5

Lab

DLL injection

Thực hành Cơ chế hoạt động của Mã độc

Lưu hành nội bộ 2024

A. TỔNG QUAN

DLL Injection đã trở thành một công cụ khá phổ biến trong việc chỉnh sửa phần mềm. Phương thức này giúp ta điều chỉnh các tính năng của một ứng dụng mà không làm ảnh hưởng trực tiếp đến mã thực thi (binary). Và càng thông dụng hơn trong ngữ cảnh khi bạn muốn "hack" một game, chỉnh sửa tài nguyên và thậm chí logic của trò chơi.

Lab này lấy ngữ cảnh "hack" tài nguyên ấy, cụ thể là game Age of Empires: Rise of Rome - một tựa game đình đám ra mắt lần đầu tiên vào khoảng hơn hai thập kỷ trước. Với injection ta sẽ hướng tới việc chỉnh sửa tài nguyên là "thức ăn" trong game.

B. CHUẨN BỊ MÔI TRƯỜNG

- Mã nguồn game Age of Empires
- Windows OS
- Visual Studio

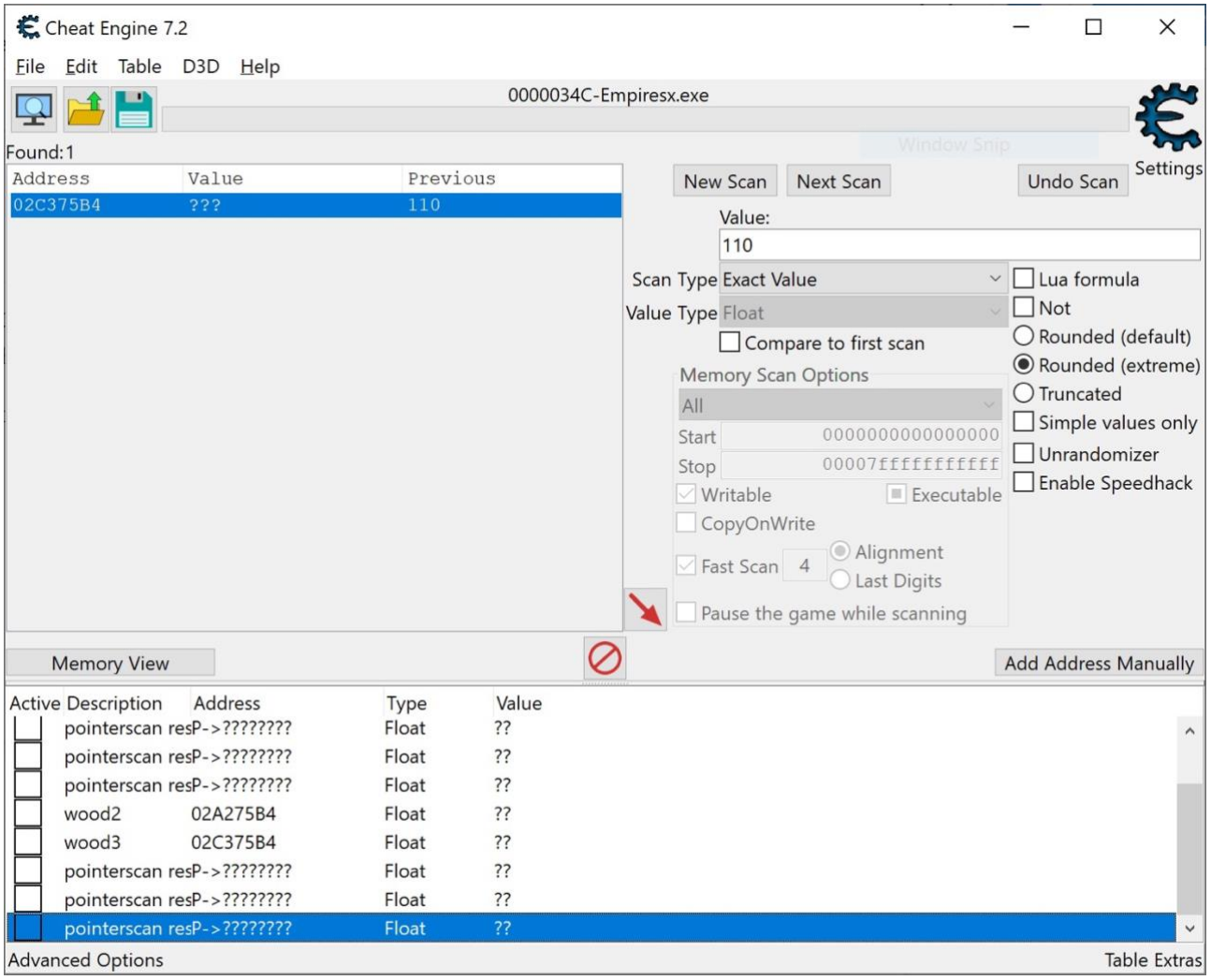
C. THỰC HÀNH

C.1 Cơ chế static pointers

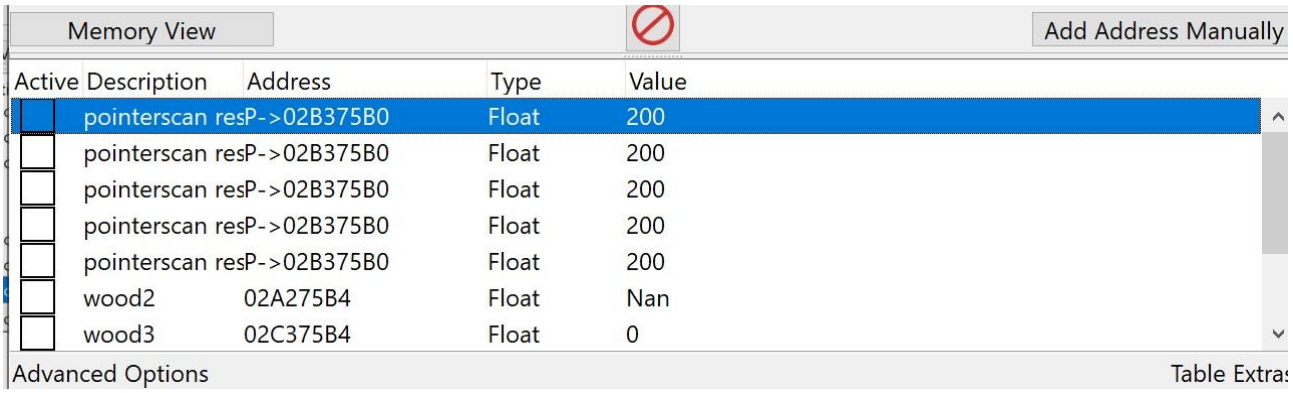
Ta biết được nếu muốn lưu trữ một đại lượng trong game, cụ thể ở đây là thức ăn, thì giá trị của nó sẽ được bỏ vào một biến. Cách đơn giản để "hack" đại lượng ấy và chỉnh thành giá trị ta mong muốn là tiếp cận bộ nhớ và ghi đè lên giá trị.

Nhưng khó khăn nằm ở việc xác định biến đấy trong không gian bộ nhớ của chương trình. Và bộ nhớ ảo thì không phải lúc nào cũng giống nhau sau khi ta khởi động lại game do cơ chế random hóa bộ nhớ của hệ điều hành.

Để xác định được vị trí của đại lượng thức ăn ta phải nhờ đến Cheat Engine, một công cụ quét bộ nhớ của chương trình.

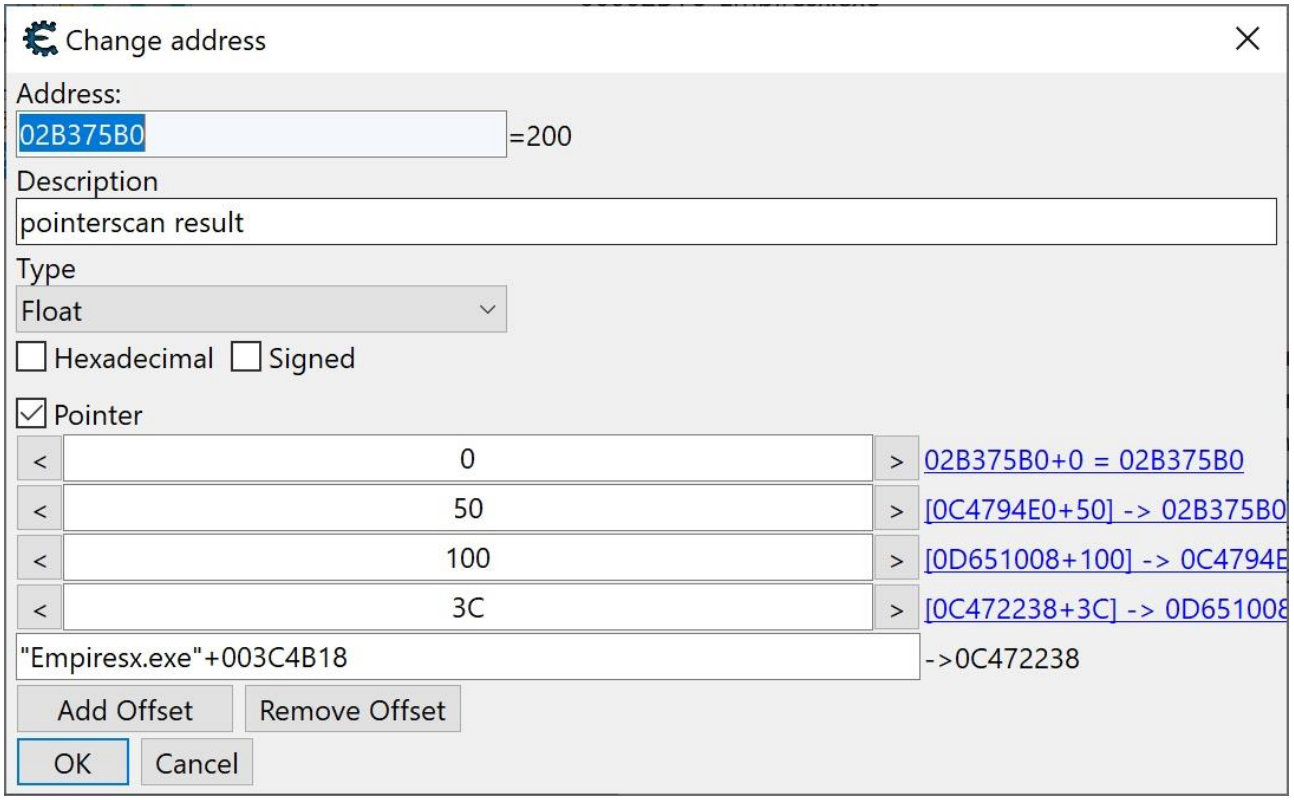


Chỉ cần ta thêm giá trị hiện tại của biến thức ăn và quét trong bộ nhớ của chương trình, Engine sẽ cho ta danh sách các địa chỉ khả nghi dẫn tới thức ăn.



Sau đấy ta sẽ tiến hành tạo pointermap để dẫn đến khác địa chỉ lưu đại lượng.

Đây là một pointermap mà ta cần tìm:

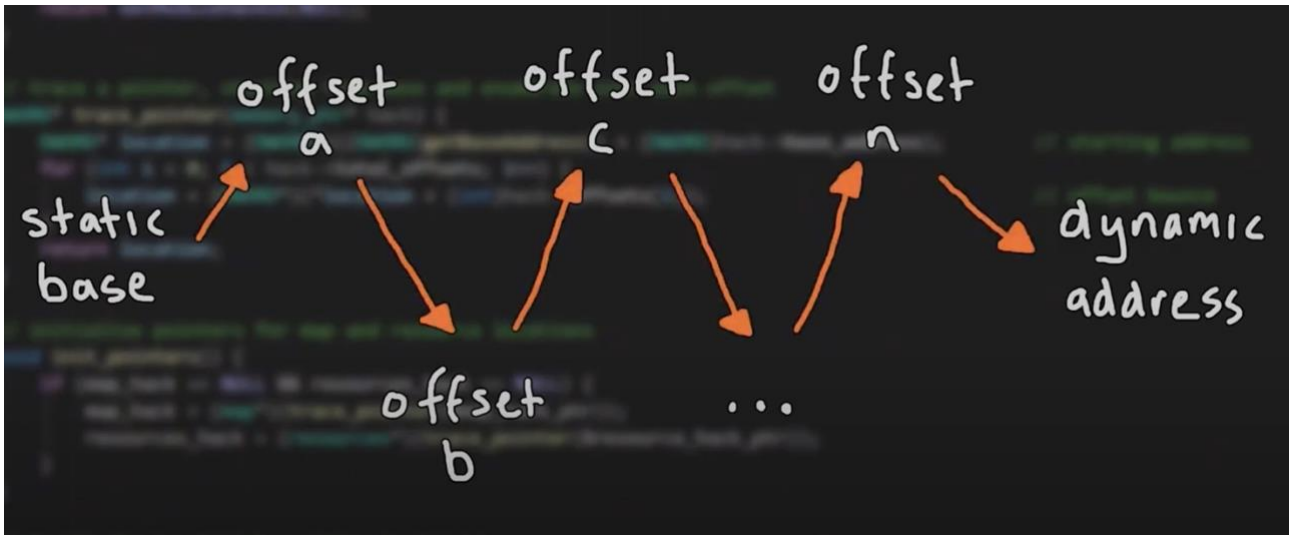


Do cơ chế random bộ nhớ ASLR (Address Space Layout Randomization), ta phải tìm vị trí của đại lượng dựa vào offsets. Ở đây Cheat Engine đã tìm được một "công thức" dẫn đến offset đó với các giá trị:

**(Empiresx.exe + base + *(offset1 + *(offset2 + offset3))) + offset4*

Với các offset là:

Base = 0x003C4B18, offset1 = 0x3C, offset2 = 0x100, offset3 = 0x50, offset4 = 0x00



Lưu ý đây chỉ là một trong nhiều "công thức" hợp lệ. Có nhiều cách để xác định địa chỉ của đại lượng. Đây chính là static pointer mà ta cần tìm.

Vậy chỉ cần xác định được Địa chỉ load của module Empiresx.exe và cộng theo công thức ta đã có thể chỉnh được đại lượng thức ăn!

C.2 Tạo DLL Injection

Ta sẽ Inject một DLL với tính năng là khi nhấn phím F6 thì chương trình sẽ tự động tăng thức ăn.

dllmain.cpp 3 ×

AOEResourceHack > dllmain.cpp > MainThread(LPVOID)

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

DWORD WINAPI MainThread(LPVOID param) {
    while (true) {
        if (GetAsyncKeyState(VK_F6) & 0x80000) {
            MessageBoxA(NULL, "F6 Pressed!", "F6 Pressed!", MB_OK);
        }
        Sleep(100);
    }
    return 0;
}


BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
        MessageBoxA(NULL, "DLL Injected!", "DLL Injected!", MB_OK);
        CreateThread(0, 0, MainThread, hModule, 0, 0);
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}

```

DLL sau khi được attached sẽ tiến hành tạo một thread để lắng nghe trạng thái của phím F6.

Hàm để patch:

```cpp
int main() {

    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;
    }

    TCHAR gameName[13];
    wcscpy_s(gameName, 13,L"Empiresx.exe");

    DWORD gameBaseAddress = GetModuleBaseAddress(gameName,pID);

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

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

Để patch giá trị, ta cần có Handle của process, địa chỉ nền của module (Empiresx.exe) và các giá trị offsets.

Quan trọng hơn hết vẫn là hàm xử lí địa chỉ và cập nhật giá trị:

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

    DWORD pointsAddress = baseAddress; //the Adress 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 << "debugginfo: 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;
    //"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);

}
```

Ở đoạn for loop, địa chỉ sẽ được tính toán như "công thức" được nêu trên và cuối cùng ta sẽ lấy giá trị hiện tại cộng thêm 100.

Vậy, với mỗi lần nhấn F6 ta sẽ được thêm 100 thức ăn!

> **Bài thực hành 1: Demo phần điểm thức ăn được nâng lên sau Inject.**

D. TÀI LIỆU THAM KHẢO

DLL injection - Wikipedia

Game Hacking 101 - YouTube

E. YÊU CẦU

Sinh viên báo cáo chi tiết PDF và quay video lại quá trình làm việc và nộp link youtube (chế độ unlisted) lên courses.

- Sinh viên tìm hiểu và thực hành theo hướng dẫn theo cá nhân.
- Báo cáo kết quả chi tiết những việc (**Report**) đã thực hiện, quan sát thấy, giải thích cho quan sát .

Báo cáo:

- Làm báo cáo trên file **mẫu**.
- File **.PDF**. Tập trung vào nội dung, không mô tả lý thuyết.
- Trong file báo cáo yêu cầu **ghi rõ** nhóm sinh viên thực hiện.
- Đặt tên theo định dạng: [Mã lớp]-Lab1_MSSV1-MSSV2.pdf
  Ví dụ: [NT330.K21.ANTN.1]-Lab2_1552xxxx-1552yyyy.pdf
- Nếu báo cáo có nhiều file, nén tất cả file vào file **.ZIP** với cùng tên file báo cáo.
- 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 được bài thực hành. Khuyến khích:

- Chuẩn bị tốt và đóng góp tích cực tại lớp.
- Có nội dung mở rộng, ứng dụng trong kịch bản phức tạp hơn, có đóng góp xây dựng bài thực hành.

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

-HẾT-