Code Injection and Thread Hijacking - Process Flow

1. Process Discovery

The function `SearchForProcess` scans all active processes and searches for a match with a specified name (e.g., mspaint.exe). It returns the Process ID (PID) of the matching process.

2. Thread Discovery

`SearchForThread` uses the PID to find and return a handle to one of the threads running in that process.

3. Payload Injection

In `InjectCTX`, memory is allocated in the target process using `VirtualAllocEx`, and a shellcode payload is written into that space using `WriteProcessMemory`.

4. Thread Hijacking

The target thread is suspended with `SuspendThread`, its context (registers) is retrieved, and the instruction pointer (`Eip` for 32-bit, `Rip` for 64-bit) is modified to point to the injected shellcode.

5. Execution

After modifying the context with `SetThreadContext`, the thread is resumed with `ResumeThread`, causing it to begin executing the injected shellcode.

6. Cleanup

Handles to the process and thread are closed after injection to clean up resources.

Architecture Handling

The code uses `#ifdef _M_IX86` to differentiate between 32-bit and 64-bit systems, ensuring the correct instruction pointer register is used (`Eip` vs `Rip`).