

# BÁO CÁO BÀI TẬP

Môn học: Cơ chế hoạt động của mã độc

Kỳ báo cáo: Buổi 05 (Session 05)

Tên chủ đề: DLL Injection

GV: Nghi Hoàng Khoa

Ngày báo cáo: 25/5/2023

## 1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT230.N21.ANTN

STT	Họ và tên	MSSV	Email
1	Võ Anh Kiệt	20520605	20520605@gm.uit.edu.vn

## 2. NỘI DUNG THỰC HIỆN:<sup>1</sup>

STT	Công việc	Kết quả tự đánh giá	Người đóng góp
1	Kịch bản 01	100%	
2			
3			
4			

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

---

<sup>1</sup> Ghi nội dung công việc, các kịch bản trong bài Thực hành

# BÁO CÁO CHI TIẾT

## 1. Kịch bản 01

Link video: <https://youtu.be/A7l9G-Bvt3M>

Giải thích code source:

Bước 1: Ở phần code Source thì ta thấy được là đầu tiên thực hiện get các thông tin về các tiến trình

```
if (hGameWindow == NULL) {
    std::cout << "Start the game!" << std::endl;
    return 0;
}

DWORD pID = NULL; // ID of our Game
GetWindowThreadProcessId(hGameWindow, &pID);
HANDLE processHandle = NULL;
processHandle = OpenProcess(PROCESS_ALL_ACCESS, FALSE, pID);
if (processHandle == INVALID_HANDLE_VALUE || processHandle == NULL) { // error handling
    std::cout << "Failed to open process" << std::endl;
    return 0;
}

TCHAR gameName[13];
wcscpy_s(gameName, 13, L"EMPIRESX.EXE");

DWORD gameBaseAddress = GetModuleBaseAddress(gameName, pID);

std::cout << "debuginfo: gameBaseAddress = " << gameBaseAddress << std::endl;

DWORD offsetGameToBaseAddress = 0x003C4B18;
std::vector<DWORD> pointsOffsets{ 0x3c, 0x100, 0x50, 0x0 };

DWORD baseAddress = NULL;
```

Bước 2: Sau đó thực hiện đọc và thao tác trên bộ nhớ

```
//Get value at gamebase+offset -> store it in baseAddress
ReadProcessMemory(processHandle, (LPVOID)(gameBaseAddress+ offsetGameToBaseAddress), &baseAddress, sizeof(baseAddress), NULL);
std::cout << "debuginfo: baseaddress = " << std::hex << baseAddress << std::endl;

DWORD pointsAddress = baseAddress; //the Address we need -> change now while going through offsets
for (int i = 0; i < pointsOffsets.size() - 1; i++) // -1 because we dont want the value at the last offset
{
    ReadProcessMemory(processHandle, (LPVOID)(pointsAddress + pointsOffsets.at(i)), &pointsAddress, sizeof(pointsAddress), NULL);
    std::cout << "debuginfo: Value at offset = " << std::hex << pointsAddress << std::endl;
}
pointsAddress += pointsOffsets.at(pointsOffsets.size() - 1); //Add Last offset -> done!!
float currentPoint = 0;

std::cout << sizeof(currentPoint) << std::endl;
ReadProcessMemory(processHandle, (LPVOID)(pointsAddress), &currentPoint, sizeof(currentPoint), NULL);
std::cout << "The last address is:" << pointsAddress << std::endl;
std::cout << "Current value is:" << currentPoint << std::endl;
```

Bước 3: tác động đến các giá trị biến từ đó có thể thay đổi được các giá trị biến trong trò chơi

```
// "UI"
std::cout << "Age of Empires Hack" << std::endl;

std::cout << "How many points you want?" << std::endl;
float newPoints = 0;
std::cin >> newPoints;
WriteProcessMemory(processHandle, (LPVOID)(pointsAddress), &newPoints, 4, 0);
```

Giải thích code Injector:

Bước 1: Tương tự như source, injector cũng thực hiện một số thao tác get các thông tin về tiến trình

```
HANDLE hProcess;
LPVOID pszLibFileRemote = NULL;
HANDLE handleThread;
const wchar_t* process = L"EMPIRESX.EXE";
int pID = getProcId(process);

char dll[] = "AOEResourceHack.dll";
if (!exist(dll)) {
    std::cout << "debuginfo: Invalid DLL path!" << std::endl;
}

char dllPath[MAX_PATH] = { 0 }; // full path of DLL
GetFullPathNameA(dll, MAX_PATH, dllPath, NULL);
std::cout << "debuginfo: Full DLL path: " << dllPath << std::endl;

hProcess = OpenProcess(PROCESS_QUERY_INFORMATION | PROCESS_CREATE_THREAD | PROCESS_VM_OPERATION | PROCESS_VM_WRITE, 0, pID);

if (hProcess) {
    pszLibFileRemote = VirtualAllocEx(hProcess, NULL, strlen(dllPath) + 1, MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
}
else {
    std::cout << "error: Could not open process handle with id:" << pID << std::endl;
}
```

Bước 2: thực hiện đọc và thao tác trên bộ nhớ

```
int isWriteOK = WriteProcessMemory(hProcess, pszLibFileRemote, dllPath, strlen(dllPath) + 1, NULL);
if (!isWriteOK) std::cout << "error: Failed to write" << std::endl;

handleThread = CreateRemoteThread(hProcess, NULL, NULL, (LPTHREAD_START_ROUTINE)LoadLibraryA, pszLibFileRemote, NULL, NULL);
if (handleThread == NULL) {
    std::cout << "error: Failed to create thread" << std::endl;
    ErrorExit(_T("CreateRemoteThread"));
}
```

Bước 3: chương trình sẽ tạo ra các thread để thực hiện việc hỗ trợ truyền các DLL độc hại vào trong chương trình trò chơi

```
WaitForSingleObject(handleThread, INFINITE);
CloseHandle(handleThread);
VirtualFreeEx(hProcess, dllPath, 0, MEM_RELEASE);
CloseHandle(hProcess);
```

Như vậy để có thể thực hiện được việc nhấn F6 tăng điểm, ta sẽ thực hiện chỉnh sửa DLL

Đầu tiên ta sẽ sử dụng code source bỏ vào DLL

```
// dllmain.cpp : Defines the entry point for the DLL application.
#include "pch.h"
#include <Windows.h>
#include <TLHelp32.h>
#include <iostream>
#include <tchar.h> // _tcsncmp
#include <vector>

DWORD GetModuleBaseAddress(TCHAR* lpzModuleName, DWORD pID) {
    DWORD dwModuleBaseAddress = 0;
    HANDLE hSnapshot = CreateToolhelp32Snapshot(TH32CS_SNAPMODULE, pID); // make snapshot of all modules within process
    MODULEENTRY32 ModuleEntry32 = { 0 };
    ModuleEntry32.dwSize = sizeof(MODULEENTRY32);

    if (Module32First(hSnapshot, &ModuleEntry32)) //store first Module in ModuleEntry32
    {
        do {
            if (_tcsncmp(ModuleEntry32.szModule, lpzModuleName) == 0) // if Found Module matches Module we look for -> done!
            {
                dwModuleBaseAddress = (DWORD)ModuleEntry32.modBaseAddr;
                break;
            }
        } while (Module32Next(hSnapshot, &ModuleEntry32)); // go through Module entries in Snapshot and store in ModuleEntry32
    }

    CloseHandle(hSnapshot);
}
```

Thực hiện một số chỉnh sửa, thực hiện đổi tên hàm

```
int AddHundred() {

    HWND hGameWindow = FindWindow(NULL, L"Age of Empires Expansion");
    if (hGameWindow == NULL) {
        std::cout << "Start the game!" << std::endl;
        return 0;
    }

    DWORD pID = NULL; // ID of our Game
    GetWindowThreadProcessId(hGameWindow, &pID);
    HANDLE processHandle = NULL;
    processHandle = OpenProcess(PROCESS_ALL_ACCESS, FALSE, pID);
    if (processHandle == INVALID_HANDLE_VALUE || processHandle == NULL) { // error handling
        std::cout << "Failed to open process" << std::endl;
        return 0;
    }
}
```

Ngoài ra ta sẽ chuyển đổi phần nhập liệu thành lấy giá trị hiện tại cộng thêm 100

```
std::cout << "How many points you want?" << std::endl;
float newPoints = currentPoint + 100;
// std::cin >> newPoints;
WriteProcessMemory(processHandle, (LPVOID)(pointsAddress), &newPoints, 4, 0);
```

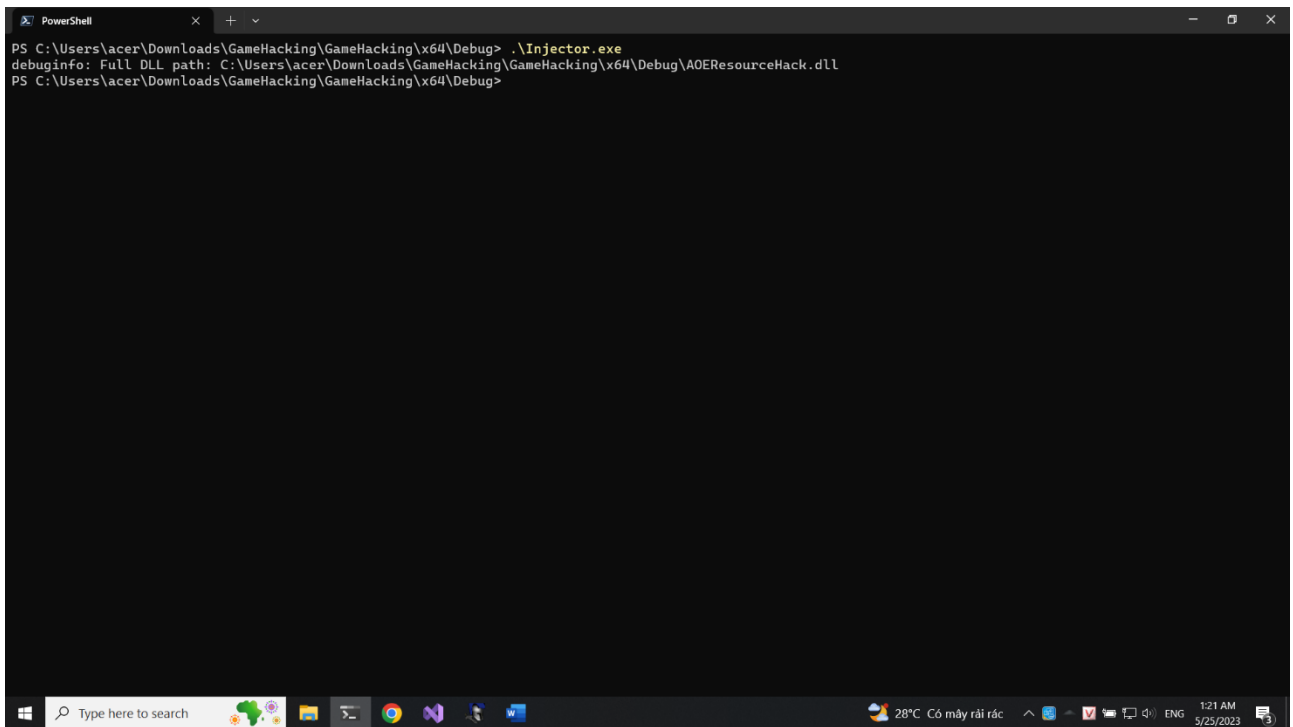
Thêm việc thực hiện hàm khi nhấn F6

```
DWORD WINAPI MainThread(LPVOID param) {  
    while (true) {  
        if (GetAsyncKeyState(VK_F6) & 0x800000) {  
            MessageBoxA(NULL, "F6 Pressed!", "F6 Pressed!", MB_OK);  
            AddHundred();  
        }  
        Sleep(100);  
    }  
    return 0;  
}
```

Sau đó ta sẽ thực thi chạy trò chơi

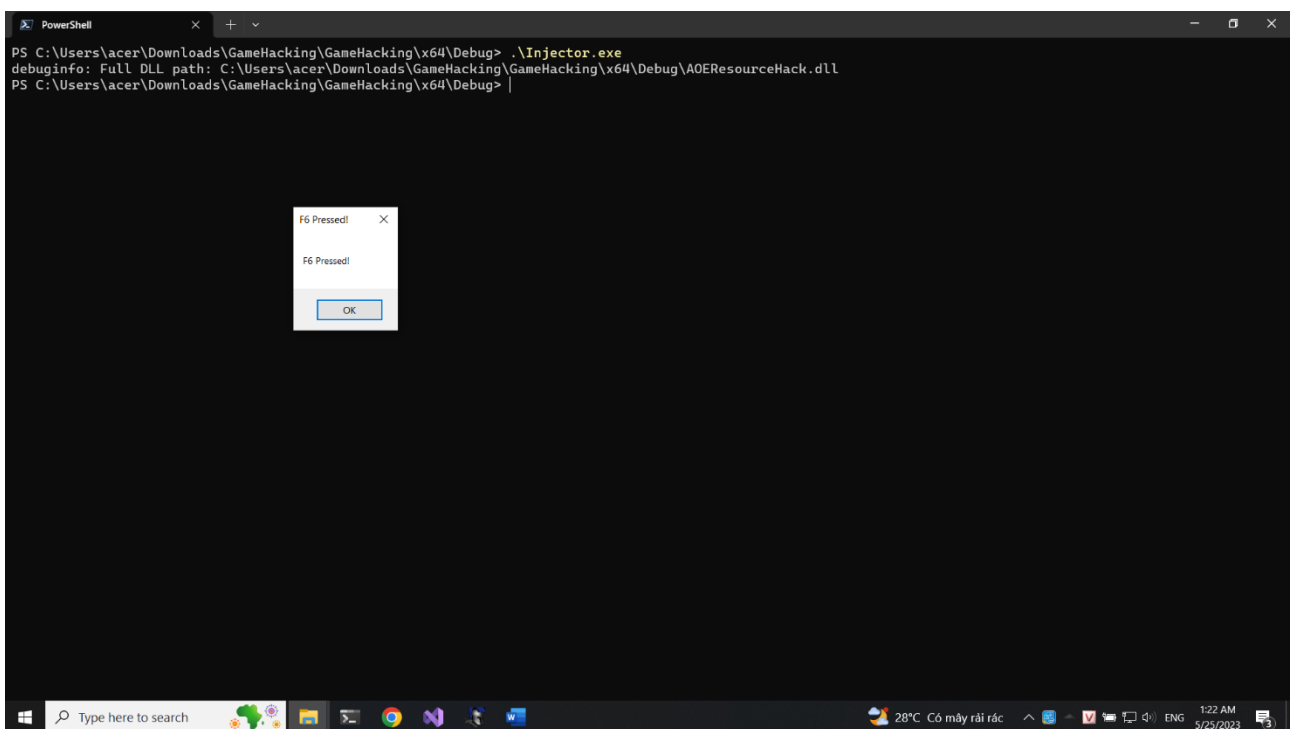


Thực hiện chạy file Injector



```
PS C:\Users\acer\Downloads\GameHacking\GameHacking\x64\Debug> .\Injector.exe
debuginfo: Full DLL path: C:\Users\acer\Downloads\GameHacking\GameHacking\x64\Debug\A0EResourceHack.dll
PS C:\Users\acer\Downloads\GameHacking\GameHacking\x64\Debug>
```

Quay lại trò chơi nhấn F6 và lúc này ta nhận được thông báo nhấn F6



```
PS C:\Users\acer\Downloads\GameHacking\GameHacking\x64\Debug> .\Injector.exe
debuginfo: Full DLL path: C:\Users\acer\Downloads\GameHacking\GameHacking\x64\Debug\A0EResourceHack.dll
PS C:\Users\acer\Downloads\GameHacking\GameHacking\x64\Debug> |
```

F6 Pressed!

F6 Pressed!

OK

Cuối cùng ta thấy thức ăn tăng lên 100 thành 300 so với 200 ban đầu đã có sự thay đổi





---

*Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này*

## YÊU CẦU CHUNG

- Sinh viên tìm hiểu và thực hành theo hướng dẫn.
- Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

### Báo cáo:

- File **.PDF**. Tập trung vào nội dung, không mô tả lý thuyết.
- Nội dung trình bày bằng **Font chữ Times New Romans/ hoặc font chữ của mẫu báo cáo này (UTM Neo Sans Intel/UTM Viet Sach)– cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.**
- Đặt tên theo định dạng: [Mã lớp]-SessionX\_GroupY. (trong đó X là Thứ tự buổi Thực hành, Y là số thứ tự Nhóm Thực hành/Tên Cá nhân đã đăng ký với GV).  
*Ví dụ: [NT101.K11.ANTT]-Session1\_Group3.*
- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- **Không đặt tên đúng định dạng – yêu cầu, sẽ KHÔNG chấm điểm.**
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

**Đánh giá:** Sinh viên hiểu và tự thực hiện. Khuyến khích:

- Chuẩn bị tốt.
- Có nội dung mở rộng, ứng dụng trong kịch bản/câu hỏi phức tạp hơn, có đóng góp xây dựng.

*Bài sao chép, trễ, ... sẽ được xử lý tùy mức độ vi phạm.*

**HẾT**