Đại học Quốc gia Thành phố Hồ Chí Minh
Trường Đại học Công nghệ Thông tin (UIT)

BÁO CÁO BÀI TẬP

Môn học: Cơ chế hoạt động của mã độc

Kỳ báo cáo: Buổi 01 (Session 01)

Tên chủ đề: Dò quét và bắt gói tin trong mạng

GV: Nghi Hoàng Khoa

Ngày báo cáo: 13/03/2023

Nhóm: 06

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 Quang Minh	20520248	20520248@gm.uit.edu.vn
2	Bùi Hải Đăng	20520173	20520173@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

STT	Công việc	Kết quả tự đánh giá	Người đóng góp
1	Bài tập 1	100%	Đăng + Minh
2	Bài tập 2	100%	Đăng + Minh
3	Bài tập 3	100%	Đăng + Minh
4	Bài tập 4	100%	Đăng + Minh
5	Bài tập 5	100%	Đăng + Minh

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

[1] Ghi nội dung công việc, các kịch bản trong bài Thực hành

BÁO CÁO CHI TIẾT

1. **BT1: Viết một đoạn chương trình tìm số nhỏ nhất trong 3 số (1 chữ số) a,b,c cho trước.**
   - Tài nguyên:

Máy ảo Linux.

- Mô tả/mục tiêu:

Tìm và in ra số nhỏ nhất trong 3 số có 1 chữ số

- Các bước thực hiện/ Phương pháp thực hiện (Ảnh chụp màn hình, có giải thích)

Chuẩn bị các dữ liệu, khởi tạo 3 giá trị có độ lớn 1 chữ số và một biến để nhận giá trị nhỏ nhất tìm được.

```
section .data

    msg db "The smallest digit is:",0xA,0xD
    len equ $- msg
    num1 dd'3'
    num2 dd'8'
    num3 dd'4'

segment .bss
    largest resb 2
```

Để tìm số nhỏ nhất, nhóm tiến hành so sánh 2 số đầu tiên với nhau sau đó lấy số nhỏ hơn vừa tìm được so sánh với số thứ 3, kết quả được lưu vào thanh ghi ecx.

```
_start:
    mov ecx,[num1] ; load so thu 1 vao ecx
    cmp ecx,[num2] ; so sanh so thu 1 voi so thu 2
    jl check_third_num
    mov ecx,[num2] ; neu so thu 2 nho hon thi load so thu 2 vao ecx

check_third_num:
    cmp ecx,[num3] ; so sanh voi so thu 3
    jl _exit
    mov ecx,[num3] ; neu so thu 3 nho hon thi load so thu 3 vao ecx
```

Sau khi kết quả được lưu ở thanh ghi ecx, tiến hành dùng các lời gọi hệ thống để in ra kết quả.

```
_exit:
    ; In cau dan
    mov [smallest], ecx
    mov ecx,msg
    mov edx,len
    mov ebx,1
    mov eax,4
    int 0x80

    ; In ket qua tim duoc
    mov ecx,smallest
    mov edx,2
    mov ebx,1
    mov eax,4
    int 0x80

    mov eax,1
    int 80h
```

Kết quả chạy chương trình:

```
The smallest digit is:

3
```

Source code:

```
section .text
        global_start

_start:
  ; so sanh so thu 1 va so thu 2
        mov ecx,[num1]
        cmp ecx,[num2]
        jl check_third_num
        mov ecx,[num2]
```

```
check_third_num:
   ; so sanh ket qua tim o tren voi so thu 3
        cmp ecx,[num3]
        jl _exit
        mov ecx,[num3]

_exit:
   ; In cau dan
        mov [smallest], ecx
        mov ecx,msg
        mov edx,len
        mov ebx,1
        mov eax,4
        int 0x80

   ; In ket qua tim duoc
        mov ecx,smallest
        mov edx,2
        mov ebx,1
        mov eax,4
        int 0x80

        mov eax,1
        int 80h

section .data

        msg db "The smallest digit is:",0xA,0xD
        len equ $- msg
        num1 dd'3'
        num2 dd'8'
        num3 dd'4'
```

segment .bss

smallest resb 2

2. **BT2: Viết chương trình chuyển đổi một số (number) 123 thành chuỗi '123' Sau đó thực hiện in ra màn hình số 123.**
   - Tài nguyên:

Máy ảo linux

- Mô tả/mục tiêu:

Viết chương trình có thể chuyển đổi một số thành chuỗi và in nó ra màng hình.

- Các bước thực hiện/ Phương pháp thực hiện (Ảnh chụp màn hình, có giải thích)

Chuẩn bị dữ liệu gồm có một số cần được chuyển thành dạng chuỗi và in ra màng hình.

```
section .data
x dd 1235463
msgX db "x="
```

Biến x lưu giá trị của số cần chuyển, hàm _printDec sẽ chuyển và in số ra. Chuyển giá trị biến x vào thanh ghi eax và tiến hành gọi hàm _printDec.

```
_start:
    mov ecx, msgX
    mov edx, 4
    call _printString
    mov eax, [x]
    call _printDec

    mov ebx, 0
    mov eax, 1
    int 0x80
```

Phân tích hàm _printDec, đầu tiên khởi tạo 1 biến trống .ct1 để đếm độ dài của số cần in ra. Biến trống .decstr lưu trữ các byte của số ở dạng string để chuẩn bị cho việc in ra. Thanh ghi eax giữ giá trị của số cần in, mỗi vòng lặp thanh ghi eax sẽ được chia cho 10, giá trị của phần thập phân được lưu trữ trong thanh ghi edx. Giá trị trong thanh ghi sẽ được lưu trữ lại tại các biến trống được khởi tạo tại .decstr, .ct1 được cộng lên để theo dõi độ dài chuỗi sẽ được in ra.

```
_printDec:
    section .bss
    .decstr resb 10 ; khoi tao mang dai 10 phan tu chua cac so se in ra
    .ct1 resd 1 ; bien luu tru do dai chuoi can in ra

    section .text
    pushad

    mov dWord[.ct1], 0 ; gan cho bien dem = 0
    mov edi, .decstr ; edi tro den mang da khoi tao
    add edi, 9 ; tro den dau mang
    xor edx, edx
.WhileNotZero:
    mov ebx, 10 ; ebx = 10
    div ebx ; thuc hien phep chia phan nguyen o eax, phan thap phan o edx
    add edx, '0' ; chuyen gia tri o edx thanh dang chuoi
    mov byte[edi], dl ; chuyen vao mang da khoi tao
    dec edi ; dich con tro len o tiep theo trong mang
    inc dWord[.ct1] ; tang bien dem do dai chuoi se in ra
    xor edx, edx
    cmp eax, 0 ; kiem tra xem so can chuyen con hay khong
    jne .WhileNotZero ; lap lai

    inc edi
    mov ecx, edi
    mov edx, [.ct1]
    mov eax, SYS_WRITE
    mov ebx, STDOUT
    int 0x80
```

Sau khi vòng lặp kết thúc, biến .decstr sẽ trỏ đến dãy byte string cần in ra, biến .ct1 trỏ đến giá trị độ dài của dãy byte cần in ra. Truyền các gia trị vào các thanh ghi và gọi hệ thống.

```
    inc edi
    mov ecx, edi
    mov edx, [.ct1]
    mov eax, SYS_WRITE
    mov ebx, STDOUT
    int 0x80

    popad
    ret
```

Kết quả:

```
x=··1235463
```

Source code:

```
%assign SYS_EXIT 1
%assign SYS_WRITE 4
%assign STDOUT 1

section .data
x dd 1235463
msgX db "x="

section .text
global _start
_start:
        mov ecx, msgX
        mov edx, 4
        call _printString
        mov eax, [x]
        call _printDec

        mov ebx, 0
        mov eax, 1
        int 0x80

_printString:
        push eax
        push ebx
        mov eax, SYS_WRITE
        mov ebx, STDOUT
        int 0x80
        pop ebx
        pop eax
        ret

_printDec:
```

```
        section .bss
        .decstr resb 10
        .ct1 resd 1

        section .text
        pushad

        mov dWord[.ct1], 0
        mov edi, .decstr
        add edi, 9
        xor edx, edx
.WhileNotZero:
        mov ebx, 10
        div ebx
        add edx, '0'
        mov byte[edi], dl
        dec edi
        inc dWord[.ct1]
        xor edx, edx
        cmp eax, 0
        jne .WhileNotZero

        inc edi
        mov ecx, edi
        mov edx, [.ct1]
        mov eax, SYS_WRITE
        mov ebx, STDOUT
        int 0x80

        popad
        ret
```

3. **BT 03: Cải tiến chương trình yêu cầu 1 sao cho tìm số nhỏ nhất trong 3 số bất kỳ (nhiều hơn 1 chữ số)**
   - Tài nguyên:

Máy ảo linux

- Mô tả/mục tiêu:

Viết chương trình tìm ra số nhỏ nhất trong 3 số có độ lớn bất kì.

- Các bước thực hiện/ Phương pháp thực hiện (Ảnh chụp màn hình, có giải thích)

Chuẩn bị dữ liệu gồm 3 số có độ lớn bất kì, các số này sẽ được biểu diễn ở dạng string đồng thời khởi tạo 3 biến lưu độ dài của chúng.

```
section .data
    msg db "The smallest digit is: ", 0xA,0xD
    len equ $- msg
    a db "1111111"
    lena equ $- a
    b db "22222222"
    lenb equ $- b
    c db "33"
    lenc equ $- c
```

Do các số đang ở dạng string, cần tạo một hàm để chuyển chúng qua thành dạng int, hàm string2int nhận vào chuỗi số và độ dài của chuỗi, kết quả là số ở dạng int và được trả vào thanh ghi eax.

Mỗi vòng lặp, sẽ lấy ra 1byte và tiến hành xét xem nó có nằm trong đoạn '0'-'9' hay không và xét xem byte đó có thuộc chuỗi số đang cần chuyển hay không. Sau khi byte được lấy ra thỏa mãn các điều kiện thì tiến hành chuyển nó thành dạng int và cộng vào thanh ghi eax, trước khi cộng thì thanh ghi eax sẽ được nhân với 10.

```
str2int:
        push ebx
        xor eax, eax ; luu gia tri cuoi cung
        xor ecx, ecx ; luu tung byte của chuoi
        xor edi, edi ; dung de theo doi do dai chuoi can lay
.top:
        xor ecx, ecx
        mov cl, byte[edx] ; chuyen 1 byte vao ecx
        inc edx ; tang edx len
        add edi, 1 ; tang bien dem len
        cmp edi, esi ; kiem tra xem da lay du chua
        jg .done
        cmp cl, '0' ; kiem tra xem byte do co hop le khong
        jb .done
        cmp cl, '9' ; kiem tra xem byte do co hop le khong
        ja .done
        sub ecx,'0' ; chuyen byte chuoi thanh so
        mov ebx, eax ; luu tam vao ebx
        shl eax, 2 ; nhan eax cho 4 -> 4*eax
        add eax, ebx ; cong voi ban than -> 5*eax
        shl eax, 1 ; nhan cho 2 -> 10*eax
        add eax, ecx ; cong voi so vua lay ra duoc vao eax
        jmp .top
.done:
        pop ebx
        ret
```

Để tìm được số nhỏ nhất, nhóm tiến hành so sánh số thứ nhất với số thứ 2 sau đó lấy kết quả so sánh tiếp với số thứ 3, kết quả được lưu vào biến trống minn.

```
_start:
    mov edx, a
    mov esi, lena
    call str2int
    mov ebx, eax
    mov edx, b
    mov esi, lenb
    call str2int
    cmp ebx, eax
    jl check_third_num
    mov ebx, eax
check_third_num:
    mov edx, c
    mov esi, lenc
    call str2int
    cmp ebx, eax
    jl _exit
    mov ebx, eax
```

Chương trình sẽ lần lượt chuyển các chuỗi số thành các số và tiến hành so sánh, thanh ghi ebx sẽ giữ kết quả cuối cùng và chuyển vào biến minn. Tiến hành in kết quả tìm được

ra màng hình bằng cách tận dụng đoạn code chuyển số thành chuỗi và in ra ở bài tập 2 ở trên.

```
_exit:
    mov [minn], ebx
    mov ecx, msg
    mov edx, len
    mov eax,SYS_WRITE
    mov ebx,STDOUT
    int 0x80


    mov eax, [minn]
    call _printDec

    mov ebx, 0
    mov eax, 1
    int 0x80

segment .bss
    minn resb 4
```

Kết quả chạy:

```
The smallest digit is:

33
```

Source code:

```
%assign SYS_EXIT 1
%assign SYS_WRITE 4
%assign STDOUT 1

section .data
        msg db "The smallest digit is: ", 0xA,0xD
        len equ $- msg
        a db "1111111"
        lena equ $- a
        b db "22222222"
        lenb equ $- b
        c db "33"
        lenc equ $- c
section .text
```

```
global _start

str2int:
        push ebx
    xor eax, eax
    xor ecx, ecx
    xor edi, edi
.top:
    xor ecx, ecx
    mov cl, byte[edx]
    inc edx
    add edi, 1
    cmp edi, esi
    jg .done
    cmp cl, '0'
    jb .done
    cmp cl, '9'
    ja .done
    sub ecx,'0'
    mov ebx, eax
    shl eax, 2
    add eax, ebx
    shl eax, 1
    add eax, ecx
    jmp .top
.done:
    pop ebx
    ret

_printDec:
        section .bss
        .decstr resb 10
```

```
        .ct1 resd 1

        section .text
        pushad

        mov dWord[.ct1], 0
        mov edi, .decstr
        add edi, 9
        xor edx, edx
.WhileNotZero:
        mov ebx, 10
        div ebx
        add edx, '0'
        mov byte[edi], dl
        dec edi
        inc dWord[.ct1]
        xor edx, edx
        cmp eax, 0
        jne .WhileNotZero

        inc edi
        mov ecx, edi
        mov edx, [.ct1]
        mov eax, SYS_WRITE
        mov ebx, STDOUT
        int 0x80

        popad
        ret
_start:
        mov edx, a
        mov esi, lena
```

```
        call str2int
        mov ebx, eax
        mov edx, b
        mov esi, lenb
        call str2int
        cmp ebx, eax
        jl check_third_num
        mov ebx, eax
check_third_num:
        mov edx, c
        mov esi, lenc
        call str2int
        cmp ebx, eax
        jl _exit
        mov ebx, eax
_exit:
        mov [minn], ebx
        mov ecx, msg
        mov edx, len
        mov eax,SYS_WRITE
        mov ebx,STDOUT
        int 0x80


        mov eax, [minn]
        call _printDec

        mov ebx, 0
        mov eax, 1
        int 0x80

segment .bss
```

minn resb 4

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

Báo cáo trên lớp thực hành.

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.

- *Tệp tin Notepad*

Sử dụng công cụ CFF Explorer để phân tích tệp tin, nhóm thu được các thông tin quan trọng phục vụ cho công việc chèn thêm chương trình vào tệp tin.

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00007748	00001000	00007800	00000400	00000000	00000000	0000	0000	60000020
.data	00001BA8	00009000	00000800	00007C00	00000000	00000000	0000	0000	C0000040
.rsrc	00008958	0000B000	00008A00	00008400	00000000	00000000	0000	0000	40000040

Mục tiêu là chèn thêm 1 một chương trình hiển thị message box khi khởi chạy Notepad.

Tiến hành viết đoạn chương trình và biên dịch thành một tệp thực thi bằng c++.

```
#include<Windows.h>
int main(int argc, char *argv[]) {
    MessageBox(NULL, L"20520173-20520248", L"MSSV", MB_OK);
    return 0;
}
```

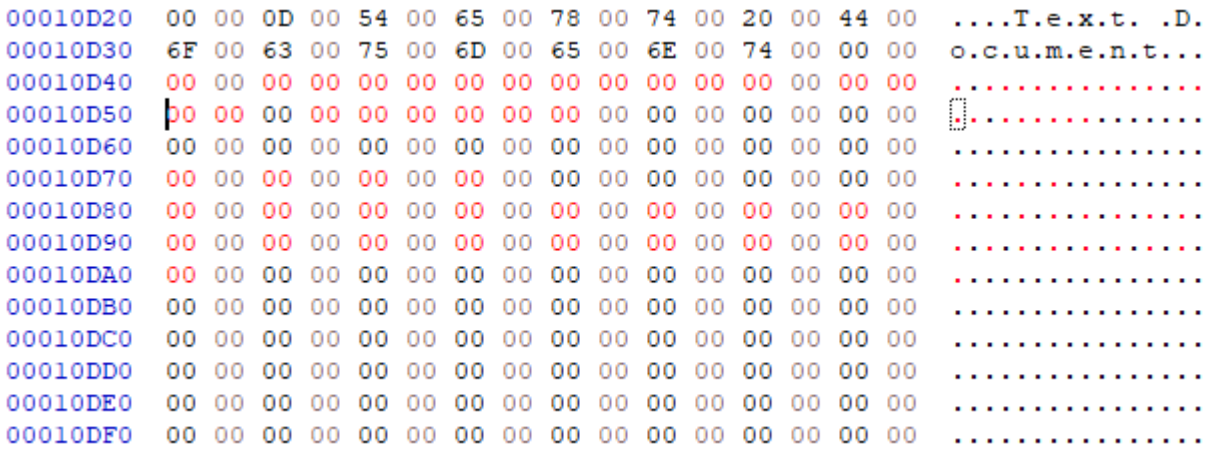
Sau khi có tệp thực thi, tiến hành dùng công cụ IDA Pro để phân tích và lấy đoạn mã Assembly cần chèn vào.

```
                sub_401000 proc near
6A 00           push    0               ; uType
68 00 21 40 00  push    offset Caption  ; "MSSV"
68 0C 21 40 00  push    offset Text     ; "20520173-20520248"
6A 00           push    0               ; hWnd
FF 15 34 20 40 00 call  ds:MessageBoxW
33 C0           xor     eax, eax
C3              retn
                sub_401000 endp
```

Ở đây chương trình gọi hàm MessageBoxW, để có thể sử dụng hàm này chúng ta cần biết địa chỉ của hàm MessageBoxW trong tệp tin thực thi Notepad. Dùng công cụ IDA pro để debug tệp tin thực thi Notepad và lấy địa chỉ này.

00000000010011AC	MessageBeep	USER32
0000000001001268	MessageBoxW	USER32
0000000001001220	MoveWindow	USER32

Dùng công cụ HxD phân tích tệp tin Notepad, nhóm thấy có một khoản trống ở cuối, đây sẽ là nơi chèn thêm đoạn mã vào.



Nhóm xác định từ địa chỉ 00010D40 sẽ là nơi để chèn đoạn mã vào, địa chỉ 00010D70 là nơi chèn đoạn title và địa chỉ 00010D80 là nơi chèn message cho MessageBox.

Đoạn mã cần chèn có cấu trúc như sau:

push 0 ; 6a 00

push Caption ; 68 X

push Text ; 68 Y

push 0 ; 6a 00

call [MessageBoxW] ; ff15 Z

jmp Origianl_Entry_Point ; e9 M

Z đã được xác định ở trên, là địa chỉ của MessageBoxW trong thư viện User32, Z = 68120001

X, Y là các Virtual address của Caption và Text sẽ hiển thị trên MessageBox và được tính dựa trên virtual address và raw address của session .rsrc (do đoạn mã đc chèn thêm vào session này) và raw address của các chuỗi nào.

M là new entry point, địa chỉ bắt đầu mà chương trình bắt đầu thực thi khi khởi chạy.

Tính X

raw address = 0x00010D70

X = virtual address = 0x00010D70 - 0x00008400 + 0x0000B000 = 0x00013970 + ImageBase = 0x01013970

Tính Y

raw address = 0x00010D80

Y = virtual address = 0x00010D80 - 0x00008400 + 0x0000B000 = 0x00013980 + ImageBase = 0x01013980

Tính M

new_entry_point = 0x00010D40 - 0x00008400 + 0x0000B000 = 0x00013940

Để sau khi hiển thị lên MessageBox chương trình sẽ vẫn hoạt động bình thường, nhóm cần tính relative virtual address và cho chương trình nhảy đến địa chỉ này. Để tính được relative_VA, nhóm tính old_entry_point_virtual_address trước và từ đó tính ra relative_VA theo công thức: relative_VA = old_entry_point_virtual_address – 5 - new_entry_point – 4*5

4*5 là để có thể quay lại relative_VA nhóm cần dùng lệnh jmp, trước lệnh jmp có 4 lệnh để gọi được MessageBox, mỗi lệnh có độ lớn 4byte.

Tính relative_VA:

old_entry_point = 0x0000739D (xem trong CFF)

old_entry_point_virtual_address = 0x0000739D + ImageBase = 0x0100739D

relative_VA = 0x0100739D - 0x00000005 - 0x00013940 - 0x00000014 = 0xFFFF3A44

Đoạn mã hoàn chỉnh:

push 0 ; 6a 00

push Caption ; 68 70390101

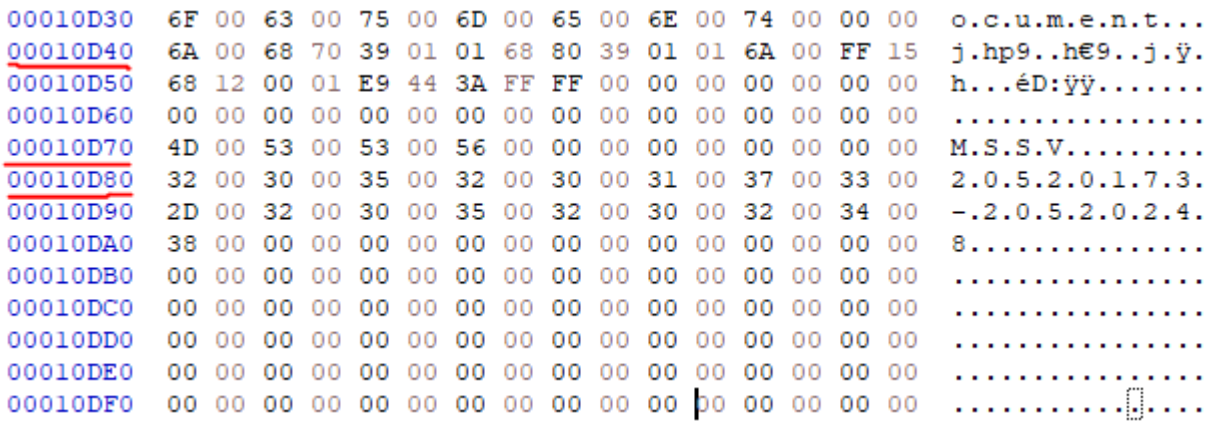
push Text ; 68 80390101

push 0 ; 6a 00

call [MessageBoxW] ; ff15 68120001

jmp Origianl_Entry_Point ; e9 443AFFFF

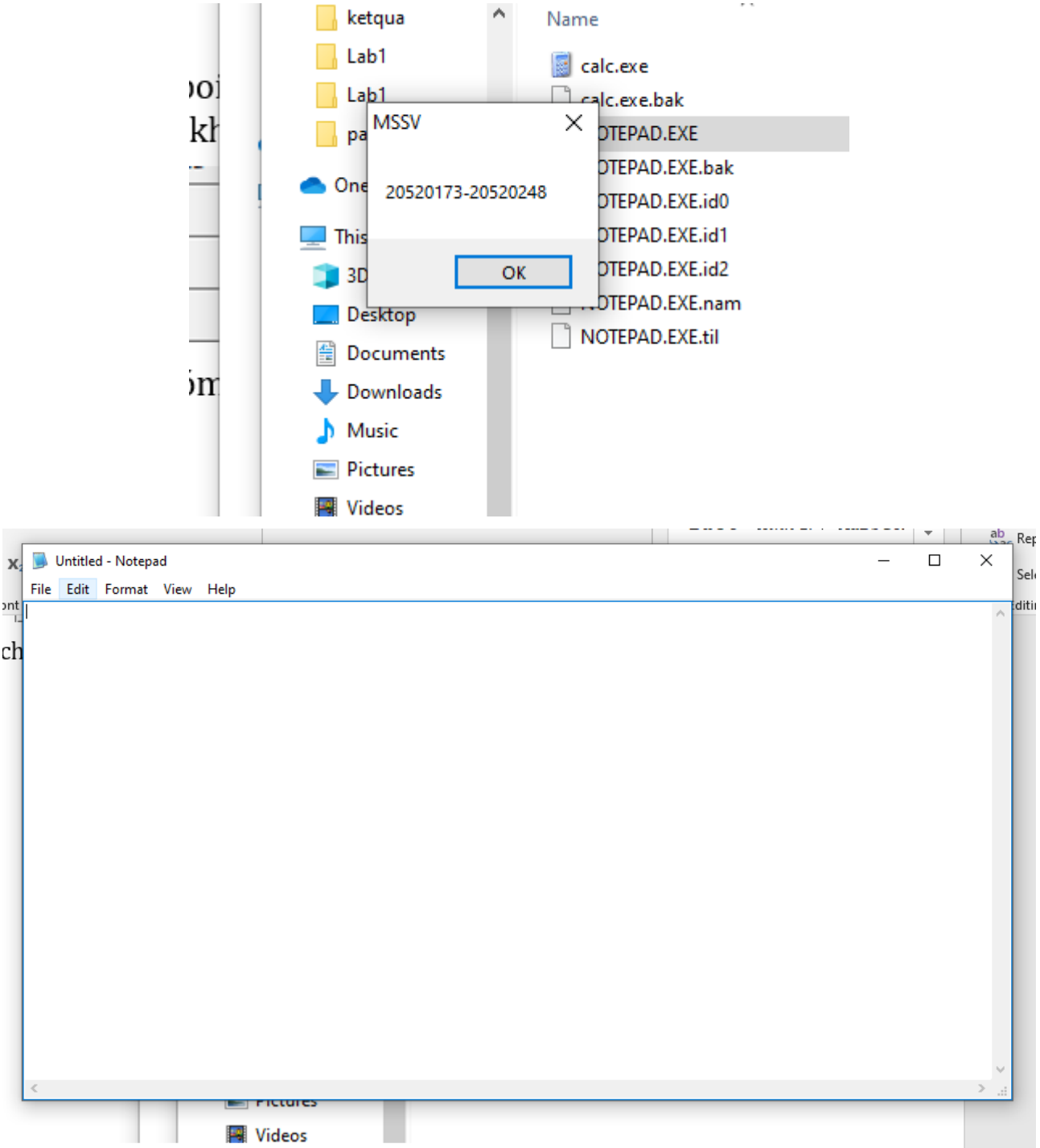
Chuyển đổi thành dãy byte và chèn vào phần trống trong tệp tin thực thi Notepad



Sau đó thay đổi entry_point bằng công cụ CFF để chương trình trỏ vào phần mã đã chèn và thực thi nó đầu tiên khi khởi chạy sau đó mới nhảy về luồng chạy bình thường.

AddressOfEntryPoint	00000108	Dword	00013940	.rsrc
BaseOfCode	0000010C	Dword	00001000	
BaseOfData	00000110	Dword	00009000	

Khởi chạy Notepad nhóm thu được kết quả



- *Tệp tin calc*

Tương tự như đã làm ở tệp thực thi Notepad, nhóm xác định các địa chỉ cần thiết hỗ trợ cho việc chèn thêm vào.

Xác định raw address và virtual address session của .rsrc

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	000126B0	00001000	00012800	00000400	00000000	00000000	0000	0000	60000020
.data	0000101C	00014000	00000A00	00012C00	00000000	00000000	0000	0000	C0000040
.rsrc	00008960	00016000	00008A00	00013600	00000000	00000000	0000	0000	40000040

Xác định địa chỉ của hàm MessageBoxW của thư viện User32 trong tệp tin thực calc.exe

0000000001001104	MessageBeep	USER32
00000000010011A8	MessageBoxW	USER32
0000000001001150	OffsetRect	USER32

Xác định địa chỉ các nơi để chèn mã và dữ liệu vào, đoạn mã được chèn tại địa chỉ 0001BF60, title được chèn tại 0001BF80 và mess được chèn tại 0001BF90.

```
0001BF20  11 00 4E 00 6F 00 74 00 20 00 45 00 6E 00 6F 00  ..N.o.t. .E.n.o.
0001BF30  75 00 67 00 68 00 20 00 4D 00 65 00 6D 00 6F 00  u.g.h. .M.e.m.o.
0001BF40  72 00 79 00 00 00 00 00 00 00 00 00 00 00 00 00  r.y.............
0001BF50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001BF60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001BF70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001BF80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001BF90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001BFA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001BFB0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001BFC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001BFD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Tính các địa chỉ cần thiết

Tính X

raw address = 0x0001bf80

virtual address = 0x0001bf80 - 0x00013600 + 0x00016000 = 0x0001E980 + ImageBase = 0x0101E980

Tính Y

raw address = 0x0001bf90

virtual address = 0x0001bf90 - 0x00013600 + 0x00016000 = 0x0001E990 + ImageBase = 0x0101E990

Tính M

new_entry_point = 0x0001bf50 - 0x00013600 + 0x00016000 = 0x0001E950

Tính relative_VA

old_entry_point = 0x00012475

old_entry_point_virtual_address = 0x00012475 + ImageBase = 0x01012475

relative_VA = 0x01012475 - 0x00000005 - 0x0001E950 - 0x00000014 = 0xFFFF3B0C

Đoạn mã hoàn chỉnh:

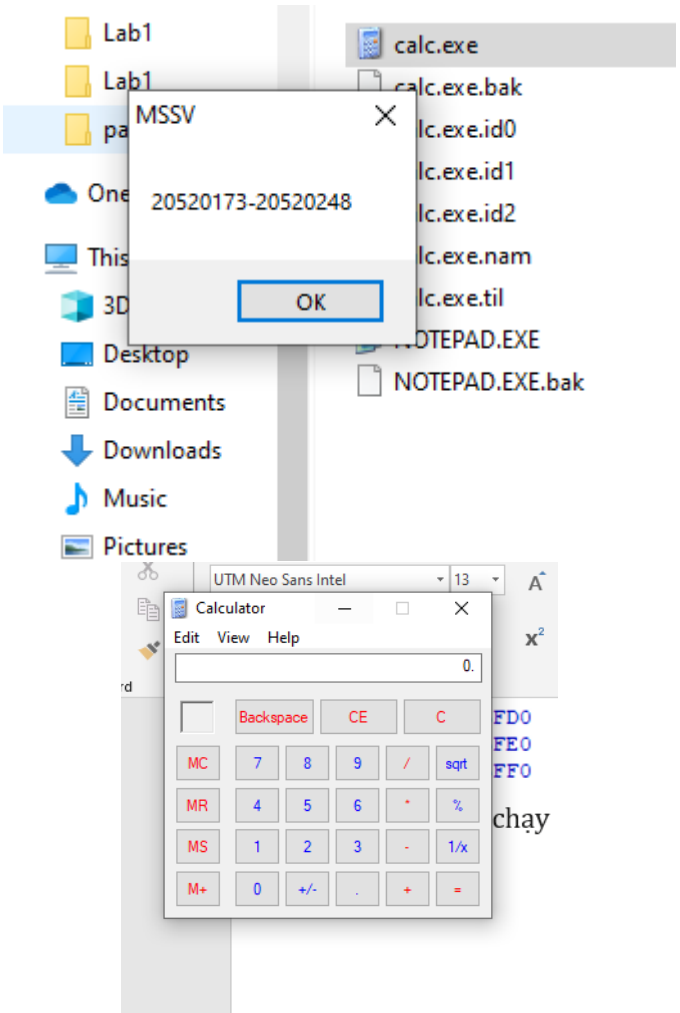
```
push 0 ; 6a 00
push Caption ; 68 80e90101
```

push Text ; 68 90e90101
push 0 ; 6a 00
call [MessageBoxW] ; ff15 a8110001
jmp Origianl_Entry_Point ; e9 0c3bffff

Chuyển thành dãy byte và chèn vào phần trống trong tệp tin thực thi calc.exe

```
0001BF30  75 00 67 00 68 00 20 00 4D 00 65 00 6D 00 6F 00  u.g.h. .M.e.m.o.
0001BF40  72 00 79 00 00 00 00 00 00 00 00 00 00 00 00 00  r.y.............
0001BF50  6A 00 68 80 E9 01 01 68 90 E9 01 01 6A 00 FF 15  j.h€é..h.é..j.ÿ.
0001BF60  A8 11 00 01 E9 0C 3B FF FF 00 00 00 00 00 00 00  ¨...é.;ÿÿ.......
0001BF70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001BF80  4D 00 53 00 53 00 56 00 00 00 00 00 00 00 00 00  M.S.S.V.........
0001BF90  32 00 30 00 35 00 32 00 30 00 31 00 37 00 33 00  2.0.5.2.0.1.7.3.
0001BFA0  2D 00 32 00 30 00 35 00 32 00 30 00 32 00 34 00  -.2.0.5.2.0.2.4.
0001BFB0  38 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  8...............
0001BFC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001BFD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001BFE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001BFF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Kết quả chạy



Link youtube demo BT5:
https://youtu.be/5U0_qKfFa0I

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