Cyberscope

Audit Report

Pepega

May 2023

SHA256 fe4d28232102e3e9120bad69e7c59d4a5e99cda26f3d3d7d829b478e601d0b72

Audited by © cyberscope

Table of Contents

Table of Contents 1
Review 2
Audit Updates 2
Source Files 2
Findings Breakdown 3
Analysis 4
ST - Stops Transactions 5
Description 5
Recommendation 5
Diagnostics 6
FAI - Fees Amount Inconsistency 7
Description 7
Recommendation 7
PTRP - Potential Transfer Revert Propagation 8
Description 8
Recommendation 8
PVC - Price Volatility Concern 9
Description 9
Recommendation 9
RSML - Redundant SafeMath Library 10
Description 10
Recommendation 10
L04 - Conformance to Solidity Naming Conventions 11
Description 11
Recommendation 12
L07 - Missing Events Arithmetic 13
Description 13
Recommendation 13
L09 - Dead Code Elimination 14
Description 14
Recommendation 14
L15 - Local Scope Variable Shadowing 16
Description 16
Recommendation 16
L16 - Validate Variable Setters 17
Description 17
Recommendation 17
L19 - Stable Compiler Version 18
Description 18

Recommendation 18
L20 - Succeeded Transfer Check 19
Description 19
Recommendation 19
Functions Analysis **20**
Inheritance Graph **24**
Flow Graph **25**
Summary **26**
Disclaimer **27**
About Cyberscope **28**

Review

Contract Name	Pepega
Testing Deploy	https://testnet.bscscan.com/address/0x60b62fdaa8aa862c2b9826c8d356a1371b0595a0
Symbol	$PEPEGA
Decimals	18
Total Supply	1,000,000,000

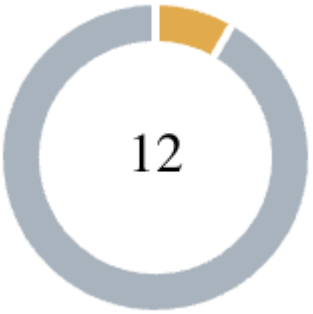
Audit Updates

Initial Audit	14 May 2023

Source Files

Filename	SHA256
contracts/testingDeploy/PEPEGA.sol	fe4d28232102e3e9120bad69e7c59d4a5e99cda26f3d3d7d829b478e601d0b72

Findings Breakdown



Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	1	0	0	0
Minor / Informative	11	0	0	0

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	Passed
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

ST - Stops Transactions

Criticality	Medium
Location	contracts/testingDeploy/PEPEGA.sol#L930
Status	Unresolved

Description

The contract's transactions are disabled by default. Once the contract owner enables them, it will not be possible to be disabled again.

```
if (!tradingActive) {
    require(
        _isExcludedFromFees[from] || _isExcludedFromFees[to],
        "Trading is not active."
    );
}
```

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. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	FAI	Fees Amount Inconsistency	Unresolved
	PTRP	Potential Transfer Revert Propagation	Unresolved
	PVC	Price Volatility Concern	Unresolved
	RSML	Redundant SafeMath Library	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L07	Missing Events Arithmetic	Unresolved
	L09	Dead Code Elimination	Unresolved
	L15	Local Scope Variable Shadowing	Unresolved
	L16	Validate Variable Setters	Unresolved
	L19	Stable Compiler Version	Unresolved
	L20	Succeeded Transfer Check	Unresolved

FAI - Fees Amount Inconsistency

Criticality	Minor / Informative
Location	contracts/testingDeploy/PEPEGA.sol#L1066
Status	Unresolved

Description

The state variable `tokensForMarketing` accumulate the transaction fees. During the `swap()` flow the accumulated fees are swapped for ETH and the `tokensForMarketing` is reset. If the contract owner executes the `withdrawToken()` method the accumulated fees will be transferred to the owner's address but the `tokensForMarketing` will not be updated.

Recommendation

The team is advised to reset the `tokensForMarketing` variable when the `withdrawToken()` is called.

PTRP - Potential Transfer Revert Propagation

Criticality	Minor / Informative
Location	contracts/testingDeploy/PEPEGA.sol#L1065
Status	Unresolved

Description

The contract sends funds to a `marketingWallet` as part of the transfer flow. This address can either be a wallet address or a contract. If the address belongs to 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.

```
payable(marketingWallet).transfer(bnbForMarketing);
```

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 achived by not allowing set contract addresses or by sending the funds in a non-revertable way.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	contracts/testingDeploy/PEPEGA.sol#L817
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `updateSwapTokensAtAmount` sets a threshold where the contract will trigger the swap functionality. If the variable is set to a big number, then the contract will swap a huge amount of tokens for ETH.

It is important to note that the price of the token representing it, can be highly volatile. This means that the value of a price volatility swap involving Ether could fluctuate significantly at the triggered point, potentially leading to significant price volatility for the parties involved.

```
function updateSwapTokensAtAmount(
    uint256 newAmount
) external onlyOwner returns (bool) {
    swapTokensAtAmount = newAmount * (10 ** 18);
    return true;
}
```

Recommendation

The contract could ensure that it will not sell more than a reasonable amount of tokens in a single transaction. A suggested implementation could check that the maximum amount should be less than a fixed percentage of the total supply. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	contracts/testingDeploy/PEPEGA.sol
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert on underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases the gas consumption unnecessarily.

```
library SafeMath {...}
```

Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/testingDeploy/PEPEGA.sol#L576,578,609,651,705,732,746,860,865,1087,1093
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
address indexed sender,
        uint256 amount0,


uint256 amount1,
...
external view returns (uint256);

    function kLast() exte

ountTokenMin,
        uint256 amountETHMin,


...
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.
Find more information on the Solidity documentation
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	contracts/testingDeploy/PEPEGA.sol#L820,833,845,861,866
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
18);
    }

    function updateMaxWallet

ludeFromMaxTransaction(
...

        buyMarketingFee = _mar

romFees(account, excluded);
    }

    "The pair cannot be removed from a
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	contracts/testingDeploy/PEPEGA.sol#L334,798,1036
Status	Unresolved

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
r(account, address(0), amount);

        _afterTokenTransfer(account, address(0), amount);
    }

    function _approve(
...
        emit Approval(owner, spender, amount);
    }

    function _beforeTokenTransfer(
        address from,
        address to,
        uint256 amou

...
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

L15 - Local Scope Variable Shadowing

Criticality	Minor / Informative
Location	contracts/testingDeploy/PEPEGA.sol#L764
Status	Unresolved

Description

Local scope variable shadowing occurs when a local variable with the same name as a variable in an outer scope is declared within a function or code block. When this happens, the local variable "shadows" the outer variable, meaning that it takes precedence over the outer variable within the scope in which it is declared.

```
e);
        excludeFromFees(address(0xdea
```

Recommendation

It's important to be aware of shadowing when working with local variables, as it can lead to confusion and unintended consequences if not used correctly. It's generally a good idea to choose unique names for local variables to avoid shadowing outer variables and causing confusion.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	contracts/testingDeploy/PEPEGA.sol#L901
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
RC20: transfer to the zero address")
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/testingDeploy/PEPEGA.sol#L16
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
/**
     * @dev Initia
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	contracts/testingDeploy/PEPEGA.sol#L1095
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

Functions Analysis

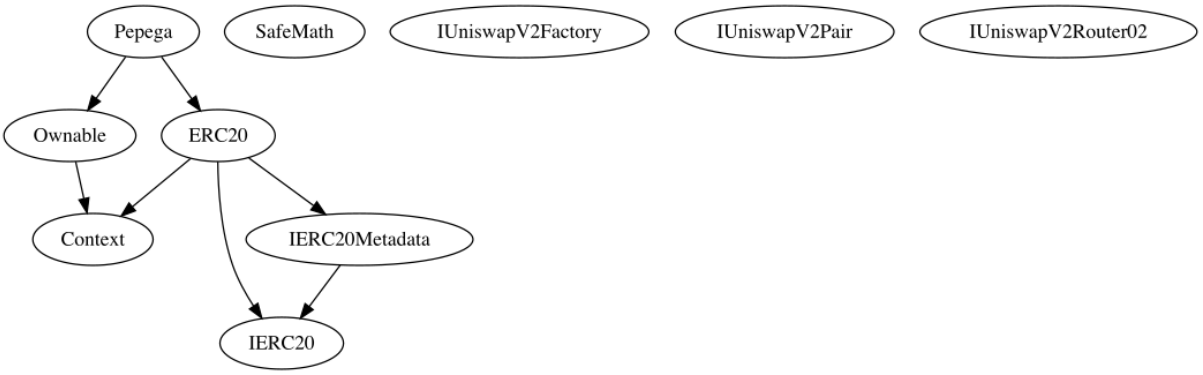
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-

IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
ERC20	Implementation	Context, IERC20, IERC20Metadata		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	

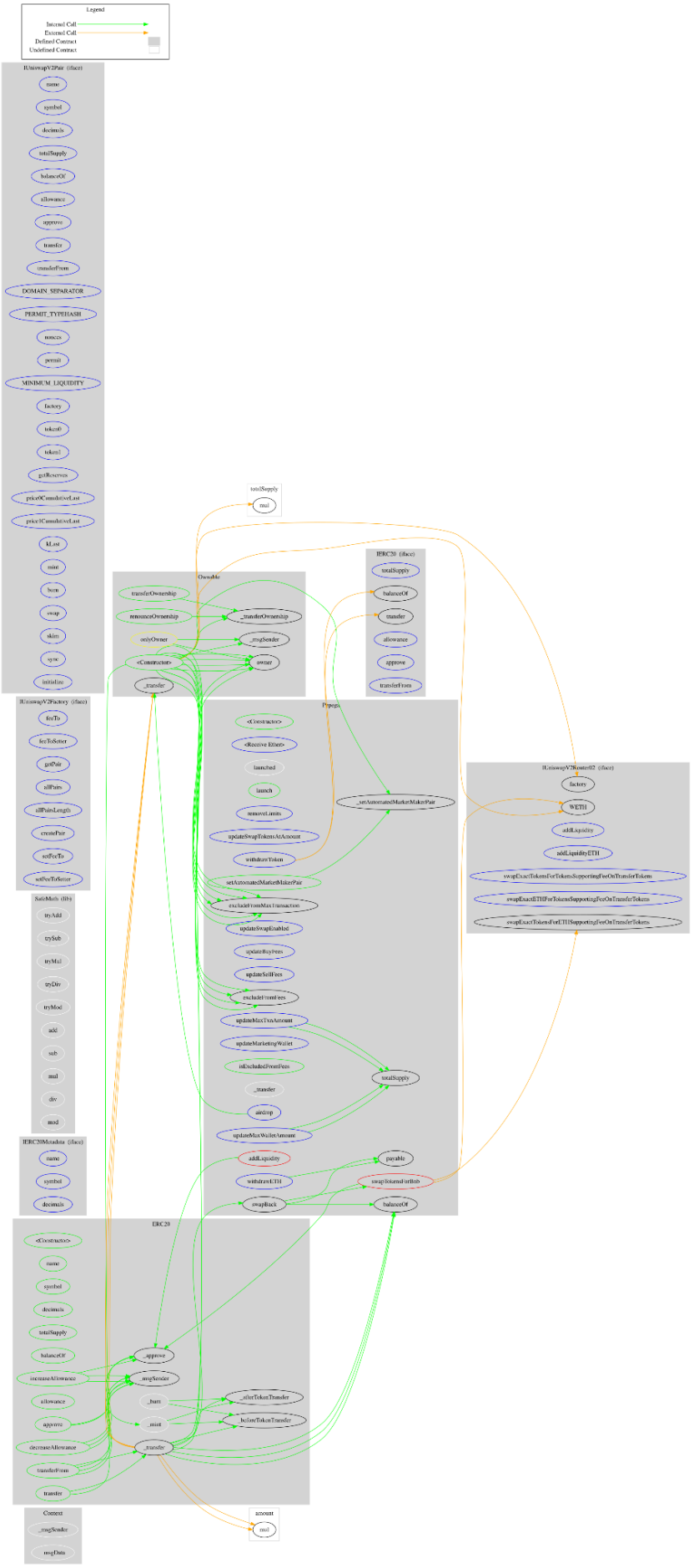
	_approve	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
SafeMath	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
Pepega	Implementation	ERC20, Ownable		
		Public	✓	ERC20
		External	Payable	-
	launched	Internal		

	launch	Public	✓	onlyOwner
	removeLimits	External	✓	onlyOwner
	updateSwapTokensAtAmount	External	✓	onlyOwner
	updateMaxTxnAmount	External	✓	onlyOwner
	updateMaxWalletAmount	External	✓	onlyOwner
	excludeFromMaxTransaction	Public	✓	onlyOwner
	updateSwapEnabled	External	✓	onlyOwner
	updateBuyFees	External	✓	onlyOwner
	updateSellFees	External	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	updateMarketingWallet	External	✓	onlyOwner
	isExcludedFromFees	Public		-
	_transfer	Internal	✓	
	swapTokensForBnb	Private	✓	
	addLiquidity	Private	✓	
	swapBack	Private	✓	
	airdrop	External	✓	onlyOwner
	withdrawETH	External	✓	onlyOwner
	withdrawToken	External	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IUniswapV2Pair (iface)
name
symbol
decimals
totalSupply
balanceOf
allowance
approve
transfer
transferFrom
DOMAIN_SEPARATOR
PERMIT_TYPEHASH
nonces
permit
MINIMUM_LIQUIDITY
factory
token0
token1
getReserves
price0CumulativeLast
price1CumulativeLast
kLast
mint
burn
swap
skim
sync
initialize
IUniswapV2Factory (iface)
feeTo
feeToSetter
getPair
allPairs
allPairsLength
createPair
setFeeTo
setFeeToSetter
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod
IERC20Metadata (iface)
name
symbol
decimals
totalSupply
mul
Ownable
transferOwnership
renounceOwnership
onlyOwner
<Constructor>
_transferOwnership
_msgSender
owner
_transfer
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Pepega
<Constructor>
<Receive Ether>
launched
launch
removeLimits
updateSwapTokensAtAmount
withdrawToken
_setAutomatedMarketMakerPair
setAutomatedMarketMakerPair
excludeFromMaxTransaction
updateSwapEnabled
updateBuyFees
updateSellFees
excludeFromFees
updateMaxTxnAmount
updateMarketingWallet
isExcludedFromFees
_transfer
airdrop
updateMaxWalletAmount
addLiquidity
withdrawETH
swapBack
totalSupply
payable
swapTokensForEth
balanceOf
IUniswapV2Router02 (iface)
factory
WETH
addLiquidity
addLiquidityETH
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
swapExactTokensForETHSupportingFeeOnTransferTokens
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
increaseAllowance
allowance
approve
decreaseAllowance
transferFrom
transfer
_approve
_msgSender
_burn
_mint
_transfer
_afterTokenTransfer
_beforeTokenTransfer
Context
_msgSender
_msgData
amount
mul

Summary

Pepega contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the owner like stop transactions. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats. There is also a limit of max 25% fees.

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