Guard

Smart contracts security assessment

Final report

Tariff: Standard

WhalesCandy

February 2023

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for **WhalesCandy**.

A ERC20 token with an auction functionality where users can buy shares and receive rewards in this

token. The token has a buy tax of 80% on the pancakeswap dex. To buy this tokens users are supposed to buy them

from the auction or users can purchase them through the buy and stake function of the contract (which will negate the 80% dex buy tax, however, upon purchase the tokens will be immediately staked). The token has also a referral system. Dev account gets 30% of the BNB

sent to the contract in auctions, other 70% are used for adding liquidity to a DEX pair. The dev address also gets 5% of the minted tokens.

A recheck was done on code provided in WhalesCandy/Token Github repo after the commit [dae3ad9](dae3ad9).

Name	WhalesCandy
Audit date	2023-02-08 - 2023-02-13
Language	Solidity
Platform	Binance Smart Chain

Contracts checked

Name	Address
WhalesCandy	

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	not 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	passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	not passed
Reentrancy	passed
Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	not 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Outdated Compiler Version	passed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. Referral system rewards can be abused (WhalesCandy)

Status: Fixed

The function `claimTokenFromSharesAndStake()` sets the flag of the `mapRefData[myRef[msg.sender]][_day + 1].hasClaimed` to false even if has already been claimed. Which leads to a possibility to abuse the referral rewards.

```
    function claimTokenFromSharesAndStake (uint256 _day, uint256 _referer) external
returns (bool) {
      require(_day < currentDay, "Day must be over to claim!");
      require(mapMemberAuction[msg.sender][_day].hasChangedShareToToken == false, "User
has already Changed his shares to Token that Day!");
      uint256 userShares = mapMemberAuction[msg.sender][_day].memberAuctionValue;

      uint256 amountUserTokens =
calculateTokenPerShareOnDay(_day).mul(userShares);

        if(refCodeToAddress[_referer] != address(0) || myRef[msg.sender] != address(0))
{
            if(myRef[msg.sender] == address(0)){
            myRef[msg.sender] = refCodeToAddress[_referer];
            }
            // eared ref token are accounted for the next day so be sure ref can claim
all past days token at once
            mapRefData[myRef[msg.sender]][_day + 1].refEarnedTokens +=
amountUserTokens.mul(5).div(100);
            mapRefData[myRef[msg.sender]][_day + 1].hasClaimed = false;
        }
        ...
  }
```

Recommendation: Modify logic of the `claimTokenFromSharesAndStake()` function to exclude the possibility to claim twice.

Medium severity issues

1. Unclear add liquidity and buyback mechanics (WhalesCandy)

Status: Fixed

The funciton updateDaily() takes eth balance of the token, mints tokens and adds them as liquidity to a pair. Then the function burnAndBuyback() is called which removes liquidity and uses ETH to buy tokens from the pair.

Recommendation: Write explicit documentation of what and why the function is supposed to do.

Team response: Basically updateDaily() is there to add liquidity with the previous days collected ETH and newly minted WC token. Also it is checking if there is leftover ETH and adds that with newly minted WC token. The burnAndBuyback() also needs to be called daily so it is called within the updateDaily() function. Its purpose is to take a share (1-4%) of the accumulated LPs and remove them, sell the ETH for WC and burn them to achieve a daily price increase.

2. Calculation without multiplier (WhalesCandy)

Status: Fixed

The fuction calculateTokenPerShareOnDay() does not use multiplier to calculate shares which may lead to rounding errors.

```
    function calculateTokenPerShareOnDay (uint256 _day) public view returns (uint256) {
      uint256 collectedThatDay = auctionEntry[_day];
      uint256 tokenPerShare = dayliAvailableToken/collectedThatDay; //@audit no
multiplier, may be different decimals
      return tokenPerShare;
    }
```

Recommendation: Use multiplier to calculate the shares.

3. Mixed usage of user function parameter and msg.sender (WhalesCandy)

Status: Fixed

The functions `calcReward()` and `calcClaim()` use `_user` and `msg.sender` params simultaneusly in calculations. This may lead to bugs if the passed `_user` param is different from the `msg.sender`.

```
    function calcReward (address _user, uint256 _stakeIndex) public view returns
(uint256) {
      if(stakes[_user][_stakeIndex].stakeTime == 0){
        return 0;
      }
      // value 11574074074074 gives 1 ether per day as multiplier!
      uint256 multiplier = (block.timestamp - stakes[_user]
[_stakeIndex].lastUpdate).mul(weiPerSfor1perDay);
      // for example: if user amount is 100 and user has staked for 365 days and not
collected so far,
      // reward would be 365, if 365 was already collected reward will be 0
      if(stakes[_user][_stakeIndex].amount.mul(multiplier).div(100
ether).add(stakes[msg.sender][_stakeIndex].collected) >
         stakes[_user][_stakeIndex].amount.mul(365).div(100)) {
        return(stakes[_user]
[_stakeIndex].amount.mul(365).div(100).sub(stakes[msg.sender]
[_stakeIndex].collected));
      }
      // in same example: below 365 days of stakeInt the reward is stakes.amount *
days/100
      return stakes[_user][_stakeIndex].amount.mul(multiplier).div(100 ether);
    }
```

Moreover, the this function is a `view` function and will give wrong results if called offchain becase a zero address would be passed for the `msg.sender` value.

Recommendation: Change `msg.sender` to `_user`.

Low severity issues

1. Lack of events (WhalesCandy)

Status: Partially fixed

The functions that change contract parameters don't emit events. Emitting events makes it easier to track changes of the parameters offchain.

Recommendation: Add events for setters in the contract.

2. Rounding errors (WhalesCandy)

Status: Fixed

The function `_transfer()` calculates tax and amount to send by multiplication and division which may give rounding errors in certain cases.

```
    function _transfer(address from, address to, uint256 amount) internal virtual {
        // For Taxed Transfer (if pair is sender (token BUY) tax of 80% applies)
        bool _isTaxedRecipient = !isExcludedFromTaxReceiver(to);
        if ( from == tradingPair && _isTaxedRecipient ) {   // if sender is pair (its a
buy tx) AND is a TaxedRecipient
        _Balances[from] = _Balances[from].sub(amount, "transfer amount exceeds
balance");
        _Balances[to] = _Balances[to].add(amount.mul(20).div(100));
        _Balances[address(0)] = _Balances[address(0)].add(amount.mul(80).div(100));
        emit Transfer(from, to, amount.mul(20).div(100));
        emit Transfer(from, address(0), amount.mul(80).div(100));
        ...
  }
```

Recommendation: We recommend to calculate tax part by multiplication and division and amount to send by subtraction.

3. Unclear calculations (WhalesCandy)

Status: Fixed

There is no information in documentation of how the claim fees should be calculated. We suppose that the current calculations might be wrong: the edge case when `totalClaimFee` is equal

to maxBUSDFee gives differend result in the following if branch:

```
    function calcClaimFee (address _user, uint256 _stakeIndex) public view returns
(uint256) {
      ...
      if(totalClaimFee > maxBUSDFee) {
        return maxBUSDFee.sub(stakes[_user]
[_stakeIndex].claimingStartFee).sub(stakes[_user][_stakeIndex].feePaid);
      }
      return stakes[_user]
[_stakeIndex].claimingStartFee.add(additionalClaimFee).sub(stakes[_user]
[_stakeIndex].feePaid);
    }
```

Recommendation: Double check the calculations

4. Gas inefficiency (WhalesCandy)

Status: Open

There are multiple reads of the same value in the functions. Each read consumes gas.

Public functions that are not called from the contract can be declared external.

Recommendation: Use memoisation to minimise gas consumption.

5. Usage of default visibility (WhalesCandy)

Status: Fixed

Some variables have no visibility specified. We recommend explicitly setting the visibility to avoid bugs.

```
    uint256 BUSDfeePerWeek = 2 ether;
    uint256 maxBUSDFee = 52*2 ether;
    uint256 weiPerSfor1perDay = 11574074074074;  // this token/wei amount need to be
accounted per second to have 1 ETH per day

    uint256 dayliAvailableToken = 2000000 ether;
```

Conclusion

WhalesCandy WhalesCandy contract was audited. 1 high, 3 medium, 5 low severity issues were found.

1 high, 3 medium, 3 low severity issues have been fixed in the update.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability)set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard 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 considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report 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 in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

Static code analysis

```
Reentrancy in WhalesCandy.buyShareFromAuction() (contracts/wc.sol#538-568):
        External calls:
        - updateDaily() (contracts/wc.sol#549)
                - IERC20(tradingPair).approve(_pancakeRouterAddress,type()
(uint256).max) (contracts/wc.sol#498)
                - _pancakeRouter.removeLiquidityETHSupportingFeeOnTransferTokens(contrAd
dr,lpBalToRemove,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#506-513)
                - _pancakeRouter.swapExactETHForTokens{value: ethGain}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#523-528)
                - IERC20(contrAddr).approve(_pancakeRouterAddress,type()(uint256).max)
(contracts/wc.sol#434)
                - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value:
ETHtoAdd}(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/
wc.sol#437-445)
                - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        External calls sending eth:
        - _dev.transfer(rawAmount.mul(30).div(100)) (contracts/wc.sol#543)
        - updateDaily() (contracts/wc.sol#549)
                - _pancakeRouter.swapExactETHForTokens{value: ethGain}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#523-528)
                - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value:
ETHtoAdd}(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/
wc.sol#437-445)
                - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        State variables written after the call(s):
        - auctionEntry[currentDay] += rawAmount (contracts/wc.sol#551)
        - lastBUYINday = currentDay (contracts/wc.sol#553)
        - liqAdded[currentDay] = false (contracts/wc.sol#554)
Reentrancy in WhalesCandy.updateDaily() (contracts/wc.sol#414-489):
        External calls:
        - IERC20(contrAddr).approve(_pancakeRouterAddress,type()(uint256).max)
(contracts/wc.sol#434)
        - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value: ETHtoAdd}
(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#437-445)
        External calls sending eth:
```

```
        - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value: ETHtoAdd}
(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#437-445)
        State variables written after the call(s):
        - _mint(contrAddr,neededToken) (contracts/wc.sol#455)
                - _Balances[_user] = _Balances[_user].add(_amount) (contracts/
wc.sol#306)
        - _mint(contrAddr,neededToken) (contracts/wc.sol#455)
                - _totalSupply = _totalSupply.add(_amount) (contracts/wc.sol#307)
        - liqAdded[lastBUYINday] = true (contracts/wc.sol#447)
Reentrancy in WhalesCandy.updateDaily() (contracts/wc.sol#414-489):
        External calls:
        - IERC20(contrAddr).approve(_pancakeRouterAddress,type()(uint256).max)
(contracts/wc.sol#434)
        - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value: ETHtoAdd}
(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#437-445)
        - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        External calls sending eth:
        - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value: ETHtoAdd}
(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#437-445)
        - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        State variables written after the call(s):
        - transferToZero(leftOverToken.sub(1000000000000000000)) (contracts/wc.sol#469)
                - _Balances[contrAddr] = _Balances[contrAddr].sub(amount,Token:
transfer amount exceeds balance) (contracts/wc.sol#269)
                - _Balances[address(0)] = _Balances[address(0)].add(amount) (contracts/
wc.sol#270)
        - currentDay = thisDay() (contracts/wc.sol#477)
Reentrancy in WhalesCandy.updateDaily() (contracts/wc.sol#414-489):
        External calls:
        - IERC20(contrAddr).approve(_pancakeRouterAddress,type()(uint256).max)
(contracts/wc.sol#434)
        - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value: ETHtoAdd}
(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#437-445)
        - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        - burnAndBuyback() (contracts/wc.sol#486)
                - IERC20(tradingPair).approve(_pancakeRouterAddress,type()
(uint256).max) (contracts/wc.sol#498)
                - _pancakeRouter.removeLiquidityETHSupportingFeeOnTransferTokens(contrAd
```

```
dr,lpBalToRemove,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#506-513)
                - _pancakeRouter.swapExactETHForTokens{value: ethGain}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#523-528)
        External calls sending eth:
        - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value: ETHtoAdd}
(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#437-445)
        - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        - burnAndBuyback() (contracts/wc.sol#486)
                - _pancakeRouter.swapExactETHForTokens{value: ethGain}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#523-528)
        State variables written after the call(s):
        - burnAndBuyback() (contracts/wc.sol#486)
                - _Balances[contrAddr] = _Balances[contrAddr].sub(amount,Token:
transfer amount exceeds balance) (contracts/wc.sol#269)
                - _Balances[address(0)] = _Balances[address(0)].add(amount) (contracts/
wc.sol#270)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities

PancakeRouter.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
(contracts/pancakeSwap/PancakeRouter.sol#136-152) ignores return value by
IPancakePair(pair).transferFrom(msg.sender,pair,liquidity) (contracts/pancakeSwap/
PancakeRouter.sol#146)
PancakeRouter01.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
 (contracts/pancakeSwap/PancakeRouter01.sol#130-146) ignores return value by
IPancakePair(pair).transferFrom(msg.sender,pair,liquidity) (contracts/pancakeSwap/
PancakeRouter01.sol#140)
WhalesCandy.claimRewards(uint256) (contracts/wc.sol#682-697) ignores return value by
IERC20(BUSDaddress).transferFrom(msg.sender,_dev1,feeToPay) (contracts/wc.sol#688)
WhalesCandy.reinvest(uint256) (contracts/wc.sol#700-732) ignores return value by
IERC20(BUSDaddress).transferFrom(msg.sender,_dev1,feeToPay) (contracts/wc.sol#706)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer

WhalesCandy.updateDaily() (contracts/wc.sol#414-489) performs a multiplication on the
result of a division:
        -currentLiqRatio = addedToken.mul(10000).div(addedEth) (contracts/wc.sol#452)
        -neededToken = currentLiqRatio.mul(ethBal).div(10000) (contracts/wc.sol#453)
WhalesCandy.calcClaimFee(address,uint256) (contracts/wc.sol#660-672) performs a
multiplication on the result of a division:
```

```
        -weeksOfStake = (block.timestamp - stakes[_user][_stakeIndex].stakeTime) /
oneWeek (contracts/wc.sol#664)
        -additionalClaimFee = weeksOfStake * BUSDfeePerWeek (contracts/wc.sol#665)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

PancakePair._safeTransfer(address,address,uint256) (contracts/pancakeSwap/
PancakePair.sol#54-64) uses a dangerous strict equality:
        - require(bool,string)(success && (data.length == 0 || abi.decode(data,
(bool))),Pancake: TRANSFER_FAILED) (contracts/pancakeSwap/PancakePair.sol#60-63)
PancakePair.mint(address) (contracts/pancakeSwap/PancakePair.sol#139-163) uses a
dangerous strict equality:
        - _totalSupply == 0 (contracts/pancakeSwap/PancakePair.sol#148)
WhalesCandy.buyShareFromAuction() (contracts/wc.sol#538-568) uses a dangerous strict
equality:
        - currentDay == 0 (contracts/wc.sol#545)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities

Reentrancy in PancakePair.burn(address) (contracts/pancakeSwap/
PancakePair.sol#166-188):
        External calls:
        - _safeTransfer(_token0,to,amount0) (contracts/pancakeSwap/PancakePair.sol#180)
                - (success,data) =
token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/pancakeSwap/
PancakePair.sol#59)
        - _safeTransfer(_token1,to,amount1) (contracts/pancakeSwap/PancakePair.sol#181)
                - (success,data) =
token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/pancakeSwap/
PancakePair.sol#59)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/pancakeSwap/
PancakePair.sol#185)
                - blockTimestampLast = blockTimestamp (contracts/pancakeSwap/
PancakePair.sol#113)
        - kLast = uint256(reserve0).mul(reserve1) (contracts/pancakeSwap/
PancakePair.sol#186)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/pancakeSwap/
PancakePair.sol#185)
                - reserve0 = uint112(balance0) (contracts/pancakeSwap/
PancakePair.sol#111)
```

```
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/pancakeSwap/
PancakePair.sol#185)
                - reserve1 = uint112(balance1) (contracts/pancakeSwap/
PancakePair.sol#112)
Reentrancy in WhalesCandy.claimRewards(uint256) (contracts/wc.sol#682-697):
        External calls:
        - IERC20(BUSDaddress).transferFrom(msg.sender,_dev1,feeToPay) (contracts/
wc.sol#688)
        State variables written after the call(s):
        - stakes[msg.sender][_stakeIndex].feePaid = stakes[msg.sender]
[_stakeIndex].feePaid.add(feeToPay) (contracts/wc.sol#689)
        - _collect(msg.sender,_stakeIndex) (contracts/wc.sol#692)
                - stakes[_user][_stakeIndex].collected = stakes[_user]
[_stakeIndex].collected.add(calcReward(_user,_stakeIndex)) (contracts/wc.sol#677)
                - stakes[_user][_stakeIndex].lastUpdate = block.timestamp (contracts/
wc.sol#678)
        - stakes[msg.sender][_stakeIndex].claimed = stakes[msg.sender]
[_stakeIndex].collected (contracts/wc.sol#694)
Reentrancy in PancakeFactory.createPair(address,address) (contracts/pancakeSwap/
PancakeFactory.sol#28-43):
        External calls:
        - IPancakePair(pair).initialize(token0,token1) (contracts/pancakeSwap/
PancakeFactory.sol#38)
        State variables written after the call(s):
        - getPair[token0][token1] = pair (contracts/pancakeSwap/PancakeFactory.sol#39)
        - getPair[token1][token0] = pair (contracts/pancakeSwap/PancakeFactory.sol#40)
Reentrancy in WhalesCandy.reinvest(uint256) (contracts/wc.sol#700-732):
        External calls:
        - IERC20(BUSDaddress).transferFrom(msg.sender,_dev1,feeToPay) (contracts/
wc.sol#706)
        State variables written after the call(s):
        - stakes[msg.sender][_stakeIndex].feePaid = stakes[msg.sender]
[_stakeIndex].feePaid.add(feeToPay) (contracts/wc.sol#707)
        - _collect(msg.sender,_stakeIndex) (contracts/wc.sol#710)
                - stakes[_user][_stakeIndex].collected = stakes[_user]
[_stakeIndex].collected.add(calcReward(_user,_stakeIndex)) (contracts/wc.sol#677)
                - stakes[_user][_stakeIndex].lastUpdate = block.timestamp (contracts/
wc.sol#678)
        - stakeInt(_amount) (contracts/wc.sol#715)
                - stakes[msg.sender][stakeNumber[msg.sender]].amount = _amount
(contracts/wc.sol#345)
```

```
                - stakes[msg.sender][stakeNumber[msg.sender]].stakeTime =
block.timestamp (contracts/wc.sol#346)
                - stakes[msg.sender][stakeNumber[msg.sender]].lastUpdate =
block.timestamp (contracts/wc.sol#347)
                - stakes[msg.sender][stakeNumber[msg.sender]].freeClaiming = false
(contracts/wc.sol#348)
                - stakes[msg.sender][stakeNumber[msg.sender]].claimingStartFee =
BUSDfeePerWeek (contracts/wc.sol#349)
        - stakes[msg.sender][_stakeIndex].claimed = stakes[msg.sender]
[_stakeIndex].collected (contracts/wc.sol#718)
Reentrancy in PancakePair.swap(uint256,uint256,address,bytes) (contracts/pancakeSwap/
PancakePair.sol#191-238):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (contracts/pancakeSwap/
PancakePair.sol#211)
                - (success,data) =
token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/pancakeSwap/
PancakePair.sol#59)
        - _safeTransfer(_token1,to,amount1Out) (contracts/pancakeSwap/
PancakePair.sol#212)
                - (success,data) =
token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/pancakeSwap/
PancakePair.sol#59)
        - IPancakeCallee(to).pancakeCall(msg.sender,amount0Out,amount1Out,data)
(contracts/pancakeSwap/PancakePair.sol#214)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/pancakeSwap/
PancakePair.sol#236)
                - blockTimestampLast = blockTimestamp (contracts/pancakeSwap/
PancakePair.sol#113)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/pancakeSwap/
PancakePair.sol#236)
                - reserve0 = uint112(balance0) (contracts/pancakeSwap/
PancakePair.sol#111)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/pancakeSwap/
PancakePair.sol#236)
                - reserve1 = uint112(balance1) (contracts/pancakeSwap/
PancakePair.sol#112)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1
```

```
PancakeRouter._swapSupportingFeeOnTransferTokens(address[],address).i (contracts/
pancakeSwap/PancakeRouter.sol#426) is a local variable never initialized
PancakeRouter._swap(uint256[],address[],address).i (contracts/pancakeSwap/
PancakeRouter.sol#283) is a local variable never initialized
PancakeRouter01._swap(uint256[],address[],address).i (contracts/pancakeSwap/
PancakeRouter01.sol#229) is a local variable never initialized
PancakeLibrary.getAmountsOut(address,uint256,address[]).i (contracts/pancakeSwap/
libraries/PancakeLibrary.sol#100) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
local-variables

PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256) (contracts/
pancakeSwap/PancakeRouter.sol#35-62) ignores return value by
IPancakeFactory(factory).createPair(tokenA,tokenB) (contracts/pancakeSwap/
PancakeRouter.sol#45)
PancakeRouter01._addLiquidity(address,address,uint256,uint256,uint256,uint256)
(contracts/pancakeSwap/PancakeRouter01.sol#32-59) ignores return value by
IPancakeFactory(factory).createPair(tokenA,tokenB) (contracts/pancakeSwap/
PancakeRouter01.sol#42)
WhalesCandy.buyAndStake() (contracts/wc.sol#388-411) ignores return value by
_pancakeRouter.swapExactETHForTokens{value: rawAmount.mul(70).div(100)}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#401-406)
WhalesCandy.updateDaily() (contracts/wc.sol#414-489) ignores return value by
IERC20(contrAddr).approve(_pancakeRouterAddress,type()(uint256).max) (contracts/
wc.sol#434)
WhalesCandy.updateDaily() (contracts/wc.sol#414-489) ignores return value by
_pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
WhalesCandy.burnAndBuyback() (contracts/wc.sol#495-535) ignores return value by
IERC20(tradingPair).approve(_pancakeRouterAddress,type()(uint256).max) (contracts/
wc.sol#498)
WhalesCandy.burnAndBuyback() (contracts/wc.sol#495-535) ignores return value by _pancake
Router.removeLiquidityETHSupportingFeeOnTransferTokens(contrAddr,lpBalToRemove,0,0,contr
Addr,block.timestamp + 100) (contracts/wc.sol#506-513)
WhalesCandy.burnAndBuyback() (contracts/wc.sol#495-535) ignores return value by
_pancakeRouter.swapExactETHForTokens{value: ethGain}(0,path,address(0),block.timestamp
+ 100) (contracts/wc.sol#523-528)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

WhalesCandy.setDevs(address,address) (contracts/wc.sol#239-242) should emit an event
for:
```

```
        - _dev = dev (contracts/wc.sol#240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
access-control

WhalesCandy.setBuyBackPerecent(uint8) (contracts/wc.sol#245-249) should emit an event
for:
        - buyBackPerecent = _buyBackPerecent (contracts/wc.sol#248)
WhalesCandy.setTaxFactor(uint8) (contracts/wc.sol#252-256) should emit an event for:
        - taxFactor = _taxFactor (contracts/wc.sol#255)
WhalesCandy.stake(uint256) (contracts/wc.sol#328-341) should emit an event for:
        - overallStakedToken += _amount (contracts/wc.sol#340)
WhalesCandy.setLaunchTime(uint256) (contracts/wc.sol#367-370) should emit an event
for:
        - LAUNCH_TIME = newTime (contracts/wc.sol#369)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

PancakeFactory.constructor(address)._feeToSetter (contracts/pancakeSwap/
PancakeFactory.sol#20) lacks a zero-check on :
                - feeToSetter = _feeToSetter (contracts/pancakeSwap/
PancakeFactory.sol#21)
PancakeFactory.setFeeTo(address)._feeTo (contracts/pancakeSwap/PancakeFactory.sol#45)
lacks a zero-check on :
                - feeTo = _feeTo (contracts/pancakeSwap/PancakeFactory.sol#47)
PancakeFactory.setFeeToSetter(address)._feeToSetter (contracts/pancakeSwap/
PancakeFactory.sol#50) lacks a zero-check on :
                - feeToSetter = _feeToSetter (contracts/pancakeSwap/
PancakeFactory.sol#52)
PancakePair.initialize(address,address)._token0 (contracts/pancakeSwap/
PancakePair.sol#83) lacks a zero-check on :
                - token0 = _token0 (contracts/pancakeSwap/PancakePair.sol#85)
PancakePair.initialize(address,address)._token1 (contracts/pancakeSwap/
PancakePair.sol#83) lacks a zero-check on :
                - token1 = _token1 (contracts/pancakeSwap/PancakePair.sol#86)
PancakeRouter.constructor(address,address)._factory (contracts/pancakeSwap/
PancakeRouter.sol#25) lacks a zero-check on :
                - factory = _factory (contracts/pancakeSwap/PancakeRouter.sol#26)
PancakeRouter.constructor(address,address)._WETH (contracts/pancakeSwap/
PancakeRouter.sol#25) lacks a zero-check on :
                - WETH = _WETH (contracts/pancakeSwap/PancakeRouter.sol#27)
PancakeRouter01.constructor(address,address)._factory (contracts/pancakeSwap/
```

```
PancakeRouter01.sol#22) lacks a zero-check on :
                - factory = _factory (contracts/pancakeSwap/PancakeRouter01.sol#23)
PancakeRouter01.constructor(address,address)._WETH (contracts/pancakeSwap/
PancakeRouter01.sol#22) lacks a zero-check on :
                - WETH = _WETH (contracts/pancakeSwap/PancakeRouter01.sol#24)
WhalesCandy.setDevs(address,address).dev (contracts/wc.sol#239) lacks a zero-check on :
                - _dev = dev (contracts/wc.sol#240)
WhalesCandy.setDevs(address,address).dev1 (contracts/wc.sol#239) lacks a zero-check
on :
                - _dev1 = dev1 (contracts/wc.sol#241)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation

Reentrancy in PancakePair.burn(address) (contracts/pancakeSwap/
PancakePair.sol#166-188):
        External calls:
        - _safeTransfer(_token0,to,amount0) (contracts/pancakeSwap/PancakePair.sol#180)
                - (success,data) =
token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/pancakeSwap/
PancakePair.sol#59)
        - _safeTransfer(_token1,to,amount1) (contracts/pancakeSwap/PancakePair.sol#181)
                - (success,data) =
token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/pancakeSwap/
PancakePair.sol#59)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/pancakeSwap/
PancakePair.sol#185)
                - price0CumulativeLast +=
uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (contracts/
pancakeSwap/PancakePair.sol#104-106)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/pancakeSwap/
PancakePair.sol#185)
                - price1CumulativeLast +=
uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (contracts/
pancakeSwap/PancakePair.sol#107-109)
Reentrancy in WhalesCandy.burnAndBuyback() (contracts/wc.sol#495-535):
        External calls:
        - IERC20(tradingPair).approve(_pancakeRouterAddress,type()(uint256).max)
(contracts/wc.sol#498)
        - _pancakeRouter.removeLiquidityETHSupportingFeeOnTransferTokens(contrAddr,lpBal
ToRemove,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#506-513)
```

```
        State variables written after the call(s):
        - usedETHforBuyBack += ethGain (contracts/wc.sol#516)
Reentrancy in WhalesCandy.burnAndBuyback() (contracts/wc.sol#495-535):
        External calls:
        - IERC20(tradingPair).approve(_pancakeRouterAddress,type()(uint256).max)
(contracts/wc.sol#498)
        - _pancakeRouter.removeLiquidityETHSupportingFeeOnTransferTokens(contrAddr,lpBal
ToRemove,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#506-513)
        - _pancakeRouter.swapExactETHForTokens{value: ethGain}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#523-528)
        External calls sending eth:
        - _pancakeRouter.swapExactETHForTokens{value: ethGain}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#523-528)
        State variables written after the call(s):
        - transferToZero(receivedToken) (contracts/wc.sol#533)
                - _Balances[contrAddr] = _Balances[contrAddr].sub(amount,Token:
transfer amount exceeds balance) (contracts/wc.sol#269)
                - _Balances[address(0)] = _Balances[address(0)].add(amount) (contracts/
wc.sol#270)
Reentrancy in WhalesCandy.buyAndStake() (contracts/wc.sol#388-411):
        External calls:
        - _pancakeRouter.swapExactETHForTokens{value: rawAmount.mul(70).div(100)}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#401-406)
        External calls sending eth:
        - _dev.transfer(rawAmount.mul(30).div(100)) (contracts/wc.sol#392)
        - _pancakeRouter.swapExactETHForTokens{value: rawAmount.mul(70).div(100)}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#401-406)
        State variables written after the call(s):
        - stakeInt(stakeAmount) (contracts/wc.sol#408)
                - overallStakedToken += _amount (contracts/wc.sol#352)
        - stakeInt(stakeAmount) (contracts/wc.sol#408)
                - stakeNumber[msg.sender] ++ (contracts/wc.sol#351)
        - stakeInt(stakeAmount) (contracts/wc.sol#408)
                - stakes[msg.sender][stakeNumber[msg.sender]].amount = _amount
(contracts/wc.sol#345)
                - stakes[msg.sender][stakeNumber[msg.sender]].stakeTime =
block.timestamp (contracts/wc.sol#346)
                - stakes[msg.sender][stakeNumber[msg.sender]].lastUpdate =
block.timestamp (contracts/wc.sol#347)
                - stakes[msg.sender][stakeNumber[msg.sender]].freeClaiming = false
(contracts/wc.sol#348)
```

```
                - stakes[msg.sender][stakeNumber[msg.sender]].claimingStartFee =
BUSDfeePerWeek (contracts/wc.sol#349)
Reentrancy in WhalesCandy.buyShareFromAuction() (contracts/wc.sol#538-568):
        External calls:
        - updateDaily() (contracts/wc.sol#549)
                - IERC20(tradingPair).approve(_pancakeRouterAddress,type()
(uint256).max) (contracts/wc.sol#498)
                - _pancakeRouter.removeLiquidityETHSupportingFeeOnTransferTokens(contrAd
dr,lpBalToRemove,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#506-513)
                - _pancakeRouter.swapExactETHForTokens{value: ethGain}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#523-528)
                - IERC20(contrAddr).approve(_pancakeRouterAddress,type()(uint256).max)
(contracts/wc.sol#434)
                - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value:
ETHtoAdd}(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/
wc.sol#437-445)
                - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        External calls sending eth:
        - _dev.transfer(rawAmount.mul(30).div(100)) (contracts/wc.sol#543)
        - updateDaily() (contracts/wc.sol#549)
                - _pancakeRouter.swapExactETHForTokens{value: ethGain}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#523-528)
                - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value:
ETHtoAdd}(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/
wc.sol#437-445)
                - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        State variables written after the call(s):
        - auctionEntry_allDays += rawAmount (contracts/wc.sol#552)
        - mapMemberAuction[msg.sender][currentDay].memberAuctionValue += rawAmount
(contracts/wc.sol#563)
        - mapMemberAuction[msg.sender][currentDay].memberAuctionEntryDay = currentDay
(contracts/wc.sol#564)
        - mapMemberAuction[msg.sender][currentDay].hasChangedShareToToken = false
(contracts/wc.sol#565)
        - mapMemberAuction_overallData[msg.sender].total_auctionEnteries += rawAmount
(contracts/wc.sol#561)
        - usersCount ++ (contracts/wc.sol#557)
        - usersCountDaily[currentDay] ++ (contracts/wc.sol#558)
Reentrancy in WhalesCandy.claimRewards(uint256) (contracts/wc.sol#682-697):
```

```
        External calls:
        - IERC20(BUSDaddress).transferFrom(msg.sender,_dev1,feeToPay) (contracts/
wc.sol#688)
        State variables written after the call(s):
        - _mint(msg.sender,reward) (contracts/wc.sol#695)
                - _Balances[_user] = _Balances[_user].add(_amount) (contracts/
wc.sol#306)
        - _mint(msg.sender,reward) (contracts/wc.sol#695)
                - _totalSupply = _totalSupply.add(_amount) (contracts/wc.sol#307)
        - overallCollectedDividends += reward (contracts/wc.sol#696)
Reentrancy in PancakeFactory.createPair(address,address) (contracts/pancakeSwap/
PancakeFactory.sol#28-43):
        External calls:
        - IPancakePair(pair).initialize(token0,token1) (contracts/pancakeSwap/
PancakeFactory.sol#38)
        State variables written after the call(s):
        - allPairs.push(pair) (contracts/pancakeSwap/PancakeFactory.sol#41)
Reentrancy in WhalesCandy.reinvest(uint256) (contracts/wc.sol#700-732):
        External calls:
        - IERC20(BUSDaddress).transferFrom(msg.sender,_dev1,feeToPay) (contracts/
wc.sol#706)
        State variables written after the call(s):
        - _mint(_dev,_amount.mul(5).div(100)) (contracts/wc.sol#721)
                - _Balances[_user] = _Balances[_user].add(_amount) (contracts/
wc.sol#306)
        - _mint(_dev,_amount.mul(5).div(100)) (contracts/wc.sol#727)
                - _Balances[_user] = _Balances[_user].add(_amount) (contracts/
wc.sol#306)
        - _mint(_dev,_amount.mul(5).div(100)) (contracts/wc.sol#721)
                - _totalSupply = _totalSupply.add(_amount) (contracts/wc.sol#307)
        - _mint(_dev,_amount.mul(5).div(100)) (contracts/wc.sol#727)
                - _totalSupply = _totalSupply.add(_amount) (contracts/wc.sol#307)
        - mapRefData[myRef[msg.sender]][currentDay + 1].refEarnedTokens +=
_amount.mul(5).div(100) (contracts/wc.sol#723)
        - mapRefData[myRef[msg.sender]][currentDay + 1].hasClaimed = false (contracts/
wc.sol#724)
        - overallCollectedDividends += _amount (contracts/wc.sol#716)
        - stakeInt(_amount) (contracts/wc.sol#715)
                - overallStakedToken += _amount (contracts/wc.sol#352)
        - stakeInt(_amount) (contracts/wc.sol#715)
                - stakeNumber[msg.sender] ++ (contracts/wc.sol#351)
```

```
Reentrancy in WhalesCandy.reinvest(uint256) (contracts/wc.sol#700-732):
        External calls:
        - IERC20(BUSDaddress).transferFrom(msg.sender,_dev1,feeToPay) (contracts/
wc.sol#706)
        - updateDaily() (contracts/wc.sol#730)
                - IERC20(tradingPair).approve(_pancakeRouterAddress,type()
(uint256).max) (contracts/wc.sol#498)
                - _pancakeRouter.removeLiquidityETHSupportingFeeOnTransferTokens(contrAd
dr,lpBalToRemove,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#506-513)
                - _pancakeRouter.swapExactETHForTokens{value: ethGain}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#523-528)
                - IERC20(contrAddr).approve(_pancakeRouterAddress,type()(uint256).max)
(contracts/wc.sol#434)
                - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value:
ETHtoAdd}(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/
wc.sol#437-445)
                - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        External calls sending eth:
        - updateDaily() (contracts/wc.sol#730)
                - _pancakeRouter.swapExactETHForTokens{value: ethGain}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#523-528)
                - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value:
ETHtoAdd}(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/
wc.sol#437-445)
                - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        State variables written after the call(s):
        - updateDaily() (contracts/wc.sol#730)
                - _allowance[msg.sender][spender] = value (contracts/wc.sol#215)
Reentrancy in PancakePair.swap(uint256,uint256,address,bytes) (contracts/pancakeSwap/
PancakePair.sol#191-238):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (contracts/pancakeSwap/
PancakePair.sol#211)
                - (success,data) =
token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/pancakeSwap/
PancakePair.sol#59)
        - _safeTransfer(_token1,to,amount1Out) (contracts/pancakeSwap/
PancakePair.sol#212)
                - (success,data) =
```

```
token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/pancakeSwap/
PancakePair.sol#59)
        - IPancakeCallee(to).pancakeCall(msg.sender,amount0Out,amount1Out,data)
(contracts/pancakeSwap/PancakePair.sol#214)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/pancakeSwap/
PancakePair.sol#236)
                - price0CumulativeLast +=
uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (contracts/
pancakeSwap/PancakePair.sol#104-106)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/pancakeSwap/
PancakePair.sol#236)
                - price1CumulativeLast +=
uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (contracts/
pancakeSwap/PancakePair.sol#107-109)
Reentrancy in WhalesCandy.updateDaily() (contracts/wc.sol#414-489):
        External calls:
        - IERC20(contrAddr).approve(_pancakeRouterAddress,type()(uint256).max)
(contracts/wc.sol#434)
        - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value: ETHtoAdd}
(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#437-445)
        - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        External calls sending eth:
        - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value: ETHtoAdd}
(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#437-445)
        - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        State variables written after the call(s):
        - lpBal = IERC20(tradingPair).balanceOf(contrAddr) (contracts/wc.sol#483)
        - tradingPair = _pancakeFactory.getPair(_pancakeRouter.WETH(),contrAddr)
(contracts/wc.sol#480)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2

Reentrancy in PancakePair.burn(address) (contracts/pancakeSwap/
PancakePair.sol#166-188):
        External calls:
        - _safeTransfer(_token0,to,amount0) (contracts/pancakeSwap/PancakePair.sol#180)
                - (success,data) =
token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/pancakeSwap/
PancakePair.sol#59)
```

```
        - _safeTransfer(_token1,to,amount1) (contracts/pancakeSwap/PancakePair.sol#181)
                - (success,data) =
token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/pancakeSwap/
PancakePair.sol#59)
        Event emitted after the call(s):
        - Burn(msg.sender,amount0,amount1,to) (contracts/pancakeSwap/
PancakePair.sol#187)
        - Sync(reserve0,reserve1) (contracts/pancakeSwap/PancakePair.sol#114)
                - _update(balance0,balance1,_reserve0,_reserve1) (contracts/pancakeSwap/
PancakePair.sol#185)
Reentrancy in WhalesCandy.burnAndBuyback() (contracts/wc.sol#495-535):
        External calls:
        - IERC20(tradingPair).approve(_pancakeRouterAddress,type()(uint256).max)
(contracts/wc.sol#498)
        - _pancakeRouter.removeLiquidityETHSupportingFeeOnTransferTokens(contrAddr,lpBal
ToRemove,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#506-513)
        - _pancakeRouter.swapExactETHForTokens{value: ethGain}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#523-528)
        External calls sending eth:
        - _pancakeRouter.swapExactETHForTokens{value: ethGain}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#523-528)
        Event emitted after the call(s):
        - Transfer(contrAddr,address(0),amount) (contracts/wc.sol#271)
                - transferToZero(receivedToken) (contracts/wc.sol#533)
Reentrancy in WhalesCandy.claimRewards(uint256) (contracts/wc.sol#682-697):
        External calls:
        - IERC20(BUSDaddress).transferFrom(msg.sender,_dev1,feeToPay) (contracts/
wc.sol#688)
        Event emitted after the call(s):
        - Transfer(address(0),_user,_amount) (contracts/wc.sol#308)
                - _mint(msg.sender,reward) (contracts/wc.sol#695)
Reentrancy in PancakeFactory.createPair(address,address) (contracts/pancakeSwap/
PancakeFactory.sol#28-43):
        External calls:
        - IPancakePair(pair).initialize(token0,token1) (contracts/pancakeSwap/
PancakeFactory.sol#38)
        Event emitted after the call(s):
        - PairCreated(token0,token1,pair,allPairs.length) (contracts/pancakeSwap/
PancakeFactory.sol#42)
Reentrancy in WhalesCandy.reinvest(uint256) (contracts/wc.sol#700-732):
        External calls:
```

```
        - IERC20(BUSDaddress).transferFrom(msg.sender,_dev1,feeToPay) (contracts/
wc.sol#706)
        Event emitted after the call(s):
        - Transfer(address(0),_user,_amount) (contracts/wc.sol#308)
                - _mint(_dev,_amount.mul(5).div(100)) (contracts/wc.sol#727)
        - Transfer(address(0),_user,_amount) (contracts/wc.sol#308)
                - _mint(_dev,_amount.mul(5).div(100)) (contracts/wc.sol#721)
Reentrancy in WhalesCandy.reinvest(uint256) (contracts/wc.sol#700-732):
        External calls:
        - IERC20(BUSDaddress).transferFrom(msg.sender,_dev1,feeToPay) (contracts/
wc.sol#706)
        - updateDaily() (contracts/wc.sol#730)
                - IERC20(tradingPair).approve(_pancakeRouterAddress,type()
(uint256).max) (contracts/wc.sol#498)
                - _pancakeRouter.removeLiquidityETHSupportingFeeOnTransferTokens(contrAd
dr,lpBalToRemove,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#506-513)
                - _pancakeRouter.swapExactETHForTokens{value: ethGain}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#523-528)
                - IERC20(contrAddr).approve(_pancakeRouterAddress,type()(uint256).max)
(contracts/wc.sol#434)
                - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value:
ETHtoAdd}(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/
wc.sol#437-445)
                - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        External calls sending eth:
        - updateDaily() (contracts/wc.sol#730)
                - _pancakeRouter.swapExactETHForTokens{value: ethGain}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#523-528)
                - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value:
ETHtoAdd}(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/
wc.sol#437-445)
                - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        Event emitted after the call(s):
        - Approval(msg.sender,spender,value) (contracts/wc.sol#216)
                - updateDaily() (contracts/wc.sol#730)
        - Transfer(contrAddr,address(0),amount) (contracts/wc.sol#271)
                - updateDaily() (contracts/wc.sol#730)
        - Transfer(address(0),_user,_amount) (contracts/wc.sol#308)
                - updateDaily() (contracts/wc.sol#730)
```

```
Reentrancy in PancakePair.swap(uint256,uint256,address,bytes) (contracts/pancakeSwap/
PancakePair.sol#191-238):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (contracts/pancakeSwap/
PancakePair.sol#211)
                - (success,data) =
token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/pancakeSwap/
PancakePair.sol#59)
        - _safeTransfer(_token1,to,amount1Out) (contracts/pancakeSwap/
PancakePair.sol#212)
                - (success,data) =
token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/pancakeSwap/
PancakePair.sol#59)
        - IPancakeCallee(to).pancakeCall(msg.sender,amount0Out,amount1Out,data)
(contracts/pancakeSwap/PancakePair.sol#214)
        Event emitted after the call(s):
        - Swap(msg.sender,amount0In,amount1In,amount0Out,amount1Out,to) (contracts/
pancakeSwap/PancakePair.sol#237)
        - Sync(reserve0,reserve1) (contracts/pancakeSwap/PancakePair.sol#114)
                - _update(balance0,balance1,_reserve0,_reserve1) (contracts/pancakeSwap/
PancakePair.sol#236)
Reentrancy in WhalesCandy.updateDaily() (contracts/wc.sol#414-489):
        External calls:
        - IERC20(contrAddr).approve(_pancakeRouterAddress,type()(uint256).max)
(contracts/wc.sol#434)
        - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value: ETHtoAdd}
(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#437-445)
        External calls sending eth:
        - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value: ETHtoAdd}
(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#437-445)
        Event emitted after the call(s):
        - Transfer(address(0),_user,_amount) (contracts/wc.sol#308)
                - _mint(contrAddr,neededToken) (contracts/wc.sol#455)
Reentrancy in WhalesCandy.updateDaily() (contracts/wc.sol#414-489):
        External calls:
        - IERC20(contrAddr).approve(_pancakeRouterAddress,type()(uint256).max)
(contracts/wc.sol#434)
        - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value: ETHtoAdd}
(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#437-445)
        - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
```

```
        External calls sending eth:
        - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value: ETHtoAdd}
(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#437-445)
        - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        Event emitted after the call(s):
        - Transfer(contrAddr,address(0),amount) (contracts/wc.sol#271)
                - transferToZero(leftOverToken.sub(1000000000000000000)) (contracts/
wc.sol#469)
Reentrancy in WhalesCandy.updateDaily() (contracts/wc.sol#414-489):
        External calls:
        - IERC20(contrAddr).approve(_pancakeRouterAddress,type()(uint256).max)
(contracts/wc.sol#434)
        - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value: ETHtoAdd}
(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#437-445)
        - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        - burnAndBuyback() (contracts/wc.sol#486)
                - IERC20(tradingPair).approve(_pancakeRouterAddress,type()
(uint256).max) (contracts/wc.sol#498)
                - _pancakeRouter.removeLiquidityETHSupportingFeeOnTransferTokens(contrAd
dr,lpBalToRemove,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#506-513)
                - _pancakeRouter.swapExactETHForTokens{value: ethGain}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#523-528)
        External calls sending eth:
        - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value: ETHtoAdd}
(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#437-445)
        - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        - burnAndBuyback() (contracts/wc.sol#486)
                - _pancakeRouter.swapExactETHForTokens{value: ethGain}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#523-528)
        Event emitted after the call(s):
        - Transfer(contrAddr,address(0),amount) (contracts/wc.sol#271)
                - burnAndBuyback() (contracts/wc.sol#486)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

PancakeERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (contracts/
pancakeSwap/PancakeERC20.sol#98-123) uses timestamp for comparisons
        Dangerous comparisons:
```

```
        - require(bool,string)(deadline >= block.timestamp,Pancake: EXPIRED) (contracts/
pancakeSwap/PancakeERC20.sol#107)
PancakePair._update(uint256,uint256,uint112,uint112) (contracts/pancakeSwap/
PancakePair.sol#90-115) uses timestamp for comparisons
        Dangerous comparisons:
        - timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (contracts/pancakeSwap/
PancakePair.sol#102)
WhalesCandy.stake(uint256) (contracts/wc.sol#328-341) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_Balances[msg.sender] >= _amount,not Enoght token to
stake) (contracts/wc.sol#329)
WhalesCandy.setLaunchTime(uint256) (contracts/wc.sol#367-370) uses timestamp for
comparisons
        Dangerous comparisons:
        - require(bool)(newTime > block.timestamp) (contracts/wc.sol#368)
WhalesCandy.updateDaily() (contracts/wc.sol#414-489) uses timestamp for comparisons
        Dangerous comparisons:
        - currentDay != thisDay() (contracts/wc.sol#416)
WhalesCandy.buyShareFromAuction() (contracts/wc.sol#538-568) uses timestamp for
comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= LAUNCH_TIME,Auctions have not starded
now!) (contracts/wc.sol#541)
        - currentDay == 0 (contracts/wc.sol#545)
WhalesCandy.claimTokenFromSharesAndStake(uint256,uint256) (contracts/wc.sol#578-601)
uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_day < currentDay,Day must be over to claim!) (contracts/
wc.sol#579)
WhalesCandy.claimRefTokensAndStake(uint256) (contracts/wc.sol#604-618) uses timestamp
for comparisons
        Dangerous comparisons:
        - require(bool,string)(_day < currentDay,Refs Day must be over to claim!)
(contracts/wc.sol#605)
WhalesCandy.calcReward(address,uint256) (contracts/wc.sol#625-639) uses timestamp for
comparisons
        Dangerous comparisons:
        - stakes[_user]
[_stakeIndex].amount.mul(multiplier).div(10000000000000000000).add(stakes[msg.sender]
[_stakeIndex].collected) > stakes[_user][_stakeIndex].amount.mul(365).div(100)
(contracts/wc.sol#633-634)
```

```
WhalesCandy.calcClaim(address,uint256) (contracts/wc.sol#644-657) uses timestamp for
comparisons
        Dangerous comparisons:
        - multiplier.mul(stakes[_user]
[_stakeIndex].amount).div(10000000000000000000).add(stakes[msg.sender]
[_stakeIndex].collected) > stakes[_user][_stakeIndex].amount.mul(365).div(100)
(contracts/wc.sol#651-652)
WhalesCandy.calcClaimFee(address,uint256) (contracts/wc.sol#660-672) uses timestamp for
comparisons
        Dangerous comparisons:
        - totalClaimFee > maxBUSDFee (contracts/wc.sol#668)
WhalesCandy.claimRewards(uint256) (contracts/wc.sol#682-697) uses timestamp for
comparisons
        Dangerous comparisons:
        - stakes[msg.sender][_stakeIndex].freeClaiming == false && feeToPay > 0
(contracts/wc.sol#685)
        - require(bool,string)(IERC20(BUSDaddress).allowance(msg.sender,contrAddr) >=
feeToPay,Approval to spend BUSD is needed!) (contracts/wc.sol#686)
        - require(bool,string)(IERC20(BUSDaddress).balanceOf(msg.sender) >=
feeToPay,Not enough BUSD in wallet!) (contracts/wc.sol#687)
WhalesCandy.reinvest(uint256) (contracts/wc.sol#700-732) uses timestamp for comparisons
        Dangerous comparisons:
        - stakes[msg.sender][_stakeIndex].freeClaiming == false && feeToPay > 0
(contracts/wc.sol#703)
        - require(bool,string)(IERC20(BUSDaddress).allowance(msg.sender,contrAddr) >=
feeToPay,Approval to spend BUSD is needed!) (contracts/wc.sol#704)
        - require(bool,string)(IERC20(BUSDaddress).balanceOf(msg.sender) >=
feeToPay,Not enough BUSD in wallet!) (contracts/wc.sol#705)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

PancakeERC20.constructor() (contracts/pancakeSwap/PancakeERC20.sol#27-43) uses assembly
        - INLINE ASM (contracts/pancakeSwap/PancakeERC20.sol#29-31)
PancakeFactory.createPair(address,address) (contracts/pancakeSwap/
PancakeFactory.sol#28-43) uses assembly
        - INLINE ASM (contracts/pancakeSwap/PancakeFactory.sol#35-37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

WhalesCandy.claimTokenFromSharesAndStake(uint256,uint256) (contracts/wc.sol#578-601)
compares to a boolean constant:
        -require(bool,string)(mapMemberAuction[msg.sender][_day].hasChangedShareToToken
```

```
== false,User has already Changed his shares to Token that Day!) (contracts/wc.sol#580)
WhalesCandy.claimRefTokensAndStake(uint256) (contracts/wc.sol#604-618) compares to a
boolean constant:
        -require(bool,string)(mapRefData[msg.sender][_day].hasClaimed == false,Ref has
already Claimed!) (contracts/wc.sol#607)
WhalesCandy.calcClaimFee(address,uint256) (contracts/wc.sol#660-672) compares to a
boolean constant:
        -stakes[_user][_stakeIndex].stakeTime == 0 || stakes[_user]
[_stakeIndex].freeClaiming == true (contracts/wc.sol#661)
WhalesCandy.claimRewards(uint256) (contracts/wc.sol#682-697) compares to a boolean
constant:
        -stakes[msg.sender][_stakeIndex].freeClaiming == false && feeToPay > 0
(contracts/wc.sol#685)
WhalesCandy.reinvest(uint256) (contracts/wc.sol#700-732) compares to a boolean
constant:
        -stakes[msg.sender][_stakeIndex].freeClaiming == false && feeToPay > 0
(contracts/wc.sol#703)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-
equality

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.15']
        - ^0.8.0 (contracts/pancakeSwap/PancakeERC20.sol#3)
        - ^0.8.0 (contracts/pancakeSwap/PancakeFactory.sol#3)
        - ^0.8.0 (contracts/pancakeSwap/PancakePair.sol#3)
        - ^0.8.0 (contracts/pancakeSwap/PancakeRouter.sol#3)
        - ^0.8.0 (contracts/pancakeSwap/PancakeRouter01.sol#3)
        - ^0.8.15 (contracts/pancakeSwap/WETH.sol#3)
        - ^0.8.0 (contracts/pancakeSwap/interfaces/IERC20.sol#3)
        - ^0.8.0 (contracts/pancakeSwap/interfaces/IPancakeCallee.sol#3)
        - ^0.8.0 (contracts/pancakeSwap/interfaces/IPancakeERC20.sol#3)
        - ^0.8.0 (contracts/pancakeSwap/interfaces/IPancakeFactory.sol#3)
        - ^0.8.0 (contracts/pancakeSwap/interfaces/IPancakePair.sol#3)
        - ^0.8.0 (contracts/pancakeSwap/interfaces/IPancakeRouter01.sol#3)
        - ^0.8.0 (contracts/pancakeSwap/interfaces/IPancakeRouter02.sol#3)
        - ^0.8.0 (contracts/pancakeSwap/interfaces/IWETH.sol#3)
        - ^0.8.15 (contracts/pancakeSwap/libraries/Math.sol#3)
        - ^0.8.15 (contracts/pancakeSwap/libraries/PancakeLibrary.sol#3)
        - ^0.8.15 (contracts/pancakeSwap/libraries/SafeMath.sol#3)
        - ^0.8.15 (contracts/pancakeSwap/libraries/TransferHelper.sol#3)
        - ^0.8.15 (contracts/pancakeSwap/libraries/UQ112x112.sol#3)
```

```
        - ^0.8.0 (contracts/wc.sol#8)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
pragma-directives-are-used

SafeMath.mod(uint256,uint256) (contracts/wc.sol#55-57) is never used and should be
removed
SafeMath.mod(uint256,uint256,string) (contracts/wc.sol#59-62) is never used and should
be removed
TransferHelper.safeApprove(address,address,uint256) (contracts/pancakeSwap/libraries/
TransferHelper.sol#7-20) is never used and should be removed
WhalesCandy._burn(address,uint256) (contracts/wc.sol#312-316) is never used and should
be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (contracts/pancakeSwap/PancakeERC20.sol#3) allows old versions
Pragma version^0.8.0 (contracts/pancakeSwap/PancakeFactory.sol#3) allows old versions
Pragma version^0.8.0 (contracts/pancakeSwap/PancakePair.sol#3) allows old versions
Pragma version^0.8.0 (contracts/pancakeSwap/PancakeRouter.sol#3) allows old versions
Pragma version^0.8.0 (contracts/pancakeSwap/PancakeRouter01.sol#3) allows old versions
Pragma version^0.8.15 (contracts/pancakeSwap/WETH.sol#3) necessitates a version too
recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (contracts/pancakeSwap/interfaces/IERC20.sol#3) allows old
versions
Pragma version^0.8.0 (contracts/pancakeSwap/interfaces/IPancakeCallee.sol#3) allows old
versions
Pragma version^0.8.0 (contracts/pancakeSwap/interfaces/IPancakeERC20.sol#3) allows old
versions
Pragma version^0.8.0 (contracts/pancakeSwap/interfaces/IPancakeFactory.sol#3) allows
old versions
Pragma version^0.8.0 (contracts/pancakeSwap/interfaces/IPancakePair.sol#3) allows old
versions
Pragma version^0.8.0 (contracts/pancakeSwap/interfaces/IPancakeRouter01.sol#3) allows
old versions
Pragma version^0.8.0 (contracts/pancakeSwap/interfaces/IPancakeRouter02.sol#3) allows
old versions
Pragma version^0.8.0 (contracts/pancakeSwap/interfaces/IWETH.sol#3) allows old versions
Pragma version^0.8.15 (contracts/pancakeSwap/libraries/Math.sol#3) necessitates a
version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.15 (contracts/pancakeSwap/libraries/PancakeLibrary.sol#3)
necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
```

```
Pragma version^0.8.15 (contracts/pancakeSwap/libraries/SafeMath.sol#3) necessitates a
version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.15 (contracts/pancakeSwap/libraries/TransferHelper.sol#3)
necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.15 (contracts/pancakeSwap/libraries/UQ112x112.sol#3) necessitates a
version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (contracts/wc.sol#8) allows old versions
solc-0.8.15 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity

Low level call in PancakePair._safeTransfer(address,address,uint256) (contracts/
pancakeSwap/PancakePair.sol#54-64):
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(contracts/pancakeSwap/PancakePair.sol#59)
Low level call in TransferHelper.safeApprove(address,address,uint256) (contracts/
pancakeSwap/libraries/TransferHelper.sol#7-20):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value))
(contracts/pancakeSwap/libraries/TransferHelper.sol#13-15)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (contracts/
pancakeSwap/libraries/TransferHelper.sol#22-35):
        - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value))
(contracts/pancakeSwap/libraries/TransferHelper.sol#28-30)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256)
(contracts/pancakeSwap/libraries/TransferHelper.sol#37-51):
        - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value))
(contracts/pancakeSwap/libraries/TransferHelper.sol#44-46)
Low level call in TransferHelper.safeTransferETH(address,uint256) (contracts/
pancakeSwap/libraries/TransferHelper.sol#53-56):
        - (success) = to.call{value: value}(new bytes(0)) (contracts/pancakeSwap/
libraries/TransferHelper.sol#54)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

WETH9 (contracts/pancakeSwap/WETH.sol#5-67) should inherit from IWETH (contracts/
pancakeSwap/interfaces/IWETH.sol#5-11)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-
inheritance

Variable PancakeERC20.DOMAIN_SEPARATOR (contracts/pancakeSwap/PancakeERC20.sol#18) is
```

```
not in mixedCase
Parameter PancakeFactory.setFeeTo(address)._feeTo (contracts/pancakeSwap/
PancakeFactory.sol#45) is not in mixedCase
Parameter PancakeFactory.setFeeToSetter(address)._feeToSetter (contracts/pancakeSwap/
PancakeFactory.sol#50) is not in mixedCase
Parameter PancakePair.initialize(address,address)._token0 (contracts/pancakeSwap/
PancakePair.sol#83) is not in mixedCase
Parameter PancakePair.initialize(address,address)._token1 (contracts/pancakeSwap/
PancakePair.sol#83) is not in mixedCase
Variable PancakeRouter.WETH (contracts/pancakeSwap/PancakeRouter.sol#18) is not in
mixedCase
Variable PancakeRouter01.WETH (contracts/pancakeSwap/PancakeRouter01.sol#15) is not in
mixedCase
Function IPancakeERC20.DOMAIN_SEPARATOR() (contracts/pancakeSwap/interfaces/
IPancakeERC20.sol#31) is not in mixedCase
Function IPancakeERC20.PERMIT_TYPEHASH() (contracts/pancakeSwap/interfaces/
IPancakeERC20.sol#33) is not in mixedCase
Function IPancakePair.DOMAIN_SEPARATOR() (contracts/pancakeSwap/interfaces/
IPancakePair.sol#31) is not in mixedCase
Function IPancakePair.PERMIT_TYPEHASH() (contracts/pancakeSwap/interfaces/
IPancakePair.sol#33) is not in mixedCase
Function IPancakePair.MINIMUM_LIQUIDITY() (contracts/pancakeSwap/interfaces/
IPancakePair.sol#59) is not in mixedCase
Function IPancakeRouter01.WETH() (contracts/pancakeSwap/interfaces/
IPancakeRouter01.sol#8) is not in mixedCase
Struct WhalesCandy.memberAuction_overallData (contracts/wc.sol#155-159) is not in
CapWords
Struct WhalesCandy.memberAuction (contracts/wc.sol#164-169) is not in CapWords
Struct WhalesCandy.refData (contracts/wc.sol#174-177) is not in CapWords
Parameter WhalesCandy.setBuyBackPerecent(uint8)._buyBackPerecent (contracts/wc.sol#245)
is not in mixedCase
Parameter WhalesCandy.setTaxFactor(uint8)._taxFactor (contracts/wc.sol#252) is not in
mixedCase
Parameter WhalesCandy.setExcludedFromTaxReceiver(address,bool)._account (contracts/
wc.sol#259) is not in mixedCase
Parameter WhalesCandy.setExcludedFromTaxReceiver(address,bool)._excluded (contracts/
wc.sol#259) is not in mixedCase
Parameter WhalesCandy.isExcludedFromTaxReceiver(address)._account (contracts/
wc.sol#264) is not in mixedCase
Parameter WhalesCandy.stake(uint256)._amount (contracts/wc.sol#328) is not in mixedCase
Parameter WhalesCandy.stakeInt(uint256)._amount (contracts/wc.sol#344) is not in
mixedCase
```

```
Parameter WhalesCandy.refStake(uint256)._amount (contracts/wc.sol#356) is not in
mixedCase
Parameter WhalesCandy.calculateTokenPerShareOnDay(uint256)._day (contracts/wc.sol#571)
is not in mixedCase
Parameter WhalesCandy.claimTokenFromSharesAndStake(uint256,uint256)._day (contracts/
wc.sol#578) is not in mixedCase
Parameter WhalesCandy.claimTokenFromSharesAndStake(uint256,uint256)._referer (contracts/
wc.sol#578) is not in mixedCase
Parameter WhalesCandy.claimRefTokensAndStake(uint256)._day (contracts/wc.sol#604) is
not in mixedCase
Parameter WhalesCandy.calcReward(address,uint256)._user (contracts/wc.sol#625) is not
in mixedCase
Parameter WhalesCandy.calcReward(address,uint256)._stakeIndex (contracts/wc.sol#625) is
not in mixedCase
Parameter WhalesCandy.calcClaim(address,uint256)._user (contracts/wc.sol#644) is not in
mixedCase
Parameter WhalesCandy.calcClaim(address,uint256)._stakeIndex (contracts/wc.sol#644) is
not in mixedCase
Parameter WhalesCandy.calcClaimFee(address,uint256)._user (contracts/wc.sol#660) is not
in mixedCase
Parameter WhalesCandy.calcClaimFee(address,uint256)._stakeIndex (contracts/wc.sol#660)
is not in mixedCase
Parameter WhalesCandy.claimRewards(uint256)._stakeIndex (contracts/wc.sol#682) is not
in mixedCase
Parameter WhalesCandy.reinvest(uint256)._stakeIndex (contracts/wc.sol#700) is not in
mixedCase
Variable WhalesCandy._dev (contracts/wc.sol#69) is not in mixedCase
Variable WhalesCandy._dev1 (contracts/wc.sol#70) is not in mixedCase
Constant WhalesCandy.BUSDaddress (contracts/wc.sol#74) is not in
UPPER_CASE_WITH_UNDERSCORES
Constant WhalesCandy._pancakeRouterAddress (contracts/wc.sol#75) is not in
UPPER_CASE_WITH_UNDERSCORES
Variable WhalesCandy._pancakeRouter (contracts/wc.sol#76) is not in mixedCase
Constant WhalesCandy._pancakeFactoryAddress (contracts/wc.sol#77) is not in
UPPER_CASE_WITH_UNDERSCORES
Variable WhalesCandy._pancakeFactory (contracts/wc.sol#78) is not in mixedCase
Variable WhalesCandy._Balances (contracts/wc.sol#102) is not in mixedCase
Variable WhalesCandy.LAUNCH_TIME (contracts/wc.sol#105) is not in mixedCase
Variable WhalesCandy.BUSDfeePerWeek (contracts/wc.sol#124) is not in mixedCase
Variable WhalesCandy.auctionEntry_allDays (contracts/wc.sol#141) is not in mixedCase
Variable WhalesCandy.mapMemberAuction_overallData (contracts/wc.sol#161) is not in
mixedCase
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions

Reentrancy in WhalesCandy.buyAndStake() (contracts/wc.sol#388-411):
        External calls:
        - _dev.transfer(rawAmount.mul(30).div(100)) (contracts/wc.sol#392)
        External calls sending eth:
        - _dev.transfer(rawAmount.mul(30).div(100)) (contracts/wc.sol#392)
        - _pancakeRouter.swapExactETHForTokens{value: rawAmount.mul(70).div(100)}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#401-406)
        State variables written after the call(s):
        - stakeInt(stakeAmount) (contracts/wc.sol#408)
                - overallStakedToken += _amount (contracts/wc.sol#352)
        - stakeInt(stakeAmount) (contracts/wc.sol#408)
                - stakeNumber[msg.sender] ++ (contracts/wc.sol#351)
        - stakeInt(stakeAmount) (contracts/wc.sol#408)
                - stakes[msg.sender][stakeNumber[msg.sender]].amount = _amount
(contracts/wc.sol#345)
                - stakes[msg.sender][stakeNumber[msg.sender]].stakeTime =
block.timestamp (contracts/wc.sol#346)
                - stakes[msg.sender][stakeNumber[msg.sender]].lastUpdate =
block.timestamp (contracts/wc.sol#347)
                - stakes[msg.sender][stakeNumber[msg.sender]].freeClaiming = false
(contracts/wc.sol#348)
                - stakes[msg.sender][stakeNumber[msg.sender]].claimingStartFee =
BUSDfeePerWeek (contracts/wc.sol#349)
Reentrancy in WhalesCandy.buyShareFromAuction() (contracts/wc.sol#538-568):
        External calls:
        - _dev.transfer(rawAmount.mul(30).div(100)) (contracts/wc.sol#543)
        State variables written after the call(s):
        - currentDay = thisDay() (contracts/wc.sol#546)
Reentrancy in WhalesCandy.buyShareFromAuction() (contracts/wc.sol#538-568):
        External calls:
        - _dev.transfer(rawAmount.mul(30).div(100)) (contracts/wc.sol#543)
        External calls sending eth:
        - _dev.transfer(rawAmount.mul(30).div(100)) (contracts/wc.sol#543)
        - updateDaily() (contracts/wc.sol#549)
                - _pancakeRouter.swapExactETHForTokens{value: ethGain}
(0,path,address(0),block.timestamp + 100) (contracts/wc.sol#523-528)
                - (addedToken,addedEth) = _pancakeRouter.addLiquidityETH{value:
ETHtoAdd}(contrAddr,tokenToAdd,0,0,contrAddr,block.timestamp + 100) (contracts/
wc.sol#437-445)
```

```
                - _pancakeRouter.addLiquidityETH{value: ethBal}
(contrAddr,neededToken,0,0,contrAddr,block.timestamp + 100) (contracts/wc.sol#457-464)
        State variables written after the call(s):
        - updateDaily() (contracts/wc.sol#549)
                - _Balances[contrAddr] = _Balances[contrAddr].sub(amount,Token:
transfer amount exceeds balance) (contracts/wc.sol#269)
                - _Balances[_user] = _Balances[_user].add(_amount) (contracts/
wc.sol#306)
                - _Balances[address(0)] = _Balances[address(0)].add(amount) (contracts/
wc.sol#270)
        - updateDaily() (contracts/wc.sol#549)
                - _allowance[msg.sender][spender] = value (contracts/wc.sol#215)
        - updateDaily() (contracts/wc.sol#549)
                - _totalSupply = _totalSupply.add(_amount) (contracts/wc.sol#307)
        - auctionEntry[currentDay] += rawAmount (contracts/wc.sol#551)
        - auctionEntry_allDays += rawAmount (contracts/wc.sol#552)
        - updateDaily() (contracts/wc.sol#549)
                - currentDay = thisDay() (contracts/wc.sol#477)
        - lastBUYINday = currentDay (contracts/wc.sol#553)
        - updateDaily() (contracts/wc.sol#549)
                - liqAdded[lastBUYINday] = true (contracts/wc.sol#447)
        - liqAdded[currentDay] = false (contracts/wc.sol#554)
        - updateDaily() (contracts/wc.sol#549)
                - lpBal = IERC20(tradingPair).balanceOf(contrAddr) (contracts/
wc.sol#483)
        - mapMemberAuction[msg.sender][currentDay].memberAuctionValue += rawAmount
(contracts/wc.sol#563)
        - mapMemberAuction[msg.sender][currentDay].memberAuctionEntryDay = currentDay
(contracts/wc.sol#564)
        - mapMemberAuction[msg.sender][currentDay].hasChangedShareToToken = false
(contracts/wc.sol#565)
        - mapMemberAuction_overallData[msg.sender].total_auctionEnteries += rawAmount
(contracts/wc.sol#561)
        - updateDaily() (contracts/wc.sol#549)
                - tradingPair =
_pancakeFactory.getPair(_pancakeRouter.WETH(),contrAddr) (contracts/wc.sol#480)
        - updateDaily() (contracts/wc.sol#549)
                - usedETHforBuyBack += ethGain (contracts/wc.sol#516)
        - usersCount ++ (contracts/wc.sol#557)
        - usersCountDaily[currentDay] ++ (contracts/wc.sol#558)
        Event emitted after the call(s):
```

```
        - Approval(msg.sender,spender,value) (contracts/wc.sol#216)
                - updateDaily() (contracts/wc.sol#549)
        - Transfer(contrAddr,address(0),amount) (contracts/wc.sol#271)
                - updateDaily() (contracts/wc.sol#549)
        - Transfer(address(0),_user,_amount) (contracts/wc.sol#308)
                - updateDaily() (contracts/wc.sol#549)
Reentrancy in WETH9.withdraw(uint256) (contracts/pancakeSwap/WETH.sol#27-32):
        External calls:
        - address(msg.sender).transfer(wad) (contracts/pancakeSwap/WETH.sol#30)
        Event emitted after the call(s):
        - Withdrawal(msg.sender,wad) (contracts/pancakeSwap/WETH.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-4

Variable PancakePair.swap(uint256,uint256,address,bytes).balance0Adjusted (contracts/
pancakeSwap/PancakePair.sol#227) is too similar to
PancakePair.swap(uint256,uint256,address,bytes).balance1Adjusted (contracts/pancakeSwap/
PancakePair.sol#228)
Variable PancakePair.price0CumulativeLast (contracts/pancakeSwap/PancakePair.sol#28) is
too similar to PancakePair.price1CumulativeLast (contracts/pancakeSwap/
PancakePair.sol#29)
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,a
ddress,uint256).amountADesired (contracts/pancakeSwap/interfaces/
IPancakeRouter01.sol#13) is too similar to IPancakeRouter01.addLiquidity(address,address
,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/pancakeSwap/
interfaces/IPancakeRouter01.sol#14)
Variable PancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,addr
ess,uint256).amountADesired (contracts/pancakeSwap/PancakeRouter.sol#67) is too similar
to PancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,ui
nt256).amountBDesired (contracts/pancakeSwap/PancakeRouter.sol#68)
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,a
ddress,uint256).amountADesired (contracts/pancakeSwap/interfaces/
IPancakeRouter01.sol#13) is too similar to PancakeRouter._addLiquidity(address,address,u
int256,uint256,uint256,uint256).amountBDesired (contracts/pancakeSwap/
PancakeRouter.sol#39)
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,a
ddress,uint256).amountADesired (contracts/pancakeSwap/interfaces/
IPancakeRouter01.sol#13) is too similar to PancakeRouter.addLiquidity(address,address,ui
nt256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/pancakeSwap/
PancakeRouter.sol#68)
Variable PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).am
```

```
ountADesired (contracts/pancakeSwap/PancakeRouter.sol#38) is too similar to IPancakeRout
er01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amoun
tBDesired (contracts/pancakeSwap/interfaces/IPancakeRouter01.sol#14)
Variable PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).am
ountADesired (contracts/pancakeSwap/PancakeRouter.sol#38) is too similar to PancakeRoute
r._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired
(contracts/pancakeSwap/PancakeRouter.sol#39)
Variable PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).am
ountADesired (contracts/pancakeSwap/PancakeRouter.sol#38) is too similar to PancakeRoute
r.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBD
esired (contracts/pancakeSwap/PancakeRouter.sol#68)
Variable PancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,addr
ess,uint256).amountADesired (contracts/pancakeSwap/PancakeRouter.sol#67) is too similar
to IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address
,uint256).amountBDesired (contracts/pancakeSwap/interfaces/IPancakeRouter01.sol#14)
Variable PancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,addr
ess,uint256).amountADesired (contracts/pancakeSwap/PancakeRouter.sol#67) is too similar
to PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBD
esired (contracts/pancakeSwap/PancakeRouter.sol#39)
Variable PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).am
ountAOptimal (contracts/pancakeSwap/PancakeRouter.sol#56) is too similar to PancakeRoute
r._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBOptimal
(contracts/pancakeSwap/PancakeRouter.sol#51)
Variable PancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,ad
dress,uint256).amountADesired (contracts/pancakeSwap/PancakeRouter01.sol#64) is too
similar to IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256
,address,uint256).amountBDesired (contracts/pancakeSwap/interfaces/
IPancakeRouter01.sol#14)
Variable PancakeRouter01._addLiquidity(address,address,uint256,uint256,uint256,uint256).
amountADesired (contracts/pancakeSwap/PancakeRouter01.sol#35) is too similar to IPancake
Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).a
mountBDesired (contracts/pancakeSwap/interfaces/IPancakeRouter01.sol#14)
Variable PancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,ad
dress,uint256).amountADesired (contracts/pancakeSwap/PancakeRouter01.sol#64) is too
similar to PancakeRouter01._addLiquidity(address,address,uint256,uint256,uint256,uint256
).amountBDesired (contracts/pancakeSwap/PancakeRouter01.sol#36)
Variable PancakeRouter01._addLiquidity(address,address,uint256,uint256,uint256,uint256).
amountADesired (contracts/pancakeSwap/PancakeRouter01.sol#35) is too similar to PancakeR
outer01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired
(contracts/pancakeSwap/PancakeRouter01.sol#36)
Variable PancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,ad
```

```
dress,uint256).amountADesired (contracts/pancakeSwap/PancakeRouter01.sol#64) is too
similar to PancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,
address,uint256).amountBDesired (contracts/pancakeSwap/PancakeRouter01.sol#65)
Variable PancakeRouter01._addLiquidity(address,address,uint256,uint256,uint256,uint256).
amountADesired (contracts/pancakeSwap/PancakeRouter01.sol#35) is too similar to PancakeR
outer01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).am
ountBDesired (contracts/pancakeSwap/PancakeRouter01.sol#65)
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,a
ddress,uint256).amountADesired (contracts/pancakeSwap/interfaces/
IPancakeRouter01.sol#13) is too similar to PancakeRouter01._addLiquidity(address,address
,uint256,uint256,uint256,uint256).amountBDesired (contracts/pancakeSwap/
PancakeRouter01.sol#36)
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,a
ddress,uint256).amountADesired (contracts/pancakeSwap/interfaces/
IPancakeRouter01.sol#13) is too similar to PancakeRouter01.addLiquidity(address,address,
uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/pancakeSwap/
PancakeRouter01.sol#65)
Variable PancakeRouter01._addLiquidity(address,address,uint256,uint256,uint256,uint256).
amountAOptimal (contracts/pancakeSwap/PancakeRouter01.sol#53) is too similar to PancakeR
outer01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBOptimal
(contracts/pancakeSwap/PancakeRouter01.sol#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
are-too-similar

PancakeFactory.createPair(address,address) (contracts/pancakeSwap/
PancakeFactory.sol#28-43) uses literals with too many digits:
        - bytecode = type()(PancakePair).creationCode (contracts/pancakeSwap/
PancakeFactory.sol#33)
PancakeFactory.slitherConstructorConstantVariables() (contracts/pancakeSwap/
PancakeFactory.sol#8-54) uses literals with too many digits:
        - INIT_CODE_PAIR_HASH = keccak256(bytes)(abi.encodePacked(type()
(PancakePair).creationCode)) (contracts/pancakeSwap/PancakeFactory.sol#9-10)
WhalesCandy.updateDaily() (contracts/wc.sol#414-489) uses literals with too many
digits:
        - ethBal > 1000000000000 (contracts/wc.sol#451)
WhalesCandy.slitherConstructorVariables() (contracts/wc.sol#65-735) uses literals with
too many digits:
        - dayliAvailableToken = 200000000000000000000000 (contracts/wc.sol#128)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-
digits
```

```
WETH9.decimals (contracts/pancakeSwap/WETH.sol#8) should be constant
WETH9.name (contracts/pancakeSwap/WETH.sol#6) should be constant
WETH9.symbol (contracts/pancakeSwap/WETH.sol#7) should be constant
WhalesCandy.BUSDfeePerWeek (contracts/wc.sol#124) should be constant
WhalesCandy.dayliAvailableToken (contracts/wc.sol#128) should be constant
WhalesCandy.maxBUSDFee (contracts/wc.sol#125) should be constant
WhalesCandy.oneDay (contracts/wc.sol#106) should be constant
WhalesCandy.oneWeek (contracts/wc.sol#107) should be constant
WhalesCandy.weiPerSfor1perDay (contracts/wc.sol#126) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-constant

quote(uint256,uint256,uint256) should be declared external:
        - PancakeRouter.quote(uint256,uint256,uint256) (contracts/pancakeSwap/
PancakeRouter.sol#516-522)
        - PancakeRouter01.quote(uint256,uint256,uint256) (contracts/pancakeSwap/
PancakeRouter01.sol#365-371)
getAmountOut(uint256,uint256,uint256) should be declared external:
        - PancakeRouter.getAmountOut(uint256,uint256,uint256) (contracts/pancakeSwap/
PancakeRouter.sol#524-530)
        - PancakeRouter01.getAmountOut(uint256,uint256,uint256) (contracts/pancakeSwap/
PancakeRouter01.sol#373-379)
getAmountIn(uint256,uint256,uint256) should be declared external:
        - PancakeRouter.getAmountIn(uint256,uint256,uint256) (contracts/pancakeSwap/
PancakeRouter.sol#532-538)
        - PancakeRouter01.getAmountIn(uint256,uint256,uint256) (contracts/pancakeSwap/
PancakeRouter01.sol#381-387)
getAmountsOut(uint256,address[]) should be declared external:
        - PancakeRouter.getAmountsOut(uint256,address[]) (contracts/pancakeSwap/
PancakeRouter.sol#540-548)
        - PancakeRouter01.getAmountsOut(uint256,address[]) (contracts/pancakeSwap/
PancakeRouter01.sol#389-396)
getAmountsIn(uint256,address[]) should be declared external:
        - PancakeRouter.getAmountsIn(uint256,address[]) (contracts/pancakeSwap/
PancakeRouter.sol#550-558)
        - PancakeRouter01.getAmountsIn(uint256,address[]) (contracts/pancakeSwap/
PancakeRouter01.sol#398-405)
withdraw(uint256) should be declared external:
        - WETH9.withdraw(uint256) (contracts/pancakeSwap/WETH.sol#27-32)
totalSupply() should be declared external:
        - WETH9.totalSupply() (contracts/pancakeSwap/WETH.sol#34-36)
```

```
approve(address,uint256) should be declared external:
        - WETH9.approve(address,uint256) (contracts/pancakeSwap/WETH.sol#38-42)
transfer(address,uint256) should be declared external:
        - WETH9.transfer(address,uint256) (contracts/pancakeSwap/WETH.sol#44-46)
transfer(address,uint256) should be declared external:
        - WhalesCandy.transfer(address,uint256) (contracts/wc.sol#275-278)
stake(uint256) should be declared external:
        - WhalesCandy.stake(uint256) (contracts/wc.sol#328-341)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
```

0x Guard