BÁO CÁO THỰC HÀNH

**Môn học: Cơ chế hoạt động của mã độc**

**Tên chủ đề: DLL Injection**

*GVHD: Tô Trọng Nghĩa*

**Nhóm: 07**

1. **THÔNG TIN CHUNG:**

*(Liệt kê tất cả các thành viên trong nhóm)*

Lớp: NT230.O21.ATCL.1

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Họ và tên | MSSV | Email |
| 1 | Chu Nguyễn Hoàng Phương | 21522483 | 21522483@gmail.com |
| 2 | Hồ Ngọc Thiện | 21522620 | 21522620@gmail.com |

1. **NỘI DUNG THỰC HIỆN:[[1]](#footnote-1)**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Nội dung | Tình trạng | Trang |
| 1 | Yêu cầu 1 | 100% |  |
| 2 | … | … | … |
| 3 | … | … | … |
| Điểm tự đánh giá | | | **9.5/10** |

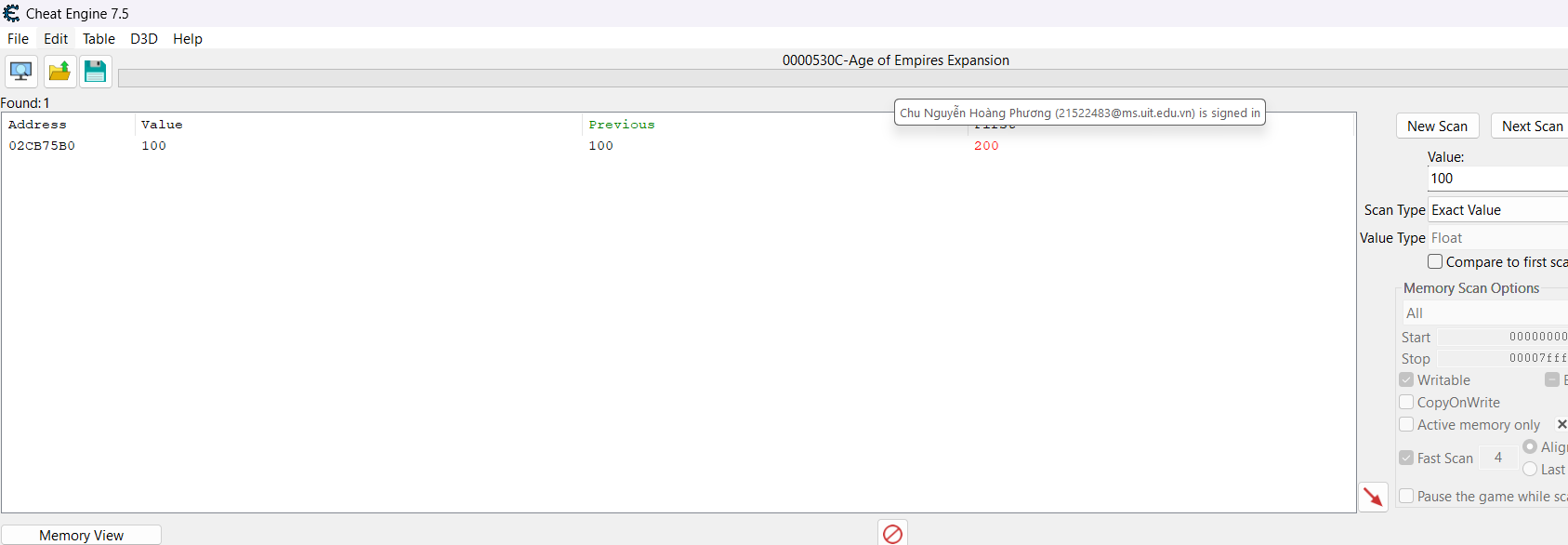
**Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.**

BÁO CÁO CHI TIẾT

-Đầu tiên tạo 2 villagers để food còn lại 100 cho dễ tìm trong cheat engine

A screenshot of a video game

Description automatically generated

-Scan giá trị 100 để tìm địa chỉ của food

-Chỉnh sửa giá trị thành 5000 lưu trong địa chỉ này và vào game xem thử.

A computer screen shot of a message

Description automatically generated-Kết quả:

A screenshot of a video game

Description automatically generated

-Tiếp theo sử dụng food để triệu hồi 8 con villagers

A screenshot of a video game

Description automatically generated

- Cùng với đó giá trị bên Cheat Engine cũng thay đổi theo, vậy đây chính xác là địa chỉ của Food

A screenshot of a computer

Description automatically generated

-Tiếp theo ta sẽ tạo Pointer Map cho vị trí lưu food đầu tiên, lưu với tên ptr1

A screenshot of a computer

Description automatically generated

- Tiến hành Scan pointer map vừa tạo và lưu lại bản scan này

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

-Kết quả được các offset dẫn đến địa chỉ biến food

A screenshot of a computer

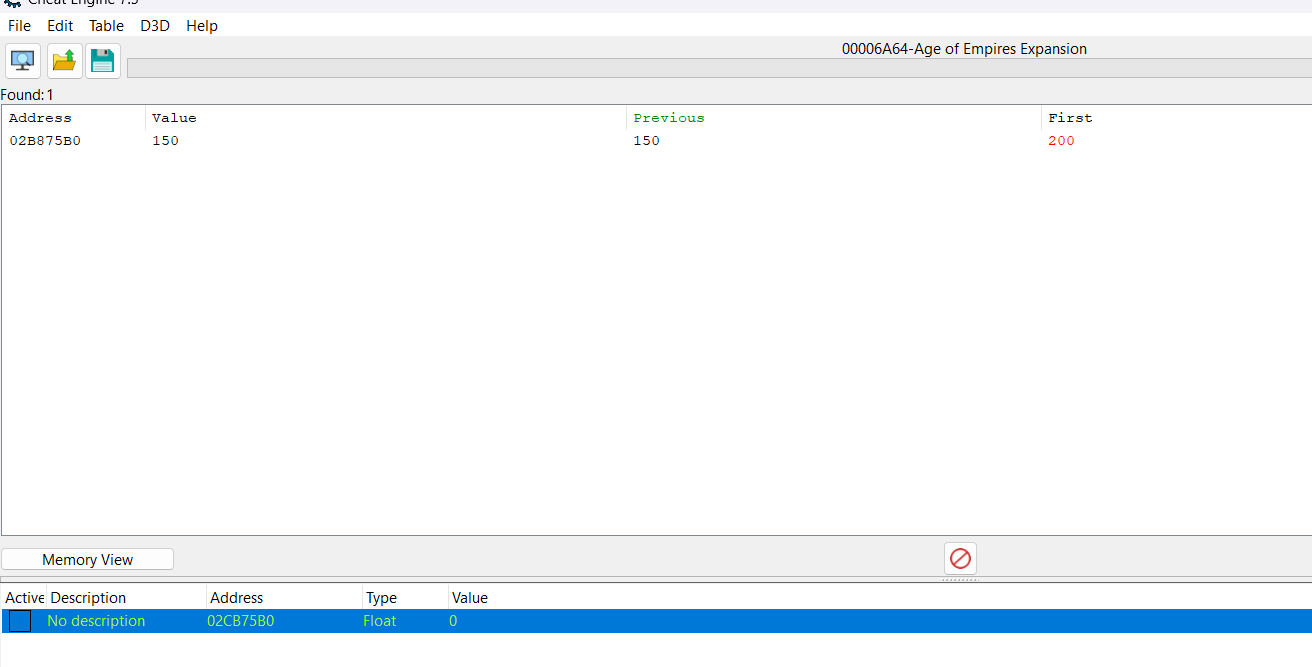
Description automatically generated

- Chọn 1 vài bản scan trong đó bằng cách double click vào và khởi động lại game thì pointermap không tìm được dynamic address của biến food.

A screenshot of a computer

Description automatically generated

- Bây giờ, ta tiếp tục tìm address của biến food thông qua scan như lúc đầu , và tạo pointermap với nó



A screenshot of a computer

Description automatically generated

-Đi so sánh với pointermap ban đầu và lưu lại

A screenshot of a computer

Description automatically generated

-Chọn 1 vài bản scan trong đó.

A screenshot of a computer

Description automatically generated

-Sau đó khởi động lại game thì đã tìm được biến và value của food

A screenshot of a computer

Description automatically generated

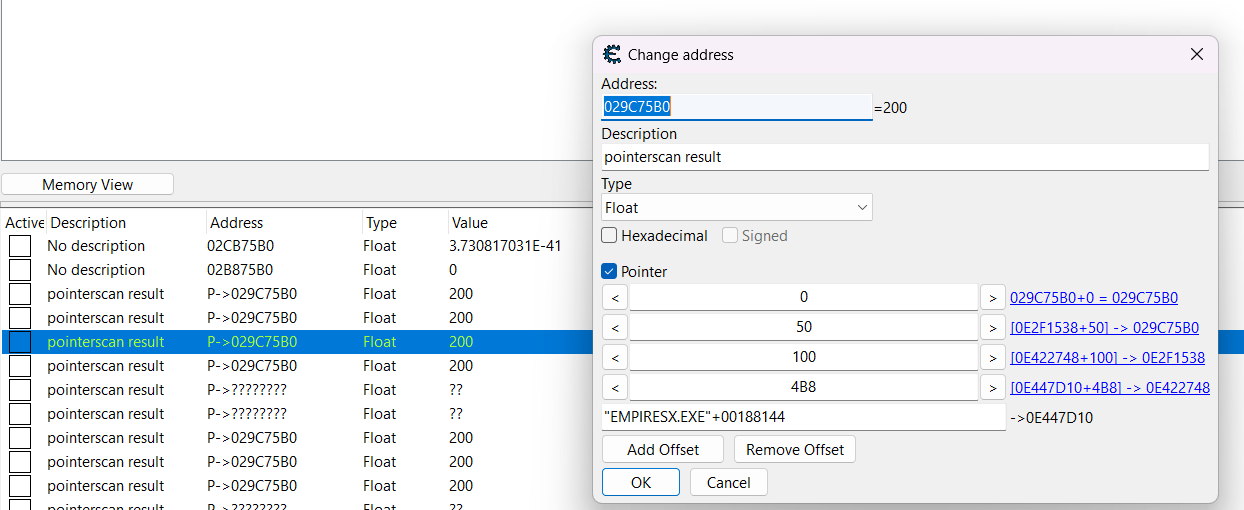
Như vậy, ta rút ra rằng:

+ Pointermap trong Cheat Engine không lưu trực tiếp base address của một biến. Thay vào đó, pointermap được sử dụng để tạo ra một danh sách các địa chỉ và offsets dẫn đến một địa chỉ cụ thể trong bộ nhớ. Từ đó, base address của biến có thể được tính toán bằng cách cộng địa chỉ load của module và static base .

+ Khi tạo một pointermap trên một địa chỉ cụ thể, nếu địa chỉ base và offsets không thay đổi sau khi khởi động lại chương trình, ta vẫn có thể có địa chỉ của biến đó sau khi khởi động lại.

+ Tuy nhiên, cần lưu ý rằng trong một số trường hợp, địa chỉ base và offsets có thể thay đổi khi chương trình khởi động lại. Điều này có thể xảy ra khi chương trình cấp phát lại vùng nhớ, thay đổi động các đối tượng trong bộ nhớ hoặc khi có sự can thiệp từ phía chương trình.

+ Quá trình "scan pointermap" trong Cheat Engine được sử dụng để tìm và xác định các con trỏ (pointers) dẫn đến một địa chỉ cụ thể trong bộ nhớ.



-Với các offset là: Base address= 0x00188144, offset1 = 0x4B8, offset2 = 0x100, offset3 = 0x50, offset4 = 0x00

- Địa chỉ gốc (base address) là địa chỉ bắt đầu của một khu vực bộ nhớ trong một chương trình , nó sẽ thay đổi khi chương trình được cấp phát lại vùng nhớ. Dù base address thay đổi, các offsets vẫn có thể truyền được vì chúng chỉ xác định sự dịch chuyển từ địa chỉ gốc đến biến. Các offsets này không phụ thuộc trực tiếp vào base address mà chỉ đóng vai trò như một lối đi để truy cập đến biến.

\*Giải thích code và Demo

+) Source.cpp:

A screenshot of a computer program

Description automatically generated

**dwModuleBaseAddress**: Biến để lưu địa chỉ cơ sở của mô-đun, khởi tạo bằng 0.

**hSnapshot**: Một handle để chụp ảnh nhanh các mô-đun trong tiến trình được chỉ định. Hàm CreateToolhelp32Snapshot được sử dụng để tạo ảnh chụp nhanh với cờ TH32CS\_SNAPMODULE, chỉ ra rằng ảnh chụp nhanh sẽ bao gồm danh sách các mô-đun trong tiến trình được chỉ định.

**ModuleEntry32**: Một cấu trúc MODULEENTRY32 sẽ chứa thông tin về từng mô-đun.

**Module32First**: Hàm này lấy thông tin về mô-đun đầu tiên trong ảnh chụp nhanh và lưu vào ModuleEntry32. Nó trả về TRUE nếu thành công.

**vòng lặp do-while**: Lặp qua danh sách các mô-đun trong ảnh chụp nhanh bằng cách sử dụng Module32Next, hàm này lấy thông tin về mô-đun tiếp theo và lưu vào ModuleEntry32.

**\_tcscmp**: So sánh tên của mô-đun hiện tại (ModuleEntry32.szModule) với tên mô-đun được chỉ định (lpszModuleName). Nếu chúng trùng khớp, nó đặt dwModuleBaseAddress là địa chỉ cơ sở của mô-đun (ModuleEntry32.modBaseAddr) và thoát khỏi vòng lặp.

* Hàm GetModuleBaseAddress nhận vào tên mô-đun và ID tiến trình, tạo ảnh chụp nhanh của tất cả các mô-đun trong tiến trình đó, lặp qua các mô-đun để tìm mô-đun có tên chỉ định, lấy địa chỉ cơ sở của nó và trả về địa chỉ này. Nếu không tìm thấy mô-đun, hàm sẽ trả về 0.

A computer screen shot of a program code

Description automatically generated

1. **Tìm cửa sổ trò chơi:**

HWND hGameWindow = FindWindow(NULL, L"Age of Empires Expansion");

if (hGameWindow == NULL) {

std::cout << "Start the game!" << std::endl;

return 0;

}

* FindWindow tìm cửa sổ của trò chơi "Age of Empires Expansion". Nếu không tìm thấy cửa sổ trò chơi, thông báo yêu cầu khởi động trò chơi được in ra và chương trình kết thúc.

2 **Lấy ID tiến trình của trò chơi:**

DWORD pID = NULL;

GetWindowThreadProcessId(hGameWindow, &pID);

HANDLE processHandle = NULL;

processHandle = OpenProcess(PROCESS\_ALL\_ACCESS, FALSE, pID);

if (processHandle == INVALID\_HANDLE\_VALUE || processHandle == NULL) {

std::cout << "Failed to open process" << std::endl;

return 0;

}

* GetWindowThreadProcessId lấy ID của tiến trình của trò chơi.
* OpenProcess mở tiến trình với tất cả các quyền truy cập (PROCESS\_ALL\_ACCESS). Nếu không thể mở tiến trình, chương trình kết thúc.

3 **Lấy địa chỉ cơ sở của mô-đun trò chơi:**

TCHAR gameName[13];

wcscpy\_s(gameName, 13, L"Empiresx.exe");

DWORD gameBaseAddress = GetModuleBaseAddress(gameName, pID);

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

* wcscpy\_s sao chép tên mô-đun trò chơi vào gameName.
* GetModuleBaseAddress lấy địa chỉ cơ sở của mô-đun "Empiresx.exe".

4 **Đọc và tính toán địa chỉ điểm số:**

DWORD offsetGameToBaseAdress = 0x003C4B18;

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

DWORD baseAddress = NULL;

ReadProcessMemory(processHandle, (LPVOID)(gameBaseAddress + offsetGameToBaseAdress), &baseAddress, sizeof(baseAddress), NULL);

std::cout << "debugginfo: baseaddress = " << std::hex << baseAddress << std::endl;

DWORD pointsAddress = baseAddress;

for (int i = 0; i < pointsOffsets.size() - 1; i++) {

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);

* ReadProcessMemory đọc giá trị tại gameBaseAddress + offsetGameToBaseAdress và lưu vào baseAddress.
* Sử dụng vòng lặp để tính toán pointsAddress dựa trên các offsets từ pointsOffsets.

5 **Đọc và ghi giá trị điểm số:**

float currentPoint = 0;

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);

* ReadProcessMemory đọc giá trị hiện tại của điểm số tại pointsAddress.
* Hiển thị giá trị hiện tại và yêu cầu người dùng nhập số điểm mới.
* WriteProcessMemory ghi giá trị điểm số mới vào pointsAddress.

+) Injector.cpp:

A screenshot of a computer program

Description automatically generated

Khai báo biến và lấy ID của tiến trình:

* HANDLE hProcess: Biến để lưu handle của tiến trình.
* LPVOID pszLibFileRemote: Con trỏ tới vùng nhớ từ xa trong tiến trình đích.
* HANDLE handleThread: Biến để lưu handle của luồng từ xa.
* const wchar\_t\* process: Tên của tiến trình đích ("Empiresx.exe").
* int pID: ID của tiến trình đích, được lấy thông qua hàm getProcId.

Kiểm tra sự tồn tại của DLL:

* char dll[]: Tên của DLL cần chèn vào tiến trình đích.
* exist(dll): Kiểm tra sự tồn tại của DLL. Nếu không tồn tại, thông báo lỗi.
* GetFullPathNameA: Lấy đường dẫn đầy đủ của DLL và lưu vào dllPath.

Mở tiến trình đích:

* OpenProcess: Mở tiến trình đích với các quyền truy cập cần thiết.
* VirtualAllocEx: Cấp phát vùng nhớ trong tiến trình đích để lưu đường dẫn của DLL.

Ghi đường dẫn của DLL vào vùng nhớ từ xa:

* WriteProcessMemory: Ghi đường dẫn của DLL vào vùng nhớ đã cấp phát trong tiến trình đích.

Tạo luồng từ xa để tải DLL:

* CreateRemoteThread: Tạo một luồng từ xa trong tiến trình đích để gọi hàm LoadLibraryA với tham số là đường dẫn của DLL.
* WaitForSingleObject: Chờ cho đến khi luồng từ xa hoàn thành.
* CloseHandle: Đóng handle của luồng và tiến trình.
* VirtualFreeEx: Giải phóng vùng nhớ đã cấp phát trong tiến trình đích.

+) dllmain.cpp:

-Đầu tiên ta sẽ bỏ toàn bộ code của Source.cpp vào, tuy nhiên thực hiện 1 số chỉnh sửa sau:

Chỉnh sửa tên hàm main() thành press(), đổi Process hiện tại thành EMPIRESX.exe thay vì Empiresx.exe như lúc đầu

A computer screen shot of a program code

Description automatically generated

-Thay vì đoạn code sau khai báo Newpoints = 0 và bắt chúng ta nhập giá trị bất kì để thay đổi biến food, thì ta sẽ lấy CurrentPoints hiện tại cộng với 100 luôn để thực hiện việc tăng 100 thức ăn.

A screen shot of a computer program

Description automatically generated

-Tiếp theo trong phần MainThread, sau khi hiện MessageBox F6 Pressed!!, ta gọi hàm Tang100thucan()

A screen shot of a computer

Description automatically generated

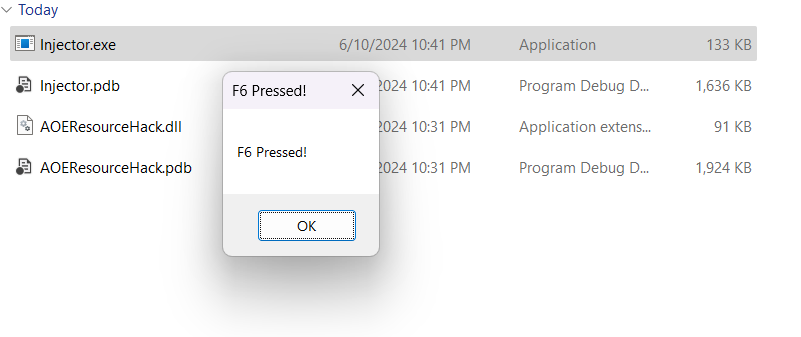
-Như vậy là xong, thực hiện build Solution , đầu tiên ta build file dllmain.cpp để có AOEResourceHack.dll cho file Injector.cpp. Sau đó ta build file Injector.cpp để ra file Injector.exe thuận tiện cho việc tiêm nhiễm vào tiến trình dll độc hại, tăng 100 thức ăn.

-Khởi động game

A screenshot of a video game

Description automatically generated

-Chạy Injector.exe, vào game nhấp F6 để tăng food



-Kết quả

A screenshot of a video game

Description automatically generated

1. Ghi nội dung công việc, các kịch bản trong bài Thực hành [↑](#footnote-ref-1)