RugFreeCoins Audit

Peperika Token
Smart Contract Security Audit

May 12th ,2023

Contents

Audit details

Audited project
Peperika Token

Contract Address
0xbA03ea394FaF07FFB47801f41E6c08a3404784ec

Client contact
Peperika Team

Blockchain
Binance smart chain

Project website
Not Available

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.

Overview

✅ No mint function found, the owner cannot mint tokens after initial deployment.

✅ The owner can set a max transaction limit minimum of up to 0.1%

✅ The owner can’t pause trading.

✅ The owner can't blacklist wallets.

✅ The owner can set a max wallet limit minimum of up to 0.1%

✅ The owner can’t claim the contract’s balance of its own token.

✅ The owner can’t set fees over 20%.

Background

Rugfreecoins was commissioned by the Peperika Team to perform an audit of the smart contract.

https://bscscan.com/token/0xbA03ea394FaF07FFB47801f41E6c08a3404784ec

The focus of this audit is to verify 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

10% tax when buying & selling

6% of trade goes to marketing wallet in BNB
4% of trade goes among all holders in BNB
0% of trade goes to buyback wallet in BNB

Target market and the concept

Target market

- 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 Peperika ecosystem.
- Anyone who’s interested in taking part in the future plans of Peperika Token.
- Anyone who’s interested in making financial transactions with any other party using Peperika Token as the currency.

Contract details

Token contract details for 12th of May 2023

Contract name	PEPERICA
Contract address	0xbA03ea394FaF07FFB47801f41E6c08a3404784ec
Token supply	420,690,000,000,000
Token ticker	$PEPER
Decimals	9
Token holders	1
Transaction count	1
Contract deployer address	0x8Be30cc514D9D151FF309f9a0570D138099F4a53
Contract's current owner address	0x8be30cc514d9d151ff309f9a0570d138099f4a53
Marketing wallet	0xf80c630e14a63e5b9b9f9589b7ad6df9bcf05200
Auto Liquidity Receiver	0x8be30cc514d9d151ff309f9a0570d138099f4a53
Buyback Receiver	0xf80c630e14a63e5b9b9f9589b7ad6df9bcf05200
Distributor	0x566a6ab7522760f3f8d67600ff03eab13dbc6157

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	pass
		SafeMath features	pass
		Fallback usage	pass
		tx.origin usage	pass
		deprecated items	pass
		Redundant code	pass
		Overriding variables	pass
2	Function call audit	Authorization of function call	pass
		Low level function (call/delegate call) security	pass
		Returned value security	Low issue
		Selfdestruct function security	pass
3	Business security	Access control of owners	pass
		Business logics	pass
		Business implementations	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
IDexFactory	**Interface**			
L	createPair	External !	🛑	NO !
IDexRouter	**Interface**			
L	factory	External !		NO !
L	WETH	External !		NO !
L	addLiquidityETH	External !	💵	NO !
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External !	💵	NO !
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External !	🛑	NO !
IERC20 Extended	**Interface**			
L	totalSupply	External !		NO !
L	decimals	External !		NO !
L	symbol	External !		NO !
L	name	External !		NO !
L	balanceOf	External !		NO !

└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
Context	**Implementation**			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	**Implementation**	**Context**		
└		Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
Idividend Distributor	**Interface**			
└	setDistributionCriteria	External ❗	🛑	NO ❗
└	setShare	External ❗	🛑	NO ❗
└	deposit	External ❗	💵	NO ❗
└	process	External ❗	🛑	NO ❗
└	claimDividend	External ❗	🛑	NO ❗
└	getPaidEarnings	External ❗		NO ❗

└	getUnpaidEarnings	External !		NO !
└	totalDistributed	External !		NO !
Dividend Distributor	**Implementation**	**IDividendDistributor**		
└		Public !	🛑	NO !
└	setDistributionCriteria	External !	🛑	onlyToken
└	setShare	External !	🛑	onlyToken
└	deposit	External !	💸	onlyToken
└	process	External !	🛑	onlyToken
└	shouldDistribute	Internal 🔒		
└	distributeDividend	Internal 🔒	🛑	
└	claimDividend	External !	🛑	NO !
└	getPaidEarnings	Public !		NO !
└	getUnpaidEarnings	Public !		NO !
└	getCumulativeDividends	Internal 🔒		
└	addShareholder	Internal 🔒	🛑	
└	removeShareholder	Internal 🔒	🛑	
PEPERICA	**Implementation**	**IERC20 Extended, Ownable**		
└		Public !	🛑	Ownable
└		External !	💸	NO !

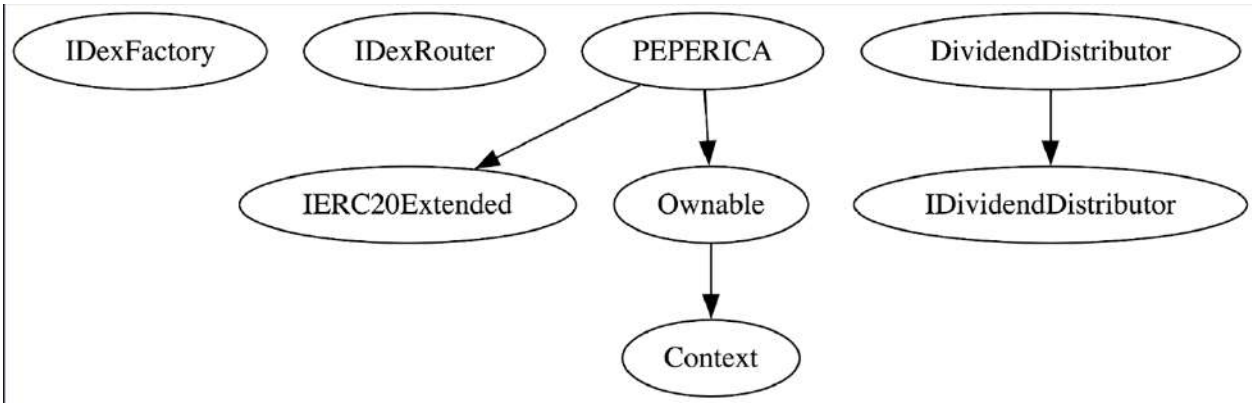
└	totalSupply	External !		NO !
└	decimals	External !		NO !
└	symbol	External !		NO !
└	name	External !		NO !
└	balanceOf	Public !		NO !
└	allowance	External !		NO !
└	approve	Public !	🛑	NO !
└	approveMax	External !	🛑	NO !
└	transfer	External !	🛑	NO !
└	transferFrom	External !	🛑	NO !
└	_transferFrom	Internal 🔒	🛑	
└	_basicTransfer	Internal 🔒	🛑	
└	takeFee	Internal 🔒	🛑	
└	setBuyAccFee	Internal 🔒	🛑	
└	setSellAccFee	Internal 🔒	🛑	
└	shouldSwapBack	Internal 🔒		
└	swapBack	Internal 🔒	🛑	swapping
└	claimDividend	External !	🛑	NO !
└	getPaidDividend	Public !		NO !
└	getUnpaidDividend	External !		NO !
└	getTotalDistributedDividend	External !		NO !

└	setIsDividendExempt	External ❗	🛑	onlyOwner
└	enableTrading	External ❗	🛑	onlyOwner
└	setMaxTxnAmount	External ❗	🛑	onlyOwner
└	setMaxWalletAmount	External ❗	🛑	onlyOwner
└	setIsFeeExempt	External ❗	🛑	onlyOwner
└	setIsLimitExempt	External ❗	🛑	onlyOwner
└	setIsWalletExempt	External ❗	🛑	onlyOwner
└	setBuyFees	Public ❗	🛑	onlyOwner
└	setSellFees	Public ❗	🛑	onlyOwner
└	setFeeReceivers	External ❗	🛑	onlyOwner
└	setSwapBackSettings	External ❗	🛑	onlyOwner
└	setDistributionCriteria	External ❗	🛑	onlyOwner
└	setDistributorSettings	External ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Inheritance Hierarchy



Security issue checking status

- **High severity issues**
  No high severity issues found

- **Medium severity issues**
  No medium severity issues found

- **Low severity issues**

There is no amount validation method in removeStuckEth function, if the contract has less BNB than the amount value this will fail and better to require statement to check contract BNB balance greater than or equal to the amount or not

Informed & fixed

```
ftrace | funcSig
function removeStuckEth(uint256 amount⬆) external onlyOwner {
    payable(owner()).transfer(amount⬆);
}
```

- **In setSwapBackSettings function, it's trying to check new threshold greater than to 0 or not but instead of _amount variable, it is checking swapThershold variable, if the owner accidentally put swapThershold to 0 then the owner can not change the swap setting after that.**

Informed & fixed

```
ftrace | funcSig
function setSwapBackSettings(
    bool _enabled⬆,
    uint256 _amount⬆
) external onlyOwner {
    require(swapThreshold > 0);
    swapEnabled = _enabled⬆;
    swapThreshold = _amount⬆;
}
```

- **autoLiquidityReceiver not used anywhere in the contract**

❖ **Centralization Risk**
No centralization issues found

Owner privileges

- Owner can include/exclude wallets from dividends

```
ftrace | funcSig
function setIsDividendExempt(
    address holder⬆,
    bool exempt⬆
) external onlyOwner {
    require(holder⬆ != address(this) && holder⬆ != pair);
    isDividendExempt[holder⬆] = exempt⬆;
    if (exempt⬆) {
        distributor.setShare(holder⬆, 0);
    } else {
        distributor.setShare(holder⬆, _balances[holder⬆]);
    }
}
```

- Owner can enable trading once enabled can not disable again

```
ftrace | funcSig
function enableTrading() external onlyOwner {
    require(!trading, "Already enabled");
    trading = true;
    swapEnabled = true;
    launchedAt = block.timestamp;
}
```

- Owner can set max transaction amount minimum up to 0.1%

```
ftrace | funcSig
function setMaxTxnAmount(uint256 amount⬆) external onlyOwner {
    require(amount⬆ >= _totalSupply / 1000);
    maxTxnAmount = amount⬆;
}
```

❖ Owner can set max wallet amount minimum up to 0.1%

```
ftrace | funcSig
function setMaxWalletAmount(uint256 amount⬆) external onlyOwner {
    require(amount⬆ >= _totalSupply / 1000);
    maxWalletAmount = amount⬆;
}
```

❖ Owner can include/exclude wallet from fees

```
ftrace | funcSig
function setIsFeeExempt(address holder⬆, bool exempt⬆) external onlyOwner {
    isFeeExempt[holder⬆] = exempt⬆;
}
```

❖ Owner can include/exclude wallets from transactions limit

```
ftrace | funcSig
function setIsLimitExempt(
    address[] memory holders⬆,
    bool exempt⬆
) external onlyOwner {
    for (uint256 i; i < holders⬆.length; i++) {
        isLimitExmpt[holders⬆[i]] = exempt⬆;
    }
}
```

❖ Owner can include/exclude wallet from max wallet limit

```
ftrace | funcSig
function setIsWalletExempt(address holder⬆, bool exempt⬆) external onlyOwner {
    isWalletExmpt[holder⬆] = exempt⬆;
}
```

❖ Owner can change buy fees maximum up to 15%

```
ftrace | funcSig
function setBuyFees(
    uint256 _reflectionFee⬆,
    uint256 _buyBackFee⬆,
    uint256 _marketingFee⬆
) public onlyOwner {
    _reflectionBuyFee = _reflectionFee⬆;
    _buyBackBuyFee = _buyBackFee⬆;
    _marketingBuyFee = _marketingFee⬆;
    totalBuyFee = _buyBackFee⬆ + (_reflectionFee⬆) + (_marketingFee⬆);
    require(
        totalBuyFee <= (feeDenominator * 15) / (100),
        "Can't be greater than 15%"
    );
}
```

❖ Owner can change sell fees maximum upto 15%

```
ftrace | funcSig
function setSellFees(
    uint256 _buyBackFee⬆,
    uint256 _reflectionFee⬆,
    uint256 _marketingFee⬆
) public onlyOwner {
    _buyBackSellFee = _buyBackFee⬆;
    _reflectionSellFee = _reflectionFee⬆;
    _marketingSellFee = _marketingFee⬆;
    totalSellFee = _buyBackFee⬆ + (_reflectionFee⬆) + (_marketingFee⬆);
    require(
        totalSellFee <= (feeDenominator * 15) / (100),
        "Can't be greater than 15%"
    );
}
```

❖ Owner can change all fee receivers

```
ftrace | funcSig
function setFeeReceivers(
    address _autoLiquidityReceiver⬆,
    address _marketingFeeReceiver⬆,
    address _buyBackFeeReceiver⬆
) external onlyOwner {
    autoLiquidityReceiver = _autoLiquidityReceiver⬆;
    marketingFeeReceiver = _marketingFeeReceiver⬆;
    buyBackFeeReceiver = _buyBackFeeReceiver⬆;
}
```

❖ Owner can enable/disable swap back setting and can change swap thershold

```
ftrace | funcSig
function setSwapBackSettings(
    bool _enabled⬆,
    uint256 _amount⬆
) external onlyOwner {
    require(swapThreshold > 0);
    swapEnabled = _enabled⬆;
    swapThreshold = _amount⬆;
}
```

❖ Owner can change minimum reward time and amount

```
ftrace | funcSig
function setDistributionCriteria(
    uint256 _minPeriod⬆,
    uint256 _minDistribution⬆
) external onlyOwner {
    distributor.setDistributionCriteria(_minPeriod⬆, _minDistribution⬆);
}
```

Audit conclusion

RugFreeCoins team has performed in-depth testings, 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: **1**

Solidity code functional issue level: **PASS**

Number of owner privileges: **12**

Centralization risk correlated to the active owner: **HIGH**

Smart contract active ownership: **ACTIVE**