QuillAudits

Audit Report
September, 2022

For

Table of Content

Executive Summary 01

Checked Vulnerabilities 03

Techniques and Methods 04

Manual Testing 05

High Severity Issues 05

Medium Severity Issues 05

Low Severity Issues 05

L.1 The Unit test cases are not present 05

L.2 Commented code is present 06

L.3 Owner should be multisig 06

Informational Issues 07

L.4 Gas optimizations and best practice 07

L.5 Missing netspec comments 07

Functional Testing 08

Automated Testing 10

Closing Summary 15

About QuillAudits 16

Executive Summary

Project Name	Pi Protocol Staking
Timeline	17 August, 2022 to 1st September, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyse Pi Protocol Staking codebase for quality, security, and correctness. *https://github.com/Pi-Protocol/PI_BSC_VAULTS/blob/master/contracts/PiStakingVault3.sol* *https://github.com/Pi-Protocol/pi-gamification_staking/blob/master/contracts/PiGamificationStaking.sol*
Fixed in	7f3341ff05110c514fcc5532e5aff8b7558aa579

High
Medium
Low
Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	2	2
Partially Resolved Issues	0	0	1	0
Resolved Issues	0	0	0	0

Types of Severities

High

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

High Severity Issues

No issues found

Medium Severity Issues

No issues found

Low Severity Issues

No issues found

L.1 The Unit test cases are not present

Description

Unit Test cases for the code are missing. We recommend to make unit test with 100% coverage. It’s good practice to cover user based scenarios in these type of tests

Status

Acknowledged

L.2 Commented code is present

Contract

PiGamificationStaking.sol
Commented code is present in the code. We recommend to remove it before finalizing the code.

Status

Resolved

L.3 Owner should be multisig

Description

We recommend multisig account address (gnosis-safe) for owner such that the onlyOwner functionalities are not compromised and the decentralization is achieved in the system

Status

Acknowledged

Informational Issues

L.4 Gas optimizations and best practice

Contract: PiGamificationStaking.sol

Description

Safe Math for version greater than 0.8 solc version are inbuilt and no need to use explicitly.

Remediation

We recommend to remove the safeMath library and it’s operation. Whenever it’s not needed then just use the uncheck flag.

The pre-increment operation is cheaper (about 5 GAS per iteration) so use ++i instead of i++ or i+= 1 in for loop. We recommend to use pre-increment in all the for loops.

In for loop the default value initialization to 0 should not be there remove from all the for loop
When the state gets updated the event should always get fired. We recommend to fire the events for all the state changes.

All internal function names should get start with _ as a good practice to include.
It’s always recommended to follow CEI pattern.

Status

Acknowledged

L. 5 Missing netspec comments

Remediation

We recommend adding netspec comments for each method and variables for better readability and understanding of code.

Status

Acknowledged

Functional Testing

Contracts

PiGamificationStaking - 0x3Abc9D47065CD9F0297022210881aF59277851E2
PiStakingVault3.sol - 0x94Fba1BB8D7E813a9C5196A3ee4A6c936A856E3A
ERC20 Test Rewarder Token - 0x4ed8071980224153075465 11055477E047e4764f
ERC721 Test Staking Token - 0x18159778F26EC53A8B1F22C89D679313061f5CBB
BLL test contract - 0x2dC8a4E452EB01d7c225a5009a7C0aEAB1D71466

Transactions

Initilize Gamification staking - 0xda4940356b0cc078dc590edeb819534f72d1705eb30fac359192b0aa39fea0a3
aproveAll - 0x3537b383aee87635d02d820858c3ab09082246c240eec46a4469b8682b493790
Initilize 721 mock token - 0x7cc85fdcb5db69c5f8c9d432c95849225d66cf86e48482419d4273cf5e4585a4
Initilize bll contract - 0xff346babdb4c2ddba84a359a90487608976d76efef779c47987495010bbd11c0
Stake 3 nft's with point's per token to be 0 - 0x6058faedcc1344577da63fe3885c4ddc7a0e7fe472f16f0a5bff13ece9f9633a
Withdraw 3 nft's with point's per token to be 0 - 0x99ab2d10f4c9f86aef409f3fb945f8616fa9ed18798422eaf5dde0f889ddc7b2
Stake 3 nft's with points per token to be id * 100 - 0x233568a04eab0b1313448c45ec4bc566731dc10f2946f3a13a91e6f62bd87667
Change Factor of BLL to 50 - 0x45ebdae8caf95d83c06c90dde06ae724a24ec975ba7d08cfe7c75c1aa401f07b
updateAllPoints - 0x52393d36095e445bdd84c3483cdcfa7fdf20c9064c6d316e685be81c748b716f
Withdraw 2 nft's - 0x4894673aa5f500977c31e35b0c3a5029a7462b961d57b715be888cd3ec6ca03f
reWithdraw not allowed - 0x7dbbbd9a687e861f7f6433747d58260e0a07689e2593dc34c75ca81c9e5efd3f
Refresh All Nft's - 0x75a7e4d4392bdac9ff28d95fdc447f75884a895a9f8aa6c57d329d2b49ec687a
Refresh Token Id's - 0x5e8ffec5fc352b7aca51fa57547314deb0457df9aa40033facd64d4492b71d6f
Get Reward - 0xd2eda97d4dbc7f7bba373c86ace04697bca94ac6b70fd5db93e3f95477af3a89

Functional Testing

Exit - 0xd19084ad7c061108d5a28884769b08925cdf8e91781822bb08c0811927a11742
Initilize Staking v3 -
0x09d67f67801c1514be4dc5c9bfa5aca80ab9a6fcf239a9931dbb36ef4e18ae65
Approve all to staking v3 -
0x317309ca452fcf0e360d2f7ca7fb99da77aa258307d113702f70a14d805f23e6
Stake 1 nft -
0xceb6ba92df9699ed7329720944ccf9ee84054c8e2f03ef050d0bc2642829bbe9
Notify reward amount to 10 -
0x7a9e55aa963caa97aba1d7a56462ad44bbde1fc06acac232dc041a164ff86889
Get reward -
0x252737cf2625d1fb5751b76e7411c8c0a565e1ebde8abac9299e6b8441fc0587
Exit - 0x09c6d3c041ef8818a37012fe1728bfd2ccc7f1a930fd4e5a751c45cf11899c9c

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity

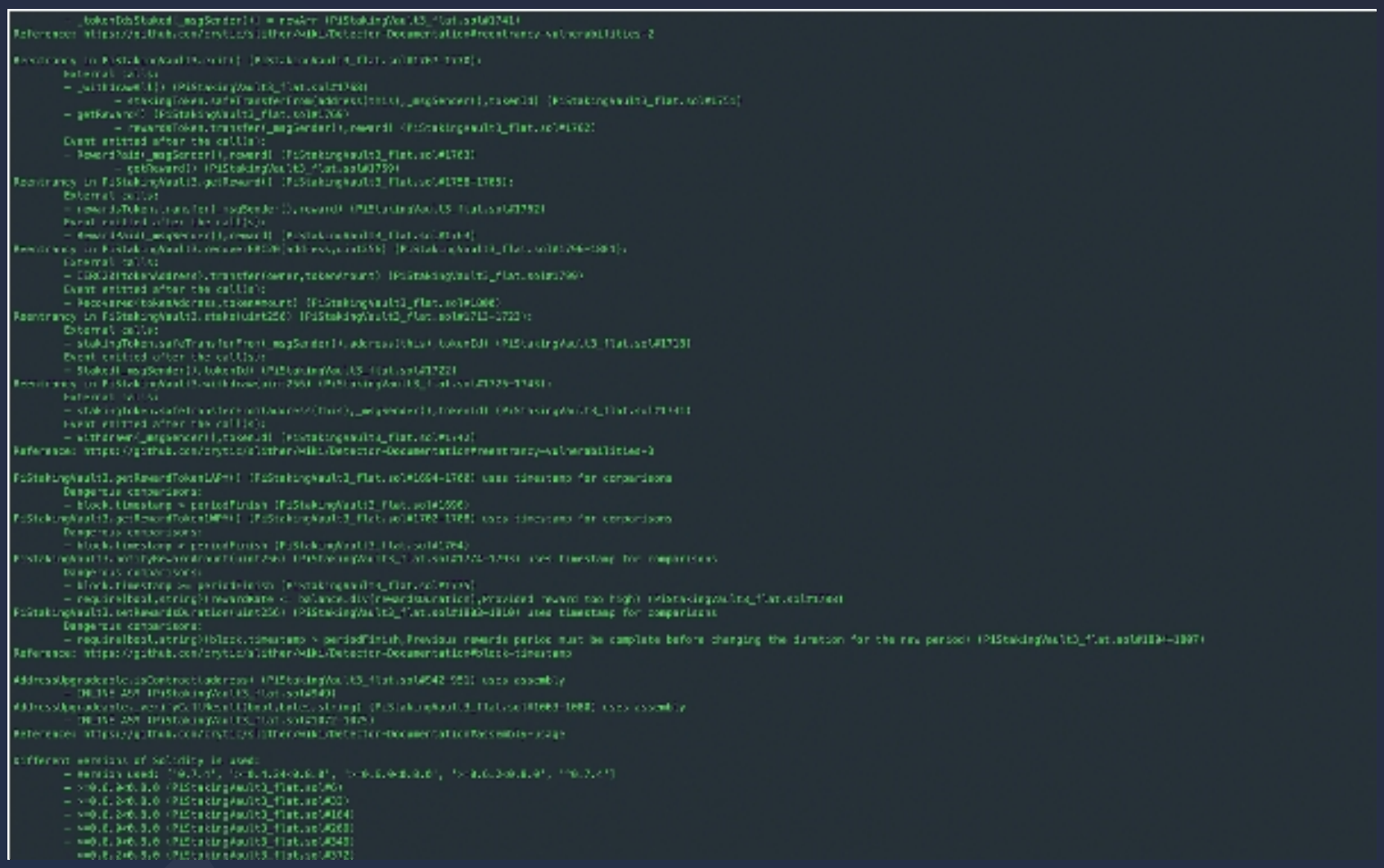
```
OwnableUpgradeable.__gap (PiGamificationStaking_flat.sol#1302) shadows:
        - ContextUpgradeable.__gap (PiGamificationStaking_flat.sol#1285)
PausableUpgradeable.__gap (PiGamificationStaking_flat.sol#2527) shadows:
        - ContextUpgradeable.__gap (PiGamificationStaking_flat.sol#1285)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

PiGamificationStaking.getReward() (PiGamificationStaking_flat.sol#1924-1931) ignores return value by rewardsToken.transfer(_msgSender(),reward) (PiGamificationStaking_flat.sol#1928)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

MathUpgradeable.mul$div(uint256,uint256,uint256) (PiGamificationStaking_flat.sol#389-403) performs a multiplication on the result of a division:
        -denominator = denominator / twos (PiGamificationStaking_flat.sol#436)
        -inverse = (3 * denominator) ^ 2 (PiGamificationStaking_flat.sol#451)
MathUpgradeable.mul$div(uint256,uint256,uint256) (PiGamificationStaking_flat.sol#389-403) performs a multiplication on the result of a division:
        -denominator = denominator / twos (PiGamificationStaking_flat.sol#436)
        -inverse *= 2 - denominator * inverse (PiGamificationStaking_flat.sol#455)
MathUpgradeable.mul$div(uint256,uint256,uint256) (PiGamificationStaking_flat.sol#389-403) performs a multiplication on the result of a division:
        -denominator = denominator / twos (PiGamificationStaking_flat.sol#436)
        -inverse *= 2 - denominator * inverse (PiGamificationStaking_flat.sol#456)
MathUpgradeable.mul$div(uint256,uint256,uint256) (PiGamificationStaking_flat.sol#389-403) performs a multiplication on the result of a division:
        -denominator = denominator / twos (PiGamificationStaking_flat.sol#436)
        -inverse *= 2 - denominator * inverse (PiGamificationStaking_flat.sol#457)
MathUpgradeable.mul$div(uint256,uint256,uint256) (PiGamificationStaking_flat.sol#389-403) performs a multiplication on the result of a division:
        -denominator = denominator / twos (PiGamificationStaking_flat.sol#436)
        -inverse *= 2 - denominator * inverse (PiGamificationStaking_flat.sol#458)
MathUpgradeable.mul$div(uint256,uint256,uint256) (PiGamificationStaking_flat.sol#389-403) performs a multiplication on the result of a division:
        -denominator = denominator / twos (PiGamificationStaking_flat.sol#436)
        -inverse *= 2 - denominator * inverse (PiGamificationStaking_flat.sol#459)
MathUpgradeable.mul$div(uint256,uint256,uint256) (PiGamificationStaking_flat.sol#389-403) performs a multiplication on the result of a division:
        -denominator = denominator / twos (PiGamificationStaking_flat.sol#436)
        -inverse *= 2 - denominator * inverse (PiGamificationStaking_flat.sol#460)
MathUpgradeable.mul$div(uint256,uint256,uint256) (PiGamificationStaking_flat.sol#389-403) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (PiGamificationStaking_flat.sol#439)
        -result = prod0 * inverse (PiGamificationStaking_flat.sol#466)
PiGamificationStaking.notifyRewardAmount(uint256) (PiGamificationStaking_flat.sol#1940-1959) performs a multiplication on the result of a division:
        -rewardRate = reward.div(rewardsDuration) (PiGamificationStaking_flat.sol#1942)
        -leftover = remaining.mul(rewardRate) (PiGamificationStaking_flat.sol#1945)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Reentrancy in PiGamificationStaking.exit() (PiGamificationStaking_flat.sol#1955-1956):
        External calls:
        - _withdrawAll() (PiGamificationStaking_flat.sol#2994)
                - stakingToken.safeTransferFrom(address(this),_msgSender(),tokenId) (PiGamificationStaking_flat.sol#1934)
        - getReward() (PiGamificationStaking_flat.sol#1935)
                - rewardsToken.transfer(_msgSender(),reward) (PiGamificationStaking_flat.sol#1928)
        State variables written after the call(s):
        - getReward() (PiGamificationStaking_flat.sol#1935)
                - lastUpdateTime = lastTimeRewardApplicable() (PiGamificationStaking_flat.sol#1902)
        - getReward() (PiGamificationStaking_flat.sol#1935)
                - rewardPerTokenStored = rewardPerToken() (PiGamificationStaking_flat.sol#1901)
        - getReward() (PiGamificationStaking_flat.sol#1935)
                - rewards[_msgSender()] = 0 (PiGamificationStaking_flat.sol#1927)
                - rewards[account] = earned(account) (PiGamificationStaking_flat.sol#1904)
        - getReward() (PiGamificationStaking_flat.sol#1935)
                - userRewardPerTokenPaid[account] = rewardPerTokenStored (PiGamificationStaking_flat.sol#1905)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

PiGamificationStaking.tokenIdsStaked(address).newArr (PiGamificationStaking_flat.sol#1750) is a local variable never initialized
PiGamificationStaking._withdrawAll().newArr (PiGamificationStaking_flat.sol#1917) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
```

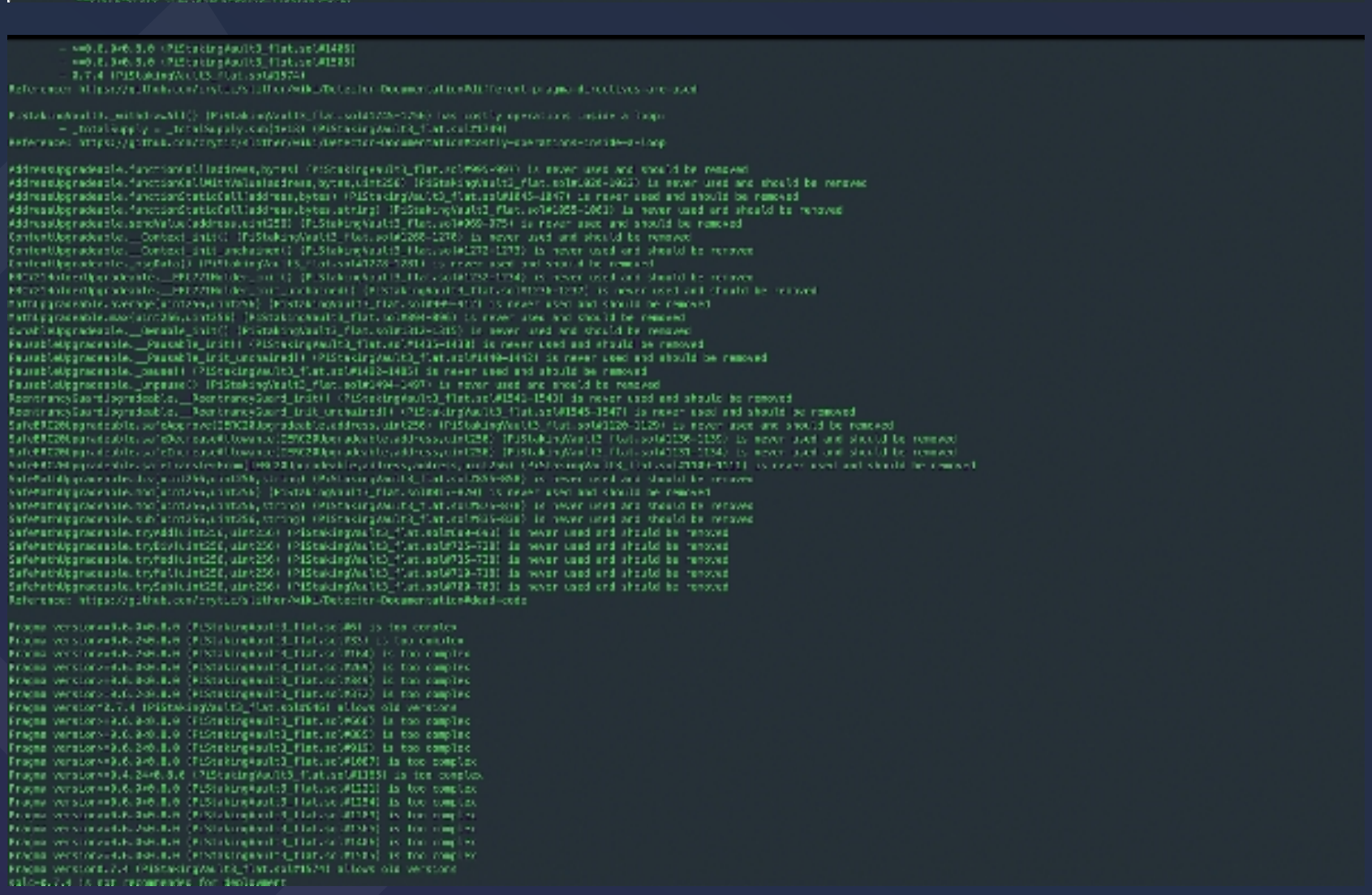
```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

RewardsDistributionRecipient.setRewardsDistribution(address) (PiGamificationStaking_flat.sol#1834-1837) should emit an event for:
        - rewardsDistribution = _rewardsDistribution (PiGamificationStaking_flat.sol#1836)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

PiGamificationStaking.initialize(address,address,address,address)._rewardsDistribution (PiGamificationStaking_flat.sol#1862) lacks a zero-check on :
        - rewardsDistribution = _rewardsDistribution (PiGamificationStaking_flat.sol#1891)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

PiGamificationStaking.stake(uint32[]) (PiGamificationStaking_flat.sol#1771-1793) has external calls inside a loop: value = BLSContract.getPointsForTokenID(tokenIds[i]).mul(1e18) (PiGamificationStaking_flat.sol#1775
)
PiGamificationStaking.stake(uint32[]) (PiGamificationStaking_flat.sol#1771-1793) has external calls inside a loop: stakingToken.safeTransferFrom(_msgSender(),address(this),tokenIds[i]) (PiGamificationStaking_flat.s
ol#1778)
PiGamificationStaking.updatePoints(uint32[]) (PiGamificationStaking_flat.sol#1796-1817) has external calls inside a loop: value = BLSContract.getPointsForTokenID(tokenId).mul(1e18) (PiGamificationStaking_flat.sol#1
805)
PiGamificationStaking.withdraw(uint32[]) (PiGamificationStaking_flat.sol#1856-1883) has external calls inside a loop: stakingToken.safeTransferFrom(address(this),_msgSender(),tokenIds[i]) (PiGamificationStaking_fla
t.sol#1865)
PiGamificationStaking._withdrawAll() (PiGamificationStaking_flat.sol#1907-1922) has external calls inside a loop: stakingToken.safeTransferFrom(address(this),_msgSender(),tokenId) (PiGamificationStaking_flat.sol#19
14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in PiGamificationStaking._withdrawAll() (PiGamificationStaking_flat.sol#1907-1922):
        External calls:
        - stakingToken.safeTransferFrom(address(this),_msgSender(),tokenId) (PiGamificationStaking_flat.sol#1914)
        State variables written after the call(s):
        - isStaked[tokenId] = false (PiGamificationStaking_flat.sol#1915)
Reentrancy in PiGamificationStaking.stake(uint32[]) (PiGamificationStaking_flat.sol#1771-1793):
        External calls:
        - stakingToken.safeTransferFrom(_msgSender(),address(this),tokenIds[i]) (PiGamificationStaking_flat.sol#1778)
        State variables written after the call(s):
        - _tokenIdsStaked[_msgSender()].push(tokenIds[i]) (PiGamificationStaking_flat.sol#1781)
        - allNFTIds.push(tokenIds[i]) (PiGamificationStaking_flat.sol#1786)
        - isStaked[tokenIds[i]] = true (PiGamificationStaking_flat.sol#1789)
        - ownerOfNFTTokenIds[i] = _msgSender() (PiGamificationStaking_flat.sol#1788)
Reentrancy in PiGamificationStaking.withdraw(uint32[]) (PiGamificationStaking_flat.sol#1856-1883):
        External calls:
        - stakingToken.safeTransferFrom(address(this),_msgSender(),tokenIds[i]) (PiGamificationStaking_flat.sol#1865)
        State variables written after the call(s):
        - isStaked[tokenIds[i]] = false (PiGamificationStaking_flat.sol#1880)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in PiGamificationStaking.exit() (PiGamificationStaking_flat.sol#1933-1936):
        External calls:
        - _withdrawAll() (PiGamificationStaking_flat.sol#1934)
                - stakingToken.safeTransferFrom(address(this),_msgSender(),tokenId) (PiGamificationStaking_flat.sol#1914)
        - getReward() (PiGamificationStaking_flat.sol#1935)
                - rewardsToken.transfer(_msgSender(),reward) (PiGamificationStaking_flat.sol#1928)
        Event emitted after the call(s):
        - RewardPaid(_msgSender(),reward) (PiGamificationStaking_flat.sol#1929)
                - getReward() (PiGamificationStaking_flat.sol#1935)
Reentrancy in PiGamificationStaking.getReward() (PiGamificationStaking_flat.sol#1924-1931):
        External calls:
        - rewardsToken.transfer(_msgSender(),reward) (PiGamificationStaking_flat.sol#1928)
        Event emitted after the call(s):
        - RewardPaid(_msgSender(),reward) (PiGamificationStaking_flat.sol#1929)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

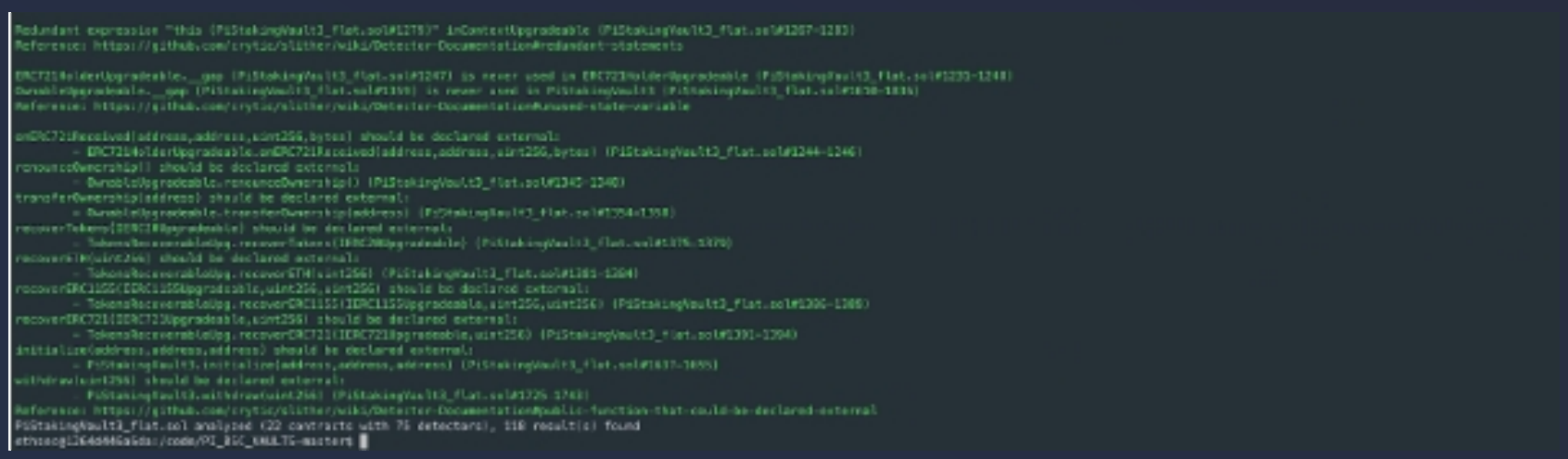
```
PiGamificationStaking.getRewardTokenIAPY() (PiGamificationStaking_flat.sol#1741-1747) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp < periodFinish (PiGamificationStaking_flat.sol#1743)
PiGamificationStaking.getRewardTokenIAPY() (PiGamificationStaking_flat.sol#1749-1755) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp < periodFinish (PiGamificationStaking_flat.sol#1751)
PiGamificationStaking.notifyRewardAmount(uint256) (PiGamificationStaking_flat.sol#1948-1960) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= periodFinish (PiGamificationStaking_flat.sol#1949)
        - require(bool,string)(rewardRate <= balance.div(rewardsDuration),Provided reward too high) (PiGamificationStaking_flat.sol#1956)
PiGamificationStaking.setRewardsDuration(uint256) (PiGamificationStaking_flat.sol#1963-1968) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > periodFinish,Previous rewards period must be complete before changing the duration for the new period) (PiGamificationStaking_flat.sol#1963-1965)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

MathUpgradeable.mulDiv(uint256,uint256,uint256) (PiGamificationStaking_flat.sol#389-408) uses assembly
        - INLINE ASM (PiGamificationStaking_flat.sol#400-402)
        - INLINE ASM (PiGamificationStaking_flat.sol#410-427)
        - INLINE ASM (PiGamificationStaking_flat.sol#434-445)
AddressUpgradeable.verifyCallResultFromTarget(address,bool,bytes,string) (PiGamificationStaking_flat.sol#967-987) uses assembly
        - INLINE ASM (PiGamificationStaking_flat.sol#979-982)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.2', '^0.8.0', '^0.8.1', '^0.8.2']
        - 0.8.2 (PiGamificationStaking_flat.sol#5)
        - ^0.8.0 (PiGamificationStaking_flat.sol#617)
        - ^0.8.0 (PiGamificationStaking_flat.sol#645)
        - ^0.8.0 (PiGamificationStaking_flat.sol#690)
        - ^0.8.0 (PiGamificationStaking_flat.sol#953)
        - ^0.8.0 (PiGamificationStaking_flat.sol#988)
        - ^0.8.0 (PiGamificationStaking_flat.sol#1567)
        - ^0.8.1 (PiGamificationStaking_flat.sol#1797)
        - ^0.8.0 (PiGamificationStaking_flat.sol#1995)
        - ^0.8.2 (PiGamificationStaking_flat.sol#2113)
        - ^0.8.0 (PiGamificationStaking_flat.sol#2253)
        - ^0.8.0 (PiGamificationStaking_flat.sol#2293)
        - 0.8.2 (PiGamificationStaking_flat.sol#3089)
        - ^0.8.0 (PiGamificationStaking_flat.sol#3416)
        - ^0.8.0 (PiGamificationStaking_flat.sol#3545)
        - 0.8.2 (PiGamificationStaking_flat.sol#3611)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

PiGamificationStaking.withdraw(uint32[]) (PiGamificationStaking_flat.sol#1856-1883) has costly operations inside a loop:
        - _totalSupply = _totalSupply.sub(value) (PiGamificationStaking_flat.sol#1863)
PiGamificationStaking._withdrawAll() (PiGamificationStaking_flat.sol#1907-1922) has costly operations inside a loop:
        - _totalSupply = _totalSupply.sub(value) (PiGamificationStaking_flat.sol#1912)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

AddressUpgradeable.functionCall(address,bytes) (PiGamificationStaking_flat.sol#878-880) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (PiGamificationStaking_flat.sol#907-913) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (PiGamificationStaking_flat.sol#940-942) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (PiGamificationStaking_flat.sol#950-959) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (PiGamificationStaking_flat.sol#853-858) is never used and should be removed
ContextUpgradeable.__Context_init() (PiGamificationStaking_flat.sol#1267-1268) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (PiGamificationStaking_flat.sol#1270-1271) is never used and should be removed
ContextUpgradeable._msgData() (PiGamificationStaking_flat.sol#1276-1278) is never used and should be removed
Initializable._disableInitializers() (PiGamificationStaking_flat.sol#1239-1245) is never used and should be removed
MathUpgradeable.average(uint256,uint256) (PiGamificationStaking_flat.sol#368-371) is never used and should be removed
MathUpgradeable.ceilDiv(uint256,uint256) (PiGamificationStaking_flat.sol#379-382) is never used and should be removed
MathUpgradeable.max(uint256,uint256) (PiGamificationStaking_flat.sol#353-355) is never used and should be removed
MathUpgradeable.mulDiv(uint256,uint256,uint256) (PiGamificationStaking_flat.sol#389-403) is never used and should be removed
MathUpgradeable.mulDiv(uint256,uint256,uint256,MathUpgradeable.Rounding) (PiGamificationStaking_flat.sol#474-485) is never used and should be removed
MathUpgradeable.sqrt(uint256) (PiGamificationStaking_flat.sol#432-540) is never used and should be removed
MathUpgradeable.sqrt(uint256,MathUpgradeable.Rounding) (PiGamificationStaking_flat.sol#553-556) is never used and should be removed
OwnableUpgradeable.__Ownable_init() (PiGamificationStaking_flat.sol#1317-1319) is never used and should be removed
PausableUpgradeable.__Pausable_init() (PiGamificationStaking_flat.sol#1445-1447) is never used and should be removed
PausableUpgradeable.__Pausable_init_unchained() (PiGamificationStaking_flat.sol#1449-1451) is never used and should be removed
PausableUpgradeable._pause() (PiGamificationStaking_flat.sol#1505-1508) is never used and should be removed
PausableUpgradeable._requirePaused() (PiGamificationStaking_flat.sol#1604-1606) is never used and should be removed
PausableUpgradeable._unpause() (PiGamificationStaking_flat.sol#1517-1520) is never used and should be removed
ReentrancyGuardUpgradeable.__ReentrancyGuard_init() (PiGamificationStaking_flat.sol#1571-1573) is never used and should be removed
ReentrancyGuardUpgradeable.__ReentrancyGuard_init_unchained() (PiGamificationStaking_flat.sol#1575-1577) is never used and should be removed
SafeERC20Upgradeable.safeApprove(IERC20Upgradeable,address,uint256) (PiGamificationStaking_flat.sol#1030-1040) is never used and should be removed
SafeERC20Upgradeable.safeDecreaseAllowance(IERC20Upgradeable,address,uint256) (PiGamificationStaking_flat.sol#1060-1073) is never used and should be removed
SafeERC20Upgradeable.safeIncreaseAllowance(IERC20Upgradeable,address,uint256) (PiGamificationStaking_flat.sol#1051-1058) is never used and should be removed
SafeERC20Upgradeable.safePermit(IERC20PermitUpgradeable,address,address,uint256,uint256,uint8,bytes32,bytes32) (PiGamificationStaking_flat.sol#1079-1087) is never used and should be removed
SafeERC20Upgradeable.safeTransferFrom(IERC20Upgradeable,address,address,uint256) (PiGamificationStaking_flat.sol#1018-1027) is never used and should be removed
SafeMathUpgradeable.div(uint256,uint256,string) (PiGamificationStaking_flat.sol#750-753) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256) (PiGamificationStaking_flat.sol#714-716) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256,string) (PiGamificationStaking_flat.sol#780-789) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256,string) (PiGamificationStaking_flat.sol#731-740) is never used and should be removed
SafeMathUpgradeable.tryAdd(uint256,uint256) (PiGamificationStaking_flat.sol#581-591) is never used and should be removed
SafeMathUpgradeable.tryDiv(uint256,uint256) (PiGamificationStaking_flat.sol#627-632) is never used and should be removed
SafeMathUpgradeable.tryMod(uint256,uint256) (PiGamificationStaking_flat.sol#639-644) is never used and should be removed
SafeMathUpgradeable.tryMul(uint256,uint256) (PiGamificationStaking_flat.sol#610-620) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (PiGamificationStaking_flat.sol#598-602) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.2 (PiGamificationStaking_flat.sol#5) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (PiGamificationStaking_flat.sol#617) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (PiGamificationStaking_flat.sol#645) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (PiGamificationStaking_flat.sol#690) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (PiGamificationStaking_flat.sol#953) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (PiGamificationStaking_flat.sol#988) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (PiGamificationStaking_flat.sol#1567) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.1 (PiGamificationStaking_flat.sol#1797) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (PiGamificationStaking_flat.sol#1995) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.2 (PiGamificationStaking_flat.sol#2113) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (PiGamificationStaking_flat.sol#2253) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (PiGamificationStaking_flat.sol#2293) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.2 (PiGamificationStaking_flat.sol#3089) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (PiGamificationStaking_flat.sol#3416) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (PiGamificationStaking_flat.sol#3545) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.2 (PiGamificationStaking_flat.sol#3611) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.2 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (PiGamificationStaking_flat.sol#853-858):
        - (success) = recipient.call{value: amount}() (PiGamificationStaking_flat.sol#856)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (PiGamificationStaking_flat.sol#921-932):
        - (success,returndata) = target.call{value: value}(data) (PiGamificationStaking_flat.sol#930)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (PiGamificationStaking_flat.sol#950-959):
        - (success,returndata) = target.staticcall(data) (PiGamificationStaking_flat.sol#957)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Pragma version^0.8.0 (PiGamificationStaking_flat.sol#1253) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (PiGamificationStaking_flat.sol#1293) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.1 (PiGamificationStaking_flat.sol#1309) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (PiGamificationStaking_flat.sol#1418) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (PiGamificationStaking_flat.sol#1515) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.1 (PiGamificationStaking_flat.sol#1911) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.2 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (PiGamificationStaking_flat.sol#873-878):
        - (success) = recipient.call{value: amount}() (PiGamificationStaking_flat.sol#876)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (PiGamificationStaking_flat.sol#926-932):
        - (success,returndata) = target.call{value: value}(data) (PiGamificationStaking_flat.sol#930)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (PiGamificationStaking_flat.sol#950-955):
        - (success,returndata) = target.staticcall(data) (PiGamificationStaking_flat.sol#953)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IERC20PermitUpgradeable.DOMAIN_SEPARATOR() (PiGamificationStaking_flat.sol#245) is not in mixedCase
Function ContextUpgradeable.__Context_init() (PiGamificationStaking_flat.sol#1367-1369) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (PiGamificationStaking_flat.sol#1372-1375) is not in mixedCase
Variable ContextUpgradeable.__gap (PiGamificationStaking_flat.sol#1385) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (PiGamificationStaking_flat.sol#1417-1419) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (PiGamificationStaking_flat.sol#1421-1423) is not in mixedCase
Variable OwnableUpgradeable.__gap (PiGamificationStaking_flat.sol#1582) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (PiGamificationStaking_flat.sol#1605-1607) is not in mixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (PiGamificationStaking_flat.sol#1609-1611) is not in mixedCase
Variable PausableUpgradeable.__gap (PiGamificationStaking_flat.sol#1717) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init() (PiGamificationStaking_flat.sol#1571-1573) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init_unchained() (PiGamificationStaking_flat.sol#1575-1577) is not in mixedCase
Variable ReentrancyGuardUpgradeable.__gap (PiGamificationStaking_flat.sol#1685) is not in mixedCase
Parameter RewardsDistributionRecipient.setRewardsDistribution(address)._rewardsDistribution (PiGamificationStaking_flat.sol#1614) is not in mixedCase
Parameter PiGamificationStaking.initialize(address,address,address,address)._rewardsDistribution (PiGamificationStaking_flat.sol#1682) is not in mixedCase
Parameter PiGamificationStaking.initialize(address,address,address,address)._rewardsToken (PiGamificationStaking_flat.sol#1683) is not in mixedCase
Parameter PiGamificationStaking.initialize(address,address,address,address)._stakingToken (PiGamificationStaking_flat.sol#1684) is not in mixedCase
Parameter PiGamificationStaking.initialize(address,address,address,address). BLLContract (PiGamificationStaking_flat.sol#1685) is not in mixedCase
Function PiGamificationStaking.refresh_AllNfTids() (PiGamificationStaking_flat.sol#1828-1840) is not in mixedCase
Parameter PiGamificationStaking.setBLLContract(IBLL)._BLLContract (PiGamificationStaking_flat.sol#1852) is not in mixedCase
Function PiGamificationStaking.refresh_tokenIdsStaked(address) (PiGamificationStaking_flat.sol#1888-1905) is not in mixedCase
Parameter PiGamificationStaking.setRewardsDuration(uint256)._rewardsDuration (PiGamificationStaking_flat.sol#2001) is not in mixedCase
Variable PiGamificationStaking._tokenBalances (PiGamificationStaking_flat.sol#1659) is not in mixedCase
Variable PiGamificationStaking.BLLContract (PiGamificationStaking_flat.sol#1677) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

OwnableUpgradeable.__gap (PiGamificationStaking_flat.sol#1582) is never used in PiGamificationStaking (PiGamificationStaking_flat.sol#1645-1885)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (PiGamificationStaking_flat.sol#1354-1356)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (PiGamificationStaking_flat.sol#1362-1365)
recoverTokens(IERC20Upgradeable) should be declared external:
        - TokensRecoverable.recoverTokens(IERC20Upgradeable) (PiGamificationStaking_flat.sol#1385-1400)
recoverNFT(uint256) should be declared external:
        - TokensRecoverable.recoverNFT(uint256) (PiGamificationStaking_flat.sol#1402-1404)
recoverERC721(IERC721Upgradeable,uint256) should be declared external:
        - TokensRecoverable.recoverERC721(IERC721Upgradeable,uint256) (PiGamificationStaking_flat.sol#1406-1408)
initialize(address,address,address,address) should be declared external:
        - PiGamificationStaking.initialize(address,address,address,address) (PiGamificationStaking_flat.sol#1681-1703)
withdraw(uint32[]) should be declared external:
        - PiGamificationStaking.withdraw(uint32[]) (PiGamificationStaking_flat.sol#1859-1883)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
PiGamificationStaking_flat.sol analyzed (17 contracts with 75 detectors), 126 result(s) found
```

```
OwnableUpgradeable.__gap (PiStakingVault3_flat.sol#1359) shadows:
        - ContextUpgradeable.__gap (PiStakingVault3_flat.sol#1202)
PausableUpgradeable.__gap (PiStakingVault3_flat.sol#1610) shadows:
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

PiStakingVault3.getReward() (PiStakingVault3_flat.sol#1758-1765) ignores return value by rewardsToken.transfer(_msgSender(),reward) (PiStakingVault3_flat.sol#1762)
PiStakingVault3.recoverERC20(address,uint256) (PiStakingVault3_flat.sol#1796-1801) ignores return value by IERC20(tokenAddress).transfer(owner,tokenAmount) (PiStakingVault3_flat.sol#1798)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

PiStakingVault3.notifyRewardAmount(uint256) (PiStakingVault3_flat.sol#1774-1785) performs a multiplication on the result of a division:
        -rewardRate = reward.div(rewardsDuration) (PiStakingVault3_flat.sol#1778)
        -leftover = remaining.mul(rewardRate) (PiStakingVault3_flat.sol#1779)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Reentrancy in PiStakingVault3.exit() (PiStakingVault3_flat.sol#1767-1770):
        External calls:
        - _withdrawAll() (PiStakingVault3_flat.sol#1768)
                - stakingToken.safeTransferFrom(address(this),_msgSender(),tokenId) (PiStakingVault3_flat.sol#1751)
          getReward() (PiStakingVault3_flat.sol#1769)
                - rewardsToken.transfer(_msgSender(),reward) (PiStakingVault3_flat.sol#1762)
        State variables written after the call(s):
        - getReward() (PiStakingVault3_flat.sol#1769)
                - lastUpdateTime = lastTimeRewardApplicable() (PiStakingVault3_flat.sol#1820)
        - getReward() (PiStakingVault3_flat.sol#1769)
                - rewardPerTokenStored = rewardPerToken() (PiStakingVault3_flat.sol#1818)
        - getReward() (PiStakingVault3_flat.sol#1769)
                - rewards[_msgSender()] = 0 (PiStakingVault3_flat.sol#1761)
                - rewards[stakingAccount] = earned(account) (PiStakingVault3_flat.sol#1822)
        - getReward() (PiStakingVault3_flat.sol#1769)
                - userRewardPerTokenPaid[account] = rewardPerTokenStored (PiStakingVault3_flat.sol#1823)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

PiStakingVault3._withdrawAll().rewAr (PiStakingVault3_flat.sol#1733) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

PiStakingVault3.recoverERC20(address,uint256).owner (PiStakingVault3_flat.sol#1788) shadows:
        - OwnableUpgradeable.owner() (PiStakingVault3_flat.sol#1326-1328) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

RewardsDistributionRecipient.setRewardsDistribution(address) (PiStakingVault3_flat.sol#1603-1606) should emit an event for:
        - rewardsDistribution = _rewardsDistribution (PiStakingVault3_flat.sol#1605)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

PiStakingVault3.initialize(address,address,address,address)._rewardsDistribution (PiStakingVault3_flat.sol#1678) lacks a zero-check on :
        - rewardsDistribution = _rewardsDistribution (PiStakingVault3_flat.sol#1694)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

PiStakingVault3._withdrawAll() (PiStakingVault3_flat.sol#1745-1758) has external calls inside a loop: stakingToken.safeTransferFrom(address(this),_msgSender(),tokenId) (PiStakingVault3_flat.sol#1751)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#calls-inside-a-loop

Reentrancy in PiStakingVault3.stake(uint256) (PiStakingVault3_flat.sol#1713-1723):
        External calls:
        - stakingToken.safeTransferFrom(_msgSender(),address(this),tokenId) (PiStakingVault3_flat.sol#1718)
        State variables written after the call(s):
        - _tokenBalances[_msgSender()] = _tokenBalances[_msgSender()].add(1) (PiStakingVault3_flat.sol#1719)
        - _tokenIdsStaked[_msgSender()].push(tokenId) (PiStakingVault3_flat.sol#1720)
Reentrancy in PiStakingVault3.withdraw(uint256) (PiStakingVault3_flat.sol#1725-1743):
        External calls:
        - stakingToken.safeTransferFrom(address(this),_msgSender(),tokenId) (PiStakingVault3_flat.sol#1733)
        State variables written after the call(s):
```







Results

No major issues in Contracts.

Some false positive errors were reported by the tool. All the other issues have been categorised above according to their level of severity.

Closing Summary

Overall, smart contracts are well written and adhere to guidelines.

Some issues of Low and Informational were discovered in the audit,which the Pi Protocol team has Acknowledged.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Pi Protocol Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Pi Protocol Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

600+
Audits Completed

$15B
Secured

600K
Lines of Code Audited

Follow Our Journey

Audit Report
September, 2022

For

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com