Cyberscope

Audit Report

PepeFace

September 2023

Network	ETH
Address	0xbc0e92fdd39d556529d69e887123b8c77829d42b
Audited by	© cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	IBC	Inconsistent Balance Calculation	Unresolved
●	ULTW	Transfers Liquidity to Team Wallet	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	FSA	Fixed Swap Address	Unresolved
●	RIC	Redundant If Condition	Unresolved
●	PTRP	Potential Transfer Revert Propagation	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	EIS	Excessively Integer Size	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved

●	L09	Dead Code Elimination	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

Table of Contents

Analysis **1**
Diagnostics **2**
Table of Contents **4**
Review **6**
Audit Updates 6
Source Files 6
Findings Breakdown **7**
ST - Stops Transactions 8
Description 8
Recommendation 8
IBC - Inconsistent Balance Calculation 9
Description 9
Recommendation 9
ULTW - Transfers Liquidity to Team Wallet 10
Description 10
Recommendation 10
PVC - Price Volatility Concern 12
Description 12
Recommendation 12
FSA - Fixed Swap Address 13
Description 13
Recommendation 13
RIC - Redundant If Condition 14
Description 14
PTRP - Potential Transfer Revert Propagation 16
Description 16
Recommendation 16
MEE - Missing Events Emission 17
Description 17
Recommendation 17
RSW - Redundant Storage Writes 18
Description 18
Recommendation 18
EIS - Excessively Integer Size 19
Description 19
Recommendation 19
RSML - Redundant SafeMath Library 20
Description 20
Recommendation 20

IDI - Immutable Declaration Improvement 21
Description 21
Recommendation 21
L02 - State Variables could be Declared Constant 22
Description 22
Recommendation 22
L04 - Conformance to Solidity Naming Conventions 23
Description 23
Recommendation 24
L07 - Missing Events Arithmetic 25
Description 25
Recommendation 25
L09 - Dead Code Elimination 26
Description 26
Recommendation 26
L20 - Succeeded Transfer Check 27
Description 27
Recommendation 27
Functions Analysis **28**
Inheritance Graph **33**
Flow Graph **34**
Summary **35**
Disclaimer **36**
About Cyberscope **37**

Review

Contract Name	Token
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	200 runs
Explorer	https://etherscan.io/address/0xbc0e92fdd39d556529d69e887123b8c77829d42b
Address	0xbc0e92fdd39d556529d69e887123b8c77829d42b
Network	ETH
Symbol	PPFACE
Decimals	18
Total Supply	810,000,000,000,000

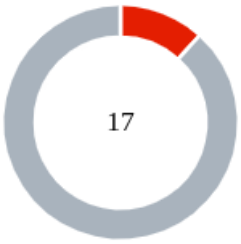
Audit Updates

Initial Audit	15 Sep 2023

Source Files

Filename	SHA256
Token.sol	16f092af4eb8c08a7cdd214791b8d95fed7610185a0fe4f79be2128dc1caac09

Findings Breakdown



Severity	Unresolved	Acknowledged	Resolved	Other
Critical	2	0	0	0
Medium	0	0	0	0
Minor / Informative	15	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	Token.sol#L365
Status	Unresolved

Description

The transactions are initially disabled for all users excluding the authorized addresses. The owner can enable the transactions for all users. Once the transactions are enable the owner will not be able to disable them again.

```
    if (isLimitedAddress(from,to)) {
            require(isTradingEnabled,"Trading is not enabled");
        }
```

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 multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

IBC - Inconsistent Balance Calculation

Criticality	Critical
Location	Token.sol#L447,482
Status	Unresolved

Description

The contract is using the `internalSwap` function to handle token swaps. This function takes `balanceBefore` as an argument, which represents the token balance before the swap. However, the contract contains a critical flaw in its calculation of amountETH. Specifically, the contract calculates the `amountETH` by subtracting the `balanceBefore` variable, which represents the token amount, from `address(this).balance`, which represents the ETH amount. This operation mixes two different types of balances (token and ETH), leading to a serious calculation inconsistency that could have unintended consequences.

```
if(is_sell(from, to) &&  !inSwap && canSwap(from, to)) {
            uint256 contractTokenBalance =
balanceOf(address(this));
            if(swapEnabled && contractTokenBalance >=
swapThreshold) { internalSwap(contractTokenBalance); }
        }
...
 function internalSwap(uint256 balanceBefore) internal
inSwapFlag {
        ...
            uint256 amountETH =
address(this).balance.sub(balanceBefore);
        ...
            uint256 amountETH =
address(this).balance.sub(balanceBefore);
```

Recommendation

It is recommended to reconsider the code implementation to ensure that calculations are consistent and accurate. If the intended implementation is to calculate the amount of ETH received from the swap, then the contract should track the ETH balance before and after the swap and subtract these to find the accurate `amountETH` variable.

ULTW - Transfers Liquidity to Team Wallet

Criticality	Minor / Informative
Location	Token.sol#L435
Status	Unresolved

Description

The contract owner has the authority to transfer funds without limit to the team wallet. These funds have been accumulated from fees collected from the contract. The owner may take advantage of it by calling the `setFeeDistribution` function and set the `sellfee` to zero.

```
function internalSwap(uint256 balanceBefore) internal
inSwapFlag {
        ...
        if (sellfee > 0) {
            ...
        } else if (balanceOf(address(this)) > 0) {

swapRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(
                balanceOf(address(this)),
                0,
                path,
                address(this),
                block.timestamp
            );
            uint256 amountETH =
address(this).balance.sub(balanceBefore);
             (success,) = marketingAddress.call{value:
amountETH, gas: 35000}("");
            require(success, "Failed to send ETH to marketing
fee receiver");
        }
```

Recommendation

The contract could embody a check for the maximum amount of funds that can be swapped, since a huge amount may volatile the token's price. 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.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	Token.sol#L371
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapThreshold` 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.

```
if(is_sell(from, to) &&  !inSwap && canSwap(from, to)) {
    uint256 contractTokenBalance = balanceOf(address(this));
    if(swapEnabled && contractTokenBalance >= swapThreshold) {
internalSwap(contractTokenBalance); }
}
```

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.

FSA - Fixed Swap Address

Criticality	Minor / Informative
Location	Token.sol#L
Status	Unresolved

Description

The swap address is assigned once and it can not be changed. It is a common practice in decentralized exchanges to create new swap versions. A contract that cannot change the swap address may not be able to catch up to the upgrade. As a result, the contract will not be able to migrate to a new liquidity pool pair or decentralized exchange.

```
lpPair =
IFactoryV2(swapRouter.factory()).createPair(swapRouter.WETH(),
address(this));
```

Recommendation

The team is advised to add the ability to change the pair and router address in order to cover potential liquidity pool migrations. It would be better to support multiple pair addresses so the token will be able to have the same behavior in all the decentralized liquidity pairs.

RIC - Redundant If Condition

Criticality	Minor / Informative
Location	Token.sol#L398
Status	Unresolved

Description

The contract is using an if statement to check if `feeAmount` is greater than zero within the `takeTaxes` function. However, this check is redundant. However the `_transfer` function already requires `amount` to be greater than zero. and if the `fee` variable is zero, the function will return the `amount` variable in the previous check. Therefore, the condition `if (feeAmount > 0)` will always evaluate to `true`, making it unnecessary and potentially confusing.

```
function takeTaxes(address from, bool isbuy, bool issell, uint256 amount)
internal returns (uint256) {
    uint256 fee;
    if (isbuy)  fee = buyfee;  else if (issell)  fee = sellfee;  else  fee =
transferfee;
    if (fee == 0)  return amount;
    uint256 feeAmount = amount * fee / fee_denominator;
    if (feeAmount > 0) {

        balance[address(this)] += feeAmount;
        emit Transfer(from, address(this), feeAmount);

    }
    return amount - feeAmount;
}
```

Recommendation

It is recommended to remove the `if (feeAmount > 0)` statement, as it does not affect the actual implementation and adds unnecessary complexity to the code. Simplifying the function by removing this redundant check will make the contract easier to read, understand, and maintain.

PTRP - Potential Transfer Revert Propagation

Criticality	Minor / Informative
Location	Token.sol#L454,459,483
Status	Unresolved

Description

The contract sends funds to `contestReceiver` and `marketingAddress` as part of the transfer flow. These address can either be a wallet address or a contract. If the addresses 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.

```
if (contestFee > 0) {
    success,) = contestReceiver.call{value: amountETHContest,
gas: 35000}("");
    require(success, "Failed to send ETH to contest receiver");
}

if (marketingFee > 0) {
    (success,) = marketingAddress.call{value:
amountETHMarketing, gas: 35000}("");
    require(success, "Failed to send ETH to marketing fee
receiver");
}
...
(success,) = marketingAddress.call{value: amountETH, gas:
35000}("");
require(success, "Failed to send ETH to marketing fee
receiver");
```

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

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	Token.sol#L311
Status	Unresolved

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
function setNoFeeWallet(address account, bool enabled) public
onlyOwner {
    _noFee[account] = enabled;
}
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	Token.sol#L311
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract updates the `_noFee` status of an account even if its current state is the same as the one passed as an argument. As a result, the contract performs redundant storage writes.

```
    function setNoFeeWallet(address account, bool enabled)
public onlyOwner {
        _noFee[account] = enabled;
    }
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

EIS - Excessively Integer Size

Criticality	Minor / Informative
Location	Token.sol#L220,223
Status	Unresolved

Description

The contract is using a bigger unsigned integer data type that the maximum size that is required. By using an unsigned integer data type larger than necessary, the smart contract consumes more storage space and requires additional computational resources for calculations and operations involving these variables. This can result in higher transaction costs, longer execution times, and potential scalability bottlenecks.

```
uint256 constant public fee_denominator = 1_000;
uint256 constant public maxFee = 150;
```

Recommendation

To address the inefficiency associated with using an oversized unsigned integer data type, it is recommended to accurately determine the required size based on the range of values the variable needs to represent.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	Token.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 to 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 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.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	Token.sol#L257,273
Status	Unresolved

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
swapRouter
lpPair
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	Token.sol#L197,221,222
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
address public MKT = 0xebb733EAB3C98975dEdd7341F0C1b5E7752F38D1
uint256 public targetLiquidity = 20
uint256 public targetLiquidityDenominator = 100
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Token.sol#L107,197,232,233,234,243,244,245,246,247,248,249,250,323,328,333,349,385,407,415,493,501,513
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.

```
function WETH() external pure returns (address);
address public MKT = 0xebb733EAB3C98975dEdd7341F0C1b5E7752F38D1
string constant private _name = "$PepeFace"
string constant private _symbol = "PPFACE"
uint8 constant private _decimals = 18
event _enableTrading();
event _setPresaleAddress(address account, bool enabled);
event _toggleCanSwapFees(bool enabled);
event _changePair(address newLpPair);
event _changeWallets(address marketing, address contest);
event _adminTokenRecovery(address tokenAddress, uint256
tokenAmount);
event _feeDistributionUpdated(uint256 liquidityFee,uint256
marketingFee,uint256 contestFee);
event _transferBuyFeeUpdated(uint256 transferFee,uint256
totalBuyFee);

...
```

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	Token.sol#L515
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.

```
swapThreshold = _amount
```

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	Token.sol#L333
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.

```
function is_transfer(address ins, address out) internal view
returns (bool) {
        bool _is_transfer = !isLpPair[out] && !isLpPair[ins];
        return _is_transfer;
    }
```

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.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	Token.sol#L503
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.

```
IERC20(_tokenAddress).transfer(address(msg.sender),
_tokenAmount)
```

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**
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
Context	Implementation			
		Public	✓	-
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner

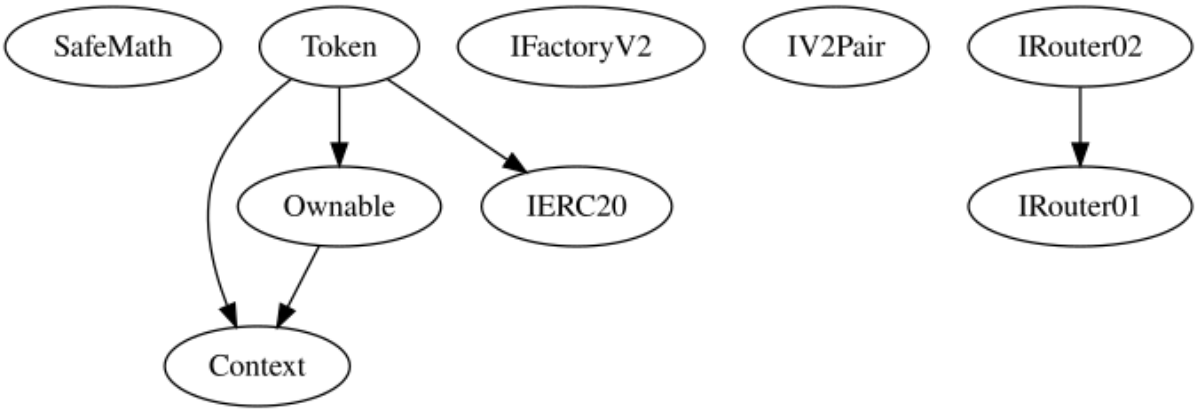
	_setOwner	Private	✓	
IFactoryV2	Interface			
	getPair	External		-
	createPair	External	✓	-
IV2Pair	Interface			
	factory	External		-
	getReserves	External		-
	sync	External	✓	-
IRouter01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
	addLiquidity	External	✓	-
	swapExactETHForTokens	External	Payable	-
	getAmountsOut	External		-
	getAmountsIn	External		-
IRouter02	Interface	IRouter01		
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-

	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokens	External	✓	-
IERC20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Token	Implementation	Context, Ownable, IERC20		
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	allowance	External		-
	balanceOf	Public		-

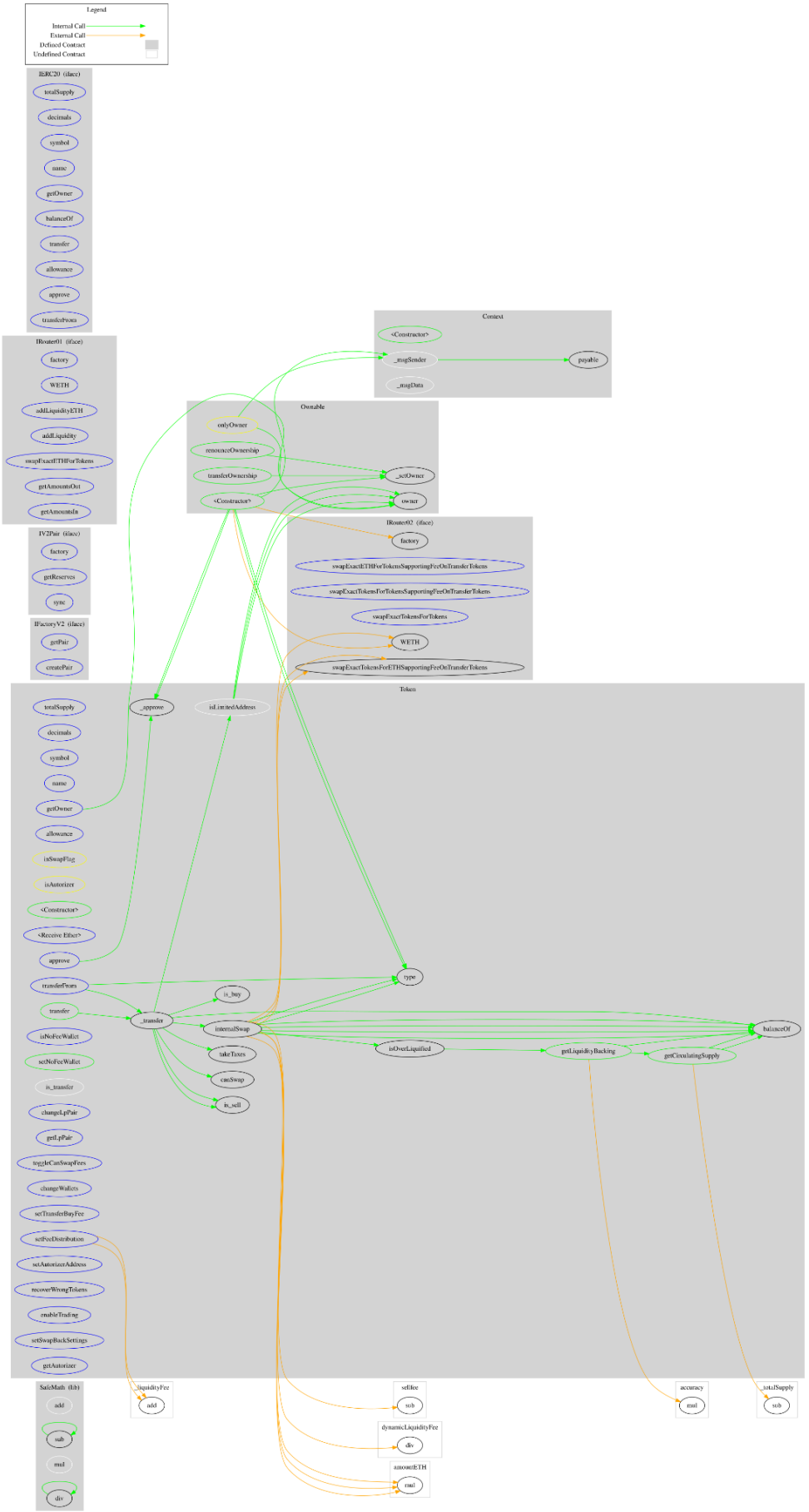
		Public	✓	-
		External	Payable	-
	transfer	Public	✓	-
	approve	External	✓	-
	_approve	Internal	✓	
	transferFrom	External	✓	-
	isNoFeeWallet	External		-
	setNoFeeWallet	Public	✓	onlyOwner
	isLimitedAddress	Internal		
	is_buy	Internal		
	is_sell	Internal		
	is_transfer	Internal		
	canSwap	Internal		
	changeLpPair	External	✓	isAutorizer
	getLpPair	External		-
	toggleCanSwapFees	External	✓	onlyOwner
	_transfer	Internal	✓	
	changeWallets	External	✓	isAutorizer
	takeTaxes	Internal	✓	
	setTransferBuyFee	External	✓	onlyOwner
	setFeeDistribution	External	✓	onlyOwner
	internalSwap	Internal	✓	inSwapFlag
	setAutorizerAddress	External	✓	isAutorizer

	recoverWrongTokens	External	✓	isAutorizer
	enableTrading	External	✓	onlyOwner
	setSwapBackSettings	External	✓	onlyOwner
	getAutorizer	External		-
	getCirculatingSupply	Public		-
	getLiquidityBacking	Public		-
	isOverLiquified	Public		-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IERC20 (iface)
totalSupply
decimals
symbol
name
getOwner
balanceOf
transfer
allowance
approve
transferFrom
IRouter01 (iface)
factory
WETH
addLiquidityETH
addLiquidity
swapExactETHForTokens
getAmountsOut
getAmountsIn
IV2Pair (iface)
factory
getReserves
sync
IFactoryV2 (iface)
getPair
createPair
Context
<Constructor>
_msgSender
payable
_msgData
Ownable
onlyOwner
renounceOwnership
transferOwnership
<Constructor>
_setOwner
owner
IRouter02 (iface)
factory
swapExactETHForTokensSupportingFeeOnTransferTokens
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactTokensForTokens
WETH
swapExactTokensForETHSupportingFeeOnTransferTokens
Token
totalSupply
decimals
symbol
name
getOwner
allowance
inSwapFlag
isAuthorizer
<Constructor>
<Receive Ether>
approve
transferFrom
transfer
isNoFeeWallet
setNoFeeWallet
is_transfer
changeLpPair
getLpPair
toggleCanSwapFees
changeWallets
setTransferBuyFee
setFeeDistribution
setAuthorizerAddress
recoverWrongTokens
enableTrading
setSwapBackSettings
getAuthorizer
_approve
isLimitedAddress
type
is_buy
_transfer
internalSwap
takeTaxes
canSwap
is_sell
isOverLiquified
getLiquidityBacking
getCirculatingSupply
balanceOf
SafeMath (lib)
add
sub
mul
div
_liquidityFee
add
sellfee
sub
dynamicLiquidityFee
div
amountETH
mul
accuracy
mul
_totalSupply
sub

Summary

PepeFace 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 15% 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