



RUGFREECOINS



Pepe Fine Token

RugfreeCoins Verified on September 24th, 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 pause trading once it's enabled
- ✗ 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 25%..
- ✓ 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 has the authority to adjust the swap threshold, allowing for changes of up to 1000% from the total supply. However, it's important to note that if the owner sets an excessively high swap threshold, it could lead to potential failures in the swap process due to significant price impact. Such failures have the potential to halt all selling transactions.

```
function updateLiquidityTreshhold(uint256 new_amount) external onlyOwner {
    require(
        new_amount <= 1e7,
        "Swap threshold amount should be lower or equal to 1% of tokens"
    );
    tokenLiquidityThreshold = new_amount * 10 ** decimals();
}
```

The launch taxes are initially set to a default value of 0, and the owner lacks any function to modify them subsequently. Consequently, all trades occurring within the specified dead block period will incur 0 taxes.

```
    } else if (useLaunchFee) {  
        feeswap = launchtax;  
        feesum = launchtax;  
    }
```

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

```
function EnableTrading() external onlyOwner {  
    require(!tradingEnabled, "Cannot re-enable trading");  
    tradingEnabled = true;  
    providingLiquidity = true;  
    genesis_block = block.number;  
}
```

Contents

Overview.....	2
Contents.....	4
Audit details.....	5
Disclaimer.....	6
Background.....	7
Tokenomics.....	8
Target market and the concept.....	9
Potential to grow with score points.....	10
Contract details.....	11
Contract code function details.....	12
Contract description table.....	13
Inheritance Hierarchy.....	16
Security issue checking status.....	17
Owner privileges.....	20
Audit conclusion.....	24

Audit details



Audited project
Pepe Fine Token



Contract Address
0xA02A2dEBf9d4e7651A73979dFF8A2dC10F94d0e8



Client contact
Pepe Fin Token Team



Blockchain
Binance Smart Chain



Project website

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 Fine Token Team** to perform an audit of the smart contract.

<https://bscscan.com/token/0xa02a2debf9d4e7651a73979dff8a2dc10f94d0e8>

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

▲ 2% tax when buying & selling (24/09/2023)












2% of trade goes to the Dev wallet in BNB

0% of trade goes to the Liquidity pool

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 Fine token ecosystem.
- ▶ Anyone who's interested in taking part in the future plans of Pepe Fine Token.
- ▶ Anyone who's interested in making financial transactions with any other party using Pepe Fine 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	6 / 10
 Smart contract functionality assessment	7 / 10
 Total Score	7.7/ 10

Contract details

Token contract details for 24th of September 2023



























Contract name	PepeFine
Contract address	0xA02A2dEBf9d4e7651A73979dFF8A2dC10F94d0e8
Token supply	1,000,000
Token ticker	PepeFine
Decimals	18
Token holders	1
Transaction count	1
Contract deployer address	0x4e2e6eb35e648168fcdE3a066b261540768ad1FC
Contract's current owner address	0x4e2e6eb35e648168fcdE3a066b261540768ad1FC
Dev wallet	0x4e2e6eb35e648168fcdE3a066b261540768ad1FC

Contract code function details

Nº	Category	Item	Result
1	Coding conventions	ERC20 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 ▾
3	Business security & centralisation	Self destruct function security	PASS ▾
		Access control of owners	HIGH ▾
		Business logics	HIGH ▾
4	Integer overflow/underflow	Business implementation	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
Context	Implementation			
L	_msgSender	Internal 		
L	_msgData	Internal 		
IERC20	Interface			
L	totalSupply	External 		NO 
L	balanceOf	External 		NO 
L	transfer	External 		NO 
L	allowance	External 		NO 
L	approve	External 		NO 
L	transferFrom	External 		NO 
IERC20 Metadata	Interface	IERC20		
L	name	External 		NO 
L	symbol	External 		NO 
L	decimals	External 		NO 
ERC20	Implementation	Context, IERC20, IERC20 Metadata		
L		Public 		NO 

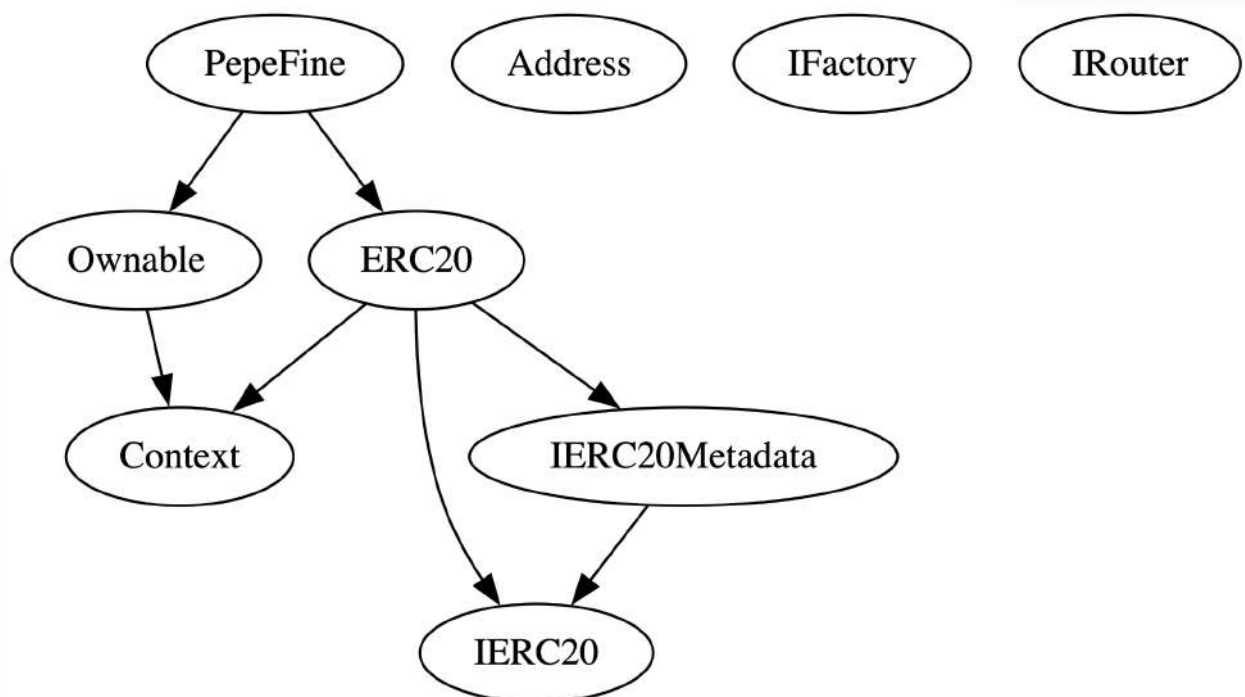
L	name	Public !		NO !
L	symbol	Public !		NO !
L	decimals	Public !		NO !
L	totalSupply	Public !		NO !
L	balanceOf	Public !		NO !
L	transfer	Public !	⛔	NO !
L	allowance	Public !		NO !
L	approve	Public !	⛔	NO !
L	transferFrom	Public !	⛔	NO !
L	increaseAllowance	Public !	⛔	NO !
L	decreaseAllowance	Public !	⛔	NO !
L	_transfer	Internal 🔒	⛔	
L	_tokengeneration	Internal 🔒	⛔	
L	_approve	Internal 🔒	⛔	
Address	Library			
L	sendValue	Internal 🔒	⛔	
Ownable	Implementation	Context		
L		Public !	⛔	NO !
L	owner	Public !		NO !
L	renounceOwnership	Public !	⛔	onlyOwner
L	transferOwnership	Public !	⛔	onlyOwner
L	_setOwner	Private 🔒🔑	⛔	
IFactory	Interface			
L	createPair	External !	⛔	NO !
IRouter	Interface			

L	factory	External !		NO !
L	WETH	External !		NO !
L	addLiquidityETH	External !		NO !
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External !		NO !
PepeFine	Implementation	ERC20, Ownable		
L		Public !		ERC20
L	approve	Public !		NO !
L	transferFrom	Public !		NO !
L	increaseAllowance	Public !		NO !
L	decreaseAllowance	Public !		NO !
L	transfer	Public !		NO !
L	_transfer	Internal		
L	Liquify	Private		lockTheSwap
L	swapTokensForETH	Private		
L	addLiquidity	Private		
L	updateLiquidityProvide	External !		onlyOwner
L	updateLiquidityTreshhold	External !		onlyOwner
L	EnableTrading	External !		onlyOwner
L	updatedeadline	External !		onlyOwner
L	updateTax	External !		onlyOwner
L	updateExemptFee	External !		onlyOwner
L	bulkExemptFee	External !		onlyOwner
L	rescueBNB	External !		onlyOwner
L	rescueBEP20	External !		onlyOwner
L		External !		NO !

Legend

Symbol	Meaning
🛑	Function can modify state
💰	Function is payable

Inheritance Hierarchy



Security issue checking status

❖ High severity issues

The owner has the authority to adjust the swap threshold, allowing for changes of up to 1000% from the total supply. However, it's important to note that if the owner sets an excessively high swap threshold, it could lead to potential failures in the swap process due to significant price impact. Such failures have the potential to halt all selling transactions.

```
function updateLiquidityTreshhold(uint256 new_amount) external onlyOwner {  
    require(  
        new_amount <= 1e7,  
        "Swap threshold amount should be lower or equal to 1% of tokens"  
    );  
    tokenLiquidityThreshold = new_amount * 10 ** decimals();  
}
```

The launch taxes are initially set to a default value of 0, and the owner lacks any function to modify them subsequently. Consequently, all trades occurring within the specified dead block period will incur 0 taxes.

```
} else if (useLaunchFee) {  
    feeswap = launchtax;  
    feesum = launchtax;  
}
```

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

```
function EnableTrading() external onlyOwner {
    require(!tradingEnabled, "Cannot re-enable trading");
    tradingEnabled = true;
    providingLiquidity = true;
    genesis_block = block.number;
}
```

❖ Medium severity issues

No medium severity issues found

❖ Low severity issues

The variables feeSwap and feesum consistently hold the same value. Therefore, there is no necessity to maintain two separate variables for handling this situation. The presence of extra variables consumes additional gas unnecessarily.

```
else if (recipient == pair && !useLaunchFee) {
    feeswap = sellTaxes.liquidity + sellTaxes.dev;
    feesum = feeswap;
    currentTaxes = sellTaxes;
} else if (!useLaunchFee) {
    feeswap = taxes.liquidity + taxes.dev;
    feesum = feeswap;
    currentTaxes = taxes;
} else if (useLaunchFee) {
    feeswap = launchtax;
    feesum = launchtax;
}
```

Since feeSwap and feesum have the same value for "fee," and feeAmount also holds the same value, there is no need to calculate feeAmount separately; it can be used directly for this purpose.

```
//rest to recipient
super._transfer(sender, recipient, amount - fee);
if (fee > 0) {
    //send the fee to the contract
    if (feeswap > 0) {
        uint256 feeAmount = (amount * feeswap) / 100;
        super._transfer(sender, address(this), feeAmount);
    }
}
```

When excluding multiple wallets from fees, the owner has the flexibility to input any number of wallets simultaneously. However, it's important to note that if the owner enters a large array of wallets, the transaction may fail due to the gas limit.

```
function bulkExemptFee(
    address[] memory accounts,
    bool state
) external onlyOwner {
    for (uint256 i = 0; i < accounts.length; i++) {
        exemptFee[accounts[i]] = state;
    }
}
```

Owner privileges

- ❖ Owner can enable/disable swapping

```
function updateLiquidityProvide(bool state) external onlyOwner {  
    providingLiquidity = state;  
}
```

- ❖ Owner can change the swap threshold up-to 1000%

```
function updateLiquidityTreshhold(uint256 new_amount) external onlyOwner {  
    require(  
        new_amount <= 1e7,  
        "Swap threshold amount should be lower or equal to 1% of tokens"  
    );  
    tokenLiquidityThreshold = new_amount * 10 ** decimals();  
}
```

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

```
function EnableTrading() external onlyOwner {  
    require(!tradingEnabled, "Cannot re-enable trading");  
    tradingEnabled = true;  
    providingLiquidity = true;  
    genesis_block = block.number;  
}
```

- ❖ Owner can change number of dead blocks up-to 4 before enabling trades (all trades within dead blocks will take launch taxes)

```
function updatedecline(uint256 _deadline) external onlyOwner {
    require(!tradingEnabled, "Can't change when trading has started");
    require(_deadline < 5, "Deadline should be less than 5 Blocks");
    deadline = _deadline;
}
```

- ❖ Owner can change all buy and sell fees total fees up-to 25% (buy tax up-to 5% and sell tax up-to 20%)

```
function updateTax(
    uint256 buyDevTax,
    uint256 buyLiquidityTax,
    uint256 sellDevTax,
    uint256 sellLiquidityTax
) external onlyOwner {
    require(
        (buyDevTax + buyLiquidityTax) <= 5,
        "Can't set tax greater than 5%"
    );
    require(
        (sellDevTax + sellLiquidityTax) <= 20,
        "Can't set tax greater than 20%"
    );
    taxes = Taxes(buyDevTax, buyLiquidityTax);
    sellTaxes = Taxes(sellDevTax, sellLiquidityTax);
}
```

- ❖ Owner can include/exclude wallets from fees

```
function updateExemptFee(address _address, bool state) external onlyOwner {  
    exemptFee[_address] = state;  
}
```

- ❖ Owner can include/exclude multiple wallets from fees

```
function bulkExemptFee(  
    address[] memory accounts,  
    bool state  
) external onlyOwner {  
    for (uint256 i = 0; i < accounts.length; i++) {  
        exemptFee[accounts[i]] = state;  
    }  
}
```

- ❖ Owner can get contract BNB balance to owner wallet

```
function rescueBNB(uint256 weiAmount) external onlyOwner {  
    payable(owner()).transfer(weiAmount);  
}
```

- ❖ Owner can get any BEP20 tokens from the contract (can not get native tokens)

```
function rescueBEP20(address tokenAdd, uint256 amount) external onlyOwner {  
    require(  
        tokenAdd != address(this),  
        "Owner can't claim contract's balance of its own tokens"  
    );  
    IERC20(tokenAdd).transfer(owner(), amount);  
}
```

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:	6
Solidity code functional issue level:	PASS ▾
Number of owner privileges:	9
Centralization risk correlated to the active owner:	HIGH ▾
Smart contract active ownership:	ACTIVE ▾