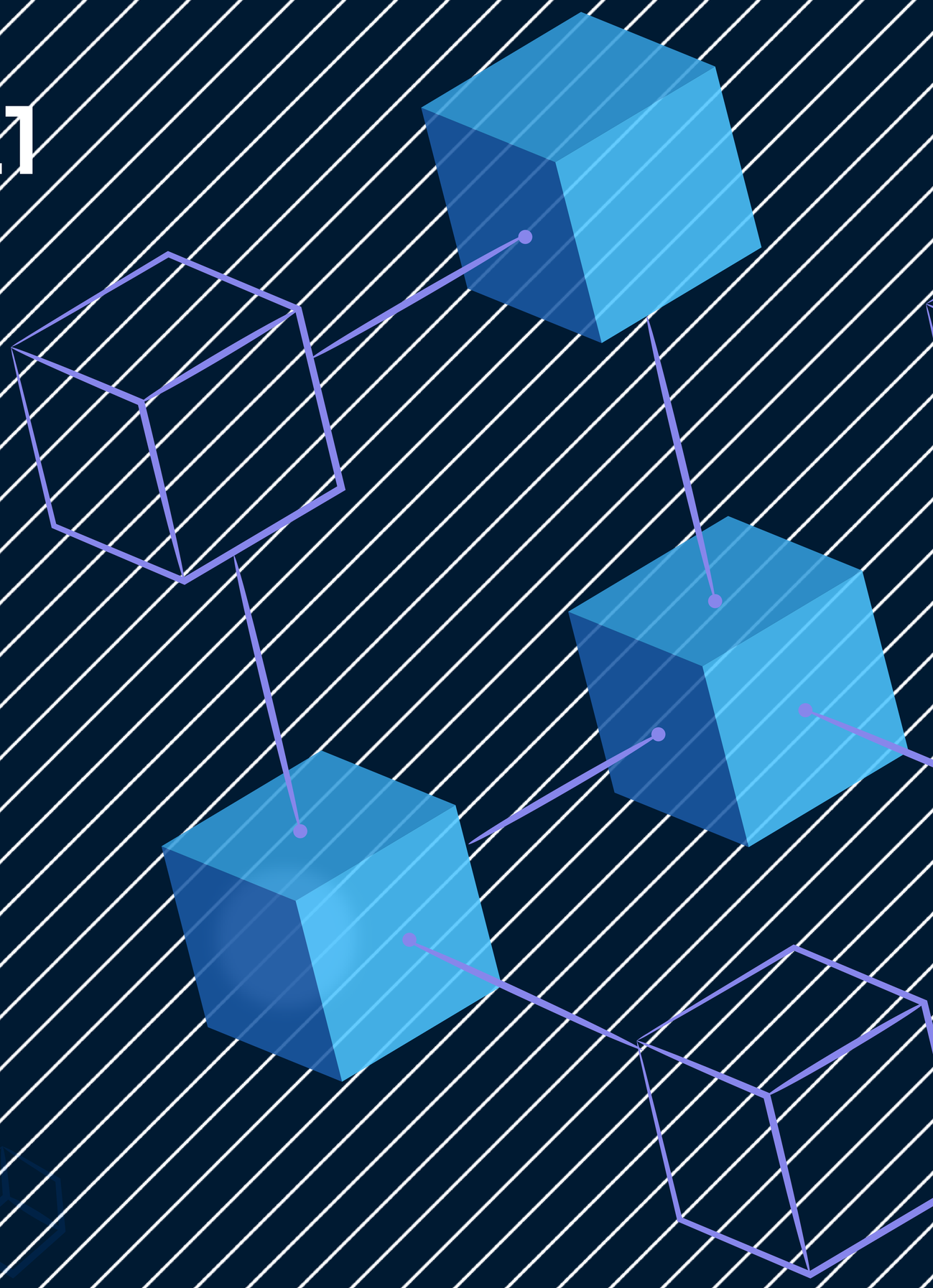
QuillAudits

# Audit Report
# November, 2021



For

# Contents

## Scope of the Audit

The scope of this audit was to analyze and document the Yoda smart contracts 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 | 2 | 1 | 1 |
| Closed | 0 | 0 | 1 | 0 |

# Introduction

During the period of **October 12, 2021, to October 21, 2021** - QuillAudits Team performed a security audit for **Yoda NFT Prediction** smart contracts.

The code for the audit was taken from the following official repo of **YODA**: https://github.com/yoda-xyz/yoda-nft-prediction-contract/

| Note | Date | Commit hash |
| --- | --- | --- |
| Version 1 | October | 091de744c2afd9947050aa624e66fb79697ddf9d |
| Version 2 | October | ea9bec4f0e431258db2d86baede23704cc3c0e6e |

# Issues Found

## A. Contract – YodaNFT

### High severity issues

No issues were found.

### Medium severity issues

#### A.1 The owner can lock the user's funds

```
Line 124:
function setEndTime(uint256 _endTime) external onlyOwner {
        require(winner == ~uint256(0), "Winner has been declared");
        require(block.timestamp < _endTime, "Time passed");
    endTime = _endTime;
        emit SetEndTime(endTime);
     }
```

**Description**

The owner has the ability to increment the endTime as much as he wants; this represents a risk for the users because in that case their funds will be locked in the smart contract.

**Remediation**

There are two possibilities to remediate the risk. The first one is using a multisig wallet to not give the owner the full power over the smart contract. The second one is setting a limit to the value that endTime can be incremented to.

**Acknowledged**

The Yoda team has acknowledged the risk knowing that they will keep the community notified about the changes in endTime.

## A.2 Usage of block.timestamp

```
Line 66:
require(_endTime > block.timestamp, "Invalid Value for timestamp");
```

```
Line 85:
require(block.timestamp < endTime, "Sale has ended");
```

```
Line 107:
require(block.timestamp >= endTime, "Not allowed");
```

```
Line 126:
require(block.timestamp < _endTime, "Time passed");
```

**Description**

Block.timestamp is used in the contract. The variable block is a set of variables. The timestamp does not always reflect the current time and may be inaccurate. The value of a block can be influenced by miners. Maximal Extractable Value attacks require a timestamp of up to 900 seconds. There is no guarantee that the value is right, all that is guaranteed is that it is higher than the timestamp of the previous block.

**Remediation**

You can use an Oracle to get the exact time or verify if a delay of 900 seconds won't impact the logic of the smart contract.

**Acknowledged**

The Yoda team has acknowledged the risk knowing that a 900 seconds delay won't affect the smart contract's logic.

# Low level severity issues

## A.3 Floating pragma

```
Line 2:
pragma solidity >=0.8.4;
```

**Description**
The contract makes use of the floating-point pragma 0.8.4 Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version that may introduce issues in the contract system.

**Remediation**
Consider locking the pragma version. It is advised that floating pragma not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

**Fixed**
The Yoda team has fixed the issue in version 2 by locking the pragma version to 0.8.4.

## A.4 Renounce Ownership

```
Line 11:
contract YodaNFT is Initializable, OwnableUpgradeable,  ERC1155SupplyUpgradeable {
```

**Description**
Typically, the contract's owner is the account that deploys the contract. As a result, the owner is able to perform certain privileged activities on his behalf. The renounceOwnership function is used in smart contracts to renounce ownership. Otherwise, if the contract's ownership has not been transferred previously, it will never have an Owner, which is risky.

**Remediation**
It is advised that the Owner cannot call renounceOwnership without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the renounceOwnership method for two or more users should be confirmed. Alternatively, the Renounce Ownership functionality can be disabled by overriding it.

**Acknowledged**
The Yoda team has acknowledged the risk.

# Informational Issues

## A.5 Incrementing with a zero value

```
Line 174:
function claimableAll() public view returns (uint256 claim) {
        if (winnerTotalLocked == 0) claim += 0;
        else if (winner == ~uint256(0)) claim += 0;
        else claim += (balanceOf(_msgSender(), winner) * ((rewardPoolLocked *
winnerRate) / 100)) / winnerTotalLocked;

        if (runnerUpTotalLocked == 0) claim += 0;
        else if (runnerUp == ~uint256(0)) claim += 0;
        else
            claim += (balanceOf(_msgSender(), runnerUp) * ((rewardPoolLocked *
runnerUpRate) / 100)) /
                runnerUpTotalLocked;
        }
```

```
Line 206:
if (balanceOf(_msgSender(), _id) < _tokenAmount) return 0;
        if (totalLocked == 0) claim += 0;
        else if (_id == ~uint256(0)) claim += 0;
        else claim += (_tokenAmount * ((rewardPoolLocked * rate) / 100)) / totalLocked;
```

**Description**
In the smart contract there are some incrementations with zero value that are not achieving a specific goal or it might be done with a wrong value.

**Remediation**
It’s recommended to remove these incrementations or change the values to a non-zero value.

**Acknowledged**
The Yoda team has acknowledged the risk.

# Automated Tests

## Slither

```
'npx hardhat compile --force' running
Downloading compiler 0.8.7
Compiling 19 files with 0.8.7
Generating typings for: 19 artifacts in dir: typechain for target: ethers-v5
Successfully generated 31 typings!
Compilation finished successfully

Solidity 0.8.7 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"



OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#77) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC1155Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#458) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC1155SupplyUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol#94) shadows:
        - ERC1155Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#458)
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

DFYNWorldCup.claimableAll() (contracts/DFYNWorldCup.sol#174-188) performs a multiplication on the result of a division:
        -claim += (balanceOf(_msgSender(),winner) * (rewardPoolLocked * winnerRate / 100)) / winnerTotalLocked (contracts/DFYNWorldCup.sol#179)
DFYNWorldCup.claimableAll() (contracts/DFYNWorldCup.sol#174-188) performs a multiplication on the result of a division:
        -claim += (balanceOf(_msgSender(),runnerUp) * (rewardPoolLocked * runnerUpRate / 100)) / runnerUpTotalLocked (contracts/DFYNWorldCup.sol#185-187)
DFYNWorldCup.claimableExact(uint256,uint256) (contracts/DFYNWorldCup.sol#192-211) performs a multiplication on the result of a division:
        -claim += _tokenAmount * (rewardPoolLocked * rate / 100) / totalLocked (contracts/DFYNWorldCup.sol#210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Reentrancy in DFYNWorldCup.claimRewardAll() (contracts/DFYNWorldCup.sol#132-151):
        External calls:
        - _mint(_msgSender(),totalTeams,balanceOf(_msgSender(),winner),0x0) (contracts/DFYNWorldCup.sol#137)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        State variables written after the call(s):
        - _burn(_msgSender(),winner,balanceOf(_msgSender(),winner)) (contracts/DFYNWorldCup.sol#139)
                - _balances[id][account] = accountBalance - amount (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#340)
        - _burn(_msgSender(),winner,balanceOf(_msgSender(),winner)) (contracts/DFYNWorldCup.sol#139)
                - _totalSupply[id] -= amount (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol#78)
Reentrancy in DFYNWorldCup.claimRewardAll() (contracts/DFYNWorldCup.sol#132-151):
        External calls:
        - _mint(_msgSender(),totalTeams,balanceOf(_msgSender(),winner),0x0) (contracts/DFYNWorldCup.sol#137)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        - _mint(_msgSender(),totalTeams + 1,balanceOf(_msgSender(),runnerUp),0x0) (contracts/DFYNWorldCup.sol#142)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        State variables written after the call(s):
        - _mint(_msgSender(),totalTeams + 1,balanceOf(_msgSender(),runnerUp),0x0) (contracts/DFYNWorldCup.sol#142)
```

```
Reentrancy in DFYNWorldCup.claimRewardAll() (contracts/DFYNWorldCup.sol#132-151):
        External calls:
        - _mint(_msgSender(),totalTeams,balanceOf(_msgSender(),winner),0x0) (contracts/DFYNWorldCup.sol#137)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        - _mint(_msgSender(),totalTeams + 1,balanceOf(_msgSender(),runnerUp),0x0) (contracts/DFYNWorldCup.sol#142)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        State variables written after the call(s):
        - _mint(_msgSender(),totalTeams + 1,balanceOf(_msgSender(),runnerUp),0x0) (contracts/DFYNWorldCup.sol#142)
                - _balances[id][account] += amount (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#281)
        - _burn(_msgSender(),runnerUp,balanceOf(_msgSender(),runnerUp)) (contracts/DFYNWorldCup.sol#144)
                - _balances[id][account] = accountBalance - amount (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#340)
        - _mint(_msgSender(),totalTeams + 1,balanceOf(_msgSender(),runnerUp),0x0) (contracts/DFYNWorldCup.sol#142)
                - _totalSupply[id] += amount (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol#51)
        - _burn(_msgSender(),runnerUp,balanceOf(_msgSender(),runnerUp)) (contracts/DFYNWorldCup.sol#144)
                - _totalSupply[id] -= amount (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol#78)
Reentrancy in DFYNWorldCup.claimRewardExact(uint256,uint256) (contracts/DFYNWorldCup.sol#154-170):
        External calls:
        - _mint(_msgSender(),totalTeams,_tokenAmount,0x0) (contracts/DFYNWorldCup.sol#159)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        - _mint(_msgSender(),totalTeams + 1,_tokenAmount,0x0) (contracts/DFYNWorldCup.sol#161)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        State variables written after the call(s):
        - _burn(_msgSender(),_id,_tokenAmount) (contracts/DFYNWorldCup.sol#164)
                - _balances[id][account] = accountBalance - amount (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#340)
        - _burn(_msgSender(),_id,_tokenAmount) (contracts/DFYNWorldCup.sol#164)
                - _totalSupply[id] -= amount (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol#78)
Reentrancy in DFYNWorldCup.withdrawTreasury() (contracts/DFYNWorldCup.sol#214-220):
        External calls:
        - token.safeTransfer(treasury,rewardPoolLocked * treasuryRate / 100) (contracts/DFYNWorldCup.sol#216)
        State variables written after the call(s):
        - treasuryClaimed = true (contracts/DFYNWorldCup.sol#217)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

ERC1155Upgradeable._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#444) is a
ocal variable never initialized
ERC1155Upgradeable._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#439) is
 local variable never initialized
ERC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#421) is a local var
able never initialized
ERC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417) is a local v
riable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#408-427) ignores return va
ue by IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
ERC1155Upgradeable._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#429-450) ignores
eturn value by IERC1155ReceiverUpgradeable(to).onERC1155BatchReceived(operator,from,ids,amounts,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#438-448)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

DFYNWorldCup.initialize(string,uint256,IERC20Upgradeable,address,uint256,uint256,uint256,uint256,uint256)._uri (contracts/DFYNWorldCup.sol#50) shadows:
        - ERC1155Upgradeable._uri (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#30) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Variable 'ERC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417)' in
 ERC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#408-427) potentially used
before declaration: response != IERC1155ReceiverUpgradeable.onERC1155Received.selector (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#418)
Variable 'ERC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#421)' in E
RC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#408-427) potentially used be
fore declaration: revert(string)(reason) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#422)
Variable 'ERC1155Upgradeable._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol
#439)' in ERC1155Upgradeable._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#429-450)
 potentially used before declaration: response != IERC1155ReceiverUpgradeable.onERC1155BatchReceived.selector (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#441)
Variable 'ERC1155Upgradeable._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#4
44)' in ERC1155Upgradeable._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#429-450) p
otentially used before declaration: revert(string)(reason) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#445)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in ERC1155SupplyUpgradeable._mint(address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol#44-52):
        External calls:
        - super._mint(account,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol#50)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        State variables written after the call(s):
        - _totalSupply[id] += amount (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol#51)
Reentrancy in ERC1155SupplyUpgradeable._mintBatch(address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol#57-67):
        External calls:
        - super._mintBatch(to,ids,amounts,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol#63)
                - IERC1155ReceiverUpgradeable(to).onERC1155BatchReceived(operator,from,ids,amounts,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#438-448)
        State variables written after the call(s):
        - _totalSupply[ids[i]] += amounts[i] (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol#65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in DFYNWorldCup.buy(uint256,uint256) (contracts/DFYNWorldCup.sol#84-92):
        External calls:
        - token.safeTransferFrom(_msgSender(),address(this),BASE_PRICE * amount) (contracts/DFYNWorldCup.sol#88)
        - _mint(_msgSender(),_id,amount,0x0) (contracts/DFYNWorldCup.sol#89)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        Event emitted after the call(s):
        - Buy(_id,amount,_msgSender()) (contracts/DFYNWorldCup.sol#91)
        - TransferSingle(operator,address(0),account,id,amount) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#282)
                - _mint(_msgSender(),_id,amount,0x0) (contracts/DFYNWorldCup.sol#89)
Reentrancy in DFYNWorldCup.claimRewardAll() (contracts/DFYNWorldCup.sol#132-151):
        External calls:
        - _mint(_msgSender(),totalTeams,balanceOf(_msgSender(),winner),0x0) (contracts/DFYNWorldCup.sol#137)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        Event emitted after the call(s):
        - TransferSingle(operator,account,address(0),id,amount) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#343)
                -  burn( msgSender(),winner,balanceOf( msgSender(),winner)) (contracts/DFYNWorldCup.sol#139)
```

```
DFYNWorldCup.initialize(string,uint256,IERC20Upgradeable,address,uint256,uint256,uint256,uint256,uint256) (contracts/DFYNWorldCup.sol#49-79) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_endTime > block.timestamp,Invalid Value for timestamp) (contracts/DFYNWorldCup.sol#66)
DFYNWorldCup.buy(uint256,uint256) (contracts/DFYNWorldCup.sol#84-92) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < endTime,Sale has ended) (contracts/DFYNWorldCup.sol#85)
DFYNWorldCup.setWinners(uint256,uint256) (contracts/DFYNWorldCup.sol#106-120) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= endTime,Not allowed) (contracts/DFYNWorldCup.sol#107)
DFYNWorldCup.setEndTime(uint256) (contracts/DFYNWorldCup.sol#124-129) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < _endTime,Time passed) (contracts/DFYNWorldCup.sol#126)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

AddressUpgradeable.isContract(address) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#26-36) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#32-34)
AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#168-188) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#180-183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['>=0.8.4', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/IERC1155ReceiverUpgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/IERC1155Upgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/IERC1155MetadataURIUpgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#3)
        - >=0.8.4 (contracts/DFYNWorldCup.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Different versions of Solidity is used:
        - Version used: ['>=0.8.4', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - >=0.8.4 (contracts/test/TestToken.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Reentrancy in DFYNWorldCup.claimRewardAll() (contracts/DFYNWorldCup.sol#132-151):
        External calls:
        - _mint(_msgSender(),totalTeams,balanceOf(_msgSender(),winner),0x0) (contracts/DFYNWorldCup.sol#137)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        - _mint(_msgSender(),totalTeams + 1,balanceOf(_msgSender(),runnerUp),0x0) (contracts/DFYNWorldCup.sol#142)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        Event emitted after the call(s):
        - TransferSingle(operator,address(0),account,id,amount) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#282)
                - _mint(_msgSender(),totalTeams + 1,balanceOf(_msgSender(),runnerUp),0x0) (contracts/DFYNWorldCup.sol#142)
        - TransferSingle(operator,account,address(0),id,amount) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#343)
                - _burn(_msgSender(),runnerUp,balanceOf(_msgSender(),runnerUp)) (contracts/DFYNWorldCup.sol#144)
Reentrancy in DFYNWorldCup.claimRewardAll() (contracts/DFYNWorldCup.sol#132-151):
        External calls:
        - _mint(_msgSender(),totalTeams,balanceOf(_msgSender(),winner),0x0) (contracts/DFYNWorldCup.sol#137)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        - _mint(_msgSender(),totalTeams + 1,balanceOf(_msgSender(),runnerUp),0x0) (contracts/DFYNWorldCup.sol#142)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        - token.safeTransfer(_msgSender(),userPoolShare) (contracts/DFYNWorldCup.sol#147)
        Event emitted after the call(s):
        - ClaimReward(_msgSender(),userPoolShare) (contracts/DFYNWorldCup.sol#149)
Reentrancy in DFYNWorldCup.claimRewardExact(uint256,uint256) (contracts/DFYNWorldCup.sol#154-170):
        External calls:
        - _mint(_msgSender(),totalTeams,_tokenAmount,0x0) (contracts/DFYNWorldCup.sol#159)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        - _mint(_msgSender(),totalTeams + 1,_tokenAmount,0x0) (contracts/DFYNWorldCup.sol#161)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        Event emitted after the call(s):
        - TransferSingle(operator,account,address(0),id,amount) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#343)
                - _burn(_msgSender(),_id,_tokenAmount) (contracts/DFYNWorldCup.sol#164)
Reentrancy in DFYNWorldCup.claimRewardExact(uint256,uint256) (contracts/DFYNWorldCup.sol#154-170):
        External calls:
        - _mint(_msgSender(),totalTeams,_tokenAmount,0x0) (contracts/DFYNWorldCup.sol#159)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        - _mint(_msgSender(),totalTeams + 1,_tokenAmount,0x0) (contracts/DFYNWorldCup.sol#161)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
        - token.safeTransfer(_msgSender(),userPoolShare) (contracts/DFYNWorldCup.sol#167)
        Event emitted after the call(s):
        - ClaimReward(_msgSender(),userPoolShare) (contracts/DFYNWorldCup.sol#168)
Reentrancy in DFYNWorldCup.emergencyWithdraw() (contracts/DFYNWorldCup.sol#223-228):
        External calls:
        - token.safeTransfer(_msgSender(),amount) (contracts/DFYNWorldCup.sol#225)
        Event emitted after the call(s):
        - EmergencyWithdraw(amount) (contracts/DFYNWorldCup.sol#227)
Reentrancy in DFYNWorldCup.withdrawTreasury() (contracts/DFYNWorldCup.sol#214-220):
        External calls:
        - token.safeTransfer(treasury,rewardPoolLocked * treasuryRate / 100) (contracts/DFYNWorldCup.sol#216)
        Event emitted after the call(s):
        - TreasuryClaimed(rewardPoolLocked * treasuryRate / 100) (contracts/DFYNWorldCup.sol#219)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
Parameter DFYNWorldCup.initialize(string,uint256,IERC20Upgradeable,address,uint256,uint256,uint256,uint256,uint256)._uri (contracts/DFYNWorldCup.sol#50) is not in mixedCase
Parameter DFYNWorldCup.initialize(string,uint256,IERC20Upgradeable,address,uint256,uint256,uint256,uint256,uint256)._basePrice (contracts/DFYNWorldCup.sol#51) is not in mixedCase
Parameter DFYNWorldCup.initialize(string,uint256,IERC20Upgradeable,address,uint256,uint256,uint256,uint256,uint256)._tokenAddr (contracts/DFYNWorldCup.sol#52) is not in mixedCase
Parameter DFYNWorldCup.initialize(string,uint256,IERC20Upgradeable,address,uint256,uint256,uint256,uint256,uint256)._treasury (contracts/DFYNWorldCup.sol#53) is not in mixedCase
Parameter DFYNWorldCup.initialize(string,uint256,IERC20Upgradeable,address,uint256,uint256,uint256,uint256,uint256)._treasuryRate (contracts/DFYNWorldCup.sol#54) is not in mixedCase
Parameter DFYNWorldCup.initialize(string,uint256,IERC20Upgradeable,address,uint256,uint256,uint256,uint256,uint256)._winnerRate (contracts/DFYNWorldCup.sol#55) is not in mixedCase
Parameter DFYNWorldCup.initialize(string,uint256,IERC20Upgradeable,address,uint256,uint256,uint256,uint256,uint256)._runnerUpRate (contracts/DFYNWorldCup.sol#56) is not in mixedCase
Parameter DFYNWorldCup.initialize(string,uint256,IERC20Upgradeable,address,uint256,uint256,uint256,uint256,uint256)._endTime (contracts/DFYNWorldCup.sol#57) is not in mixedCase
Parameter DFYNWorldCup.initialize(string,uint256,IERC20Upgradeable,address,uint256,uint256,uint256,uint256,uint256)._totalTeams (contracts/DFYNWorldCup.sol#58) is not in mixedCase
Parameter DFYNWorldCup.buy(uint256,uint256)._id (contracts/DFYNWorldCup.sol#84) is not in mixedCase
Parameter DFYNWorldCup.setWinners(uint256,uint256)._idWinner (contracts/DFYNWorldCup.sol#106) is not in mixedCase
Parameter DFYNWorldCup.setWinners(uint256,uint256)._idRunnerUp (contracts/DFYNWorldCup.sol#106) is not in mixedCase
Parameter DFYNWorldCup.setEndTime(uint256)._endTime (contracts/DFYNWorldCup.sol#124) is not in mixedCase
Parameter DFYNWorldCup.claimRewardExact(uint256,uint256)._id (contracts/DFYNWorldCup.sol#154) is not in mixedCase
Parameter DFYNWorldCup.claimRewardExact(uint256,uint256)._tokenAmount (contracts/DFYNWorldCup.sol#154) is not in mixedCase
Parameter DFYNWorldCup.claimableExact(uint256,uint256)._id (contracts/DFYNWorldCup.sol#192) is not in mixedCase
Parameter DFYNWorldCup.claimableExact(uint256,uint256)._tokenAmount (contracts/DFYNWorldCup.sol#192) is not in mixedCase
Variable DFYNWorldCup.BASE_PRICE (contracts/DFYNWorldCup.sol#19) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

ERC1155SupplyUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol#94) is never used in DFYNWorldCup (contracts/DFYNWorldCup.sol#11-229)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#59-61)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#67-70)
uri(uint256) should be declared external:
        - ERC1155Upgradeable.uri(uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#65-67)
balanceOfBatch(address[],uint256[]) should be declared external:
        - ERC1155Upgradeable.balanceOfBatch(address[],uint256[]) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#88-104)
setApprovalForAll(address,bool) should be declared external:
        - ERC1155Upgradeable.setApprovalForAll(address,bool) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#109-114)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155Upgradeable.safeTransferFrom(address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#126-138)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155Upgradeable.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#143-155)
exists(uint256) should be declared external:
        - ERC1155SupplyUpgradeable.exists(uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol#37-39)
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#61-63)
symbol() should be declared external:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#69-71)
decimals() should be declared external:
        - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#86-88)
totalSupply() should be declared external:
        - ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#93-95)
balanceOf(address) should be declared external:

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.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-upgradeable/proxy/utils/Initializable.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-upgradeable/token/ERC1155/ERC1155Upgradeable.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-upgradeable/token/ERC1155/IERC1155ReceiverUpgradeable.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-upgradeable/token/ERC1155/IERC1155Upgradeable.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-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.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-upgradeable/token/ERC1155/extensions/IERC1155MetadataURIUpgradeable.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-upgradeable/token/ERC20/IERC20Upgradeable.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-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.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-upgradeable/utils/AddressUpgradeable.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-upgradeable/utils/ContextUpgradeable.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-upgradeable/utils/introspection/ERC165Upgradeable.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-upgradeable/utils/introspection/IERC165Upgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.8.4 (contracts/DFYNWorldCup.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.7 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>=0.8.4 (contracts/test/TestToken.sol#3) 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

Low level call in AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#54-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#57)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#122-133):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#131)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#151-160):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#158)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function OwnableUpgradeable.__Ownable_init() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#28-31) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#33-35) is not in mixedCase
Variable OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#77) is not in mixedCase
Function ERC1155Upgradeable.__ERC1155_init(string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#35-39) is not in mixedCase
Function ERC1155Upgradeable.__ERC1155_init_unchained(string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#41-43) is not in mixedCase
Variable ERC1155Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#458) is not in mixedCase
Function ERC1155SupplyUpgradeable.__ERC1155Supply_init() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol#17-21) is not in mixedCase
Function ERC1155SupplyUpgradeable.__ERC1155Supply_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol#23-24) is not in mixedCase
Variable ERC1155SupplyUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol#94) is not in mixedCase
Function ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#17-19) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#23-25) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#27-28) is not in mixedCase
Variable ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35) is not in mixedCase
```

```
name() should be declared external:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#61-63)
symbol() should be declared external:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#69-71)
decimals() should be declared external:
        - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#86-88)
totalSupply() should be declared external:
        - ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#93-95)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#100-102)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#112-115)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#120-122)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#131-134)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#149-163)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#177-180)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#196-204)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
. analyzed (19 contracts with 75 detectors), 115 result(s) found
```

## Results

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

# Unit Tests

```
root@SecTests:~/nft-prediction-contract-main# npx hardhat test
No need to generate any newer typings.


  E2E ERC20 - Two EVM Chains
    ✓ e2e: 3 users buy and ONE winner claims. [No RunnerUp] (888ms)
    ✓ e2e: 3 users buy and TWO winners claims. [No RunnerUp] (852ms)
    ✓ e2e: 3 users buy and ONE winner claims, with exact [No RunnerUp] (560ms)
    ✓ e2e: 3 users buy all, and THREE winners claims, with exact and all [No RunnerUp] (1665ms)


  4 passing (8s)
```

# Functional test

| Function Names | Technical Result | Logical Result | Overall |
|---|---|---|---|
| initialize | PASS | PASS | PASS |
| buy | PASS | PASS | PASS |
| addTeams | PASS | PASS | PASS |
| setWinners | PASS | PASS | PASS |
| setEndTime | PASS | PASS | PASS |
| claimRewardAll | PASS | PASS | PASS |
| claimRewardExact | PASS | PASS | PASS |
| claimableAll | PASS | PASS | PASS |
| claimableExact | PASS | PASS | PASS |
| withdrawTreasury | PASS | PASS | PASS |
| emergencyWithdraw | PASS | PASS | PASS |

# Closing Summary

Overall, smart contracts are very well written and adhere to guidelines. Many issues were discovered during the Final audit; it is recommended to fix them before deployment.

# Disclaimer

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

# Audit Report
# November, 2021

For

QuillAudits


Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com