```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts v4.4.1 (proxy/beacon/UpgradeableBeacon.sol)

pragma solidity ^0.8.0;

import "./IBeacon.sol";
import "../../access/Ownable.sol";
import "../../utils/Address.sol";

/**
 * @dev This contract is used in conjunction with one or more instances of {BeaconProxy} to determine their
 * implementation contract, which is where they will delegate all function calls.
 *
 * An owner is able to change the implementation the beacon points to, thus upgrading the proxies that use this beacon.
 */
contract UpgradeableBeacon is IBeacon, Ownable {
    address private _implementation;

    /**
     * @dev Emitted when the implementation returned by the beacon is changed.
     */
    event Upgraded(address indexed implementation);

    /**
     * @dev Sets the address of the initial implementation, and the deployer account as the owner who can upgrade the
     * beacon.
     */
    constructor(address implementation_) {
        _setImplementation(implementation_);
    }

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

    /**
     * @dev Upgrades the beacon to a new implementation.
     *
     * Emits an {Upgraded} event.
     *
     * Requirements:
     *
     * - msg.sender must be the owner of the contract.
     * - `newImplementation` must be a contract.
     */
    function upgradeTo(address newImplementation) public virtual onlyOwner {
        _setImplementation(newImplementation);
        emit Upgraded(newImplementation);
    }

    /**
     * @dev Sets the implementation contract address for this beacon
     *
     * Requirements:
     *
     * - `newImplementation` must be a contract.
     */
    function _setImplementation(address newImplementation) private {
        require(Address.isContract(newImplementation), "UpgradeableBeacon: implementation is not a contract");
        _implementation = newImplementation;
    }
}

```