0x Guard

Smart contracts security assessment

Final report

Tariff: Standard

Saiyan Pepe

May 2023

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for **Saiyan Pepe**.

Name	Saiyan Pepe
Audit date	2023-05-03 - 2023-05-06
Language	Solidity
Platform	Polygon Network

Contracts checked

Name	Address
SPepe	0xfcA466F2fA8E667a517C9C6cfa99Cf985be5d9B1

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed
Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	not passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed
Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed

Floating Pragma	passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

1. The owner can block token sells for users (SPepe)
Status: Fixed

The contract swaps commission tokens to ETH and sends them to the marketing address when tokens are sold on a DEX.

```
function _transfer(address from, address to, uint256 amount) internal override {
    require(from != address(0), "ERC20: transfer from the zero address");
    require(to != address(0), "ERC20: transfer to the zero address");
    require(balanceOf(from) >= amount, "ERC20: transfer amount exceeds balance");
```

```
        if ((isLiquidityPair[from] || isLiquidityPair[to]) && !inSwapAndLiquify) {
            if (!isLiquidityPair[from]) {
                uint256 contractLiquidityBalance = balanceOf(address(this)) -
_marketingReserves;
                if (contractLiquidityBalance >= numTokensSellToAddToLiquidity) {
                    _swapAndLiquify(numTokensSellToAddToLiquidity);
                }
                if ((_marketingReserves) >= numTokensSellToAddToETH) {
                    _swapTokensForEth(numTokensSellToAddToETH);
                    _marketingReserves -= numTokensSellToAddToETH;
                    bool sent = payable(marketingWallet).send(address(this).balance);
                    require(sent, "Failed to send ETH");
                }
            }
            ...
    }
```

The owner has a possibility to block all sells except his own sell by setting a marketing wallet to a contract which will revert all ETH transfers but not when the owner performs a sell.

```
    function changeMarketingWallet(address newWallet) external onlyOwner
    {
        require(newWallet != DEAD, "LP Pair cannot be the Dead wallet, or 0!");
        require(newWallet != address(0), "LP Pair cannot be the Dead wallet, or 0!");
        marketingWallet = newWallet;
        emit MarketingWalletUpdated(marketingWallet);
    }
```

Also, transfers could be blocked by setting `numTokensSellToAddToLiquidity` variable to zero

```
    function changeSwapThresholds(uint256 _numTokensSellToAddToLiquidity, uint256
_numTokensSellToAddToETH)
        external
        onlyOwner
    {
        require(_numTokensSellToAddToLiquidity < _supply / 98, "Cannot liquidate more
than 2% of the supply at once!");
```

```
        require(_numTokensSellToAddToETH < _supply / 98, "Cannot liquidate more than 2%
of the supply at once!");
        numTokensSellToAddToLiquidity = _numTokensSellToAddToLiquidity * 10**_decimals;
        numTokensSellToAddToETH = _numTokensSellToAddToETH * 10**_decimals;
        emit SwapThresholdsChanged(numTokensSellToAddToLiquidity,
numTokensSellToAddToETH);
    }
```

Recommendation: Given the token is deployed the way to address the issue is either renouncing ownership or writing a wrapper contract which will restrict to set `numTokensSellToAddToLiquidity` and `numTokensSellToAddToETH` to a reasonable minimum value and also restrict changing the marketing wallet address.

Update: The ownership of the contract was transferred to a 24-hour minimum Timelock contract at address 0x00864d467aa182702cd0692ccae04d5d65a6be75 in transaction 0xa8a098c28d28f9e39b8ca37b8537b2a4cbef210d9ba5ac08abacc462f14c8f53. The owner of the contract is a multisig Gnosis-safe wallet. The issue is marked as resolved although users must monitor timelock Timelock transactions.

Medium severity issues

No issues were found

Low severity issues

1. Gas optimizations (SPepe)

Status: Open

- `_name`, `_symbol`, `_decimals`, `_supply` variables can be declared constant

- approve in `_swapTokensForEth()`, `_addLiquidity()` can be made only once in `updatePrimaryPair()`

2. Wrong swap threshold calculations (SPepe)

Status: Open

Wrong calculations in changeSwapThresholds() function. The comment says 2%, but the actual value is 1.02%.

```
    function changeSwapThresholds(uint256 _numTokensSellToAddToLiquidity, uint256 
_numTokensSellToAddToETH)
        external
        onlyOwner
    {
        require(_numTokensSellToAddToLiquidity < _supply / 98, "Cannot liquidate more 
than 2% of the supply at once!");
        require(_numTokensSellToAddToETH < _supply / 98, "Cannot liquidate more than 2% 
of the supply at once!");
        numTokensSellToAddToLiquidity = _numTokensSellToAddToLiquidity * 10**_decimals;
        numTokensSellToAddToETH = _numTokensSellToAddToETH * 10**_decimals;
        emit SwapThresholdsChanged(numTokensSellToAddToLiquidity, 
numTokensSellToAddToETH);
    }
```

3. Excluded from fees addresses may transfer more than max amount (SPepe)

Status: Open

The max transfer amount is not checked for excluded from fees addresses. Also, the max wallet amount is not checked for these addresses.

```
    function _transfer(address from, address to, uint256 amount) internal override {]
        ...
              uint256 transferAmount;
            if (isExcludedFromFee[from] || isExcludedFromFee[to]) {
                transferAmount = amount;
            }
            else {
                require(amount <= maxTxAmount, "ERC20: transfer amount exceeds the max 
transaction amount");
                if (!isLiquidityPair[to]) {
                    require(amount + balanceOf(to) <= maxWalletAmount, "ERC20: balance
amount exceeded max wallet amount limit");
                }

                uint256 marketingShare = ((amount * taxForMarketing) / 100);
                uint256 liquidityShare = ((amount * taxForLiquidity) / 100);
                transferAmount = amount - (marketingShare + liquidityShare);
                _marketingReserves += marketingShare;

                super._transfer(from, address(this), (marketingShare +
liquidityShare));
            }
        ...
  }
```

4. Unnecessary update pair function (SPepe)

Status: Open

The contract has `updatePrimaryPair()` function which changes the pair address, but calling it does not have an effect on the token. To change the pair the router address must be changed.

Conclusion

Saiyan Pepe SPepe contract was audited. 1 high, 4 low severity issues were found.
1 high severity issue has been fixed in the update.

Update. The ownership of the contract was transferred to a 24-hour minimum Timelock contract. The owner of the contract is a multisig Gnosis-safe wallet. The issue is marked as resolved although users must monitor timelock Timelock transactions to ensure no malicious transaction is sent to it.

Disclaimer

Static code analysis

```
INFO:Detectors:
Reentrancy in SPepe._transfer(address,address,uint256) (contracts/SPepe.sol#1010-1051):
        External calls:
        - _swapAndLiquify(numTokensSellToAddToLiquidity) (contracts/SPepe.sol#1019)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (contracts/
SPepe.sol#1087-1094)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tok
enAmount,0,path,address(this),block.timestamp) (contracts/SPepe.sol#1075-1081)
        - _swapTokensForEth(numTokensSellToAddToETH) (contracts/SPepe.sol#1022)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tok
enAmount,0,path,address(this),block.timestamp) (contracts/SPepe.sol#1075-1081)
        External calls sending eth:
        - _swapAndLiquify(numTokensSellToAddToLiquidity) (contracts/SPepe.sol#1019)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (contracts/
SPepe.sol#1087-1094)
        State variables written after the call(s):
        - _marketingReserves -= numTokensSellToAddToETH (contracts/SPepe.sol#1023)
        SPepe._marketingReserves (contracts/SPepe.sol#955) can be used in cross
function reentrancies:
        - SPepe._marketingReserves (contracts/SPepe.sol#955)
        - SPepe._transfer(address,address,uint256) (contracts/SPepe.sol#1010-1051)
        - _swapTokensForEth(numTokensSellToAddToETH) (contracts/SPepe.sol#1022)
                - inSwapAndLiquify = true (contracts/SPepe.sol#991)
                - inSwapAndLiquify = false (contracts/SPepe.sol#993)
        SPepe.inSwapAndLiquify (contracts/SPepe.sol#982) can be used in cross function
reentrancies:
        - SPepe._transfer(address,address,uint256) (contracts/SPepe.sol#1010-1051)
        - SPepe.lockTheSwap() (contracts/SPepe.sol#990-994)
Reentrancy in SPepe._transfer(address,address,uint256) (contracts/SPepe.sol#1010-1051):
        External calls:
        - _swapAndLiquify(numTokensSellToAddToLiquidity) (contracts/SPepe.sol#1019)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (contracts/
SPepe.sol#1087-1094)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tok
enAmount,0,path,address(this),block.timestamp) (contracts/SPepe.sol#1075-1081)
        - _swapTokensForEth(numTokensSellToAddToETH) (contracts/SPepe.sol#1022)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tok
enAmount,0,path,address(this),block.timestamp) (contracts/SPepe.sol#1075-1081)
        External calls sending eth:
        - _swapAndLiquify(numTokensSellToAddToLiquidity) (contracts/SPepe.sol#1019)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (contracts/
SPepe.sol#1087-1094)
        - sent = address(marketingWallet).send(address(this).balance) (contracts/
SPepe.sol#1024)
        State variables written after the call(s):
        - super._transfer(from,address(this),(marketingShare + liquidityShare))
(contracts/SPepe.sol#1044)
                - _balances[from] = fromBalance - amount (contracts/SPepe.sol#934)
                - _balances[to] += amount (contracts/SPepe.sol#937)
        ERC20._balances (contracts/SPepe.sol#626) can be used in cross function
reentrancies:
        - ERC20._mint(address,uint256) (contracts/SPepe.sol#834-843)
        - ERC20._transfer(address,address,uint256) (contracts/SPepe.sol#920-941)
        - ERC20.balanceOf(address) (contracts/SPepe.sol#657-665)
        - super._transfer(from,to,transferAmount) (contracts/SPepe.sol#1046)
                - _balances[from] = fromBalance - amount (contracts/SPepe.sol#934)
                - _balances[to] += amount (contracts/SPepe.sol#937)
        ERC20._balances (contracts/SPepe.sol#626) can be used in cross function
reentrancies:
        - ERC20._mint(address,uint256) (contracts/SPepe.sol#834-843)
        - ERC20._transfer(address,address,uint256) (contracts/SPepe.sol#920-941)
        - ERC20.balanceOf(address) (contracts/SPepe.sol#657-665)
        - _marketingReserves += marketingShare (contracts/SPepe.sol#1042)
        SPepe._marketingReserves (contracts/SPepe.sol#955) can be used in cross
function reentrancies:
        - SPepe._marketingReserves (contracts/SPepe.sol#955)
        - SPepe._transfer(address,address,uint256) (contracts/SPepe.sol#1010-1051)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities
INFO:Detectors:
SPepe._name (contracts/SPepe.sol#945) shadows:
        - ERC20._name (contracts/SPepe.sol#631)
SPepe._symbol (contracts/SPepe.sol#946) shadows:
        - ERC20._symbol (contracts/SPepe.sol#632)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-
shadowing
INFO:Detectors:
SPepe._addLiquidity(uint256,uint256) (contracts/SPepe.sol#1084-1095) ignores return
value by uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (contracts/
SPepe.sol#1087-1094)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in SPepe._swapAndLiquify(uint256) (contracts/SPepe.sol#1053-1066):
        External calls:
        - _swapTokensForEth(half) (contracts/SPepe.sol#1059)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tok
enAmount,0,path,address(this),block.timestamp) (contracts/SPepe.sol#1075-1081)
        - _addLiquidity(otherHalf,newBalance) (contracts/SPepe.sol#1063)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (contracts/
SPepe.sol#1087-1094)
        External calls sending eth:
        - _addLiquidity(otherHalf,newBalance) (contracts/SPepe.sol#1063)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (contracts/
SPepe.sol#1087-1094)
        State variables written after the call(s):
        - _addLiquidity(otherHalf,newBalance) (contracts/SPepe.sol#1063)
                - _allowances[owner][spender] = amount (contracts/SPepe.sol#891)
        - _addLiquidity(otherHalf,newBalance) (contracts/SPepe.sol#1063)
                - inSwapAndLiquify = true (contracts/SPepe.sol#991)
                - inSwapAndLiquify = false (contracts/SPepe.sol#993)
Reentrancy in SPepe._transfer(address,address,uint256) (contracts/SPepe.sol#1010-1051):
        External calls:
        - _swapAndLiquify(numTokensSellToAddToLiquidity) (contracts/SPepe.sol#1019)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (contracts/
SPepe.sol#1087-1094)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tok
enAmount,0,path,address(this),block.timestamp) (contracts/SPepe.sol#1075-1081)
        - _swapTokensForEth(numTokensSellToAddToETH) (contracts/SPepe.sol#1022)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tok
enAmount,0,path,address(this),block.timestamp) (contracts/SPepe.sol#1075-1081)
        External calls sending eth:
```

```
        - _swapAndLiquify(numTokensSellToAddToLiquidity) (contracts/SPepe.sol#1019)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (contracts/
SPepe.sol#1087-1094)
        State variables written after the call(s):
        - _swapTokensForEth(numTokensSellToAddToETH) (contracts/SPepe.sol#1022)
                - _allowances[owner][spender] = amount (contracts/SPepe.sol#891)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2
INFO:Detectors:
Reentrancy in SPepe._swapAndLiquify(uint256) (contracts/SPepe.sol#1053-1066):
        External calls:
        - _swapTokensForEth(half) (contracts/SPepe.sol#1059)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tok
enAmount,0,path,address(this),block.timestamp) (contracts/SPepe.sol#1075-1081)
        - _addLiquidity(otherHalf,newBalance) (contracts/SPepe.sol#1063)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (contracts/
SPepe.sol#1087-1094)
        External calls sending eth:
        - _addLiquidity(otherHalf,newBalance) (contracts/SPepe.sol#1063)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (contracts/
SPepe.sol#1087-1094)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/SPepe.sol#892)
                - _addLiquidity(otherHalf,newBalance) (contracts/SPepe.sol#1063)
        - SwapAndLiquify(half,newBalance,otherHalf) (contracts/SPepe.sol#1065)
Reentrancy in SPepe._transfer(address,address,uint256) (contracts/SPepe.sol#1010-1051):
        External calls:
        - _swapAndLiquify(numTokensSellToAddToLiquidity) (contracts/SPepe.sol#1019)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (contracts/
SPepe.sol#1087-1094)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tok
enAmount,0,path,address(this),block.timestamp) (contracts/SPepe.sol#1075-1081)
        - _swapTokensForEth(numTokensSellToAddToETH) (contracts/SPepe.sol#1022)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tok
enAmount,0,path,address(this),block.timestamp) (contracts/SPepe.sol#1075-1081)
        External calls sending eth:
        - _swapAndLiquify(numTokensSellToAddToLiquidity) (contracts/SPepe.sol#1019)
```

```
                - uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (contracts/
SPepe.sol#1087-1094)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/SPepe.sol#892)
                - _swapTokensForEth(numTokensSellToAddToETH) (contracts/SPepe.sol#1022)
Reentrancy in SPepe._transfer(address,address,uint256) (contracts/SPepe.sol#1010-1051):
        External calls:
        - _swapAndLiquify(numTokensSellToAddToLiquidity) (contracts/SPepe.sol#1019)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (contracts/
SPepe.sol#1087-1094)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tok
enAmount,0,path,address(this),block.timestamp) (contracts/SPepe.sol#1075-1081)
        - _swapTokensForEth(numTokensSellToAddToETH) (contracts/SPepe.sol#1022)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tok
enAmount,0,path,address(this),block.timestamp) (contracts/SPepe.sol#1075-1081)
        External calls sending eth:
        - _swapAndLiquify(numTokensSellToAddToLiquidity) (contracts/SPepe.sol#1019)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (contracts/
SPepe.sol#1087-1094)
        - sent = address(marketingWallet).send(address(this).balance) (contracts/
SPepe.sol#1024)
        Event emitted after the call(s):
        - Transfer(from,to,amount) (contracts/SPepe.sol#940)
                - super._transfer(from,address(this),(marketingShare + liquidityShare))
(contracts/SPepe.sol#1044)
        - Transfer(from,to,amount) (contracts/SPepe.sol#940)
                - super._transfer(from,to,transferAmount) (contracts/SPepe.sol#1046)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3
INFO:Detectors:
ERC20._burn(address,uint256) (contracts/SPepe.sol#856-868) is never used and should be
removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
SPepe.maxTxAmount (contracts/SPepe.sol#951) is set pre-construction with a non-constant
function or state variable:
        - 6969111691 * 10 ** _decimals
SPepe.maxWalletAmount (contracts/SPepe.sol#952) is set pre-construction with a non-
```

```
constant function or state variable:
        - 6969111691 * 10 ** _decimals
SPepe.numTokensSellToAddToLiquidity (contracts/SPepe.sol#956) is set pre-construction
with a non-constant function or state variable:
        - 139382233 * 10 ** _decimals
SPepe.numTokensSellToAddToETH (contracts/SPepe.sol#957) is set pre-construction with a
non-constant function or state variable:
        - 69691116 * 10 ** _decimals
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-
initializing-state
INFO:Detectors:
Pragma version0.8.16 (contracts/SPepe.sol#47) allows old versions
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (contracts/SPepe.sol#112) is not in
mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (contracts/SPepe.sol#114) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (contracts/SPepe.sol#145) is not in
mixedCase
Function IUniswapV2Router01.WETH() (contracts/SPepe.sol#191) is not in mixedCase
Parameter SPepe.updatePrimaryPair(address)._pair (contracts/SPepe.sol#1097) is not in
mixedCase
Parameter SPepe.changeTaxForLiquidityAndMarketing(uint256,uint256)._taxForLiquidity
(contracts/SPepe.sol#1113) is not in mixedCase
Parameter SPepe.changeTaxForLiquidityAndMarketing(uint256,uint256)._taxForMarketing
(contracts/SPepe.sol#1113) is not in mixedCase
Parameter SPepe.changeSwapThresholds(uint256,uint256)._numTokensSellToAddToLiquidity
(contracts/SPepe.sol#1123) is not in mixedCase
Parameter SPepe.changeSwapThresholds(uint256,uint256)._numTokensSellToAddToETH
(contracts/SPepe.sol#1123) is not in mixedCase
Parameter SPepe.updatePairStatus(address,bool)._pair (contracts/SPepe.sol#1134) is not
in mixedCase
Parameter SPepe.updatePairStatus(address,bool)._status (contracts/SPepe.sol#1134) is
not in mixedCase
Parameter SPepe.excludeFromFee(address,bool)._address (contracts/SPepe.sol#1139) is not
in mixedCase
Parameter SPepe.excludeFromFee(address,bool)._status (contracts/SPepe.sol#1139) is not
in mixedCase
Parameter SPepe.changeMaxTxAmount(uint256)._maxTxAmount (contracts/SPepe.sol#1144) is
```

```
not in mixedCase
Parameter SPepe.changeMaxWalletAmount(uint256)._maxWalletAmount (contracts/
SPepe.sol#1150) is not in mixedCase
Function SPepe.KAMEHAMEHA() (contracts/SPepe.sol#1156-1158) is not in mixedCase
Variable SPepe.DEAD (contracts/SPepe.sol#954) is not in mixedCase
Variable SPepe._marketingReserves (contracts/SPepe.sol#955) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions
INFO:Detectors:
Reentrancy in SPepe._transfer(address,address,uint256) (contracts/SPepe.sol#1010-1051):
        External calls:
        - sent = address(marketingWallet).send(address(this).balance) (contracts/
SPepe.sol#1024)
        External calls sending eth:
        - _swapAndLiquify(numTokensSellToAddToLiquidity) (contracts/SPepe.sol#1019)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (contracts/
SPepe.sol#1087-1094)
        - sent = address(marketingWallet).send(address(this).balance) (contracts/
SPepe.sol#1024)
        State variables written after the call(s):
        - super._transfer(from,address(this),(marketingShare + liquidityShare))
(contracts/SPepe.sol#1044)
                - _balances[from] = fromBalance - amount (contracts/SPepe.sol#934)
                - _balances[to] += amount (contracts/SPepe.sol#937)
        - super._transfer(from,to,transferAmount) (contracts/SPepe.sol#1046)
                - _balances[from] = fromBalance - amount (contracts/SPepe.sol#934)
                - _balances[to] += amount (contracts/SPepe.sol#937)
        - _marketingReserves += marketingShare (contracts/SPepe.sol#1042)
        Event emitted after the call(s):
        - Transfer(from,to,amount) (contracts/SPepe.sol#940)
                - super._transfer(from,address(this),(marketingShare + liquidityShare))
(contracts/SPepe.sol#1044)
        - Transfer(from,to,amount) (contracts/SPepe.sol#940)
                - super._transfer(from,to,transferAmount) (contracts/SPepe.sol#1046)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-4
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256
,address,uint256).amountADesired (contracts/SPepe.sol#196) is too similar to IUniswapV2R
outer01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).am
```

```
ountBDesired (contracts/SPepe.sol#197)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
too-similar
INFO:Detectors:
SPepe.DEAD (contracts/SPepe.sol#954) should be constant
SPepe._decimals (contracts/SPepe.sol#947) should be constant
SPepe._name (contracts/SPepe.sol#945) should be constant
SPepe._supply (contracts/SPepe.sol#948) should be constant
SPepe._symbol (contracts/SPepe.sol#946) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-constant
INFO:Slither:. analyzed (10 contracts with 85 detectors), 42 result(s) found
```

Guard