



Pepe Of Wallstreet Token

RugfreeCoins Verified on September 06th, 2023

Overview

- ✓ No mint function found, the owner cannot mint tokens after initial deployment.
- ✓ The owner can't set a max transaction limit.
- ✓ The owner can't enable or disable trading.
- ✗ The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.
- ✓ The owner can't change fees over 20%.
- ✓ The owner can't blacklist wallets.
- ✓ The owner can't set a max wallet limit.
- ✓ The owner can't claim the contract's balance of its own token.

! High severity issues

The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.

```
function enableTrading() public onlyOwner {  
    require(!tradingEnabled, "Trading already enabled!");  
    require(hasLiqBeenAdded, "Liquidity must be added.");  
    tradingEnabled = true;  
    swapEnabled = true;  
    allowedPresaleExclusion = false;  
}
```

Contents

Overview.....	2
Contents.....	3
Audit details.....	4
Disclaimer.....	5
Background.....	6
Tokenomics.....	7
Target market and the concept.....	8
Potential to grow with score points.....	9
Contract details.....	10
Contract code function details.....	11
Contract description table.....	12
Security issue checking status.....	17
Owner privileges.....	19
Audit conclusion.....	24

Audit details



Audited project

Pepe Of Wallstreet Token



Contract Address

0x8cD372e27C42cDD00F588DBC610920ccb8d82d8D



Client contact

PepeOf Wallstreet Token Team



Blockchain

Binance Smart Chain



Project website

<https://pepeofwallstreet.info/>

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 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 **RugfreeCoins and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (RugfreeCoins) owe no duty of care towards you or any other person**, nor does RugfreeCoins 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 RugfreeCoins 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, RugfreeCoins hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against RugfreeCoins, 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

RugfreeCoins was commissioned by the **Pepe Of Wallstreet Token Team** to perform an audit of the smart contract.

<https://bscscan.com/token/0x8cd372e27c42cdd00f588dbc610920ccb8d82d8d>

This audit focuses on verifying that the smart contract is secure, resilient, and working according to the specifications.

The information in this report should be used to understand the risk exposure of the smart contract, project feasibility, and long-term sustainability, and as a guide to improving the smart contract's security posture by remediating the identified issues.

Tokenomics












▲ 3% tax when buying & selling

3% of trade goes to the dev wallet in BNB

Target market and the concept

- ▶ Anyone who's interested in the Crypto space with long-term investment plans.
- ▶ Anyone who's ready to earn a passive income by holding tokens.
- ▶ Anyone who's interested in trading tokens.
- ▶ Anyone who's interested in taking part in the Pepe Of Wallstreet Token ecosystem.
- ▶ Anyone who's interested in taking part in the future plans of Pepe Of Wallstreet Token.
- ▶ Anyone who's interested in making financial transactions with any other party using Pepe Of Wallstreet Token as the currency.

Potential to grow with score points

 Project efficiency	8 / 10
 Project uniqueness	7 / 10
 Information quality	8 / 10
 Service quality	8 / 10
 System quality	8 / 10
 Impact on the community	8 / 10
 Impact on the business	9 / 10
 Preparing for the future	8 / 10
 Smart contract security	9 / 10
 Smart contract functionality assessment	9 / 10
 Total Score	8.2 / 10

Contract details

Token contract details for 6th of September 2023

Contract name	Pepe Of Wallstreet (POW)
Contract address	0x8cD372e27C42cDD00F588DBC610920ccb8d82d8D
Token supply	420,000,000,000,000,000
Token ticker	POW
Decimals	6
Token holders	1
Transaction count	1
Contract deployer address	0x76b47db59d6934D4981Bac703A0ca6f1D05fE83e
Contract's current owner address	0x76b47db59d6934D4981Bac703A0ca6f1D05fE83e









Contract code function details



Nº	Category	Item	Result
1	Coding conventions	BRC20 Token standards	PASS ▾
		Compile errors	PASS ▾
		Compiler version security	PASS ▾
		Visibility specifiers	PASS ▾
		Gas consumption	LOW ▾
		SafeMath features	PASS ▾
		Fallback usage	PASS ▾
		tx.origin usage	PASS ▾
		Deprecated items	PASS ▾
		Redundant code	PASS ▾
2	Function call audit	Overriding variables	PASS ▾
		Authorization of function call	PASS ▾
		Low level function (call/delegate call) security	PASS ▾
		Returned value security	PASS ▾
		Self destruct function security	PASS ▾
3	Business security & centralisation	Access control of owners	HIGH ▾
		Business logics	PASS ▾
		Business implementation	PASS ▾
4	Integer overflow/underflow		PASS ▾
5	Reentrancy		PASS ▾
6	Exceptional reachable state		PASS ▾
7	Transaction ordering dependence		PASS ▾
8	Block properties dependence		PASS ▾
9	Pseudo random number generator (PRNG)		PASS ▾
10	DoS (Denial of Service)		PASS ▾
11	Token vesting implementation		PASS ▾
12	Fake deposit		PASS ▾
13	Event security		PASS ▾























Contract description table

The below table represents the summary of the contracts and methods in the token contract. We scanned the whole contract and listed down all the Interfaces, functions, and implementations with their visibility and mutability.



Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
SafeMath	Library			
L	add	Internal 🔒		
L	sub	Internal 🔒		
L	sub	Internal 🔒		
L	mul	Internal 🔒		
L	div	Internal 🔒		
L	div	Internal 🔒		
IERC20	Interface			
L	totalSupply	External !		NO !
L	decimals	External !		NO !
L	symbol	External !		NO !
L	name	External !		NO !
L	getOwner	External !		NO !
L	balanceOf	External !		NO !

L	transfer	External !		NO !
L	allowance	External !		NO !
L	approve	External !		NO !
L	transferFrom	External !		NO !
IFactoryV2	Interface			
L	getPair	External !		NO !
L	createPair	External !		NO !
IV2Pair	Interface			
L	factory	External !		NO !
L	getReserves	External !		NO !
L	sync	External !		NO !
IRouter01	Interface			
L	factory	External !		NO !
L	WETH	External !		NO !
L	addLiquidityETH	External !		NO !
L	addLiquidity	External !		NO !
L	swapExactETHForTokens	External !		NO !
L	getAmountsOut	External !		NO !

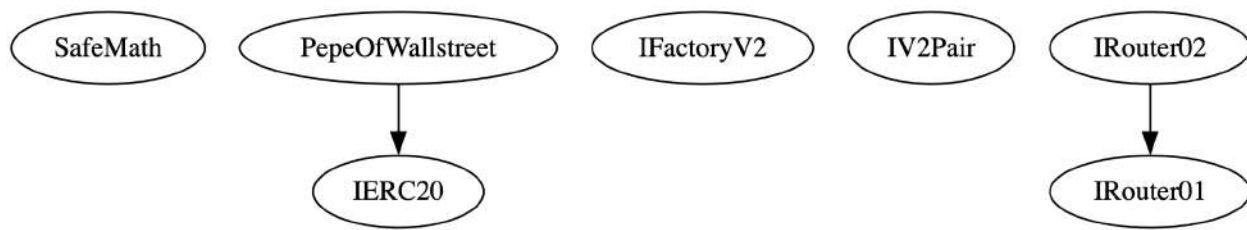
L	getAmountsIn	External !		NO !
IRouter02	Interface	IRouter01		
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External !		NO !
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External !		NO !
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External !		NO !
L	swapExactTokensForTokens	External !		NO !
PepeOfWallstreet	Implementation	IERC20		
L		Public !		NO !
L		External !		NO !
L	totalSupply	External !		NO !
L	decimals	External !		NO !
L	symbol	External !		NO !
L	name	External !		NO !
L	getOwner	External !		NO !
L	allowance	Public !		NO !
L	balanceOf	Public !		NO !
L	transfer	Public !		NO !
L	approve	External !		NO !
L	_approve	Internal 		

L	approveContractContingency	External !		onlyOwner
L	transferFrom	External !		NO !
L	_hasLimits	Internal 		
L	_transfer	Internal 		
L	finalizeTransfer	Internal 		
L	_transferAmount	Internal 		
L	_checkLiquidityAdd	Internal 		
L	swapTokensForEth	Private 		lockThe Swap
L	sendETHToFee	Private 		
L	manualswap	External !		NO !
L	manualsend	External !		NO !
L	transferOwner	External !		onlyOwner
L	renounceOwnership	External !		onlyOwner
L	excludePresaleAddresses	External !		onlyOwner
L	setDevAddress	External !		onlyOwner
L	excludeMultipleAccountsFromFees	Public !		onlyOwner
L	enableTrading	Public !		onlyOwner
L	setFee	Public !		onlyOwner
L	updateSwapEnabled	External !		onlyOwner

Legend

Symbol	Meaning
	Function can modify state
	Function is payable

Inheritance hierarchy



Security issue checking status

❖ High severity issues

The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.

```
function enableTrading() public onlyOwner {
    require(!tradingEnabled, "Trading already enabled!");
    require(hasLiqBeenAdded, "Liquidity must be added.");
    tradingEnabled = true;
    swapEnabled = true;
    allowedPresaleExclusion = false;
}
```

❖ Medium severity issues

No Medium severity issues found

❖ Low severity issues

When excluding multiple wallets from fees, the owner can input any number of wallets at a time. However, if the owner enters a large number of wallets, this function might fail due to the block gas limit.

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

❖ Centralization issues

No Centralization issues found

Owner privileges

- ❖ Owner can enable/disable swapping

```
function updateSwapEnabled(bool _enabled) external onlyOwner {  
    swapEnabled = _enabled;  
}
```

- ❖ Owner can change buy and sell fees maximum up-to 10% (buy max 5% and sell max 5%)

```
function setFee(uint256 _taxFeeOnBuy, uint256 _taxFeeOnSell)  
    public  
    onlyOwner  
{  
    require(_taxFeeOnBuy <= 50, "Tax cannot be more than 5.");  
    require(_taxFeeOnSell <= 50, "Tax cannot be more than 5.");  
    taxFeeOnBuy = _taxFeeOnBuy;  
    taxFeeOnSell = _taxFeeOnSell;  
}
```

- ❖ Owner can enable trading, but once enabled can not disable it again

```
function enableTrading() public onlyOwner {  
    require(!tradingEnabled, "Trading already enabled!");  
    require(hasLiqBeenAdded, "Liquidity must be added.");  
    tradingEnabled = true;  
    swapEnabled = true;  
    allowedPresaleExclusion = false;  
}
```

- ❖ Owner can include/exclude wallets from fees

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

- ❖ Owner can change the dev wallet address

```
function setDevAddress(address _newAddress) external onlyOwner {  
    require(devAddress != address(0), "address cannot be 0");  
    devAddress = payable(_newAddress);  
}
```

- ❖ Owner can exclude pre-sale address from the contract

```
function excludePresaleAddresses(address _router, address _presale)
    external
    onlyOwner
{
    require(allowedPresaleExclusion);
    require(
        _router != address(this) &&
        _presale != address(this) &&
        lpPair != _router &&
        lpPair != _presale,
        "Just don't."
    );
    if (_router == _presale) {
        _liquidityHolders[_presale] = true;
    } else {
        _liquidityHolders[_router] = true;
        _liquidityHolders[_presale] = true;
    }
}
```

- ❖ Owner can renounce the ownership

```
function renounceOwnership() external onlyOwner {
    address oldOwner = _owner;
    _owner = address(0);
    emit OwnershipTransferred(oldOwner, address(0));
}
```

❖ Owner can transfer ownership

```
function transferOwner(address _newOwner) external onlyOwner {
    require(
        _newOwner != address(0),
        "Call renounceOwnership to transfer owner to the zero address."
    );
    require(
        _newOwner != DEAD,
        "Call renounceOwnership to transfer owner to the zero address."
    );
    if (balanceOf(_owner) > 0) {
        finalizeTransfer(_owner, _newOwner, balanceOf(_owner));
    }

    address oldOwner = _owner;
    _owner = _newOwner;
    _isExcludedFromFee[_owner] = true;

    emit OwnershipTransferred(oldOwner, _newOwner);
}
```

❖ Owner and dev wallet can get contract BNB balance to dev wallet

```
function manualSend() external {
    require(msg.sender == devAddress || msg.sender == _owner);
    uint256 contractETHBalance = address(this).balance;
    sendETHToFee(contractETHBalance);
}
```

- ❖ Owner and dev wallet can manually trigger the swap

```
function manualswap() external {  
    require(msg.sender == devAddress || msg.sender == _owner);  
    uint256 contractBalance = balanceOf(address(this));  
    swapTokensForEth(contractBalance);  
}
```

Audit conclusion

RugFreeCoins team has performed in-depth testing, line-by-line manual code review, and automated audit of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, manipulations, and hacks. According to the smart contract audit.

Smart contract functional Status:	PASS ▾
Number of risk issues:	2
Solidity code functional issue level:	PASS ▾
Number of owner privileges:	10
Centralization risk correlated to the active owner:	HIGH ▾
Smart contract active ownership:	ACTIVE ▾