```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v4.7.0) (security/Pausable.sol)

pragma solidity ^0.8.0;

import "../utils/Context.sol";

/**
 * @dev Contract module which allows children to implement an emergency stop
 * mechanism that can be triggered by an authorized account.
 *
 * This module is used through inheritance. It will make available the
 * modifiers `whenNotPaused` and `whenPaused`, which can be applied to
 * the functions of your contract. Note that they will not be pausable by
 * simply including this module, only once the modifiers are put in place.
 */
abstract contract Pausable is Context {
    /**
     * @dev Emitted when the pause is triggered by `account`.
     */
    event Paused(address account);

    /**
     * @dev Emitted when the pause is lifted by `account`.
     */
    event Unpaused(address account);

    bool private _paused;

    /**
     * @dev Initializes the contract in unpaused state.
     */
    constructor() {
        _paused = false;
    }

    /**
     * @dev Modifier to make a function callable only when the contract is not paused.
     *
     * Requirements:
     *
     * - The contract must not be paused.
     */
    modifier whenNotPaused() {
        _requireNotPaused();
        _;
    }

    /**
     * @dev Modifier to make a function callable only when the contract is paused.
     *
     * Requirements:
     *
     * - The contract must be paused.
     */
    modifier whenPaused() {
        _requirePaused();
        _;
    }

    /**
     * @dev Returns true if the contract is paused, and false otherwise.
     */
    function paused() public view virtual returns (bool) {
        return _paused;
    }

    /**
     * @dev Throws if the contract is paused.
     */
    function _requireNotPaused() internal view virtual {
        require(!paused(), "Pausable: paused");
    }
```

```
74
75          /**
76           * @dev Throws if the contract is not paused.
77           */
78          function _requirePaused() internal view virtual {
79              require(paused(), "Pausable: not paused");
80          }
81
82          /**
83           * @dev Triggers stopped state.
84           *
85           * Requirements:
86           *
87           * - The contract must not be paused.
88           */
89          function _pause() internal virtual whenNotPaused {
90              _paused = true;
91              emit Paused(_msgSender());
92          }
93
94          /**
95           * @dev Returns to normal state.
96           *
97           * Requirements:
98           *
99           * - The contract must be paused.
100          */
101         function _unpause() internal virtual whenPaused {
102             _paused = false;
103             emit Unpaused(_msgSender());
104         }
105     }
106
```