PPID Spoofing

```cpp
C++ Code
#include <windows.h>
#include <iostream>
#include <tlhelp32.h>

int getPIDbyProcName(const char* procName) {
    int pid = 0;
    HANDLE hSnap = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
    PROCESSENTRY32 pe32;
    pe32.dwSize = sizeof(PROCESSENTRY32);
    if (Process32First(hSnap, &pe32) != FALSE) {
        while (pid == 0 && Process32Next(hSnap, &pe32) != FALSE) {
            if (strcmp(pe32.szExeFile, procName) == 0) {
                pid = pe32.th32ProcessID;
            }
        }
    }
    CloseHandle(hSnap);
    return pid;
}

int main() {
    const char* exePath = "C:\\Windows\\System32\\notepad.exe";
    DWORD parentPID = getPIDbyProcName("mspaint.exe");

    STARTUPINFOEX info = { sizeof(STARTUPINFOEX) }; // Changed sizeof(0) to sizeof(STARTUPINFOEX)
    PROCESS_INFORMATION processInfo;
    SIZE_T cbAttributeListSize = 0;
    PPROC_THREAD_ATTRIBUTE_LIST pAttributeList = NULL;
    HANDLE hParentProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE, parentPID); // Open the parent process with all access

    InitializeProcThreadAttributeList(NULL, 1, 0, &cbAttributeListSize); // Get the size of the attribute list
    pAttributeList = (PPROC_THREAD_ATTRIBUTE_LIST)HeapAlloc(GetProcessHeap(), 0, cbAttributeListSize);  // Allocate memory for the attribute list
    InitializeProcThreadAttributeList(pAttributeList, 1, 0, &cbAttributeListSize); // Initialize the attribute list

    UpdateProcThreadAttribute(pAttributeList, 0, PROC_THREAD_ATTRIBUTE_PARENT_PROCESS, &hParentProcess, sizeof(HANDLE), NULL, NULL); // Update the attribute list with the parent process

    info.lpAttributeList = pAttributeList; // Set the attribute list in the startup info

    CreateProcess(NULL, (LPSTR)exePath, NULL, NULL, TRUE, EXTENDED_STARTUPINFO_PRESENT | CREATE_NO_WINDOW, NULL, NULL, reinterpret_cast<STARTUPINFO*>(&info), &processInfo);

    printf("Process created with PID: %d\n", processInfo.dwProcessId);
    printf("Parent PID: %d\n", parentPID);
    return 0;
}
```

This C++ code appears to be creating a new process in Windows with specific attributes, particularly specifying a parent process. Let's break down the code step by step:

1. Includes:
   - `#include <windows.h>`: This header file includes various Windows API functions and data types required for system-level programming.
   - `#include <iostream>`: This header file provides input and output stream functionality.
   - `#include <tlhelp32.h>`: This header file includes functions and structures for working with processes and threads.
2. `getPIDbyProcName` Function:
   - This function takes the name of a process as input (`procName`) and returns the Process ID (PID) of the first process with a matching name.
   - It uses the Windows ToolHelp API to enumerate running processes and find the PID of the specified process name.
   - It iterates through the running processes, comparing the names until it finds a match or reaches the end of the process list.
3. `main` Function:
   - The `main` function is the entry point of the program.
   - It defines the path to the executable (`exePath`) that it intends to run (`notepad.exe` in this case).
   - It calls `getPIDbyProcName` to find the PID of the parent process (`mspaint.exe` in this case).
   - It sets up the `STARTUPINFOEX` structure, which provides extended information about how the new process should be created.
   - It allocates memory for a process thread attribute list (`pAttributeList`) using the `HeapAlloc` function.
   - It initializes and updates the process thread attribute list to specify the parent process (the one with PID `parentPID`).
   - Finally, it creates a new process using `CreateProcess`, passing the executable path, the attribute list, and other parameters. This will start the new process.
   - The program then prints the PID of the newly created process and the PID of the parent process.