Homework 7 Nadine Chancay

Security

Look at the following contract, there are a number of vulnerabilities and flaws. In your teams try to find all of the problems.
You do not need to fix any of the problems.

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.4;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract BadLotteryGame {
    uint256 public prizeAmount;         // payout amount
    address payable[] public players;
    uint256 public num_players;
    address payable[] public prize_winners;
    event winnersPaid(uint256);

    constructor() {}

    function addNewPlayer(address payable _playerAddress) public payable {
        if (msg.value == 500000) {
            players.push(_playerAddress);
        }
        num_players++;
        if (num_players > 50) {
            emit winnersPaid(prizeAmount);
        }
    }

    function pickWinner(address payable _winner) public {
        if ( block.timestamp % 15 == 0){    // use timestamp for random number
            prize_winners.push(_winner);
        }
    }

    function payout() public {
        if (address(this).balance == 500000 * 100) {
            uint256 amountToPay = prize_winners.length / 100;
            distributePrize(amountToPay);
        }
    }

    function distributePrize(uint256 _amount) public {
        for (uint256 i = 0; i <= prize_winners.length; i++) {
            prize_winners[i].transfer(_amount);
```

```
        }
    }
}
```

Problems

- The contract imports the ERC20 contract from the OpenZeppelin library, but it does not use any of its functions or variables. This may cause unnecessary gas cost.
- `prizeAmount` is never used in any of the contract's functions.
- The pickWinner function uses the block timestamp to generate a random number, which is not truly random.
- The `addNewPlayer` function does not validate the input parameter `_playerAddress`, which can potentially allow an attacker to add a non-payable address or a contract address that may execute malicious code.
- The pragma has been stated as ^0.8.4. It is generally recommended to lock the pragma version for production contracts to ensure that the code behaves consistently across different compiler versions.
- The compiler version used in the contract is outdated and may have security vulnerabilities and other issues.
- Some functions have been specified as public that should be private. Anyone can call the functions in the contract when only authorized parties should access or manipulate sensitive data.