A region is captured if it is not surrounded by 'X'. The task is to modify the board in-place to capture the surrounded regions.
Here's a common approach to solving this problem:

- **Traversal and Marking:**Start by traversing the border of the board, and if you encounter 'O', perform a Depth-First Search (DFS) or Breadth-First Search (BFS) to mark all connected 'O' cells as safe. You can use a marker (e.g., change 'O' to 'S') to indicate these safe cells.
- **Capture Surrounded Regions:**After marking all safe regions, iterate through the entire board. If you encounter 'O', it means it is not on the border, and if it is not marked as safe ('S'), then it is surrounded. In this case, change 'O' to 'X'.
- **Restore Unsurrounded Regions:**Finally, iterate through the board again and restore the marked safe regions back to 'O'. Change the markers ('S') back to 'O'.

CODE

```
def solve(board):

    if not board or not board[0]:
        return
    rows, cols = len(board), len(board[0])
    # Step 1: Mark safe regions connected to the border
    for i in range(rows):
        if board[i][0] == 'O':
            dfs(board, i, 0)
        if board[i][cols - 1] == 'O':
            dfs(board, i, cols - 1)
    for j in range(cols):
        if board[0][j] == 'O':
            dfs(board, 0, j)
        if board[rows - 1][j] == 'O':
            dfs(board, rows - 1, j)
    # Step 2: Capture surrounded regions
    for i in range(rows):
        for j in range(cols):
            if board[i][j] == 'O':
                board[i][j] = 'X'
            elif board[i][j] == 'S':
                board[i][j] = 'O'
def dfs(board, i, j):
    if i < 0 or i >= len(board) or j < 0 or j >= len(board[0]) or board[i][j] != 'O':
        return
    board[i][j] = 'S'
    dfs(board, i - 1, j)
    dfs(board, i + 1, j)
    dfs(board, i, j - 1)
    dfs(board, i, j + 1)
```