PACT

COMMUNITY TOKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

April 2021

This document may contain confidential information about IT systems and the customer's intellectual property and information about potential vulnerabilities and exploitation methods. The report contains confidential information. This information can be used internally by the customer. The customer can release the information after fixing all vulnerabilities.

Customer: PACT
Date: 22/04/21
Platform: EVM
Language: Solidity

Table of contents

Introduction

This report presents the Customer`s smart contract's security assessment findings and its code review conducted between PACT 2021 – PACT 2021.

Scope

The scope of the project is PACT smart contract. We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the widely known vulnerabilities that are considered (the full list includes them but does not limit by them):

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference
- Implicit visibility level

Executive Summary

According to the assessment, Customer' smart contracts are well-secured.

Our team performed an analysis of code functionality, manual audit, and automated checks with Slither and remix IDE (see Appendix B pic 1-9). All issues found during automated analysis have been reviewed manually, and application vulnerabilities are presented in the Audit overview section. A general overview is presented in the AS-IS section, and all found issues can be found in the Audit overview section.

We found one low issue in a smart contract which couldn't have any significant security impact.

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also significantly impact smart contract execution, e.g., public access to crucial functions.
Medium	Medium-level vulnerabilities are essential to fix; however, they can't lead to tokens loss.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc., code snippets that can't significantly impact execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored.

PACT AS-IS overview

PACT contract consists of the next smart contracts:

1. **PACT.sol** contracts - Openzeppelin
2. **SafeMath.sol, Ownable.sol, DelegableToken.sol** contracts interfaces
3. **IDelegableERC20.sol** contract interface

PACT.sol contract inherits the class IDelegableERC20, DelegableToken, Ownable.

PACT.sol contract **init** functions:

bridgeRegistration function was called with the following parameters:

- address(newBridge)

bridgeDisable function was called with the following parameters:

- address(oldBridge)

mintByBridge function was called with the following parameters:

- address(account)
- uint(amount)

burnByBridge function was called with the following parameters:

- address(account)
- uint(amount)

mint function was called with the following parameters:

- uint(amount)
- address(account)

burn function was called with the following parameters:

- uint(amount)

bridgesList function was called without parameters.

Audit PACT overview

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

No low severity vulnerabilities were found.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, a high-level description of functionality was presented in the report's As-is overview section.
The audit report contains all found security vulnerabilities and other issues in the reviewed code.
The overall quality of the reviewed contracts is well-secure. Security engineers didn't find any vulnerability, which could have any security impact.

GovernorPACT AS-IS overview

GovernorPACT contract consists of the next smart contracts:

1. **GovernorPACT.sol, AbstractGovernor.sol, GovernanceOwnable.sol** contracts - Openzeppelin
2. **SafeMath.sol**

Contracts from point two were compared to original "Openzeppelin" templates. We found no logic differences. They are considered secure.

GovernorPACT.sol contract inherits the class AbstractGovernor, GovernanceOwnable.

GovernorPACT.sol contract **init** functions:

createDefaultPropose function was called with the following parameters:

- address[](memory targets)
- uint[](memory values)
- string[](memory signatures)
- bytes[](memory calldatas)
- string(memory description)

createFastPropose function was called with the following parameters:

- address[](memory targets)
- uint[](memory values)
- string[](memory signatures)
- bytes[](memory calldatas)
- string(memory description)

createMultiExecutablePropose function was called with the following parameters:

- address[](memory targets)
- uint[](memory values)
- string[](memory signatures)

- bytes[](memory calldatas)
- string(memory description)

addAllowedTarget function was called with the following parameters:

- address(target)

getAllowedTargets function was called without parameters.

Audit GovernorPACT overview

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

No low severity vulnerabilities were found.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, a high-level description of functionality was presented in the report's As-is overview section.
The audit report contains all found security vulnerabilities and other issues in the reviewed code.
The overall quality of the reviewed contracts is well-secure. Security engineers didn't find any vulnerability, which could have any security impact.

MockERC20 AS-IS overview

MockERC20 contract consists of the next smart contracts:

1. **MockERC20.sol** contracts - Openzeppelin
2. **SafeMath.sol, WETH9.sol** contracts - interfaces
3. **ERC20.sol**

MockERC20.sol contract inherits the class IERC20.
MockERC20.sol contract **init** functions:
balanceOf function was called with the following parameters:

- address(account)

transfer function was called with the following parameters:

- address(to)
- uint256(amount)

allowance function was called with the following parameters:

- address(owner)
- address(spender)

approve function was called with the following parameters:

- address(spender)
- uint256(amount)

transferFrom function was called with the following parameters:

- address(from)
- address(to)
- uint256(amount)

mint function was called with the following parameters:

- address(account)

- uint256(amount)

burn function was called with the following parameters:

- address(account)
- uint256(amount)

_transfer function was called with the following parameters:

- address(from)
- address(to)
- uint256(amount)

_mint function was called with the following parameters:

- address(account)
- uint256(amount)

_burn function was called with the following parameters:

- address(account)
- uint256(amount)

_approve function was called with the following parameters:

- address(owner)
- address(spender)
- uint256(amount)

name function was called without parameters.

symbol function was called without parameters.

decimals function was called without parameters.

totalSupply function was called without parameters.

Audit MockERC20 overview

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

No low severity vulnerabilities were found.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, a high-level description of functionality was presented in the report's As-is overview section.
The audit report contains all found security vulnerabilities and other issues in the reviewed code.
The overall quality of the reviewed contracts is well-secure. Security engineers didn't find any vulnerability, which could have any security impact.

PactBasePool AS-IS overview

PactBasePool contract consists of the next smart contracts:

1. **PactBasePool.sol** contracts - Openzeppelin
2. **SafeMath.sol, SafeERC20.sol, TransferHelper.sol, TxStorage.sol, Whitelist.sol** contracts - interfaces
3. **IERC20.sol, IUniswapOracle.sol**

PactBasePool.sol contract inherits the class Whitelist and TxStorage.
PactBasePool.sol contract **init** functions:
buylimitsUpdate function was called with the following parameters:

- uint(minLimit)

changeOracleAddress function was called with the following parameters:

- address(oracleAddress)

calcPriceEthPact function was called with the following parameters:

- uint(amountInEth)

calcPricePactEth function was called with the following parameters:

- uint(amountInPact)

returnToken function was called with the following parameters:

- uint(index)

withdrawEthForExpiredTransaction function was called with the following parameters:

- address(to)

changeEthToToken function was called without parameters.

Audit PactBasePool overview

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

No low severity vulnerabilities were found.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, a high-level description of functionality was presented in the report's As-is overview section.
The audit report contains all found security vulnerabilities and other issues in the reviewed code.
The overall quality of the reviewed contracts is well-secure. Security engineers didn't find any vulnerability, which could have any security impact.

FarmPACT AS-IS overview

FarmPACT contract consists of the next smart contracts:

1. **FarmPACT.sol, AbstractFarm.sol** contracts - Openzeppelin
2. **SafeMath.sol, SafeERC20.sol, GovernanceOwnable.sol, LpTokensStorage.sol, StagesStorage.sol, UsersStorage.sol** contracts interfaces
3. **IDelegableERC20.sol, IUniswapV2Pair.sol, IERC20.sol**

FarmPACT.sol contract inherits the class GovernanceOwnable and AbstractFarm.

FarmPACT.sol contract **init** functions:

startFarming function was called with the following parameters:

- uint256(startBlock)

addLpToken function was called with the following parameters:

- uint256(_allocPoint)
- address(_lpToken)
- bool(_withUpdate)

updateLpToken function was called with the following parameters:

- uint256(poolId)
- uint256(allocPoint)
- bool(withUpdate)

blockGenerationFrequency function was called without parameters.

AbstractFarm.sol contract inherits the class UsersStorage, StagesStorage.

AbstractFarm.sol contract init functions:

updatePool function was called with the following parameters:

- uint256(poolId)

_updatePool function was called with following parameters:

- PoolInfo storage(pool)

_updatePoolInfoInFarmStage function was called with the following parameters:

- StageInfo storage(stage)
- PoolInfo storage(pool)
- uint256(lpSupply)

pending function was called with the following parameters:

- uint256(poolId)
- address(account)

_addLpToken function was called with the following parameters:

- uint256(allocPoint)
- IUniswapV2Pair(lpToken)
- bool(withUpdate)

_updateLpToken function was called with the following parameters:

- uint256(poolId)
- uint256(allocPoint)
- bool(withUpdate)

_beforeBalanceChange function was called with the following parameters:

- PoolInfo storage(pool)
- address(account)

_afterBalanceChange function was called with the following parameters:

- PoolInfo storage(pool)
- address(account)

_updateUserRewardDebtAndPending function was called with the following parameters:

- PoolInfo storage(pool)
- address(account)

harvest function was called with the following parameters:

- uint256(poolId)

massUpdatePools function was called without parameters.

Audit FarmPACT overview

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

1. Unused return. The return value of an external call is not stored in a local or state variable.(see Appendix A pic. 1 for evidence)

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, a high-level description of functionality was presented in the report's As-is overview section.
The audit report contains all found security vulnerabilities and other issues in the reviewed code.
The overall quality of the reviewed contracts is well-secure. Security engineers found one low vulnerability, which would not have security impact.

PactSwapRouter AS-IS overview

PactSwapRouter contract consists of the next smart contracts:

1. **PactSwapRouter.sol, PactSwapFactory.sol** contracts Openzeppelin
2. **SafeMath.sol,RequiredTokenValidator.sol, GovernanceOwnable.sol,TransferHelper.sol, IUniswapV2Factory.sol, PactSwapERC20.sol, PactSwapPair.sol** contracts interfaces
3. **IWithIncentivesPool.sol, IWETH.sol, IPactSwapFactory.sol, IPactSwapRouter.sol**

Description: Based on the code of the Openseppelin libraries, in **PactSwapERC20.sol** and **PactSwapPair.sol** description is required.

PactSwapRouter.sol contract inherits the class IPactSwapRouter.
PactSwapRouter.sol contract **init** functions:
_addLiquidity function was called with the following parameters:

- address(tokenA)
- address(tokenB)
- uint(amountADesired)
- uint(amountBDesired)
- uint(amountAMin)
- uint(amountBMin)

addLiquidity function was called with the following parameters:

- address(tokenA)
- address(tokenB)
- uint(amountADesired)
- uint(amountBDesired)
- uint(amountAMin)
- uint(amountBMin)
- address(to)
- uint(deadline)

addLiquidityETH function was called with the following parameters:

- address(token)
- uint(amountTokenDesired)
- uint(amountTokenMin)
- uint(amountETHMin)
- address(to)
- uint(deadline)

removeLiquidity function was called with the following parameters:

- address(tokenA)
- address(tokenB)
- uint(liquidity)
- uint(amountAMin)
- uint(amountBMin)
- address(to)
- uint(deadline)

removeLiquidityETH function was called with the following parameters:

- address(token)
- uint(liquidity)
- uint(amountTokenMin)
- uint(amountETHMin)
- address(to)
- uint(deadline)

removeLiquidityWithPermit function was called with the following parameters:

- address(tokenA)
- address(tokenB)
- uint(liquidity)
- uint(amountAMin)

- uint(amountBMin)
- address(to)
- uint(deadline)
- bool(approveMax)
- uint8(v)
- bytes32(r)
- bytes32(s)

removeLiquidityETHWithPermit function was called with the following parameters:

- address(token)
- uint(liquidity)
- uint(amountTokenMin)
- uint(amountETHMin)
- address(to)
- uint(deadline)
- bool(approveMax)
- uint8(v)
- bytes32(r)
- bytes32(s)

_swap function was called with the following parameters:

- uint[](memory amounts)
- address[](memory path)
- address(to)

swapExactTokensForTokens function was called with following parameters:

- uint(amountIn)
- uint(amountOutMin)
- address[](calldata path)
- address(to)
- uint(deadline)

swapTokensForExactTokens function was called with following parameters:

- uint(amountOut)
- uint(amountInMax)
- address[](calldata path)
- address(to)
- uint(deadline)

swapExactETHForTokens function was called with following parameters:

- uint(amountOutMin)
- address[](calldata path)
- address(to)
- uint(deadline)

swapTokensForExactETH function was called with following parameters:

- uint(amountOut)
- uint(amountInMax)
- address[](calldata path)
- address(to)
- uint(deadline)

swapExactTokensForETH function was called with following parameters:

- uint(amountIn)
- uint(amountOutMin)
- address[](calldata path)
- address(to)
- uint(deadline)

swapETHForExactTokens function was called with following parameters:

- uint(amountIn)
- address[](calldata path)
- address(to)

- uint(deadline)

getAmountOut function was called with the following parameters:

- uint(amountIn)
- uint(reserveIn)
- uint(reserveOut)

getAmountIn function was called with the following parameters:

- uint(amountOut)
- uint(reserveIn)
- uint(reserveOut)

getAmountsOut function was called with the following parameters:

- uint(amountIn)
- address[](memory path)

getAmountsIn function was called with the following parameters:

- uint(amountOut)
- address[](memory path)

removeIncentivesPoolLiquidity function was called with the following parameters:

- address(token)
- uint(amountTokenMin)
- uint(amountPactMin)
- uint(deadline)

bridgesList function was called without parameters.

Audit PactSwapRouter overview

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

No low severity vulnerabilities were found.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, a high-level description of functionality was presented in the report's As-is overview section.
The audit report contains all found security vulnerabilities and other issues in the reviewed code.
The overall quality of the reviewed contracts is well-secure. Security engineers didn't find any vulnerability, which could have any security impact.

PactSwapFactory AS-IS overview

PactSwapFactory contract inherits the class IUniswapV2Factory, IWithIncentivesPool, GovernanceOwnable, RequiredTokenValidator.

PactSwapFactory.sol contract init functions:

setIncentivesPool function was called with following parameters:

- address(newIncentivesPool)

createPair function was called with following parameters:

- address(tokenA)
- address(tokenB)

pairCodeHash function was called without parameters.

allPairsLength function was called without parameters.

Audit PactSwapFactory overview

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

No low severity vulnerabilities were found.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, a high-level description of functionality was presented in the report's As-is overview section.
The audit report contains all found security vulnerabilities and other issues in the reviewed code.
The overall quality of the reviewed contracts is well-secure. Security engineers didn't find any vulnerability, which could have any security impact.

UniswapOracle AS-IS overview

UniswapOracle contract consists of the next smart contracts:

1. **UniswapOracle.sol** contracts - Openzeppelin
2. **SafeMath.sol, UniswapV2OracleLibrary.sol, FixedPoint.sol, UniswapV2Library.sol** contracts - interfaces
3. **IUniswapOracle.sol**

UniswapOracle.sol contract inherits the class IUniswapOracle.
UniswapOracle.sol contract **init** functions:
observationIndexOf function was called with following parameters:
uint(timestamp)

getFirstObservationInWindow function was called with the following parameters:

- address(pair)

getFirstObservationInWindow2 function was called with the following parameters:

- address(pair)

computeAmountOut function was called with the following parameters:

- uint(priceCumulativeStart)
- uint(priceCumulativeEnd)
- uint(timeElapsed)
- uint(amountIn)

consult function was called with the following parameters:

- address(tokenIn)
- uint(amountIn)
- address(tokenOut)

consultBA function was called with the following parameters:

- uint(amountIn)

getTimeElapsed function was called with the following parameters:

- address(tokenIn)
- address(tokenOut)

update function was called without parameters.

getPair function was called without parameters.

getPrice function was called without parameters.

Audit UniswapOracle overview

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

No low severity vulnerabilities were found.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, a high-level description of functionality was presented in the report's As-is overview section.
The audit report contains all found security vulnerabilities and other issues in the reviewed code.
The overall quality of the reviewed contracts is well-secure. Security engineers didn't find any vulnerability, which could have any security impact.

TeamPoolPACT AS-IS overview

TeamPoolPACT contract consists of the next smart contracts:

1. **TeamPoolPACT.sol** contracts - Openzeppelin
2. **SafeMath.sol, SafeERC20.sol, Ownable.sol** contracts - interfaces
3. **IDelegableERC20.sol, IERC20.sol**

TeamPoolPACT.sol contract inherits the class Ownable.
TeamPoolPACT.sol contract init functions:
withdraw function was called with following parameters:

- address(to)
- uint(amount)

getReleases function was called without parameters.

Audit TeamPoolPACT overview

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

No low severity vulnerabilities were found.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, a high-level description of functionality was presented in the report's As-is overview section.
The audit report contains all found security vulnerabilities and other issues in the reviewed code.
The overall quality of the reviewed contracts is well-secure. Security engineers didn't find any vulnerability, which could have any security impact.

Disclaimers

The smart contracts given for audit had been analyzed following the best industry practices at the date of this report, concerning: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).
The audit makes no statements or warranties on the security of the code. It can also not be considered a sufficient assessment regarding the code's utility and safety, bug-free status, or any other contract statements. While we have done our best to conduct the analysis and produce this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, programming language, and other software related to the smart contract can have their vulnerabilities leading to hacks. Thus, the audit can’t guarantee the explicit security of the audited smart contracts.

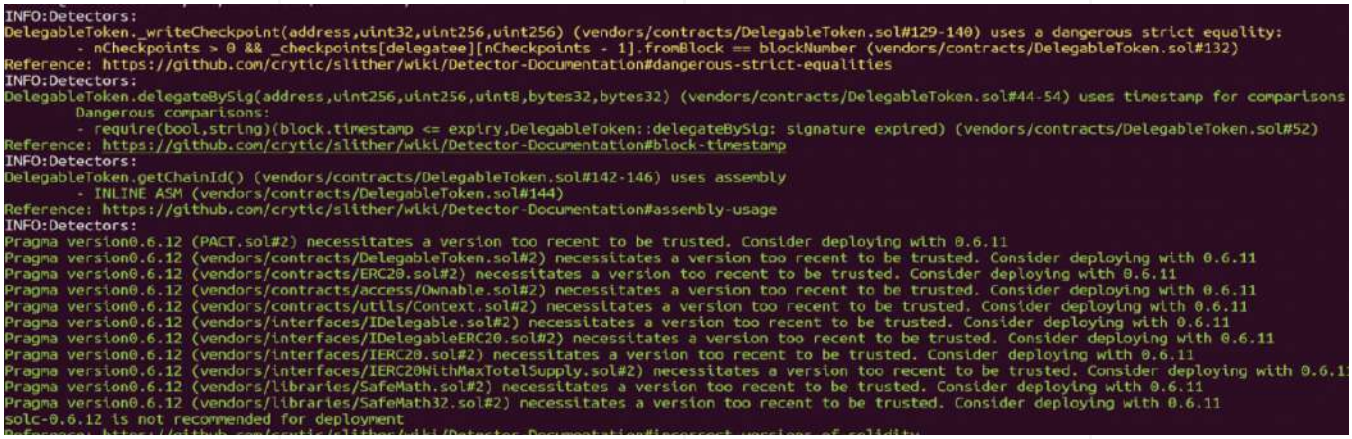
Appendix A. Evidences

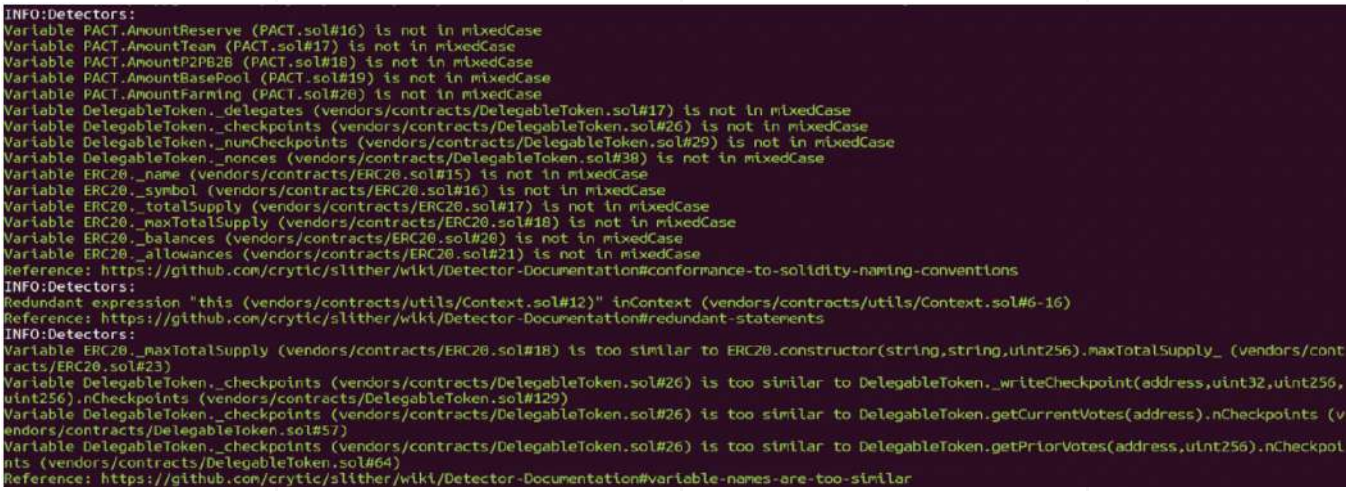
Pic 1. FarmPACT Slither automated report:

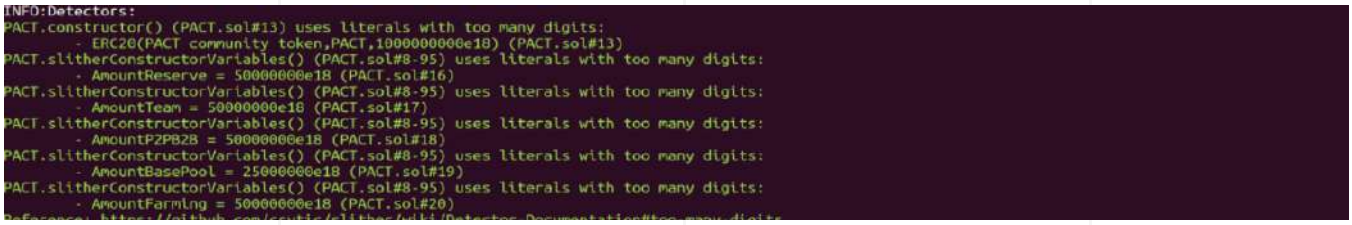
```
INFO:Detectors:
AbstractFarm._updatePoolInfoInFarmStage(StagesStorage.StageInfo,LpTokensStorage.PoolInfo,uint256) (AbstractFarm.sol#57-84) performs a multiplication on
the result of a division:
        -erc20Reward = nrOfBlocks.mul(stage.rewardPerBlock).mul(pool.allocPoint).div(_totalAllocPoint) (AbstractFarm.sol#79)
        -poolInFarmStage.accERC20PerShare = poolInFarmStage.accERC20PerShare.add(erc20Reward.mul(1e36).div(lpSupply)) (AbstractFarm.sol#81)
AbstractFarm.pending(uint256,address) (AbstractFarm.sol#88-121) performs a multiplication on the result of a division:
        -erc20Reward = nrOfBlocks.mul(stage.rewardPerBlock).mul(pool.allocPoint).div(_totalAllocPoint) (AbstractFarm.sol#111)
        -accERC20PerShare = accERC20PerShare.add(erc20Reward.mul(1e36).div(lpSupply)) (AbstractFarm.sol#113)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
StagesStorage._addFirstStage(uint256,uint256,uint256) (StagesStorage.sol#44-56) uses a dangerous strict equality:
        - require(bool,string)(_lastStageEndBlock == 0,_addFirstStage: first stage is already installed) (StagesStorage.sol#49)
AbstractFarm._updatePoolInfoInFarmStage(StagesStorage.StageInfo,LpTokensStorage.PoolInfo,uint256) (AbstractFarm.sol#57-84) uses a dangerous strict equal
ity:
        - lpSupply == 0 (AbstractFarm.sol#73)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in AbstractFarm.harvest(uint256) (AbstractFarm.sol#234-248):
        External calls:
        - _pact.transfer(msg.sender,user.rewardPending) (AbstractFarm.sol#242)
        State variables written after the call(s):
        - _totalRewardPending = _totalRewardPending.sub(user.rewardPending) (AbstractFarm.sol#244)
        - user.rewardPending = 0 (AbstractFarm.sol#247)
Reentrancy in AbstractFarm.withdrawAndHarvest(uint256,uint256) (AbstractFarm.sol#211-232):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),amount) (AbstractFarm.sol#221)
        - _pact.transfer(msg.sender,user.rewardPending) (AbstractFarm.sol#224)
        State variables written after the call(s):
        - _totalRewardPending = _totalRewardPending.sub(user.rewardPending) (AbstractFarm.sol#226)
        - user.rewardPending = 0 (AbstractFarm.sol#229)
        - _afterBalanceChange(pool,msg.sender) (AbstractFarm.sol#231)
                - _userRewardDebt[stage.id][pool.id][account] = user.amount.mul(poolInFarmStage.accERC20PerShare).div(1e36) (AbstractFarm.sol#179)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
AbstractFarm.withdrawAndHarvest(uint256,uint256) (AbstractFarm.sol#211-232) ignores return value by _pact.transfer(msg.sender,user.rewardPending) (Abstr
actFarm.sol#224)
AbstractFarm.harvest(uint256) (AbstractFarm.sol#234-248) ignores return value by _pact.transfer(msg.sender,user.rewardPending) (AbstractFarm.sol#242)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

Appendix B. Automated tools report

Pic 1. PACT Slither automated report:





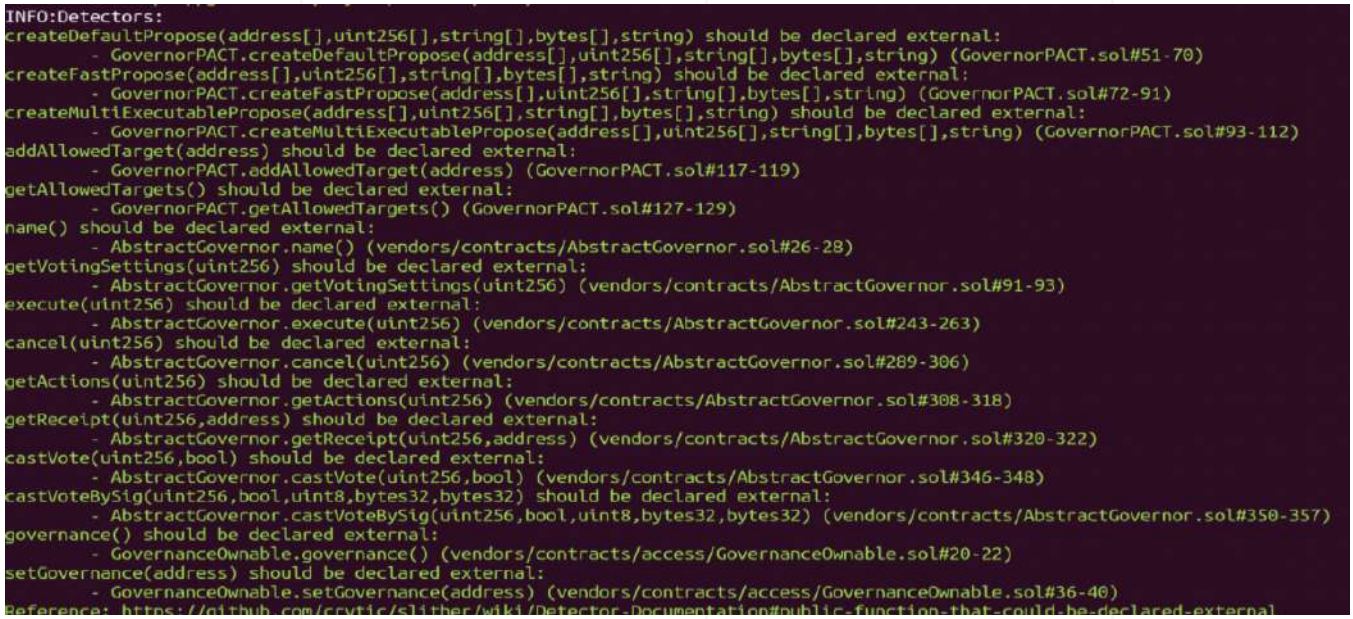


```
INFO:Detectors:
bridgesList() should be declared external:
        - PACT.bridgesList() (PACT.sol#57-59)
bridgeRegistration(address) should be declared external:
        - PACT.bridgeRegistration(address) (PACT.sol#64-69)
bridgeDisable(address) should be declared external:
        - PACT.bridgeDisable(address) (PACT.sol#70-74)
delegate(address) should be declared external:
        - DelegableToken.delegate(address) (vendors/contracts/DelegableToken.sol#40-42)
delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - DelegableToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (vendors/contracts/DelegableToken.sol#44-54)
getPriorVotes(address,uint256) should be declared external:
        - DelegableToken.getPriorVotes(address,uint256) (vendors/contracts/DelegableToken.sol#61-93)
name() should be declared external:
        - ERC20.name() (vendors/contracts/ERC20.sol#29-31)
symbol() should be declared external:
        - ERC20.symbol() (vendors/contracts/ERC20.sol#33-35)
decimals() should be declared external:
        - ERC20.decimals() (vendors/contracts/ERC20.sol#37-39)
totalSupply() should be declared external:
        - ERC20.totalSupply() (vendors/contracts/ERC20.sol#41-43)
maxTotalSupply() should be declared external:
        - ERC20.maxTotalSupply() (vendors/contracts/ERC20.sol#45-47)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (vendors/contracts/ERC20.sol#50-52)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (vendors/contracts/ERC20.sol#54-57)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (vendors/contracts/ERC20.sol#59-61)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (vendors/contracts/ERC20.sol#63-66)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (vendors/contracts/ERC20.sol#68-77)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (vendors/contracts/access/Ownable.sol#44-47)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (vendors/contracts/access/Ownable.sol#53-57)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:./PACT.sol analyzed (11 contracts with 72 detectors), 59 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

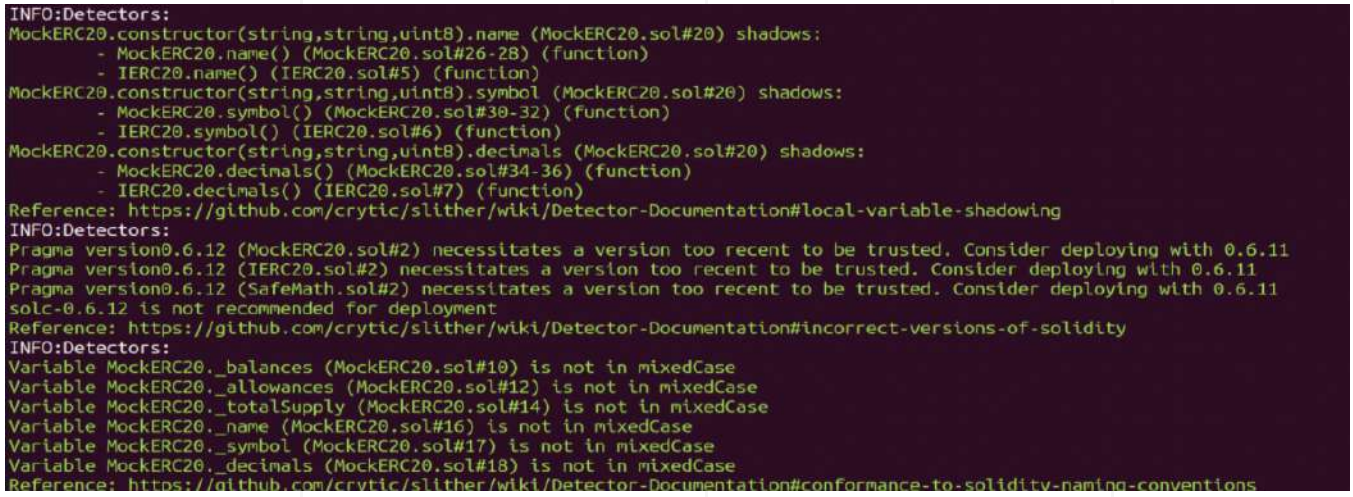
Pic 2. GovernorPACT Slither automated report:

```
GovernorPACT (GovernorPACT.sol#9-132) contract sets array length with a user-controlled value:
        - allowedTargetsList.push(target) (GovernorPACT.sol#123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#array-length-assignment
INFO:Detectors:
AbstractGovernor.cancel(uint256).state (vendors/contracts/AbstractGovernor.sol#290) shadows:
        - AbstractGovernor.state(uint256) (vendors/contracts/AbstractGovernor.sol#324-344) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Reentrancy in AbstractGovernor._executeTransaction(address,uint256,string,bytes) (vendors/contracts/AbstractGovernor.sol#265-287):
        External calls:
        - (success,returnData) = target.call{value: value}(callData) (vendors/contracts/AbstractGovernor.sol#281)
        Event emitted after the call(s):
        - ExecuteTransaction(target,value,signature,data) (vendors/contracts/AbstractGovernor.sol#284)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
AbstractGovernor.getChainId() (vendors/contracts/AbstractGovernor.sol#379-383) uses assembly
        - INLINE ASM (vendors/contracts/AbstractGovernor.sol#381)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
GovernorPACT._addAllowedTarget(address) (GovernorPACT.sol#120-125) compares to a boolean constant:
        -allowedTargets[target] == false (GovernorPACT.sol#121)
AbstractGovernor._setVotingSettings(uint256,uint256,uint256,uint256,uint256,uint256) (vendors/contracts/AbstractGovernor.sol#62-89) compares to a boolea
n constant:
        -require(bool,string)(votingSettings.isValue == false,Governor::setVotingSettings: incorrect votingSettingsId) (vendors/contracts/AbstractGovern
or.sol#71)
AbstractGovernor._castVote(address,uint256,bool) (vendors/contracts/AbstractGovernor.sol#359-377) compares to a boolean constant:
        -require(bool,string)(receipt.hasVoted == false,Governor::_castVote: voter already voted) (vendors/contracts/AbstractGovernor.sol#363)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

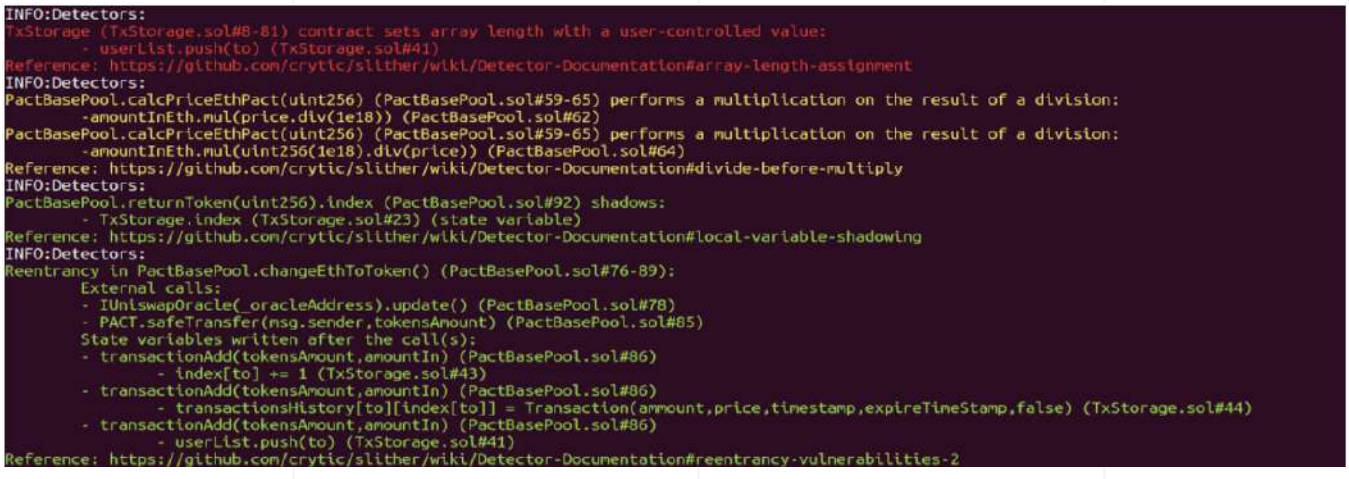
```
INFO:Detectors:
Pragma version0.6.12 (GovernorPACT.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (vendors/contracts/AbstractGovernor.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (vendors/contracts/access/GovernanceOwnable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (vendors/contracts/utils/Context.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (vendors/interfaces/IDelegable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (vendors/interfaces/IDelegableERC20.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (vendors/interfaces/IERC20.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (vendors/interfaces/IERC20WithMaxTotalSupply.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (vendors/interfaces/IGovernanceOwnable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (vendors/libraries/SafeMath.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
solc-0.6.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in AbstractGovernor._executeTransaction(address,uint256,string,bytes) (vendors/contracts/AbstractGovernor.sol#265-287):
        - (success,returnData) = target.call{value: value}(callData) (vendors/contracts/AbstractGovernor.sol#281)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Variable AbstractGovernor._name (vendors/contracts/AbstractGovernor.sol#15) is not in mixedCase
Variable AbstractGovernor._token (vendors/contracts/AbstractGovernor.sol#16) is not in mixedCase
Variable AbstractGovernor._votingSettings (vendors/contracts/AbstractGovernor.sol#31) is not in mixedCase
Variable AbstractGovernor._proposalCount (vendors/contracts/AbstractGovernor.sol#96) is not in mixedCase
Variable AbstractGovernor._proposals (vendors/contracts/AbstractGovernor.sol#152) is not in mixedCase
Variable AbstractGovernor._proposalsExecutionBlocks (vendors/contracts/AbstractGovernor.sol#154) is not in mixedCase
Variable AbstractGovernor._latestProposalIds (vendors/contracts/AbstractGovernor.sol#156) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (vendors/contracts/utils/Context.sol#12)" inContext (vendors/contracts/utils/Context.sol#6-16)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
Redundant expression "this (vendors/contracts/utils/Context.sol#12)" inContext (vendors/contracts/utils/Context.sol#6-16)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```



Pic 3. MockERC20 Slither automated report:



Pic 4. PactBasePool Slither automated report:



```
INFO:Detectors:
Reentrancy in PactBasePool.changeEthToToken() (PactBasePool.sol#76-89):
        External calls:
        - IUniswapOracle(_oracleAddress).update() (PactBasePool.sol#78)
        - PACT.safeTransfer(msg.sender,tokensAmount) (PactBasePool.sol#85)
        Event emitted after the call(s):
        - Deposit(tokensAmount,amountIn) (PactBasePool.sol#88)
Reentrancy in PactBasePool.returnToken(uint256) (PactBasePool.sol#92-105):
        External calls:
        - PACT.safeTransferFrom(msg.sender,amount) (PactBasePool.sol#101)
        - TransferHelper.safeTransferETH(msg.sender,price) (PactBasePool.sol#102)
        Event emitted after the call(s):
        - Withdraw(amount,price) (PactBasePool.sol#104)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
TxStorage.amountOfActualTransactions() (TxStorage.sol#48-57) uses timestamp for comparisons
        Dangerous comparisons:
        - transactionsHistory[userList[i]][a].expireTimeStamp > block.timestamp (TxStorage.sol#51)
TxStorage.checkTrransaction(address,uint256) (TxStorage.sol#69-74) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= transaction.expireTimeStamp,TRANSACTION TIME EXPIRED) (TxStorage.sol#73)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Pragma version0.6.12 (PactBasePool.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (GovernanceOwnable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IERC20.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IGovernanceOwnable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IUniswapOracle.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (SafeERC20.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (SafeMath.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (TransferHelper.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (TxStorage.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (Whitelist.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
solc-0.6.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

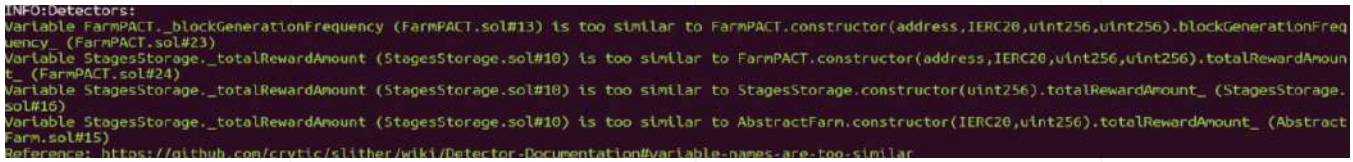
```
INFO:Detectors:
Low level call in SafeERC20.safeSymbol(IERC20) (SafeERC20.sol#7-10):
        - (success,data) = address(token).staticcall(abi.encodeWithSelector(0x95d89b41)) (SafeERC20.sol#8)
Low level call in SafeERC20.safeName(IERC20) (SafeERC20.sol#12-15):
        - (success,data) = address(token).staticcall(abi.encodeWithSelector(0x06fdde03)) (SafeERC20.sol#13)
Low level call in SafeERC20.safeDecimals(IERC20) (SafeERC20.sol#17-20):
        - (success,data) = address(token).staticcall(abi.encodeWithSelector(0x313ce567)) (SafeERC20.sol#18)
Low level call in SafeERC20.safeTransfer(IERC20,address,uint256) (SafeERC20.sol#22-25):
        - (success,data) = address(token).call(abi.encodeWithSelector(0xa9059cbb,to,amount)) (SafeERC20.sol#23)
Low level call in SafeERC20.safeApprove(IERC20,address,uint256) (SafeERC20.sol#27-31):
        - (success,data) = address(token).call(abi.encodeWithSelector(0x095ea7b3,to,value)) (SafeERC20.sol#29)
Low level call in SafeERC20.safeTransferFrom(IERC20,address,uint256) (SafeERC20.sol#33-36):
        - (success,data) = address(token).call(abi.encodeWithSelector(0x23b872dd,from,address(this),amount)) (SafeERC20.sol#34)
Low level call in TransferHelper.safeApprove(address,address,uint256) (TransferHelper.sol#7-11):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (TransferHelper.sol#9)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (TransferHelper.sol#13-17):
        - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (TransferHelper.sol#15)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (TransferHelper.sol#19-23):
        - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (TransferHelper.sol#21)
Low level call in TransferHelper.safeTransferETH(address,uint256) (TransferHelper.sol#25-28):
        - (success) = to.call{value: value}(new bytes(0)) (TransferHelper.sol#26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Variable PactBasePool._oracleAddress (PactBasePool.sol#19) is not in mixedCase
Variable PactBasePool._PACT (PactBasePool.sol#20) is not in mixedCase
Variable PactBasePool._minBuy (PactBasePool.sol#22) is not in mixedCase
Variable PactBasePool._price (PactBasePool.sol#23) is not in mixedCase
Parameter TxStorage.setExparePeriud(uint256)._epirePeriod (TxStorage.sol#26) is not in mixedCase
Parameter TxStorage.closedTransaction(address,uint256)._index (TxStorage.sol#35) is not in mixedCase
Parameter TxStorage.getTransaction(address,uint256)._index (TxStorage.sol#59) is not in mixedCase
Parameter TxStorage.checkTrransaction(address,uint256)._index (TxStorage.sol#69) is not in mixedCase
Parameter Whitelist.whitelistAdd(address)._address (Whitelist.sol#17) is not in mixedCase
Parameter Whitelist.whitelistRemove(address)._address (Whitelist.sol#22) is not in mixedCase
Parameter Whitelist.isWhitelisted(address)._address (Whitelist.sol#27) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable TxStorage.setExparePeriud(uint256)._epirePeriod (TxStorage.sol#26) is too similar to TxStorage.expirePeriod (TxStorage.sol#11)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

```
INFO:Detectors:
buyLimitsUpdate(uint256) should be declared external:
        - PactBasePool.buyLimitsUpdate(uint256) (PactBasePool.sol#45-47)
changeOracleAddress(address) should be declared external:
        - PactBasePool.changeOracleAddress(address) (PactBasePool.sol#50-56)
calcPricePactEth(uint256) should be declared external:
        - PactBasePool.calcPricePactEth(uint256) (PactBasePool.sol#67-73)
changeEthToToken() should be declared external:
        - PactBasePool.changeEthToToken() (PactBasePool.sol#76-89)
withdrawEthForExpiredTransaction(address) should be declared external:
        - PactBasePool.withdrawEthForExpiredTransaction(address) (PactBasePool.sol#108-113)
governance() should be declared external:
        - GovernanceOwnable.governance() (GovernanceOwnable.sol#20-22)
setGovernance(address) should be declared external:
        - GovernanceOwnable.setGovernance(address) (GovernanceOwnable.sol#36-40)
safeDecimals(IERC20) should be declared external:
        - SafeERC20.safeDecimals(IERC20) (SafeERC20.sol#17-20)
whitelistAdd(address) should be declared external:
        - Whitelist.whitelistAdd(address) (Whitelist.sol#17-20)
whitelistRemove(address) should be declared external:
        - Whitelist.whitelistRemove(address) (Whitelist.sol#22-25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

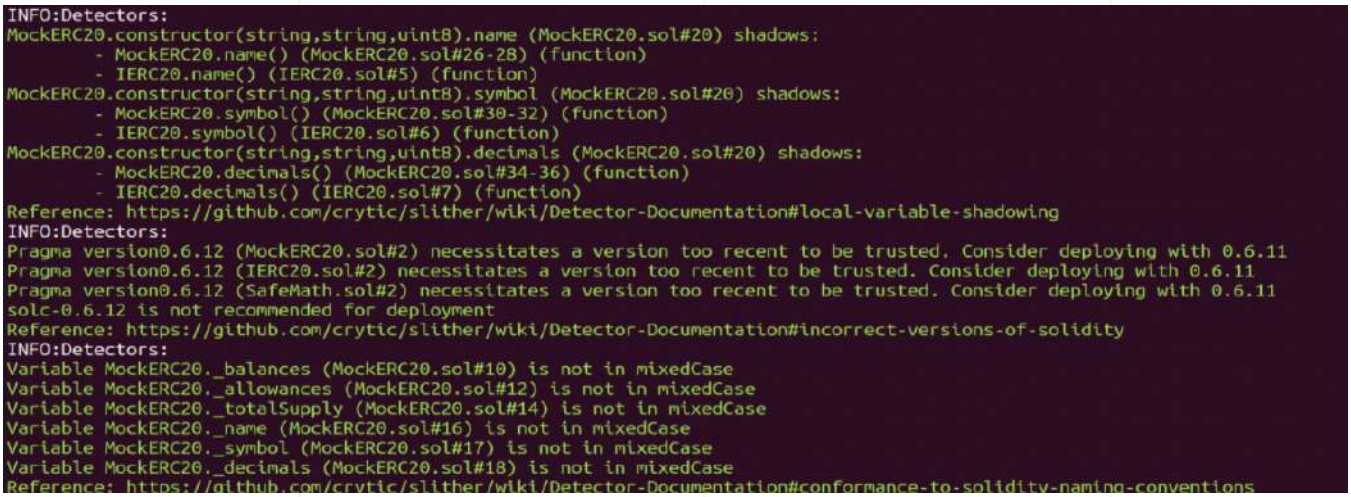
Pic 5. FarmPACT Slither automated report:

```
INFO:Detectors:
Reentrancy in AbstractFarm.harvest(uint256) (AbstractFarm.sol#234-248):
        External calls:
        - _pact.transfer(msg.sender,user.rewardPending) (AbstractFarm.sol#242)
        State variables written after the call(s):
        - _paidOut = _paidOut.add(user.rewardPending) (AbstractFarm.sol#243)
Reentrancy in AbstractFarm.withdrawAndHarvest(uint256,uint256) (AbstractFarm.sol#211-232):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),amount) (AbstractFarm.sol#221)
        - _pact.transfer(msg.sender,user.rewardPending) (AbstractFarm.sol#224)
        State variables written after the call(s):
        - _paidOut = _paidOut.add(user.rewardPending) (AbstractFarm.sol#225)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in UsersStorage.deposit(uint256,uint256) (UsersStorage.sol#26-40):
        External calls:
        - pool.lpToken.safeTransferFrom(address(msg.sender),amount) (UsersStorage.sol#36)
        Event emitted after the call(s):
        - Deposit(msg.sender,poolId,amount) (UsersStorage.sol#37)
Reentrancy in AbstractFarm.harvest(uint256) (AbstractFarm.sol#234-248):
        External calls:
        - _pact.transfer(msg.sender,user.rewardPending) (AbstractFarm.sol#242)
        Event emitted after the call(s):
        - Harvest(msg.sender,poolId,user.rewardPending) (AbstractFarm.sol#246)
Reentrancy in UsersStorage.withdraw(uint256,uint256) (UsersStorage.sol#42-56):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),amount) (UsersStorage.sol#52)
        Event emitted after the call(s):
        - Withdraw(msg.sender,poolId,amount) (UsersStorage.sol#53)
Reentrancy in AbstractFarm.withdrawAndHarvest(uint256,uint256) (AbstractFarm.sol#211-232):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),amount) (AbstractFarm.sol#221)
        Event emitted after the call(s):
        - Withdraw(msg.sender,poolId,amount) (AbstractFarm.sol#222)
Reentrancy in AbstractFarm.withdrawAndHarvest(uint256,uint256) (AbstractFarm.sol#211-232):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),amount) (AbstractFarm.sol#221)
        - _pact.transfer(msg.sender,user.rewardPending) (AbstractFarm.sol#224)
        Event emitted after the call(s):
        - Harvest(msg.sender,poolId,user.rewardPending) (AbstractFarm.sol#228)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
INFO:Detectors:
LpTokensStorage._addLpToken(uint256,IUniswapV2Pair) (LpTokensStorage.sol#46-59) compares to a boolean constant:
        -require(bool,string)(_lpTokensList[address(lpToken)] == false,_addLpToken: LP Token exists) (LpTokensStorage.sol#47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Pragma version0.6.12 (FarmPACT.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (AbstractFarm.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (GovernanceOwnable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IERC20.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IGovernanceOwnable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IUniswapV2ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IUniswapV2Pair.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (LpTokensStorage.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (SafeERC20.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (SafeMath.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (StagesStorage.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (UsersStorage.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
solc-0.6.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in SafeERC20.safeSymbol(IERC20) (SafeERC20.sol#7-10):
        - (success,data) = address(token).staticcall(abi.encodeWithSelector(0x95d89b41)) (SafeERC20.sol#8)
Low level call in SafeERC20.safeName(IERC20) (SafeERC20.sol#12-15):
        - (success,data) = address(token).staticcall(abi.encodeWithSelector(0x06fdde03)) (SafeERC20.sol#13)
Low level call in SafeERC20.safeDecimals(IERC20) (SafeERC20.sol#17-20):
        - (success,data) = address(token).staticcall(abi.encodeWithSelector(0x313ce567)) (SafeERC20.sol#18)
Low level call in SafeERC20.safeTransfer(IERC20,address,uint256) (SafeERC20.sol#22-25):
        - (success,data) = address(token).call(abi.encodeWithSelector(0xa9059cbb,to,amount)) (SafeERC20.sol#23)
Low level call in SafeERC20.safeApprove(IERC20,address,uint256) (SafeERC20.sol#27-31):
        - (success,data) = address(token).call(abi.encodeWithSelector(0x095ea7b3,to,value)) (SafeERC20.sol#29)
Low level call in SafeERC20.safeTransferFrom(IERC20,address,uint256) (SafeERC20.sol#33-36):
        - (success,data) = address(token).call(abi.encodeWithSelector(0x23b872dd,from,address(this),amount)) (SafeERC20.sol#34)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

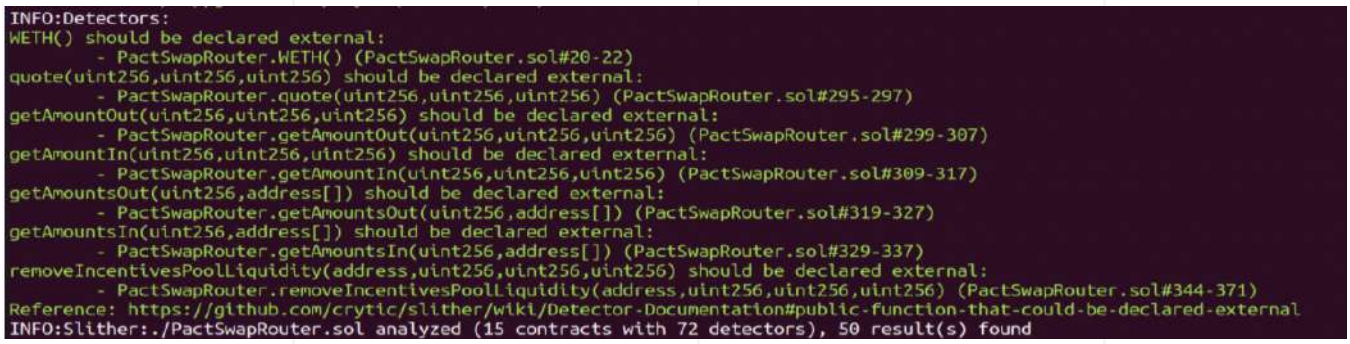


Pic 6. PactSwapRouter Slither automated report:



```
INFO:Detectors:
UniswapV2Library.getAmountsOut(address,uint256,address[]).i (UniswapV2Library.sol#67) is a local variable never initialized
PactSwapRouter._swap(uint256[],address[],address).i (PactSwapRouter.sol#188) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
PactSwapRouter.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256) (PactSwapRouter.sol#114-133) ignores return value by IUniswapV2
air(pair).transferFrom(msg.sender,pair,liquidity) (PactSwapRouter.sol#127)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
PactSwapRouter.constructor(address,address).factory_ (PactSwapRouter.sol#28) lacks a zero-check on :
        - _factory = factory_ (PactSwapRouter.sol#29)
PactSwapRouter.constructor(address,address).WETH_ (PactSwapRouter.sol#28) lacks a zero-check on :
        - _WETH = WETH_ (PactSwapRouter.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
PactSwapRouter._swap(uint256[],address[],address) (PactSwapRouter.sol#187-198) has external calls inside a loop: IUniswapV2Pair(UniswapV2Library.pairFo
(_factory,input,output)).swap(amount0Out,amount1Out,to,new bytes(0)) (PactSwapRouter.sol#194-196)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Pragma version0.6.12 (PactSwapRouter.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IERC20.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IGovernanceOwnable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IPactSwapFactory.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IPactSwapRouter.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IRequiredTokenValidator.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IUniswapV2ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IUniswapV2Factory.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IUniswapV2Pair.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IUniswapV2Router01.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IWETH.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IWithIncentivesPool.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (SafeMath.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (TransferHelper.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (UniswapV2Library.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
solc-0.6.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

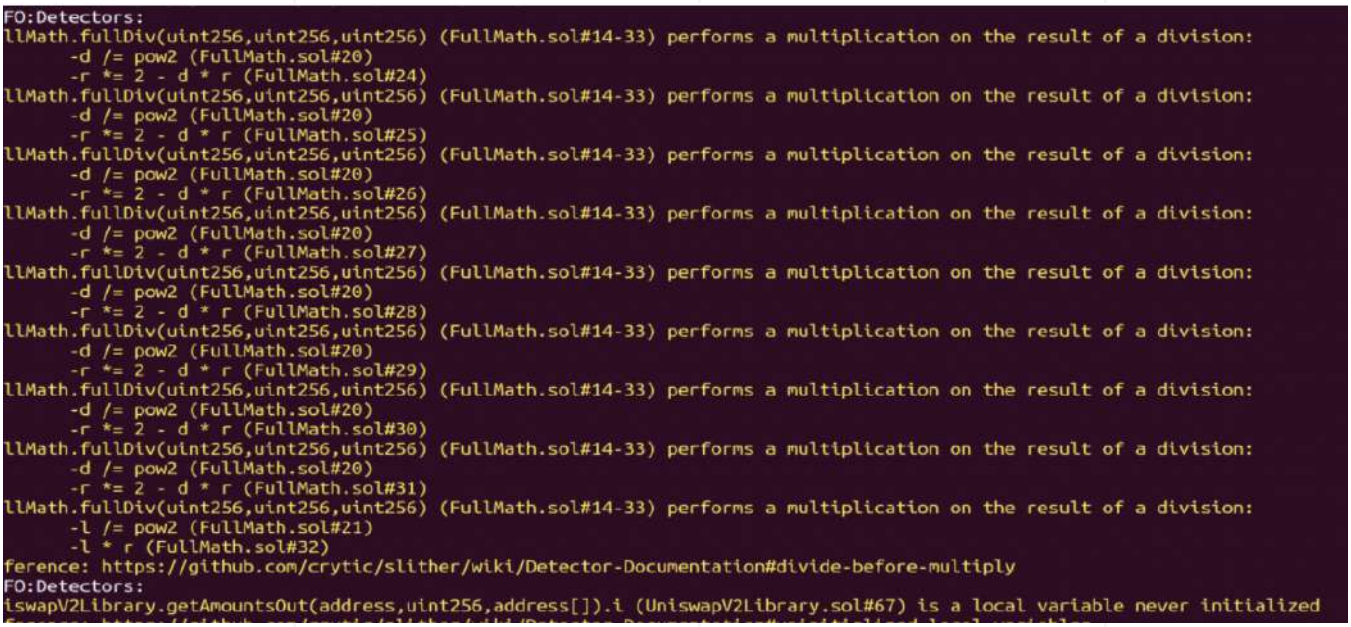
```
INFO:Detectors:
Low level call in TransferHelper.safeApprove(address,address,uint256) (TransferHelper.sol#7-11):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (TransferHelper.sol#9)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (TransferHelper.sol#13-17):
        - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (TransferHelper.sol#15)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (TransferHelper.sol#19-23):
        - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (TransferHelper.sol#21)
Low level call in TransferHelper.safeTransferETH(address,uint256) (TransferHelper.sol#25-28):
        - (success) = to.call{value: value}(new bytes(0)) (TransferHelper.sol#26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function PactSwapRouter.WETH() (PactSwapRouter.sol#20-22) is not in mixedCase
Variable PactSwapRouter._factory (PactSwapRouter.sol#15) is not in mixedCase
Variable PactSwapRouter._WETH (PactSwapRouter.sol#19) is not in mixedCase
Function IUniswapV2ERC20.DOMAIN_SEPARATOR() (IUniswapV2ERC20.sol#8) is not in mixedCase
Function IUniswapV2ERC20.PERMIT_TYPEHASH() (IUniswapV2ERC20.sol#9) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (IUniswapV2Pair.sol#20) is not in mixedCase
Function IUniswapV2Router01.WETH() (IUniswapV2Router01.sol#7) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable PactSwapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PactSwapRouter.sol#74) is too simi
lar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (IUniswapV2Router01.sol#13)
Variable PactSwapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (PactSwapRouter.sol#47) is too similar to PactSwap
Router._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (PactSwapRouter.sol#48)
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (IUniswapV2Router01.sol#12) is
too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (IUniswapV2Router01.sol#1
3)
Variable PactSwapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (PactSwapRouter.sol#47) is too similar to PactSwap
Router.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (PactSwapRouter.sol#75)
Variable PactSwapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PactSwapRouter.sol#74) is too simi
lar to PactSwapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (PactSwapRouter.sol#75)
Variable PactSwapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (PactSwapRouter.sol#47) is too similar to IUniswap
V2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (IUniswapV2Router01.sol#13)
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (IUniswapV2Router01.sol#12) is
too similar to PactSwapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (PactSwapRouter.sol#48)
Variable PactSwapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PactSwapRouter.sol#74) is too simi
lar to PactSwapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (PactSwapRouter.sol#48)
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (IUniswapV2Router01.sol#12) is
too similar to PactSwapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (PactSwapRouter.sol#75)
Variable PactSwapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountAOptimal (PactSwapRouter.sol#64) is too similar to PactSwap
Router._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBOptimal (PactSwapRouter.sol#59)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

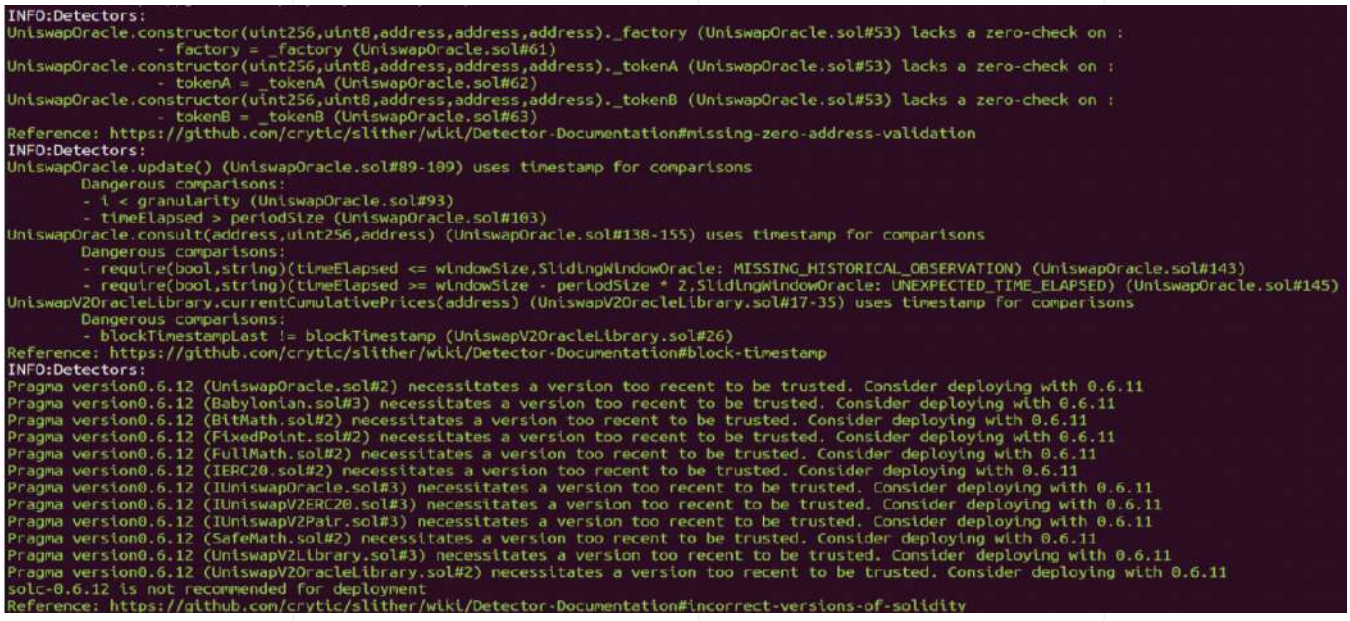


```
INFO:Detectors:
WETH() should be declared external:
        - PactSwapRouter.WETH() (PactSwapRouter.sol#20-22)
quote(uint256,uint256,uint256) should be declared external:
        - PactSwapRouter.quote(uint256,uint256,uint256) (PactSwapRouter.sol#295-297)
getAmountOut(uint256,uint256,uint256) should be declared external:
        - PactSwapRouter.getAmountOut(uint256,uint256,uint256) (PactSwapRouter.sol#299-307)
getAmountIn(uint256,uint256,uint256) should be declared external:
        - PactSwapRouter.getAmountIn(uint256,uint256,uint256) (PactSwapRouter.sol#309-317)
getAmountsOut(uint256,address[]) should be declared external:
        - PactSwapRouter.getAmountsOut(uint256,address[]) (PactSwapRouter.sol#319-327)
getAmountsIn(uint256,address[]) should be declared external:
        - PactSwapRouter.getAmountsIn(uint256,address[]) (PactSwapRouter.sol#329-337)
removeIncentivesPoolLiquidity(address,uint256,uint256,uint256) should be declared external:
        - PactSwapRouter.removeIncentivesPoolLiquidity(address,uint256,uint256,uint256) (PactSwapRouter.sol#344-371)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:./PactSwapRouter.sol analyzed (15 contracts with 72 detectors), 50 result(s) found
```

Pic 7. PactSwapFactory Slither automated report:

Pic 8. UniswapOracle Slither automated report:





```
INFO:Detectors:
Struct FixedPoint.uq112x112 (FixedPoint.sol#12-14) is not in CapWords
Struct FixedPoint.uq144x112 (FixedPoint.sol#18-20) is not in CapWords
Function IUniswapV2ERC20.DOMAIN_SEPARATOR() (IUniswapV2ERC20.sol#8) is not in mixedCase
Function IUniswapV2ERC20.PERMIT_TYPEHASH() (IUniswapV2ERC20.sol#9) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (IUniswapV2Pair.sol#20) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable UniswapOracle.update().price0Cumulative (UniswapOracle.sol#104) is too similar to UniswapOracle.update().price1Cumulative (UniswapOracle.sol#10
4)
Variable UniswapOracle.getPrice().price0Cumulative (UniswapOracle.sol#117) is too similar to UniswapOracle.getPrice().price1Cumulative (UniswapOracle.so
l#117)
Variable UniswapOracle.getPrice().price0Cumulative (UniswapOracle.sol#117) is too similar to UniswapOracle.update().price1Cumulative (UniswapOracle.sol#
104)
Variable UniswapOracle.update().price0Cumulative (UniswapOracle.sol#104) is too similar to UniswapOracle.getPrice().price1Cumulative (UniswapOracle.sol#
117)
Variable UniswapOracle.consult(address,uint256,address).price0Cumulative (UniswapOracle.sol#147) is too similar to UniswapOracle.getPrice().price1Cumula
tive (UniswapOracle.sol#117)
Variable UniswapOracle.getPrice().price0Cumulative (UniswapOracle.sol#117) is too similar to UniswapOracle.consult(address,uint256,address).price1Cumula
tive (UniswapOracle.sol#147)
Variable UniswapOracle.consult(address,uint256,address).price0Cumulative (UniswapOracle.sol#147) is too similar to UniswapOracle.update().price1Cumulati
ve (UniswapOracle.sol#104)
Variable UniswapOracle.update().price0Cumulative (UniswapOracle.sol#104) is too similar to UniswapOracle.consult(address,uint256,address).price1Cumulati
ve (UniswapOracle.sol#147)
Variable UniswapOracle.consult(address,uint256,address).price0Cumulative (UniswapOracle.sol#147) is too similar to UniswapOracle.consult(address,uint256
,address).price1Cumulative (UniswapOracle.sol#147)
Variable UniswapV2OracleLibrary.currentCumulativePrices(address).price0Cumulative (UniswapV2OracleLibrary.sol#19) is too similar to UniswapV2OracleLibra
ry.currentCumulativePrices(address).price1Cumulative (UniswapV2OracleLibrary.sol#19)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

```
INFO:Detectors:
Babylonian.sqrt(uint256) (Babylonian.sol#10-52) uses literals with too many digits:
        - xx >= 0x100000000000000000000000000000000 (Babylonian.sol#16)
Babylonian.sqrt(uint256) (Babylonian.sol#10-52) uses literals with too many digits:
        - xx >= 0x10000000000000000 (Babylonian.sol#20)
Babylonian.sqrt(uint256) (Babylonian.sol#10-52) uses literals with too many digits:
        - xx >= 0x100000000 (Babylonian.sol#24)
BitMath.mostSignificantBit(uint256) (BitMath.sol#7-39) uses literals with too many digits:
        - x >= 0x100000000000000000000000000000000 (BitMath.sol#10)
BitMath.mostSignificantBit(uint256) (BitMath.sol#7-39) uses literals with too many digits:
        - x >= 0x10000000000000000 (BitMath.sol#14)
BitMath.mostSignificantBit(uint256) (BitMath.sol#7-39) uses literals with too many digits:
        - x >= 0x100000000 (BitMath.sol#18)
FixedPoint.slitherConstructorConstantVariables() (FixedPoint.sol#9-146) uses literals with too many digits:
        - Q112 = 0x10000000000000000000000000000000 (FixedPoint.sol#23)
FixedPoint.slitherConstructorConstantVariables() (FixedPoint.sol#9-146) uses literals with too many digits:
        - Q224 = 0x100000000000000000000000000000000000000000000000000000000000000000 (FixedPoint.sol#24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
UniswapOracle.tmp (UniswapOracle.sol#51) is never used in UniswapOracle (UniswapOracle.sol#15-172)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
UniswapOracle.tmp (UniswapOracle.sol#51) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Slither:./UniswapOracle.sol analyzed (12 contracts with 72 detectors), 59 result(s) found
```

```
INFO:Detectors:
Reentrancy in PactSwapPair.burn(address) (PactSwapPair.sol#141-163):
        External calls:
        - _safeTransfer(_token0,to,amount0) (PactSwapPair.sol#155)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PactSwapPair.sol#48)
        - _safeTransfer(_token1,to,amount1) (PactSwapPair.sol#156)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PactSwapPair.sol#48)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (PactSwapPair.sol#160)
                - blockTimestampLast = blockTimestamp (PactSwapPair.sol#87)
        - kLast = uint256(reserve0).mul(reserve1) (PactSwapPair.sol#161)
        - _update(balance0,balance1,_reserve0,_reserve1) (PactSwapPair.sol#160)
                - reserve0 = uint112(balance0) (PactSwapPair.sol#85)
        - _update(balance0,balance1,_reserve0,_reserve1) (PactSwapPair.sol#160)
                - reserve1 = uint112(balance1) (PactSwapPair.sol#86)
Reentrancy in PactSwapFactory.createPair(address,address) (PactSwapFactory.sol#40-58):
        External calls:
        - PactSwapPair(pair).initialize(token0,token1) (PactSwapFactory.sol#53)
        State variables written after the call(s):
        - getPair[token0][token1] = pair (PactSwapFactory.sol#54)
        - getPair[token1][token0] = pair (PactSwapFactory.sol#55)
Reentrancy in PactSwapPair.swap(uint256,uint256,address,bytes) (PactSwapPair.sol#166-194):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (PactSwapPair.sol#177)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PactSwapPair.sol#48)
        - _safeTransfer(_token1,to,amount1Out) (PactSwapPair.sol#178)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PactSwapPair.sol#48)
        - IUniswapV2Callee(to).uniswapV2Call(msg.sender,amount0Out,amount1Out,data) (PactSwapPair.sol#179)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (PactSwapPair.sol#192)
                - blockTimestampLast = blockTimestamp (PactSwapPair.sol#87)
        - _update(balance0,balance1,_reserve0,_reserve1) (PactSwapPair.sol#192)
                - reserve0 = uint112(balance0) (PactSwapPair.sol#85)
        - _update(balance0,balance1,_reserve0,_reserve1) (PactSwapPair.sol#192)
                - reserve1 = uint112(balance1) (PactSwapPair.sol#86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

Pic 9. TeamPoolPACT Slither automated report:

```
INFO:Detectors:
TeamPoolPACT.withdraw(address,uint256).i (TeamPoolPACT.sol#61) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
TeamPoolPACT.withdraw(address,uint256) (TeamPoolPACT.sol#57-69) uses timestamp for comparisons
        Dangerous comparisons:
        - annualSupplyPoints[l][1] >= amount && block.timestamp >= annualSupplyPoints[l][0] (TeamPoolPACT.sol#62)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Pragma version0.6.12 (TeamPoolPACT.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (vendors/contracts/access/Ownable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (vendors/contracts/utils/Context.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (vendors/interfaces/IDelegable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (vendors/interfaces/IDelegableERC20.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (vendors/interfaces/IERC20.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (vendors/interfaces/IERC20WithMaxTotalSupply.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (vendors/libraries/SafeERC20.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (vendors/libraries/SafeMath.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
solc-0.6.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in SafeERC20.safeSymbol(IERC20) (vendors/libraries/SafeERC20.sol#7-10):
        - (success,data) = address(token).staticcall(abi.encodeWithSelector(0x95d89b41)) (vendors/libraries/SafeERC20.sol#8)
Low level call in SafeERC20.safeName(IERC20) (vendors/libraries/SafeERC20.sol#12-15):
        - (success,data) = address(token).staticcall(abi.encodeWithSelector(0x06fdde03)) (vendors/libraries/SafeERC20.sol#13)
Low level call in SafeERC20.safeDecimals(IERC20) (vendors/libraries/SafeERC20.sol#17-20):
        - (success,data) = address(token).staticcall(abi.encodeWithSelector(0x313ce567)) (vendors/libraries/SafeERC20.sol#18)
Low level call in SafeERC20.safeTransfer(IERC20,address,uint256) (vendors/libraries/SafeERC20.sol#22-25):
        - (success,data) = address(token).call(abi.encodeWithSelector(0xa9059cbb,to,amount)) (vendors/libraries/SafeERC20.sol#23)
Low level call in SafeERC20.safeApprove(IERC20,address,uint256) (vendors/libraries/SafeERC20.sol#27-31):
        - (success,data) = address(token).call(abi.encodeWithSelector(0x095ea7b3,to,value)) (vendors/libraries/SafeERC20.sol#29)
Low level call in SafeERC20.safeTransferFrom(IERC20,address,uint256) (vendors/libraries/SafeERC20.sol#33-36):
        - (success,data) = address(token).call(abi.encodeWithSelector(0x23b872dd,from,address(this),amount)) (vendors/libraries/SafeERC20.sol#34)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
Variable TeamPoolPACT._PACT (TeamPoolPACT.sol#20) is not in mixedCase
Constant TeamPoolPACT.oneYear (TeamPoolPACT.sol#21) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (vendors/contracts/utils/Context.sol#12)" inContext (vendors/contracts/utils/Context.sol#6-16)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
TeamPoolPACT.slitherConstructorVariables() (TeamPoolPACT.sol#13-72) uses literals with too many digits:
        - annualSupplyPoints = ((block.timestamp,12500000e18),(block.timestamp.add(oneYear.mul(1)),12500000e18),(block.timestamp.add(oneYear.mul(2)),125
00000e18),(block.timestamp.add(oneYear.mul(3)),12500000e18)) (TeamPoolPACT.sol#23-28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (vendors/contracts/access/Ownable.sol#44-47)
safeDecimals(IERC20) should be declared external:
        - SafeERC20.safeDecimals(IERC20) (vendors/libraries/SafeERC20.sol#17-20)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
INFO:Detectors:
RequiredTokenValidator.constructor(address).requiredToken_ (RequiredTokenValidator.sol#9) lacks a zero-check on :
                - _requiredTokenAddress = requiredToken_ (RequiredTokenValidator.sol#10)
PactSwapFactory.constructor(address,address,address).incentivesPool_ (PactSwapFactory.sol#18) lacks a zero-check on :
                - _incentivesPoolAddress = incentivesPool_ (PactSwapFactory.sol#19)
PactSwapFactory.setIncentivesPool(address).newIncentivesPool (PactSwapFactory.sol#27) lacks a zero-check on :
                - _incentivesPoolAddress = newIncentivesPool (PactSwapFactory.sol#29)
PactSwapPair.initialize(address,address)._token0 (PactSwapPair.sol#69) lacks a zero-check on :
                - token0 = _token0 (PactSwapPair.sol#71)
PactSwapPair.initialize(address,address)._token1 (PactSwapPair.sol#69) lacks a zero-check on :
                - token1 = _token1 (PactSwapPair.sol#72)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in PactSwapPair.burn(address) (PactSwapPair.sol#141-163):
        External calls:
        - _safeTransfer(_token0,to,amount0) (PactSwapPair.sol#155)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PactSwapPair.sol#48)
        - _safeTransfer(_token1,to,amount1) (PactSwapPair.sol#156)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PactSwapPair.sol#48)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (PactSwapPair.sol#160)
                - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (PactSwapPair.sol#82)
        - _update(balance0,balance1,_reserve0,_reserve1) (PactSwapPair.sol#160)
                - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (PactSwapPair.sol#83)
Reentrancy in PactSwapFactory.createPair(address,address) (PactSwapFactory.sol#40-58):
        External calls:
        - PactSwapPair(pair).initialize(token0,token1) (PactSwapFactory.sol#53)
        State variables written after the call(s):
        - allPairs.push(pair) (PactSwapFactory.sol#56)
Reentrancy in PactSwapPair.swap(uint256,uint256,address,bytes) (PactSwapPair.sol#166-194):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (PactSwapPair.sol#177)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PactSwapPair.sol#48)
        - _safeTransfer(_token1,to,amount1Out) (PactSwapPair.sol#178)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PactSwapPair.sol#48)
        - IUniswapV2Callee(to).uniswapV2Call(msg.sender,amount0Out,amount1Out,data) (PactSwapPair.sol#179)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (PactSwapPair.sol#192)
                - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (PactSwapPair.sol#82)
        - _update(balance0,balance1,_reserve0,_reserve1) (PactSwapPair.sol#192)
                - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (PactSwapPair.sol#83)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
INFO:Detectors:
Reentrancy in PactSwapPair.burn(address) (PactSwapPair.sol#141-163):
        External calls:
        - _safeTransfer(_token0,to,amount0) (PactSwapPair.sol#155)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PactSwapPair.sol#48)
        - _safeTransfer(_token1,to,amount1) (PactSwapPair.sol#156)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PactSwapPair.sol#48)
        Event emitted after the call(s):
        - Burn(msg.sender,amount0,amount1,to) (PactSwapPair.sol#162)
        - Sync(reserve0,reserve1) (PactSwapPair.sol#88)
                - _update(balance0,balance1,_reserve0,_reserve1) (PactSwapPair.sol#160)
Reentrancy in PactSwapFactory.createPair(address,address) (PactSwapFactory.sol#40-58):
        External calls:
        - PactSwapPair(pair).initialize(token0,token1) (PactSwapFactory.sol#53)
        Event emitted after the call(s):
        - PairCreated(token0,token1,pair,allPairs.length) (PactSwapFactory.sol#57)
Reentrancy in PactSwapPair.swap(uint256,uint256,address,bytes) (PactSwapPair.sol#166-194):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (PactSwapPair.sol#177)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PactSwapPair.sol#48)
        - _safeTransfer(_token1,to,amount1Out) (PactSwapPair.sol#178)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PactSwapPair.sol#48)
        - IUniswapV2Callee(to).uniswapV2Call(msg.sender,amount0Out,amount1Out,data) (PactSwapPair.sol#179)
        Event emitted after the call(s):
        - Swap(msg.sender,amount0In,amount1In,amount0Out,amount1Out,to) (PactSwapPair.sol#193)
        - Sync(reserve0,reserve1) (PactSwapPair.sol#88)
                - _update(balance0,balance1,_reserve0,_reserve1) (PactSwapPair.sol#192)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
PactSwapERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (PactSwapERC20.sol#83-95) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,UniswapV2: EXPIRED) (PactSwapERC20.sol#84)
PactSwapPair._update(uint256,uint256,uint112,uint112) (PactSwapPair.sol#76-89) uses timestamp for comparisons
        Dangerous comparisons:
        - timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (PactSwapPair.sol#80)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
INFO:Detectors:
PactSwapFactory.createPair(address,address) (PactSwapFactory.sol#40-58) uses assembly
        - INLINE ASM (PactSwapFactory.sol#50-52)
PactSwapERC20.constructor() (PactSwapERC20.sol#26-40) uses assembly
        - INLINE ASM (PactSwapERC20.sol#28-30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Pragma version0.6.12 (PactSwapFactory.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (GovernanceOwnable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IERC20.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IGovernanceOwnable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IPactSwapFactory.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IRequiredTokenValidator.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IUniswapV2Callee.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IUniswapV2ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IUniswapV2Factory.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IUniswapV2Pair.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (IWithIncentivesPool.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (Math.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (PactSwapERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (PactSwapPair.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (RequiredTokenValidator.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (SafeMath.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (UQ112x112.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
solc-0.6.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in PactSwapPair._safeTransfer(address,address,uint256) (PactSwapPair.sol#47-50):
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PactSwapPair.sol#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
PactSwapFactory (PactSwapFactory.sol#11-60) should inherit from IPactSwapFactory (IPactSwapFactory.sol#10-11)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance
INFO:Detectors:
Function IUniswapV2ERC20.DOMAIN_SEPARATOR() (IUniswapV2ERC20.sol#8) is not in mixedCase
Function IUniswapV2ERC20.PERMIT_TYPEHASH() (IUniswapV2ERC20.sol#9) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (IUniswapV2Pair.sol#20) is not in mixedCase
Variable PactSwapERC20.DOMAIN_SEPARATOR (PactSwapERC20.sol#18) is not in mixedCase
Parameter PactSwapPair.initialize(address,address)._token0 (PactSwapPair.sol#69) is not in mixedCase
Parameter PactSwapPair.initialize(address,address)._token1 (PactSwapPair.sol#69) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
INFO:Detectors:
Variable PactSwapPair.swap(uint256,uint256,address,bytes).balance0Adjusted (PactSwapPair.sol#187) is too similar to PactSwapPair.swap(uint256,uint256,ad
dress,bytes).balance1Adjusted (PactSwapPair.sol#188)
Variable PactSwapPair.price0CumulativeLast (PactSwapPair.sol#29) is too similar to PactSwapPair.price1CumulativeLast (PactSwapPair.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
PactSwapFactory.pairCodeHash() (PactSwapFactory.sol#36-38) uses literals with too many digits:
        - keccak256(bytes)(type()(PactSwapPair).creationCode) (PactSwapFactory.sol#37)
PactSwapFactory.createPair(address,address) (PactSwapFactory.sol#40-58) uses literals with too many digits:
        - bytecode = type()(PactSwapPair).creationCode (PactSwapFactory.sol#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
incentivesPool() should be declared external:
        - PactSwapFactory.incentivesPool() (PactSwapFactory.sol#24-26)
setIncentivesPool(address) should be declared external:
        - PactSwapFactory.setIncentivesPool(address) (PactSwapFactory.sol#27-30)
governance() should be declared external:
        - GovernanceOwnable.governance() (GovernanceOwnable.sol#20-22)
setGovernance(address) should be declared external:
        - GovernanceOwnable.setGovernance(address) (GovernanceOwnable.sol#36-40)
requiredToken() should be declared external:
        - RequiredTokenValidator.requiredToken() (RequiredTokenValidator.sol#13-15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:./PactSwapFactory.sol analyzed (17 contracts with 72 detectors), 57 result(s) found
```