```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v4.7.0) (proxy/ERC1967/ERC1967Proxy.sol)

pragma solidity ^0.8.0;

import "../Proxy.sol";
import "./ERC1967Upgrade.sol";

/**
 * @dev This contract implements an upgradeable proxy. It is upgradeable because calls are delegated to an
 * implementation address that can be changed. This address is stored in storage in the location specified by
 * https://eips.ethereum.org/EIPS/eip-1967[EIP1967], so that it doesn't conflict with the storage layout of the
 * implementation behind the proxy.
 */
contract ERC1967Proxy is Proxy, ERC1967Upgrade {
    /**
     * @dev Initializes the upgradeable proxy with an initial implementation specified by `_logic`.
     *
     * If `_data` is nonempty, it's used as data in a delegate call to `_logic`. This will typically be an encoded
     * function call, and allows initializing the storage of the proxy like a Solidity constructor.
     */
    constructor(address _logic, bytes memory _data) payable {
        _upgradeToAndCall(_logic, _data, false);
    }

    /**
     * @dev Returns the current implementation address.
     */
    function _implementation() internal view virtual override returns (address impl) {
        return ERC1967Upgrade._getImplementation();
    }
}

```