QuillAudits

Audit Report
April, 2022

For

POLYSPORTS

Contents

Scope of the Audit

The scope of this audit was to analyze and document the Polysports Token and Vesting 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
- ERC20 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 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, Theo.

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	0	0
Closed	0	0	0	2

Introduction

During the period of **March 29, 2021 to April 6, 2021** - QuillAudits Team performed a security audit for Polysports smart contracts.

The code for the audit was taken from following the official link:
https://github.com/abhijeet029/PS1-contract

V	Date	Commit ID
1	06 April	478e73f166757b63a06cf2de4090d7915b15cde4
2	07 April	9bd99acbb8d0ce72f4e6a7fd1162750937e8dac1

Issues Found – Code Review / Manual Testing

A. Contract - PolysportsToken.sol

High severity issues

No issues found

Medium severity issues

No issues found

Low severity issues

No issues found

Informational Issues

1. **Missing Events for Significant Transactions**

   Description

   The missing event makes it difficult to track off-chain state variable changes. An event should be emitted for significant transactions calling the functions :
   - setGovernance

   Remediation

   We recommend emitting the appropriate events.

   Status: Fixed

   In version 2, the team fixed the issue with the recommended changes.

2. Ownable contract has no use

   Line#14 contract PolysportsToken is ERC20Permit, Ownable {

   Description

   Ownable contract has been imported and inherited by the PolysportsToken contract, but none of its functions/modifiers have been used in the contract. In place of owner, an entity governance has been used and functions have been restricted using onlyGovernance modifier.

Remediation
It's recommended that you remove the inherited Ownable contract, to decrease the contract size.

Status: Fixed
In version 2, the team fixed the issue with the recommended changes.

B. Contract - PolysportsTokenVesting.sol

High severity issues

No issues found

Medium severity issues

3. Centralization Risks

Description
The role owner has the authority to :
- Transfer out the vested tokens from the contract at any point of time using the recoverExcessToken function

Remediation
We advise the client to handle the governance account carefully to avoid any potential hack. We also advise the client to consider the following solutions:

- Time-lock with reasonable latency for community awareness on privileged operations;
- Multisig with community-voted 3rd-party independent co-signers.
- DAO or Governance module, increasing transparency and community involvement.

Status: Acknowledged

Low severity issues

No issues found

Informational Issues

No issues found

Functional Tests

- Should not be withdrawable before initial timestamp PASS
- Should be withdrawable according to vesting timelines PASS
- Should not be able to withdraw again, if once withdrawn PASS
- totalSupply should equal the specified amount PASS
- Users should be able to burn tokens PASS

Automated Tests

Slither

```
BokkyPooBahsDateTimeLibrary._daysFromDate(uint256,uint256,uint256) (contracts/BokkyPooBahsDateTimeLibrary.sol#57-79) performs a multiplication on the result of a division:
	-__days = _day - 32075 + (1461 * (_year + 4800 + (_month - 14) / 12)) / 4 + (367 * (_month - 2 - ((_month - 14) / 12) * 12)) / 12 - (3 * ((_year + 4900 + (_month - 14) / 12) / 100)) / 4 - OFFSET19700101 (contracts/BokkyPooBahsDateTimeLibrary.sol#67-76)
BokkyPooBahsDateTimeLibrary._daysToDate(uint256) (contracts/BokkyPooBahsDateTimeLibrary.sol#98-123) performs a multiplication on the result of a division:
	-N = (4 * L) / 146097 (contracts/BokkyPooBahsDateTimeLibrary.sol#110)
	-L = L - (146097 * N + 3) / 4 (contracts/BokkyPooBahsDateTimeLibrary.sol#111)
BokkyPooBahsDateTimeLibrary._daysToDate(uint256) (contracts/BokkyPooBahsDateTimeLibrary.sol#98-123) performs a multiplication on the result of a division:
	-_year = (4000 * (L + 1)) / 1461001 (contracts/BokkyPooBahsDateTimeLibrary.sol#112)
	-L = L - (1461 * _year) / 4 + 31 (contracts/BokkyPooBahsDateTimeLibrary.sol#113)
BokkyPooBahsDateTimeLibrary._daysToDate(uint256) (contracts/BokkyPooBahsDateTimeLibrary.sol#98-123) performs a multiplication on the result of a division:
	-_month = (80 * L) / 2447 (contracts/BokkyPooBahsDateTimeLibrary.sol#114)
	-_day = L - (2447 * _month) / 80 (contracts/BokkyPooBahsDateTimeLibrary.sol#115)
BokkyPooBahsDateTimeLibrary._daysToDate(uint256) (contracts/BokkyPooBahsDateTimeLibrary.sol#98-123) performs a multiplication on the result of a division:
	-L = _month / 11 (contracts/BokkyPooBahsDateTimeLibrary.sol#116)
	-_month = _month + 2 - 12 * L (contracts/BokkyPooBahsDateTimeLibrary.sol#117)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

BokkyPooBahsDateTimeLibrary._daysToDate(uint256).L (contracts/BokkyPooBahsDateTimeLibrary.sol#109) is written in both
	L = L - (1461 * _year) / 4 + 31 (contracts/BokkyPooBahsDateTimeLibrary.sol#113)
	L = _month / 11 (contracts/BokkyPooBahsDateTimeLibrary.sol#116)
BokkyPooBahsDateTimeLibrary.subMonths(uint256,uint256).year (contracts/BokkyPooBahsDateTimeLibrary.sol#331) is written in both
	(year,month,day) = _daysToDate(timestamp / SECONDS_PER_DAY) (contracts/BokkyPooBahsDateTimeLibrary.sol#331)
	year = yearMonth / 12 (contracts/BokkyPooBahsDateTimeLibrary.sol#333)
BokkyPooBahsDateTimeLibrary.subMonths(uint256,uint256).month (contracts/BokkyPooBahsDateTimeLibrary.sol#331) is written in both
	(year,month,day) = _daysToDate(timestamp / SECONDS_PER_DAY) (contracts/BokkyPooBahsDateTimeLibrary.sol#331)
	month = (yearMonth % 12) + 1 (contracts/BokkyPooBahsDateTimeLibrary.sol#334)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#write-after-write

PolysportsToken.setGovernance(address) (contracts/PolysportsToken.sol#36-38) should emit an event for:
	- governance = _governance (contracts/PolysportsToken.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

PolysportsTokenVesting.setInitialTimestamp(uint256) (contracts/PolysportsTokenVesting.sol#260-263) should emit an event for:
	- _initialTimestamp = _timestamp (contracts/PolysportsTokenVesting.sol#262)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

PolysportsToken.setGovernance(address)._governance (contracts/PolysportsToken.sol#36) lacks a zero-check on :
		- governance = _governance (contracts/PolysportsToken.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in PolysportsTokenVesting.recoverExcessToken(address,uint256) (contracts/PolysportsTokenVesting.sol#327-330):
	External calls:
	- IERC20(_token).safeTransfer(_msgSender(),amount) (contracts/PolysportsTokenVesting.sol#328)
	Event emitted after the call(s):
	- RecoverToken(_token,amount) (contracts/PolysportsTokenVesting.sol#329)
Reentrancy in PolysportsToken.recoverToken(address,address,uint256) (contracts/PolysportsToken.sol#47-55):
	External calls:
	- require(bool,string)(IERC20(token).transfer(destination,amount),Retrieve failed) (contracts/PolysportsToken.sol#53)
	Event emitted after the call(s):
	- RecoverToken(token,destination,amount) (contracts/PolysportsToken.sol#54)
Reentrancy in PolysportsTokenVesting.withdrawTokens(uint256) (contracts/PolysportsTokenVesting.sol#245-256):
	External calls:
	- _ps1Token.safeTransfer(distribution.beneficiary,tokensAvailable) (contracts/PolysportsTokenVesting.sol#253)
	Event emitted after the call(s):
	- WithdrawnTokens(_msgSender(),tokensAvailable) (contracts/PolysportsTokenVesting.sol#255)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ERC20Permit.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (contracts/ERC20Permit.sol#40-61) uses timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(block.timestamp <= deadline,Permit: expired deadline) (contracts/ERC20Permit.sol#49)
PolysportsTokenVesting._calculateAvailablePercentage(PolysportsTokenVesting.DistributionType) (contracts/PolysportsTokenVesting.sol#280-318) uses timestamp for comparisons
	Dangerous comparisons:
	- currentTimeStamp > _initialTimestamp (contracts/PolysportsTokenVesting.sol#289)
	- noOfMonths > 12 (contracts/PolysportsTokenVesting.sol#291)
	- noOfMonths > 22 (contracts/PolysportsTokenVesting.sol#294)
	- noOfMonths > 11 (contracts/PolysportsTokenVesting.sol#297)
	- noOfMonths > 4 (contracts/PolysportsTokenVesting.sol#300)
	- noOfMonths > 22 (contracts/PolysportsTokenVesting.sol#303)
	- noOfMonths > 22 (contracts/PolysportsTokenVesting.sol#306)
	- noOfMonths > 12 (contracts/PolysportsTokenVesting.sol#309)
	- noOfMonths > 12 (contracts/PolysportsTokenVesting.sol#312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#201-221) uses assembly
	- INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#213-216)
ERC20Permit.constructor() (contracts/ERC20Permit.sol#19-34) uses assembly
	- INLINE ASM (contracts/ERC20Permit.sol#21-23)
console._sendLogPayload(bytes) (node_modules/hardhat/console.sol#7-14) uses assembly
	- INLINE ASM (node_modules/hardhat/console.sol#10-13)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
	- Version used: ['0.8.1', '>=0.4.22<0.9.0', '^0.8.0', '^0.8.1']
	- ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
	- ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Counters.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#4)
	- 0.8.1 (contracts/BokkyPooBahsDateTimeLibrary.sol#3)
	- 0.8.1 (contracts/ERC20Permit.sol#3)
	- 0.8.1 (contracts/IERC2612Permit.sol#3)
	- 0.8.1 (contracts/PolysportsToken.sol#3)
	- 0.8.1 (contracts/PolysportsTokenVesting.sol#3)
	- >=0.4.22<0.9.0 (node_modules/hardhat/console.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

BokkyPooBahsDateTimeLibrary._daysFromDate(uint256,uint256,uint256) (contracts/BokkyPooBahsDateTimeLibrary.sol#57-79) is never used and should be removed
BokkyPooBahsDateTimeLibrary._daysToDate(uint256) (contracts/BokkyPooBahsDateTimeLibrary.sol#98-123) is never used and should be removed
BokkyPooBahsDateTimeLibrary._getDaysInMonth(uint256,uint256) (contracts/BokkyPooBahsDateTimeLibrary.sol#233-241) is never used and should be removed
BokkyPooBahsDateTimeLibrary._isLeapYear(uint256) (contracts/BokkyPooBahsDateTimeLibrary.sol#216-218) is never used and should be removed
BokkyPooBahsDateTimeLibrary.addDays(uint256,uint256) (contracts/BokkyPooBahsDateTimeLibrary.sol#299-302) is never used and should be removed
```

```
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Counters.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#4) allows old versions
Pragma version0.8.1 (contracts/BokkyPooBahsDateTimeLibrary.sol#3) allows old versions
Pragma version0.8.1 (contracts/ERC20Permit.sol#3) allows old versions
Pragma version0.8.1 (contracts/IERC2612Permit.sol#3) allows old versions
Pragma version0.8.1 (contracts/PolysportsToken.sol#3) allows old versions
Pragma version0.8.1 (contracts/PolysportsTokenVesting.sol#3) allows old versions
Pragma version>=0.4.22<0.9.0 (node_modules/hardhat/console.sol#2) is too complex
solc-0.8.1 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#60-65):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#63)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#157-166):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#164)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#184-193):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter BokkyPooBahsDateTimeLibrary.addYears(uint256,uint256)._years (contracts/BokkyPooBahsDateTimeLibrary.sol#275) is not in mixedCase
Parameter BokkyPooBahsDateTimeLibrary.addMonths(uint256,uint256)._months (contracts/BokkyPooBahsDateTimeLibrary.sol#286) is not in mixedCase
Parameter BokkyPooBahsDateTimeLibrary.addDays(uint256,uint256)._days (contracts/BokkyPooBahsDateTimeLibrary.sol#299) is not in mixedCase
Parameter BokkyPooBahsDateTimeLibrary.addHours(uint256,uint256)._hours (contracts/BokkyPooBahsDateTimeLibrary.sol#304) is not in mixedCase
Parameter BokkyPooBahsDateTimeLibrary.addMinutes(uint256,uint256)._minutes (contracts/BokkyPooBahsDateTimeLibrary.sol#309) is not in mixedCase
Parameter BokkyPooBahsDateTimeLibrary.addSeconds(uint256,uint256)._seconds (contracts/BokkyPooBahsDateTimeLibrary.sol#314) is not in mixedCase
Parameter BokkyPooBahsDateTimeLibrary.subYears(uint256,uint256)._years (contracts/BokkyPooBahsDateTimeLibrary.sol#319) is not in mixedCase
Parameter BokkyPooBahsDateTimeLibrary.subMonths(uint256,uint256)._months (contracts/BokkyPooBahsDateTimeLibrary.sol#330) is not in mixedCase
Parameter BokkyPooBahsDateTimeLibrary.subDays(uint256,uint256)._days (contracts/BokkyPooBahsDateTimeLibrary.sol#343) is not in mixedCase
Parameter BokkyPooBahsDateTimeLibrary.subHours(uint256,uint256)._hours (contracts/BokkyPooBahsDateTimeLibrary.sol#348) is not in mixedCase
Parameter BokkyPooBahsDateTimeLibrary.subMinutes(uint256,uint256)._minutes (contracts/BokkyPooBahsDateTimeLibrary.sol#353) is not in mixedCase
Parameter BokkyPooBahsDateTimeLibrary.subSeconds(uint256,uint256)._seconds (contracts/BokkyPooBahsDateTimeLibrary.sol#358) is not in mixedCase
Variable ERC20Permit.DOMAIN_SEPARATOR (contracts/ERC20Permit.sol#17) is not in mixedCase
Parameter PolysportsToken.setGovernance(address)._governance (contracts/PolysportsToken.sol#36) is not in mixedCase
Parameter PolysportsTokenVesting.withdrawTokens(uint256)._distributionType (contracts/PolysportsTokenVesting.sol#245) is not in mixedCase
Parameter PolysportsTokenVesting.setInitialTimestamp(uint256)._timestamp (contracts/PolysportsTokenVesting.sol#260) is not in mixedCase
Parameter PolysportsTokenVesting.recoverExcessToken(address,uint256)._token (contracts/PolysportsTokenVesting.sol#327) is not in mixedCase
Variable PolysportsTokenVesting._preSeedVesting (contracts/PolysportsTokenVesting.sol#53-66) is not in mixedCase
Variable PolysportsTokenVesting._private1Vesting (contracts/PolysportsTokenVesting.sol#68-80) is not in mixedCase
Variable PolysportsTokenVesting._publicVesting (contracts/PolysportsTokenVesting.sol#82) is not in mixedCase
Variable PolysportsTokenVesting._gamingVesting (contracts/PolysportsTokenVesting.sol#84-107) is not in mixedCase
Variable PolysportsTokenVesting._advisorsVesting (contracts/PolysportsTokenVesting.sol#109-132) is not in mixedCase
Variable PolysportsTokenVesting._valueAdvisorsVesting (contracts/PolysportsTokenVesting.sol#134-147) is not in mixedCase
Variable PolysportsTokenVesting._teamVesting (contracts/PolysportsTokenVesting.sol#149-172) is not in mixedCase
```

```
PolysportsTokenVesting.constructor(address,address) (contracts/PolysportsTokenVesting.sol#202-218) uses literals with too many digits:
        - _addDistribution(_treasury,DistributionType.PRE_SEED,187500000000000000000000,0) (contracts/PolysportsTokenVesting.sol#206)
PolysportsTokenVesting.constructor(address,address) (contracts/PolysportsTokenVesting.sol#202-218) uses literals with too many digits:
        - _addDistribution(_treasury,DistributionType.SEED,85000000000000000000000,0) (contracts/PolysportsTokenVesting.sol#207)
PolysportsTokenVesting.constructor(address,address) (contracts/PolysportsTokenVesting.sol#202-218) uses literals with too many digits:
        - _addDistribution(_treasury,DistributionType.PRIVATE1,150000000000000000000000,0) (contracts/PolysportsTokenVesting.sol#208)
PolysportsTokenVesting.constructor(address,address) (contracts/PolysportsTokenVesting.sol#202-218) uses literals with too many digits:
        - _addDistribution(_treasury,DistributionType.PRIVATE2,165714290000000000000000,0) (contracts/PolysportsTokenVesting.sol#209)
PolysportsTokenVesting.constructor(address,address) (contracts/PolysportsTokenVesting.sol#202-218) uses literals with too many digits:
        - _addDistribution(_treasury,DistributionType.PUBLIC,196785710000000000000000,983928550000000000000000) (contracts/PolysportsTokenVesting.sol#210)
PolysportsTokenVesting.constructor(address,address) (contracts/PolysportsTokenVesting.sol#202-218) uses literals with too many digits:
        - _addDistribution(_treasury,DistributionType.COMMUNITY,100000000000000000000000,0) (contracts/PolysportsTokenVesting.sol#211)
PolysportsTokenVesting.constructor(address,address) (contracts/PolysportsTokenVesting.sol#202-218) uses literals with too many digits:
        - _addDistribution(_treasury,DistributionType.ECOSYSTEM,170000000000000000000000,0) (contracts/PolysportsTokenVesting.sol#212)
PolysportsTokenVesting.constructor(address,address) (contracts/PolysportsTokenVesting.sol#202-218) uses literals with too many digits:
        - _addDistribution(_treasury,DistributionType.GAMING,200000000000000000000000,0) (contracts/PolysportsTokenVesting.sol#213)
PolysportsTokenVesting.constructor(address,address) (contracts/PolysportsTokenVesting.sol#202-218) uses literals with too many digits:
        - _addDistribution(_treasury,DistributionType.ADVISORS,50000000000000000000000,0) (contracts/PolysportsTokenVesting.sol#214)
PolysportsTokenVesting.constructor(address,address) (contracts/PolysportsTokenVesting.sol#202-218) uses literals with too many digits:
        - _addDistribution(_treasury,DistributionType.VALUE_ADVISORS,100000000000000000000000,0) (contracts/PolysportsTokenVesting.sol#215)
PolysportsTokenVesting.constructor(address,address) (contracts/PolysportsTokenVesting.sol#202-218) uses literals with too many digits:
        - _addDistribution(_treasury,DistributionType.TEAM,120000000000000000000000,0) (contracts/PolysportsTokenVesting.sol#216)
PolysportsTokenVesting.constructor(address,address) (contracts/PolysportsTokenVesting.sol#202-218) uses literals with too many digits:
        - _addDistribution(_treasury,DistributionType.TREASURY,100000000000000000000000,833333333330000000000000) (contracts/PolysportsTokenVesting.sol#217)
PolysportsTokenVesting.slitherConstructorVariables() (contracts/PolysportsTokenVesting.sol#12-331) uses literals with too many digits:
        - _preSeedVesting = (0,909090909090000000,1818181818180000000,2727272727270000000,3636363636360000000,4545454545450000000,5454545454540000000,6363636363630000000,7272727272720000000,8181818181810000000,9090909090900000000,10000000000000000000) (contracts/PolysportsTokenVesting.sol#53-66)
PolysportsTokenVesting.slitherConstructorVariables() (contracts/PolysportsTokenVesting.sol#12-331) uses literals with too many digits:
        - _privateVesting = (0,1000000000000000000,2000000000000000000,3000000000000000000,4000000000000000000,5000000000000000000,6000000000000000000,7000000000000000000,8000000000000000000,9000000000000000000,10000000000000000000) (contracts/PolysportsTokenVesting.sol#68-80)
PolysportsTokenVesting.slitherConstructorVariables() (contracts/PolysportsTokenVesting.sol#12-331) uses literals with too many digits:
        - _publicVesting = (250000000000000000,500000000000000000,750000000000000000,1000000000000000000) (contracts/PolysportsTokenVesting.sol#82)
PolysportsTokenVesting.slitherConstructorVariables() (contracts/PolysportsTokenVesting.sol#12-331) uses literals with too many digits:
        - _gamingVesting = (0,0,0,526315789470000000,1052631578940000000,1578947368410000000,2105263157880000000,2631578947350000000,3157894736820000000,3684210526290000000,4210526315760000000,4736842105230000000,5263157894700000000,5789473684170000000,6315789473640000000,6842105263110000000,7368421052580000000,7894736842050000000,8421052631520000000,8947368420990000000,9473684210460000000,10000000000000000000) (contracts/PolysportsTokenVesting.sol#84-107)
PolysportsTokenVesting.slitherConstructorVariables() (contracts/PolysportsTokenVesting.sol#12-331) uses literals with too many digits:
        - _advisorsVesting = (0,0,0,0,0,0,0,625000000000000000,1250000000000000000,1875000000000000000,2500000000000000000,3125000000000000000,3750000000000000000,4375000000000000000,5000000000000000000,5625000000000000000,6250000000000000000,6875000000000000000,7500000000000000000,8125000000000000000,8750000000000000000,9375000000000000000,10000000000000000000) (contracts/PolysportsTokenVesting.sol#109-132)
PolysportsTokenVesting.slitherConstructorVariables() (contracts/PolysportsTokenVesting.sol#12-331) uses literals with too many digits:
        - _valueAdvisorsVesting = (0,0,0,1111111111110000000,2222222222220000000,3333333333330000000,4444444444440000000,5555555555550000000,6666666666660000000,7777777777770000000,8888888888880000000,10000000000000000000) (contracts/PolysportsTokenVesting.sol#134-147)
PolysportsTokenVesting.slitherConstructorVariables() (contracts/PolysportsTokenVesting.sol#12-331) uses literals with too many digits:
        - _teamVesting = (0,0,0,0,0,0,0,0,0,769230769230000000,1538461538460000000,2307692307690000000,3076923076920000000,3846153846150000000,4615384615380000000,5384615384610000000,6153846153840000000,6923076923070000000,7692307692300000000,8461538461530000000,9230769230760000000,10000000000000000000)
(contracts/PolysportsTokenVesting.sol#149-172)
PolysportsTokenVesting.slitherConstructorVariables() (contracts/PolysportsTokenVesting.sol#12-331) uses literals with too many digits:
        - _treasuryVesting = (833333333330000000,1666666666660000000,2499999999990000000,3333333333320000000,4166666666650000000,4999999999980000000,5833333333310000000,6666666666640000000,7499999999970000000,8333333333300000000,9166666666630000000,10000000000000000000) (contracts/PolysportsTokenVesting.s
ol#174-187)
console.slitherConstructorConstantVariables() (node_modules/hardhat/console.sol#4-1532) uses literals with too many digits:
        - CONSOLE_ADDRESS = address(0x000000000000000000636F6e736F6c652e6c6f67) (node_modules/hardhat/console.sol#5)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

BokkyPooBahsDateTimeLibrary.DOW_MON (contracts/BokkyPooBahsDateTimeLibrary.sol#36) is never used in BokkyPooBahsDateTimeLibrary (contracts/BokkyPooBahsDateTimeLibrary.sol#30-396)
BokkyPooBahsDateTimeLibrary.DOW_TUE (contracts/BokkyPooBahsDateTimeLibrary.sol#37) is never used in BokkyPooBahsDateTimeLibrary (contracts/BokkyPooBahsDateTimeLibrary.sol#30-396)
BokkyPooBahsDateTimeLibrary.DOW_WED (contracts/BokkyPooBahsDateTimeLibrary.sol#38) is never used in BokkyPooBahsDateTimeLibrary (contracts/BokkyPooBahsDateTimeLibrary.sol#30-396)
BokkyPooBahsDateTimeLibrary.DOW_THU (contracts/BokkyPooBahsDateTimeLibrary.sol#39) is never used in BokkyPooBahsDateTimeLibrary (contracts/BokkyPooBahsDateTimeLibrary.sol#30-396)
```

```
BokkyPooBahsDateTimeLibrary.DOW_SUN (contracts/BokkyPooBahsDateTimeLibrary.sol#42) is never used in BokkyPooBahsDateTimeLibrary (contracts/BokkyPooBahsDateTimeLibrary.sol#30-396)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
symbol() should be declared external:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
        - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)
totalSupply() should be declared external:
        - ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#113-117)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#136-140)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#158-167)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#181-185)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#201-210)
permit(address,address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - ERC20Permit.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (contracts/ERC20Permit.sol#40-61)
nonces(address) should be declared external:
        - ERC20Permit.nonces(address) (contracts/ERC20Permit.sol#66-68)
setGovernance(address) should be declared external:
        - PolysportsToken.setGovernance(address) (contracts/PolysportsToken.sol#36-38)
burn(uint256) should be declared external:
        - PolysportsToken.burn(uint256) (contracts/PolysportsToken.sol#61-63)
getInitialTimestamp() should be declared external:
        - PolysportsTokenVesting.getInitialTimestamp() (contracts/PolysportsTokenVesting.sol#221-223)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
. analyzed (15 contracts with 77 detectors), 146 result(s) found
```

Results

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.

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines. One Medium and Two Informational severity Issues have been reported and all the issues have been acknowledged or fixed by the Auditee.

The contracts are good to deploy on the public mainnet.

Disclaimer

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

Audit Report
April, 2022

For

POLYSPORTS

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com