```
import math

# Variables to count nodes explored
nodes_explored_minimax = 0
nodes_explored_alpha_beta = 0

# Basic Minimax Function
def minimax(board, depth, maximizing):
    global nodes_explored_minimax
    nodes_explored_minimax += 1
    # Count each node visited

    # Terminal conditions
    if check_winner(board, 'Osaid'):
        return 1
    elif check_winner(board, 'Abubakar'):
        return -1
    elif board_full(board):
        return 0

    # Maximizing player
    if maximizing:
        max_eval = -math.inf
        for i in range(9):
            if board[i] == ' ':
                board[i] = 'Osaid'
                eval = minimax(board, depth + 1, False)
                board[i] = ' '
                max_eval = max(max_eval, eval)
        return max_eval
    # Minimizing player
    else:
        min_eval = math.inf
        for i in range(9):
            if board[i] == ' ':
                board[i] = 'Abubakar'
                eval = minimax(board, depth + 1, True)
                board[i] = ' '
                min_eval = min(min_eval, eval)
        return min_eval

# Minimax with Alpha-Beta Pruning
def minimax_alpha_beta(board, depth, alpha, beta, maximizing):
    global nodes_explored_alpha_beta
    nodes_explored_alpha_beta += 1  # Count each node visited

    # Terminal conditions
    if check_winner(board, 'Osaid'):
        return 1
    elif check_winner(board, 'Abubakar'):
        return -1
    elif board_full(board):
        return 0

    # Maximizing player
    if maximizing:
        max_eval = -math.inf
        for i in range(9):
            if board[i] == ' ':
                board[i] = 'Osaid'
                eval = minimax_alpha_beta(board, depth + 1, alpha, beta, False)
                board[i] = ' '
                max_eval = max(max_eval, eval)
                alpha = max(alpha, eval)
                if beta <= alpha:
                    break  # Prune the branch
        return max_eval
    # Minimizing player
    else:
        min_eval = math.inf
        for i in range(9):
            if board[i] == ' ':
                board[i] = 'Abubakar'
                eval = minimax_alpha_beta(board, depth + 1, alpha, beta, True)
                board[i] = ' '
                min_eval = min(min_eval, eval)
                beta = min(beta, eval)
                if beta <= alpha:
                    break  # Prune the branch
        return min_eval

# Example Test Run
# Initialize board to an empty Tic-Tac-Toe board
board = [' ' for _ in range(9)]

# Reset node counters
nodes_explored_minimax = 0
nodes_explored_alpha_beta = 0

# Run Minimax without Alpha-Beta Pruning
minimax(board, 0, True)
print("Nodes explored by Minimax:", nodes_explored_minimax)

# Run Minimax with Alpha-Beta Pruning
minimax_alpha_beta(board, 0, -math.inf, math.inf, True)
print("Nodes explored by Alpha-Beta Pruning:", nodes_explored_alpha_beta)
```

```
Nodes explored by Minimax: 549946
Nodes explored by Alpha-Beta Pruning: 18297
```