QuillAudits

Audit Report
January, 2022

For

PATHFUND

Contents

Contents

Scope of the Audit

The scope of this audit was to analyze and document the Pathfund smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- BEP20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

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 BEP-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.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
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.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	1	2	0
Closed	1	0	1	9

Introduction

During the period of **December 1, 2021 to January 6, 2022** - QuillAudits Team performed a security audit for **LPADStake** smart contracts.

The code for the audit was taken from following the official link:
Codebase: Shared a zip file of the smart contract

V	Date	Commit ID/Contract address	Network
1	3 December	0x16689eE75493337d3d5D a4e992D38f8a6B68558f	Goerli

Issues Found – Code Review / Manual Testing

A.Contract - LPADStake

High severity issues

1. Missing check for the amount in claimReward and getPathFundTokens

In claimReward method
There is no check for the `_amount` which a user needs to claim. The `signature` can be created using the ECDSA contract of openzeppelin, and a user can input a wish of amount in the claimReward method to verify the signature

In getPathFundTokens method
There is no check for the `_pathFundAmount` which a user will get after sending ETH to the contract.

Recommendation
For the *claimReward*, calculate the reward of the user according to the staked amount in the contract and add a require check for input _amount.

For the *getPathFundTokens*, calculate the _pathFundAmount according to the ETH transferred to the contract.

Status: Fixed

Medium severity issues

2. Missing method to transfer accumulated fee

While the user stakes the token, the contract is deducting a percentage of the amount for the fee. The fee gets accumulated but there is no way to claim/transfer the accumulated fees to the owner/recipient.

Recommendation
Add a method to claim/transfer the accumulated fee to the owner/ recipient.

Status: Acknowledged

This issue is acknowledged and Path Fund team reported it as the fee is for staking contract and that fee is charged by the PathFund token.

Low severity issues

3. Missing method to transfer ETH

While transferring the _pathFundAmount to the user in return of ETH received, there should be a method to transfer/claim the ETH via another method after accumulation of it in the contract

```
payable(_owner).transfer(msg.value);
```

Recommendation
Add a method to claim/transfer the accumulated ETH in the contract to the owner/recipient to reduce the gas cost of calling the `getPathFundTokens`.

Status: Acknowledged

The Path Fund team acknowledges the issue and reported it as the BNB will be transferred.

4. Use of block.timestamp for stake time calculation and stake time comparison

The value of block.timestamp can be manipulated by the miner.
And conditions with strict equality is difficult to achieve -
block.timestamp <= _stakeTime[_user]

Recommendation
Avoid use of block.timestamp

Status: Acknowledged

The issue is acknowledged and the PathFund team reposted as the staking contract uses this time validation because the contract has only way to validate the staking duration.

5. Missing method and variable for staking fee

The stake fee (9%) is hardcoded here, and there is no way to update the stake fee in the future.

```
uint256 _finalAmount = _amount.sub(_amount.mul(9).div(100));
```

Recommendation

Add a variable to store the staking fee and a method to update the stake fee whenever necessary.

Status: Fixed

Informational issues

6. Length calculation within the loop

Reading from the state and fetching its length is a costly operation and doing such within a loop should be avoided

```
for (uint256 index = 0; index < _stakeHolders.length; index++) {}
```

Recommendation

_stakeHolders.length should be calculated and stored in a local variable before using in the for loop inside `**removeStakeHolder**`, `**isStakeHolder**`, `**removeTierID**`, `**updateTiersConfigurations**` and `**getTierForUser**` method.

Example:

```
uint256 arrayLength = _stakeHolders.length;
for (uint256 index = 0; index < arrayLength; index++) {}
```

Status: Fixed

7. Unnecessary use of Openzeppelin SafeMath

Solidity version 0.8 was released with SafeMath checks inbuilt, we can avoid using an explicit safe math library

Status: Fixed

8. Unnecessary use of bool checks in require method:

For a mapping that is returning a bool value, the equality check is not necessary.

```
require(
      isUniqueTierName[_tierName] == false,
      "LPADStake: Tier name already exist!"
);
```

Recommendation
Remove the equality check of bool returning mappings and add the use of “!”.

```
require(
      !isUniqueTierName[_tierName],
      "LPADStake: Tier name already exist!"
);
```

Status: Fixed

9. Unnecessary initialization of finePercentage

The variable `_finePercentage` is used once in the `if` block of `unStakeCoins()` method

```
if (block.timestamp <= _stakeTime[_user]) {
          uint16 _finePercentage = userTier.finePercentage;
          _amountToUnstake = _amount.sub(
              _amount.mul(_finePercentage).div(10000)
          );
      }
```

Recommendation
We recommend to remove the local variable initialization and directly use the `userTier.finePercentage` value in the calculation

```
if (block.timestamp <= _stakeTime[_user]) {
          _amountToUnstake = _amount.sub(
              _amount.mul(userTier.finePercentage).div(10000)
          );
      }
```

Status: Fixed

10. Naming Conventions

The contract should follow a consistent naming convention where the private variables with leading “_” and public variables without it.
But we have missed complying to the condition for certain variable names - “**allTierLimitConfigurations**”, “**isUniqueTierName**” and “**tierIDs**” which are private

Recommendation
Add “_” to private variable names.

Status: Fixed

11. Incorrect return comment on `isStakeHolder` method

“isStakeHolder” method has return comment stating,

```
* @return 'true' and 'index' if valid or 'false' and '0' if not
```

but only return bool value.

Recommendation
Replace the comment.

Status: Fixed

12. Documentation tag @notice is not valid for non-public state variables.

`@notice` tag is not valid for non-public state variables, it will cause `DocstringParsingError` while compiling the code.

Status: Fixed

13. Unnecessary use of “if” and “else” block

In `isValidSigner` method of “Verify.sol” contract there is a use of “IF” and “ELSE” blocks which is unnecessary for single line code

```
if(_amount2 == 0) _messageHash = messageHash1(_amount1);
else _messageHash = messageHash2(_amount1, _amount2);
```

Recommendation

Use the ternary operation,

```
_messageHash = _amount2 == 0 ? messageHash1(_amount1) :
messageHash2(_amount1, _amount2);
```

Status: Fixed

14. Length calculation within the loop

Reading from the state and fetching its length is a costly operation and doing such within a loop should be avoided

Recommendation

_excluded.length should be calculated and stored in a variable before using in the for loop inside `**_getCurrentSupply**` and `**includeInReward**` method.

Status: Fixed

Goerli Testnet Test Contract

ERC20: 0xFf3d839697224B66c7E7eD6610115e967ad7d6B7
LPADStake: 0x16689eE75493337d3d5Da4e992D38f8a6B68558f
Verify: 0x80c17B250b3274f2Ec2416e31d063aB5d2a83d81

Functional Tests

- Add Tier
- Stake Coins
- Unstake Coins
- Update Tiers Configurations

Automated Tests

Slither

```
▶slither .
'npx hardhat compile --force' running
Compiling 10 files with 0.8.0
Compilation finished successfully
 ·----------------|-------------·
 |  Contract Name  ·  Size (Kb)  |
 ·----------------|-------------·
 |  ECDSA          ·       0.08  |
 ·----------------|-------------·
 |  ERC20          ·       2.47  |
 ·----------------|-------------·
 |  LPADStake      ·      11.83  |
 ·----------------|-------------·
 |  SafeMath       ·       0.08  |
 ·----------------|-------------·
 |  Strings        ·       0.08  |
 ·----------------|-------------·
 |  Token          ·       2.47  |
 ·----------------|-------------·
 |  Verify         ·       2.13  |
 ·----------------|-------------·
Creating Typechain artifacts in directory types for target ethers-v5
Successfully generated Typechain artifacts!

Solidity 0.8.0 is not fully supported yet. You can still use Hardhat, but some features, like stack traces, might not work correctly.

Learn more at https://hardhat.org/reference/solidity-support"



Reentrancy in LPADStake.unstakeCoins(uint256) (contracts/LPADStakeFlattened.sol#591-622):
        External calls:
        - require(bool)(IERC20(_tokenAddress).transfer(_user,_amountToUnstake)) (contracts/LPADStakeFlattened.sol#613)
        State variables written after the call(s):
        - removeStakeHolder(_user) (contracts/LPADStakeFlattened.sol#617)
                - _stakeHolders[index] = _stakeHolders[_stakeHolders.length - 1] (contracts/LPADStakeFlattened.sol#640)
                - _stakeHolders.pop() (contracts/LPADStakeFlattened.sol#641)
        - delete _stakeTime[_user] (contracts/LPADStakeFlattened.sol#618)
        - _stakes[_user] = _stakes[msg.sender].sub(_amount) (contracts/LPADStakeFlattened.sol#614)
        - userTier = getTierForUser(_stakes[_user],_user) (contracts/LPADStakeFlattened.sol#615)
                - userTiers[_user] = tierIDs[0] (contracts/LPADStakeFlattened.sol#511)
                - userTiers[_user] = _tierID (contracts/LPADStakeFlattened.sol#552)
        - delete userTiers[_user] (contracts/LPADStakeFlattened.sol#619)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

LPADStake.getTierForUser(uint256,address)._tierID (contracts/LPADStakeFlattened.sol#515) is a local variable never initialized
LPADStake.addTier(string,string,uint256,uint16,uint16).newTier (contracts/LPADStakeFlattened.sol#445) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

LPADStake.getPathFundTokens(uint256,bytes)._owner (contracts/LPADStakeFlattened.sol#707) shadows:
        - Ownable._owner (contracts/LPADStakeFlattened.sol#315) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Token.constructor(string,string).name (contracts/tokens/Token.sol#7) shadows:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#60-62) (function)
```

```
Token.constructor(string,string).name (contracts/tokens/Token.sol#7) shadows:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#60-62) (function)
        - IERC20Metadata.name() (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#16) (function)
Token.constructor(string,string).symbol (contracts/tokens/Token.sol#7) shadows:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#68-70) (function)
        - IERC20Metadata.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#21) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

LPADStake.constructor(address,address).tokenAddress_ (contracts/LPADStakeFlattened.sol#389) lacks a zero-check on :
                - _tokenAddress = tokenAddress_ (contracts/LPADStakeFlattened.sol#390)
LPADStake.constructor(address,address).verifyAddress_ (contracts/LPADStakeFlattened.sol#389) lacks a zero-check on :
                - _verifyAddress = verifyAddress_ (contracts/LPADStakeFlattened.sol#391)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in LPADStake.stakeCoins(uint256) (contracts/LPADStakeFlattened.sol#560-585):
        External calls:
        - require(bool)(IERC20(_tokenAddress).transferFrom(msg.sender,address(this),_amount)) (contracts/LPADStakeFlattened.sol#566-572)
        State variables written after the call(s):
        - _stakeHolders.push(msg.sender) (contracts/LPADStakeFlattened.sol#575)
        - _stakeTime[msg.sender] = block.timestamp.add(_days.mul(86400)) (contracts/LPADStakeFlattened.sol#583)
        - _stakes[msg.sender] = _stakes[msg.sender].add(_finalAmount) (contracts/LPADStakeFlattened.sol#577)
        - userTier = getTierForUser(_stakes[msg.sender],msg.sender) (contracts/LPADStakeFlattened.sol#578-581)
                - userTiers[_user] = tierIDs[0] (contracts/LPADStakeFlattened.sol#511)
                - userTiers[_user] = _tierID (contracts/LPADStakeFlattened.sol#552)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in LPADStake.claimReward(uint256,bytes) (contracts/LPADStakeFlattened.sol#664-682):
        External calls:
        - require(bool)(IERC20(_tokenAddress).transfer(msg.sender,_amount)) (contracts/LPADStakeFlattened.sol#680)
        Event emitted after the call(s):
        - ClaimReward(msg.sender,_amount) (contracts/LPADStakeFlattened.sol#681)
Reentrancy in LPADStake.getPathFundTokens(uint256,bytes) (contracts/LPADStakeFlattened.sol#684-710):
        External calls:
        - require(bool)(IERC20(_tokenAddress).transfer(msg.sender,_pathFundAmount)) (contracts/LPADStakeFlattened.sol#706)
        External calls sending eth:
        - address(_owner).transfer(msg.value) (contracts/LPADStakeFlattened.sol#708)
        Event emitted after the call(s):
        - PurchaseTokens(_pathFundAmount,msg.value,msg.sender) (contracts/LPADStakeFlattened.sol#709)
Reentrancy in LPADStake.stakeCoins(uint256) (contracts/LPADStakeFlattened.sol#560-585):
        External calls:
        - require(bool)(IERC20(_tokenAddress).transferFrom(msg.sender,address(this),_amount)) (contracts/LPADStakeFlattened.sol#566-572)
        Event emitted after the call(s):
        - Stake(msg.sender,_finalAmount) (contracts/LPADStakeFlattened.sol#584)
Reentrancy in LPADStake.unstakeCoins(uint256) (contracts/LPADStakeFlattened.sol#591-622):
        External calls:
        - require(bool)(IERC20(_tokenAddress).transfer(_user,_amountToUnstake)) (contracts/LPADStakeFlattened.sol#613)
        Event emitted after the call(s):
        - Unstake(_user,_amount) (contracts/LPADStakeFlattened.sol#621)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

LPADStake.unstakeCoins(uint256) (contracts/LPADStakeFlattened.sol#591-622) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp <= _stakeTime[_user] (contracts/LPADStakeFlattened.sol#607)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
ECDSA.recover(bytes32,bytes) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#26-59) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#39-43)
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#48-53)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

LPADStake.addTier(string,string,uint256,uint16,uint16) (contracts/LPADStakeFlattened.sol#417-454) compares to a boolean constant:
        -require(bool,string)(isUniqueTierName[_tierName] == false,LPADStake: Tier name already exist!) (contracts/LPADStakeFlattened.sol#436-439)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Context._msgData() (contracts/LPADStakeFlattened.sol#297-299) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/LPADStakeFlattened.sol#179-188) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/LPADStakeFlattened.sol#139-141) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/LPADStakeFlattened.sol#205-214) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (contracts/LPADStakeFlattened.sol#156-165) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (contracts/LPADStakeFlattened.sol#10-16) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (contracts/LPADStakeFlattened.sol#52-57) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (contracts/LPADStakeFlattened.sol#64-69) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (contracts/LPADStakeFlattened.sol#35-45) is never used and should be removed
SafeMath.trySub(uint256,uint256) (contracts/LPADStakeFlattened.sol#23-28) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (contracts/LPADStakeFlattened.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/tokens/Token.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Verify.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter LPADStake.addTier(string,string,uint256,uint16,uint16)._tierID (contracts/LPADStakeFlattened.sol#418) is not in mixedCase
Parameter LPADStake.addTier(string,string,uint256,uint16,uint16)._tierName (contracts/LPADStakeFlattened.sol#419) is not in mixedCase
Parameter LPADStake.addTier(string,string,uint256,uint16,uint16)._limit (contracts/LPADStakeFlattened.sol#420) is not in mixedCase
Parameter LPADStake.addTier(string,string,uint256,uint16,uint16)._fineDays (contracts/LPADStakeFlattened.sol#421) is not in mixedCase
Parameter LPADStake.addTier(string,string,uint256,uint16,uint16)._finePercentage (contracts/LPADStakeFlattened.sol#422) is not in mixedCase
Parameter LPADStake.removeTierID(string)._tierID (contracts/LPADStakeFlattened.sol#456) is not in mixedCase
Parameter LPADStake.updateTiersConfigurations(string[],string[],uint16[],uint16[],uint256[])._tierIDs (contracts/LPADStakeFlattened.sol#478) is not in mixedCase
Parameter LPADStake.updateTiersConfigurations(string[],string[],uint16[],uint16[],uint256[])._tierNames (contracts/LPADStakeFlattened.sol#479) is not in mixedCase
Parameter LPADStake.updateTiersConfigurations(string[],string[],uint16[],uint16[],uint256[])._fineDays (contracts/LPADStakeFlattened.sol#480) is not in mixedCase
Parameter LPADStake.updateTiersConfigurations(string[],string[],uint16[],uint16[],uint256[])._finePercentages (contracts/LPADStakeFlattened.sol#481) is not in mixedCase
Parameter LPADStake.updateTiersConfigurations(string[],string[],uint16[],uint16[],uint256[])._limits (contracts/LPADStakeFlattened.sol#482) is not in mixedCase
Parameter LPADStake.getTierForUser(uint256,address)._amount (contracts/LPADStakeFlattened.sol#506) is not in mixedCase
Parameter LPADStake.getTierForUser(uint256,address)._user (contracts/LPADStakeFlattened.sol#506) is not in mixedCase
Parameter LPADStake.stakeCoins(uint256)._amount (contracts/LPADStakeFlattened.sol#560) is not in mixedCase
Parameter LPADStake.unstakeCoins(uint256)._amount (contracts/LPADStakeFlattened.sol#591) is not in mixedCase
Parameter LPADStake.stakeOf(address)._stakeHolder (contracts/LPADStakeFlattened.sol#629) is not in mixedCase
Parameter LPADStake.removeStakeHolder(address)._stakeHolder (contracts/LPADStakeFlattened.sol#637) is not in mixedCase
Parameter LPADStake.isStakeHolder(address)._stakeHolder (contracts/LPADStakeFlattened.sol#652) is not in mixedCase
```


```
Parameter LPADStake.addTier(string,string,uint256,uint16,uint16)._tierName (contracts/LPADStakeFlattened.sol#419) is not in mixedCase
Parameter LPADStake.addTier(string,string,uint256,uint16,uint16)._limit (contracts/LPADStakeFlattened.sol#420) is not in mixedCase
Parameter LPADStake.addTier(string,string,uint256,uint16,uint16)._fineDays (contracts/LPADStakeFlattened.sol#421) is not in mixedCase
Parameter LPADStake.addTier(string,string,uint256,uint16,uint16)._finePercentage (contracts/LPADStakeFlattened.sol#422) is not in mixedCase
Parameter LPADStake.removeTierID(string)._tierID (contracts/LPADStakeFlattened.sol#456) is not in mixedCase
Parameter LPADStake.updateTiersConfigurations(string[],string[],uint16[],uint16[],uint256[])._tierIDs (contracts/LPADStakeFlattened.sol#478) is not in mixedCase
Parameter LPADStake.updateTiersConfigurations(string[],string[],uint16[],uint16[],uint256[])._tierNames (contracts/LPADStakeFlattened.sol#479) is not in mixedCase
Parameter LPADStake.updateTiersConfigurations(string[],string[],uint16[],uint16[],uint256[])._fineDays (contracts/LPADStakeFlattened.sol#480) is not in mixedCase
Parameter LPADStake.updateTiersConfigurations(string[],string[],uint16[],uint16[],uint256[])._finePercentages (contracts/LPADStakeFlattened.sol#481) is not in mixedCase
Parameter LPADStake.updateTiersConfigurations(string[],string[],uint16[],uint16[],uint256[])._limits (contracts/LPADStakeFlattened.sol#482) is not in mixedCase
Parameter LPADStake.getTierForUser(uint256,address)._amount (contracts/LPADStakeFlattened.sol#506) is not in mixedCase
Parameter LPADStake.getTierForUser(uint256,address)._user (contracts/LPADStakeFlattened.sol#506) is not in mixedCase
Parameter LPADStake.stakeCoins(uint256)._amount (contracts/LPADStakeFlattened.sol#560) is not in mixedCase
Parameter LPADStake.unstakeCoins(uint256)._amount (contracts/LPADStakeFlattened.sol#591) is not in mixedCase
Parameter LPADStake.stakeOf(address)._stakeHolder (contracts/LPADStakeFlattened.sol#629) is not in mixedCase
Parameter LPADStake.removeStakeHolder(address)._stakeHolder (contracts/LPADStakeFlattened.sol#637) is not in mixedCase
Parameter LPADStake.isStakeHolder(address)._stakeHolder (contracts/LPADStakeFlattened.sol#652) is not in mixedCase
Parameter LPADStake.claimReward(uint256,bytes)._amount (contracts/LPADStakeFlattened.sol#664) is not in mixedCase
Parameter LPADStake.claimReward(uint256,bytes)._signature (contracts/LPADStakeFlattened.sol#664) is not in mixedCase
Parameter LPADStake.getPathFundTokens(uint256,bytes)._pathFundAmount (contracts/LPADStakeFlattened.sol#684) is not in mixedCase
Parameter LPADStake.getPathFundTokens(uint256,bytes)._signature (contracts/LPADStakeFlattened.sol#684) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Constant Strings.alphabet (node_modules/@openzeppelin/contracts/utils/Strings.sol#9) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter Verify.isValidSigner(uint256,uint256,address,bytes)._amount1 (contracts/Verify.sol#10) is not in mixedCase
Parameter Verify.isValidSigner(uint256,uint256,address,bytes)._amount2 (contracts/Verify.sol#10) is not in mixedCase
Parameter Verify.isValidSigner(uint256,uint256,address,bytes)._caller (contracts/Verify.sol#10) is not in mixedCase
Parameter Verify.isValidSigner(uint256,uint256,address,bytes)._signature (contracts/Verify.sol#10) is not in mixedCase
Parameter Verify.messageHash1(uint256)._amount (contracts/Verify.sol#20) is not in mixedCase
Parameter Verify.messageHash2(uint256,uint256)._pathFundAmuont (contracts/Verify.sol#26) is not in mixedCase
Parameter Verify.messageHash2(uint256,uint256)._bnbAmount (contracts/Verify.sol#26) is not in mixedCase
Parameter Verify.messageSigner(bytes32,bytes)._messageHash (contracts/Verify.sol#33) is not in mixedCase
Parameter Verify.messageSigner(bytes32,bytes)._signature (contracts/Verify.sol#33) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (node_modules/@openzeppelin/contracts/utils/Context.sol#21)" inContext (node_modules/@openzeppelin/contracts/utils/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Reentrancy in LPADStake.getPathFundTokens(uint256,bytes) (contracts/LPADStakeFlattened.sol#684-710):
        External calls:
        - address(_owner).transfer(msg.value) (contracts/LPADStakeFlattened.sol#708)
        Event emitted after the call(s):
        - PurchaseTokens(_pathFundAmount,msg.value,msg.sender) (contracts/LPADStakeFlattened.sol#709)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable LPADStake._verifyAddress (contracts/LPADStakeFlattened.sol#378) is too similar to LPADStake.constructor(address,address).verifyAddress_ (contracts/LPADStakeFlattened.sol#389)
Variable LPADStake._tokenAddress (contracts/LPADStakeFlattened.sol#377) is too similar to LPADStake.constructor(address,address).tokenAddress_ (contracts/LPADStakeFlattened.sol#389)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

Token.constructor(string,string) (contracts/tokens/Token.sol#7-9) uses literals with too many digits:
        - _mint(msg.sender,100000000000000000000000000000000) (contracts/tokens/Token.sol#8)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

LPADStake._stakingDays (contracts/LPADStakeFlattened.sol#375) is never used in LPADStake (contracts/LPADStakeFlattened.sol#372-711)
LPADStake._unStakingChargePercentage (contracts/LPADStakeFlattened.sol#376) is never used in LPADStake (contracts/LPADStakeFlattened.sol#372-711)
```

```
LPADStake._stakingDays (contracts/LPADStakeFlattened.sol#375) is never used in LPADStake (contracts/LPADStakeFlattened.sol#372-711)
LPADStake._unStakingChargePercentage (contracts/LPADStakeFlattened.sol#376) is never used in LPADStake (contracts/LPADStakeFlattened.sol#372-711)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

LPADStake._stakingDays (contracts/LPADStakeFlattened.sol#375) should be constant
LPADStake._unStakingChargePercentage (contracts/LPADStakeFlattened.sol#376) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (contracts/LPADStakeFlattened.sol#348-350)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (contracts/LPADStakeFlattened.sol#356-359)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

name() should be declared external:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#60-62)
symbol() should be declared external:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#68-70)
decimals() should be declared external:
        - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#85-87)
totalSupply() should be declared external:
        - ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#92-94)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#99-101)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#111-114)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#119-121)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#130-133)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#148-156)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#170-173)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#189-195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
. analyzed (14 contracts with 75 detectors), 90 result(s) found
```

Results

Major issues were found. We strictly recommend resolving the high and medium severity issues.

Closing Summary

Overall, the smart contract looks good and well written. Majority of the issues are fixed by the Pathfund team.
One Medium and two low severity issues has been Acknowledged by the Pathfund Team.
The contracts can be deployed to Mainnet now, considering the fix of the High severity issue.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or endorsement of the **PathFund**. 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 **PathFund** Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
January, 2022

For

PATHFUND

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com