BBÁO CÁO THỰC HÀNH

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

**Lab 1: Assembly PE injection**

*GVHD: Nguyễn Hữu Quyền*

1. **THÔNG TIN CHUNG:**

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

Lớp: NT230.O22.ATCL.1- Nhóm 3

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Nguyễn Ngọc Trà My | 21520353 | [21520353@gm.uit.edu.vn](mailto:21520353@gm.uit.edu.vn) |
| 2 | Bùi Hoàng Trúc Anh | 21521817 | [21521817@gm.uit.edu.vn](mailto:21521817@gm.uit.edu.vn) |
| 3 | Lê Hoàng Oanh | 21521253 | [21521253@gm.uit.edu.vn](mailto:21521253@gm.uit.edu.vn) |
| 4 | Bùi Nguyên Phúc | 21522469 | 21522469@gm.uit.edu.vn |

1. **NỘI DUNG THỰC HIỆN:**

|  |  |  |
| --- | --- | --- |
| **STT** | **Công việc** | **Kết quả tự đánh giá** |
| 1 | **Yêu cầu 1** | 100% |
| 2 | **Yêu cầu 2** | 100% |
| 3 | **Yêu cầu 3** | 100% |
| 4 | **Yêu cầu 4** | 100% |
| 5 | **Yêu cầu 5** | 100% |

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

BÁO CÁO CHI TIẾT

**Bài 1. Viết một đoạn chương trình tìm số nhỏ nhất trong 3 số (1 chữ số) a,b,c**

**cho trước.**

section .text: ;tell linker to put this section in the text segment

global \_start ;tell linker entry point

\_start: ;tell linker entry point

mov ecx, [num1] ;move first number to ecx

cmp ecx, [num2] ;compare first number to second number

jle check\_third\_num ;if first number is less than or equal to second number, check third number

mov ecx, [num2] ;if first number is greater than second number, move second number to ecx

check\_third\_num: ;check third number

cmp ecx, [num3] ;compare ecx to third number

jle \_exit ;if ecx is less than or equal to third number, exit

mov ecx, [num3] ;if ecx is greater than third number, move third number to ecx

\_exit: ;exit

mov [smallest], ecx ;move smallest number to smallest

mov ecx,msg ;message to display

mov edx, len ;length of message

mov ebx,1 ;file descriptor (stdout)

mov eax,4 ;system call number (sys\_write)

int 0x80 ;call kernel

mov ecx,smallest ;smallest number to display

mov edx, 2 ;length of smallest number

mov ebx,1 ;file descriptor (stdout)

mov eax,4 ;system call number (sys\_write)

int 0x80 ;call kernel

mov eax, 1 ;system call number (sys\_exit)

int 80h ;exit

section .data ;tell linker to put this section in the data segment

msg db "The smallest digit is: ", 0xA,0xD ;message to display

len equ $- msg ;length of message

num1 dd '1' ;first number

num2 dd '2' ;second number

num3 dd '7' ;third number

segment .bss ;tell linker to put this section in the bss segment

smallest resb 10 ;smallest number

**Bài 2. Viết chương trình chuyển đổi một số (number) 123 thành chuổi ‘123’**

**Sau đó thực hiện in ra màn hình số 123.**

%assign SYS\_EXIT 1

%assign SYS\_WRITE 4

%assign STDOUT 1

section .data ;tell linker data section

x db 123 ;x = 123

msgX db "x = " ;message

section .text ;tell linker text section

global \_start ;tell linker entry point

\_start: ;entry point

mov ecx, msgX ;move address of msgX to ecx

mov edx, 4 ;move length of msgX to edx

call \_printString ;call printString

mov eax, 0 ;move 0 to eax

mov al, byte[x] ;move x to al

call \_printDec ;call printDec

mov ebx, 0 ;move 0 to ebx

mov eax, 1 ;move 1 to eax

int 0x80 ;call kernel

\_printString: ;printString function

push eax ;save eax

push ebx ;save ebx

mov eax,SYS\_WRITE ;move SYS\_WRITE to eax

mov ebx,STDOUT ;move STDOUT to ebx

int 0x80 ;call kernel

pop ebx ;restore ebx

pop eax ;restore eax

ret ;return

\_println:

section .data ;tell linker data section

.nl db 10 ;newline

section .text ;tell linker text section

push ecx ;save ecx

push edx ;save edx

mov ecx, .nl ;move address of .nl to ecx

mov edx, 1 ;move length of .nl to edx

call \_printString ;call printString

pop edx ;restore edx

pop ecx ;restore ecx

ret ;return

\_printDec:

;;;saves all the registers so that they are not changed by the function

section .bss

.decstr resb 10

.ct1 resd 1 ;to keep track of the size of the string

section .text

pushad ;save all registers

mov dword[.ct1],0 ;assume initially 0

mov edi,.decstr ;edi points to decstring

add edi,9 ;moved to the last element of string

xor edx,edx ;clear edx for 64-bit division

.whileNotZero:

mov ebx,10 ;get ready to divide by 10

div ebx ;divide by 10

add edx,'0' ;converts to ascii char

mov byte[edi],dl ;put it in sring

dec edi ;mov to next char in string

inc dword[.ct1] ;increment char counter

xor edx,edx ;clear edx

cmp eax,0 ;is remainder of division 0?

jne .whileNotZero ;no, keep on looping

inc edi ;conversion, finish, bring edi

mov ecx, edi ;back to beg of string. make ecx

mov edx, [.ct1] ;point to it, and edx gets # chars

mov eax, SYS\_WRITE ;and print!

mov ebx, STDOUT ;print to stdout

int 0x80 ;call the kernel

popad ;restore all registers

ret

**Bài 3. Cải tiến chương trình yêu cầu 1 sao cho tìm số nhỏ nhất trong 3 số bất**

**kỳ (nhiều hơn 1 chữ số)**

section .text ;tell linker to put this section in the text segment

global \_start ;tell linker to start at \_start

\_start: ;start of the program

mov eax, [num1] ;put the first number in eax

mov ebx, [num2] ;put the second number in ebx

cmp eax, ebx ;compare the first number and the second number

jle .case1 ;if the first number is less than or equal to the second number, jump to .else

mov eax, [num2] ;put the second number in eax

mov ebx, [num3] ;put the third number in ebx

cmp eax, ebx ;compare the second number and the third number

jle .case2 ;if the second number is less than or equal to the third number, jump to .case2

mov eax, [num3] ;put the third number in eax

jmp .exit ;jump to .exit

.case1: ;if the first number is greater than the second number

mov eax, [num1] ;put the first number in eax

mov ebx, [num3] ;put the third number in ebx

cmp eax, ebx ;compare the first number and the third number

jle .case3 ;if the first number is less than or equal to the third number, jump to .case3

mov eax, [num3] ;put the third number in eax

jmp .exit ;jump to .exit

.case2: ;if the second number is greater than the third number

mov eax, [num1] ;put the first number in eax

mov ebx, [num2] ;put the second number in ebx

cmp eax, ebx ;compare the first number and the second number

jle .case3 ;if the first number is less than or equal to the second number, jump to .case3

mov eax, [num2] ;put the second number in eax

jmp .exit ;jump to .exit

.case3: ;if the first number is greater than the third number

mov eax, [num1] ;put the first number in eax

.exit: ;exit of the program

mov [smallest], eax ;put the minimum number in the smallest variable

mov eax, 4 ;put the system call number 4 in eax

mov ebx, 1 ;put the file descriptor 1 in ebx

mov ecx, msg1 ;put the address of the message 4 in ecx

mov edx, len1 ;put the length of the message 4 in edx

int 0x80 ;call the system call

mov eax, 4 ;put the system call number 4 in eax

mov ebx, 1 ;put the file descriptor 1 in ebx

mov ecx, smallest ;put the address of the smallest variable in ecx

mov edx, 4 ;put the length of the smallest variable in edx

int 0x80 ;call the system call

mov eax, 1 ;put the system call number 1 in eax

mov ebx, 0 ;put the exit code 0 in ebx

int 0x80 ;call the system call

section .data ;tell linker to put this section in the data segment

msg1 db "The minimum number is: ", 0xa, 0x0 ;announce the minimum number

len1 equ $ - msg1 ;get the length of the message 4

num1 dd '123' ;first number

num2 dd '234' ;second number

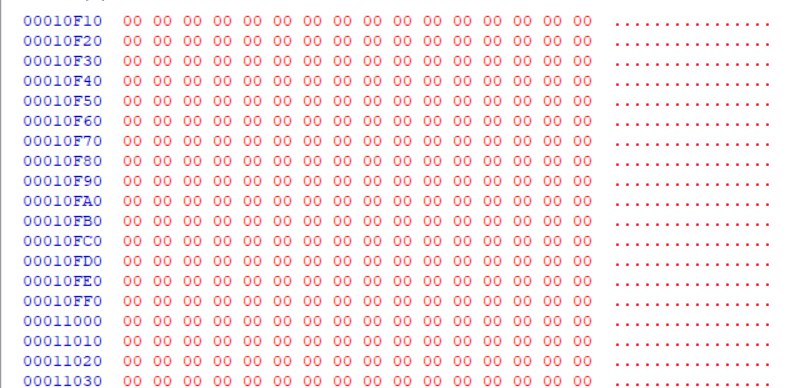
num3 dd '345' ;third number

section .bss ;tell linker to put this section in the bss segment

smallest resd 1 ;reserve 4 bytes for the minimum number

**Bài 4: Thực hiện lại các bước trên thay đổi phần Text là MSSV.**

* Tạo vùng nhớ trong tập tin PE



* Trong HxD, ta chọn địa chỉ 0x00011000 trong vùng nhớ đã được mở rộng để lưu trữ mã hợp ngữ, 0x00011040 để lưu trữ caption, 0x00011060 để lưu trữ Text

Offset = RA – Section RA = VA – Section VA (1)

Ta sử dụng công thức (1) trên để tính X, Y

0x00011040 - 0x00008400 = X – 0x000B000 -> X = 0x00013C40

Cộng thêm ImageBase, suy ra X = 0x01013C40.

Tương tự,

0x00011060 - 0x00008400 = Y – 0x000B000 -> Y = 0x00013C40 => Y=0x01013C60.

0x00011000 - 0x00008400 = new\_entry\_point – 0x000B000 -> new\_entry\_point = 0x00013C00

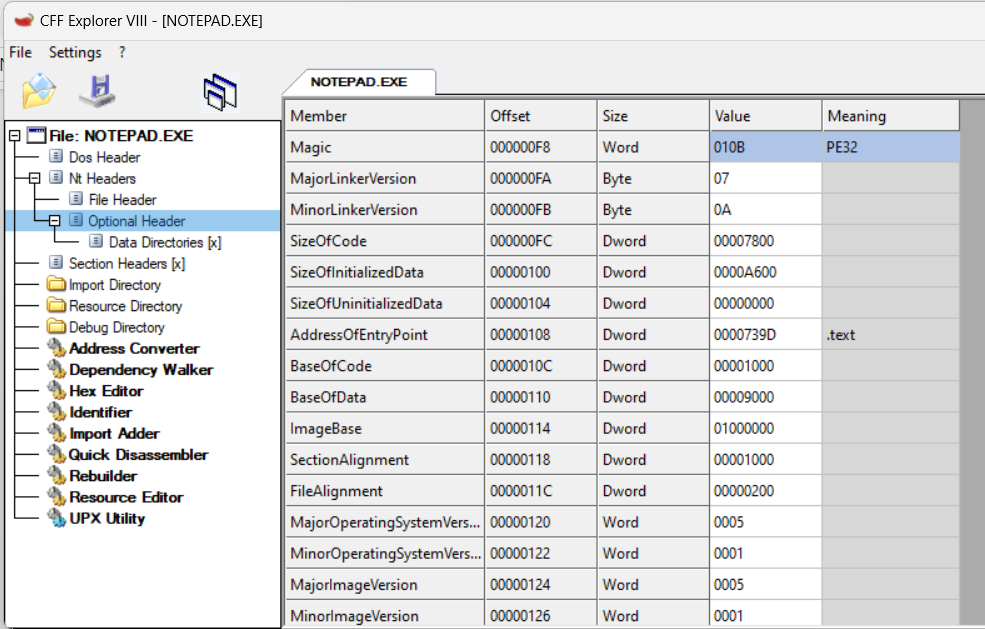
* Thiết lập quay về AddresssOfEntrypoint ban đầu

Ta cần chèn dòng lệnh quay về AddressOfEntrypoint ban đầu ngay sau lệnh call jmp relative\_VA

old\_entry\_point = jmp\_instruction\_VA + 5 + relative\_VA (2)

jmp\_instruction\_VA = 0x01013C14

old\_entry\_point = AddressOfEntryPoint + ImageBase = 0x0100739D



=> relative\_VA = 0xFFFF3784.

|  |
| --- |
| push 0 ; 6a 00  push Caption ; 68 403C0101  push Text ; 68 603C0101  push 0 ; 6a00 call [MessageBoxW] ; ff15 68120001  jmp Origianl\_Entry\_Point ; e9 8437FFFF |

Ta insert những bytes sau vào vị trí 0x00011000: 6a 00 68 60 3C 01 01 68 40 3C 01 01 ff 15 68 12 00 01 e9 84 37 FF FF

Vị trí 0x00011060: 35 00 33 00 20 00 35 00 33 00 20 00 31 00 37 00 20 00 36 00 39

Vị trí 0x00011040: 6e 00 68 00 6f 00 6d 00 33

Sử dụng CFF Explorer để thay đổi các giá trị sau:

Trong Section Headers, thay đoi .rsrc SectionHeader.

Trong Optional Headers, tăng SizeOfImage lên 0x1000.

Trong Optional Headers, chı̉nh sửa AddressOfEntryPoint thành 0x00013C00. Lưu lại file.

Kết quả:

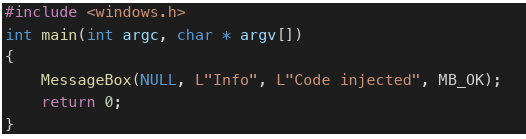
Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, Hình chữ nhật

Description automatically generated

**Bài 5: Bằng cách không tạo thêm vùng nhớ mở rộng vào tập tin PE, tận dụng vùng nhớ trống để chèn chương trình cần chèn trên tập tin Notepad và calc.**

* **Notepad**

Ta có một chương trình hiển thị MessageBox như sau:



Biên dịch chương trình dưới chế độ Release, Not Using Precompiled Headers. Sử dụng IDA Pro để mở file PE và xem mã hợp ngữ của chương trình vừa biên dịch.

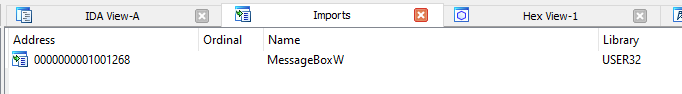
Về cơ bản, chương trình gồm 5 dòng lệnh (sử dụng chức năng hexview để xem mã hex của từng lệnh).

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, số

Description automatically generated

Để chèn đoạn code này vào Notepad.exe, ta phải đi tìm các giá trị (X, Y, Z) phù hợp.

Giá trị Z chính là địa chỉ của hàm MessageBoxW được import từ thư viện USER32.dll. Trong IDA Pro, mở Notepad.exe, chọn View -> Open Subviews -> Imports và ta thấy địa chỉ của hàm MessageBoxW chính là 01001268.



Trong HxD, ta chọn địa chı̉ 0x00010DA0 trong vùng nhớ đã được mở rộng (bước C1) để lưu trữ mã hợp ngữ , 0x00010D50 để lưu trữ Caption và 0x00010D60 để lưu trữ Text. Ta có thể tùy chọn những vị trı́ khác tùy thı́ch.

Đối với mỗi section, loader sẽ copy section tại RA trong PE file sang bộ nhớ ảo tại VA trong khi vẫn giữ đúng offset bên trong section đó .

Giá trị X có thể được tìm dựa vào công thức:

Offset = RA – Section RA = VA – Section VA

0x00010D50 – 0x00008400 = X – 0x000B000 => X = 0x00013950

Cộng thêm ImageBase, suy ra: X = 0x01013950. Tương tự , Y = 0x01013960.

Như vậy, đoạn code này thực hiện chức năng như mong đợi và có địa chı̉ mới là:

new\_entry\_point = 0x00010DA0 – 0x00008400 + 0x000B000 = 0x000139A0

Để chương trình Notepad.exe tiếp tục được thực thi sau khi đã chạy đoạn code trên, ta cần chèn dòng lệnh quay về AddressOfEntryPoint cũ ngay sau đoạn code ở bước 2 jmp relative\_VA

Đối với lệnh jmp, đích đến (old\_entry\_point) sẽ được tính bằng cách cộng giá trị relative\_VA vào thanh ghi PC khi lệnh được thực thi. Bởi vì PC luôn trỏ đến vị trí đầu của câu lệnh kế tiếp, cho nên cần phải tính 5 bytes của câu lệnh jmp nữa. Ta có công thức sau:

old\_entry\_point = jmp\_instruction\_VA + 5 + relative\_VA

Nếu đặt jmp sau 5 câu lệnh ở bước 2 thì jmp\_instruction\_VA = 0x01013950.

old\_entry\_point = 0x0100739D chính là giá trị AddressOfEntryPoint ban đầu đã cộng ImageBase.

Suy ra, relative\_VA = 0x0100739D – 5 – 0x01013950 = 0xFFFF39E4.

Đến đây, ta đã có một đoạn hợp ngữ hoàn chỉnh để chèn vào Notepad.exe. Các địa chỉ được biểu diễn theo thứ tự little endian (x86)

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Description automatically generated

Sử dụng HxD để chèn đoạn mã cùng với giá trị Caption và Text vào Notepad.exe. Lưu lại file.

A number of numbers on a white background

Description automatically generated

Sử dụng CFF explorer để thay đổi các giá trị sau:



Ra được kết quả như sau:

A screenshot of a computer error

Description automatically generated

* **Calc**

Ta có một chương trình hiển thị MessageBox như sau:

A black background with white text

Description automatically generated

Biên dịch chương trình dưới chế độ Release, Not Using Precompiled Headers. Sử dụng IDA Pro để mở file PE và xem mã hợp ngữ của chương trình vừa biên dịch.

Về cơ bản, chương trình gồm 5 dòng lệnh (sử dụng chức năng hexview để xem mã hex của từng lệnh).

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, số

Description automatically generated

Để chèn đoạn code này vào Notepad.exe, ta phải đi tìm các giá trị (X, Y, Z) phù hợp.

Giá trị Z chính là địa chỉ của hàm MessageBoxW được import từ thư viện USER32.dll. Trong IDA Pro, mở Notepad.exe, chọn View -> Open Subviews -> Imports và ta thấy địa chỉ của hàm MessageBoxW chính là 010011A8.

A white rectangular box with blue text

Description automatically generated with medium confidence

Trong HxD, ta chọn địa chı̉ 0x0001BF50 để lưu trữ mã hợp ngữ , 0x0001BF70 để lưu trữ Caption và 0x0001BF80 để lưu trữ Text. Ta có thể tùy chọn những vị trı́ khác tùy thı́ch.

Đối với mỗi section, loader sẽ copy section tại RA trong PE file sang bộ nhớ ảo tại VA trong khi vẫn giữ đúng offset bên trong section đó .

Giá trị X có thể được tìm dựa vào công thức:

Offset = RA – Section RA = VA – Section VA

0x0001BF70 – 0x00013600 = X – 0x00016000

Cộng thêm ImageBase, suy ra: X = 0x0101E970. Tương tự, Y = 0x001E980.

Như vậy, đoạn code này thực hiện chức năng như mong đợi và có địa chı̉ mới là:

new\_entry\_point = 0x0001BF50 – 0x00013600 + 0x00016000 = 0x0001E950

Để chương trình Notepad.exe tiếp tục được thực thi sau khi đã chạy đoạn code trên, ta cần chèn dòng lệnh quay về AddressOfEntryPoint cũ ngay sau đoạn code ở bước 2 jmp relative\_VA

Đối với lệnh jmp, đích đến (old\_entry\_point) sẽ được tính bằng cách cộng giá trị relative\_VA vào thanh ghi PC khi lệnh được thực thi. Bởi vì PC luôn trỏ đến vị trí đầu của câu lệnh kế tiếp, cho nên cần phải tính 5 bytes của câu lệnh jmp nữa. Ta có công thức sau:

old\_entry\_point = jmp\_instruction\_VA + 5 + relative\_VA

Nếu đặt jmp sau 5 câu lệnh ở bước 2 thì jmp\_instruction\_VA = 0x0101E970.

old\_entry\_point = 0x01012475 chính là giá trị AddressOfEntryPoint ban đầu đã cộng ImageBase.

Suy ra, relative\_VA = 0x01012475 – 5 – 0x0101E970 = 0xFFFF3B0C.

Đến đây, ta đã có một đoạn hợp ngữ hoàn chỉnh để chèn vào Notepad.exe. Các địa chỉ được biểu diễn theo thứ tự little endian (x86)

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Description automatically generated

Sử dụng HxD để chèn đoạn mã cùng với giá trị Caption và Text vào Notepad.exe. Lưu lại file.

A screenshot of a computer code

Description automatically generated

Sử dụng CFF explorer để thay đổi các giá trị sau:



Ra được kết quả như sau:

A screenshot of a computer error

Description automatically generated