503672733fe3e85951fffa999d3772e13c1af753882b259

Audit Updates

Initial Audit	3rd December 2022 https://github.com/cyberscope-io/audits/tree/main/trumptweets/v1/audit.pdf
Corrected	5th December 2022

Contract Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OCTD	Transfers Contract's Tokens	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Unresolved
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Unresolved

ST - Stops Transactions

Criticality	critical
Location	Trump_Tweets.sol
Status	Unresolved

Description

The contract owner has the authority to stop the sales for all users excluding the owner. As a result, the contract may operate as a honeypot. The owner may take advantage of it by setting:

- The `cooldownTimerInterval` to a huge value.
- The `_maxSellTxAmount` to zero.
- The `_maxBuyTxAmount` to zero.
- The `tradingOpen` to false.
- The `_maxWalletToken` to zero.

```
require(cooldownTimer[recipient] < block.timestamp,"Please wait between two
buys");
cooldownTimer[recipient] = block.timestamp + cooldownTimerInterval;

if(isSell){
    require(amount <= _maxSellTxAmount || isTxLimitExempt[sender] ||
isTxLimitExempt[recipient], "TX Limit Exceeded");
} else {
    require(amount <= _maxBuyTxAmount || isTxLimitExempt[sender] ||
isTxLimitExempt[recipient], "TX Limit Exceeded");
}


if(!authorizations[sender] && !authorizations[recipient]){
    require(tradingOpen,"Trading not enabled yet");
}
```

Recommendation

The contract could embody a check for not allowing setting the `_maxSellTxAmount`, `_maxBuyTxAmount` and `_maxWalletToken` less than a reasonable amount. A suggested implementation could check that the maximum amount should be more than a fixed percentage of the total supply. The `cooldownTimerInterval` should not be able to set more than a reasonable period. The `tradingOpen` should be allowed to be enabled once.

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

ELFM - Exceeds Fees Limit

Criticality	critical
Location	Trump_Tweets.sol#L904,915
Status	Unresolved

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `setBuyFees` or `setSellFees` function with a high percentage value.

```
// Set our sell fees
function setSellFees(uint256 _liquidityFeeSell, uint256 _buybackFeeSell, uint256 _reflectionFeeSell, uint256 _marketingFeeSell, uint256 _devFeeSell, uint256 _feeDenominator) external authorized {
    liquidityFeeSell = _liquidityFeeSell;
    buybackFeeSell = _buybackFeeSell;
    reflectionFeeSell = _reflectionFeeSell;
    marketingFeeSell = _marketingFeeSell;
    devFeeSell = _devFeeSell;
    totalFeeSell = _liquidityFeeSell.add(_buybackFeeSell).add(_reflectionFeeSell).add(_marketingFeeSell).add(_devFeeSell);
    feeDenominator = _feeDenominator;
}
```

Recommendation

The contract could embody a check for the maximum acceptable value.

The team should carefully manage the private keys of the owner’s account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

BC - Blacklists Addresses

Criticality	medium
Location	Trump_Tweets.sol#L553
Status	Unresolved

Description

The contract owner has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `blacklistAddress` function.

```
require(!isBlacklisted[recipient] && !isBlacklisted[sender], 'Address is blacklisted');
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

Contract Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	PTRP	Potential Transfer Revert Propagation	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	ZD	Zero Division	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L05	Unused State Variable	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L09	Dead Code Elimination	Unresolved

PTRP - Potential Transfer Revert Propagation

Criticality	minor / informative
Location	Trump_Tweets.sol#L769
Status	Unresolved

Description

The contract sends funds to a marketingFeeReceiver as part of the transfer flow. This address can either be a wallet address or a contract. If the address is a contract then it may revert from incoming payment. As a result, the error will propagate to the token's contract and revert the transfer.

```
(bool successMarketing, /* bytes memory data */) =
payable(marketingFeeReceiver).call{value: amountBNBMarketing, gas: 30000}("");
(bool successDev, /* bytes memory data */) = payable(DEV).call{value:
amountBNBDev, gas: 30000}("");
require(successMarketing, "marketing receiver rejected ETH transfer");
```

Recommendation

The contract should tolerate the potential revert from the underlying contracts when the interaction is part of the main transfer flow. This could be archived by not allowing set contract addresses or by sending the funds in a non-revertable way.

RSML - Redundant SafeMath Library

Criticality	minor / informative
Location	Trump_Tweets.sol#L367
Status	Unresolved

Description

The Solidity versions that are greater than or equal to 0.8.0 do not need the use of SafeMath Library. The usage of the SafeMath library produces unnecessary additional gas.

```
using SafeMath for uint256;
```

Recommendation

The team is advised to remove the SafeMath library as it is safe to do math operations without it.

ZD - Zero Division

Criticality	critical
Location	Trump_Tweets.sol
Status	Unresolved

Description

The contract is using variables that may be set to zero as denominators. As a result, the transactions will revert.

```
if(launchedAt + deadBlocks >= block.number){ return feeDenominator.sub(1); }
...
if (GREEDMode){
    if(isSell){
        // We are selling so up the selling tax to 1.5x
        feeAmount = amount.mul(totalFee).mul(3).div(2).div(feeDenominator);
    } else {
        // We are buying so cut our taxes in half
        feeAmount = amount.mul(totalFee).div(2).div(feeDenominator);
    }
} else {
    feeAmount = amount.mul(totalFee).div(feeDenominator);
}
```

Recommendation

The contract should prevent those variables to be set to zero or should not allow to execute the corresponding statements.

L02 - State Variables could be Declared Constant

Criticality	minor / informative
Location	Trump_Tweets.sol#L374,372,432,386,373,370,371,224,211
Status	Unresolved

Description

Constant state variables should be declared constant to save gas.

```
DEV
DEAD
deadBlocks
_totalSupply
ZERO
BUSD
WBNB
dividendsPerShareAccuracyFactor
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	Trump_Tweets.sol#L737,720,372,143,950,394,932,904,382,926,383,915,428,395,366,210,839,938,390,374,249,391,389,381,683,714,429,370,386,201,211,373,371,731
Status	Unresolved

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
_address
_seconds
DEAD
WETH
_minDistribution
_minPeriod
_balances
_amount
_feeDenominator
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-conventions.

L05 - Unused State Variable

Criticality	minor / informative
Location	Trump_Tweets.sol#L370
Status	Unresolved

Description

There are segments that contain unused state variables.

```
BUSD
```

Recommendation

Remove unused state variables.

L07 - Missing Events Arithmetic

Criticality	minor / informative
Location	Trump_Tweets.sol#L850,249,720,868,904,839,932,938,538,915,873,731
Status	Unresolved

Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.

```
buybackMultiplierNumerator = numerator
minPeriod = _minPeriod
GREEDDuration = _seconds
_maxBuyTxAmount = _totalSupply.mul(maxBuyTxPercent).div(10000)
liquidityFeeBuy = _liquidityFeeBuy
autoBuybackCap = _cap
swapThreshold = _totalSupply * _amount / 10000
targetLiquidity = _target
_maxWalletToken = _totalSupply.mul(maxWallPercent).div(10000)
...
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor / informative
Location	Trump_Tweets.sol#L858,788,813
Status	Unresolved

Description

Functions that are not used in the contract, and make the code's size bigger.

```
launched
shouldAutoBuyback
triggerAutoBuyback
```

Recommendation

Remove unused functions.

Contract Functions

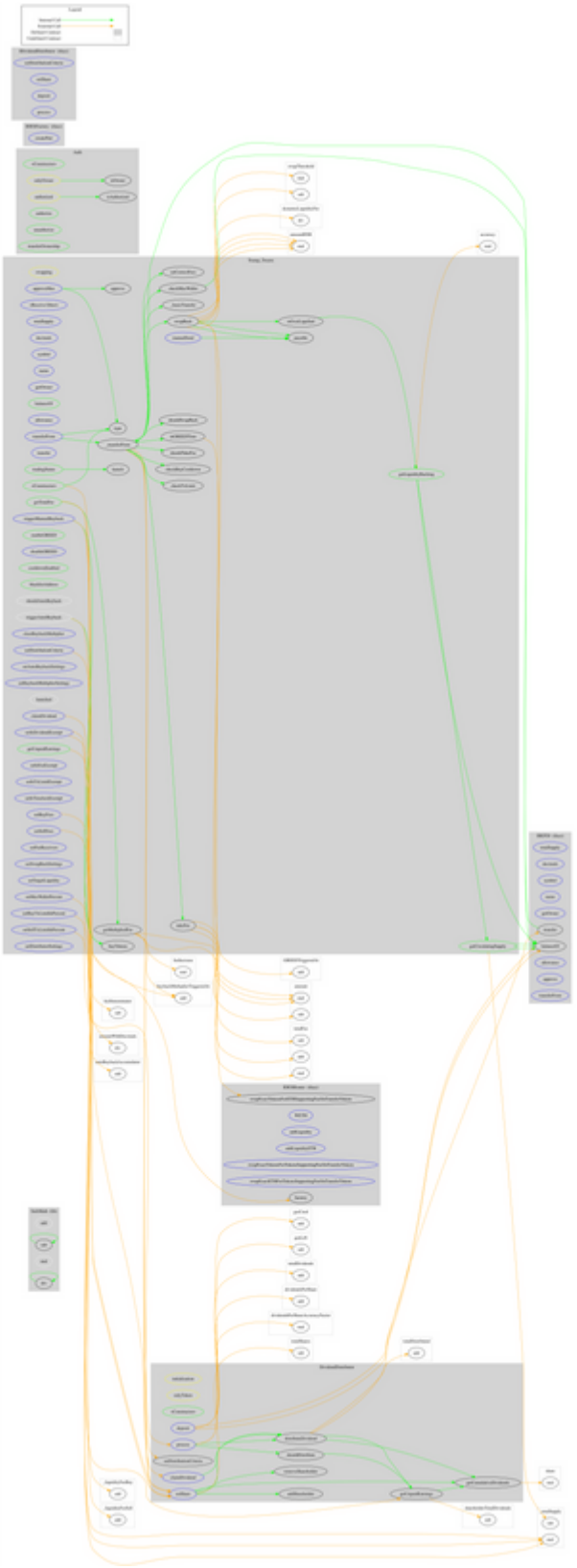
Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
IBEP20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Auth	Implementation			
	<Constructor>	Public	✓	-
	authorize	Public	✓	onlyOwner
	unauthorize	Public	✓	onlyOwner
	isOwner	Public		-
	isAuthorized	Public		-
	transferOwnership	Public	✓	onlyOwner
IDEXFactory	Interface			

	createPair	External	✓	-
IDEXRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IDividendDistributor	Interface			
	setDistributionCriteria	External	✓	-
	setShare	External	✓	-
	deposit	External	Payable	-
	process	External	✓	-
DividendDistributor	Implementation	IDividendDistributor		
	<Constructor>	Public	✓	-
	setDistributionCriteria	External	✓	onlyToken
	setShare	External	✓	onlyToken
	deposit	External	Payable	onlyToken
	process	External	✓	onlyToken
	shouldDistribute	Internal		
	distributeDividend	Internal	✓	
	claimDividend	External	✓	onlyToken
	getUnpaidEarnings	Public		-
	getCumulativeDividends	Internal		
	addShareholder	Internal	✓	
	removeShareholder	Internal	✓	
Trump_Tweets	Implementation	IBEP20,		

		Auth		
	<Constructor>	Public	✓	Auth
	<Receive Ether>	External	Payable	-
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	Public		-
	allowance	External		-
	approve	Public	✓	-
	approveMax	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	setMaxWalletPercent	External	✓	onlyOwner
	_transferFrom	Internal	✓	
	_basicTransfer	Internal	✓	
	setCorrectFees	Internal	✓	
	inGREEDTime	Public		-
	checkTxLimit	Internal		
	checkBuyCooldown	Internal	✓	
	checkMaxWallet	Internal		
	shouldTakeFee	Internal		
	getTotalFee	Public		-
	getMultipliedFee	Public		-
	takeFee	Internal	✓	
	shouldSwapBack	Internal		
	tradingStatus	Public	✓	onlyOwner
	enableGREED	Public	✓	authorized
	disableGREED	External	✓	authorized
	cooldownEnabled	Public	✓	authorized
	blacklistAddress	Public	✓	authorized
	swapBack	Internal	✓	swapping
	shouldAutoBuyback	Internal		
	triggerManualBuyback	External	✓	authorized

	clearBuybackMultiplier	External	✓	authorized
	triggerAutoBuyback	Internal	✓	
	buyTokens	Internal	✓	swapping
	setAutoBuybackSettings	External	✓	authorized
	setBuybackMultiplierSettings	External	✓	authorized
	launched	Internal		
	launch	Internal	✓	
	setBuyTxLimitInPercent	External	✓	authorized
	setSellTxLimitInPercent	External	✓	authorized
	setIsDividendExempt	External	✓	authorized
	setIsFeeExempt	External	✓	authorized
	setIsTxLimitExempt	External	✓	authorized
	setIsTimelockExempt	External	✓	authorized
	setBuyFees	External	✓	authorized
	setSellFees	External	✓	authorized
	setFeeReceivers	External	✓	authorized
	setSwapBackSettings	External	✓	authorized
	setTargetLiquidity	External	✓	authorized
	manualSend	External	✓	authorized
	setDistributionCriteria	External	✓	authorized
	claimDividend	External	✓	-
	getUnpaidEarnings	Public		-
	setDistributorSettings	External	✓	authorized
	getCirculatingSupply	Public		-
	getLiquidityBacking	Public		-
	isOverLiquified	Public		-

Contract Flow



Summary

There are some functions that can be abused by the owner like stopping transactions and blacklisting addresses. The contract can be converted into a honeypot and prevent users from selling if the owner abuses the admin functions. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io