BÁO CÁO THỰC HÀNH

Môn: Lập trình hệ thống

Buổi báo cáo: Lab 04

Lớp: NT209.O22.ANTT.2

THÔNG TIN CHUNG

STT	Họ và Tên	MSSV	Lớp
1	Trần Tuấn Anh	22520080	ATTT2022.1
2	Nguyễn Khắc Hậu	22520410	ATTT2022.1

Báo Cáo Chi Tiết

Câu 1

```
.text:08048669                sub     esp, 0Ch
.text:0804866C                push    offset aEnterTheHardCo ; "Enter the hard-coded password (option 1"...
.text:08048671                call    _puts
.text:08048676                add     esp, 10h
.text:08048679                sub     esp, 8
.text:0804867C                lea     eax, [ebp+s1]
.text:08048682                push    eax
.text:08048683                push    offset asc_804916A ; "%[^\n]"
.text:08048688                call    ___isoc99_scanf
.text:0804868D                add     esp, 10h
.text:08048690                sub     esp, 8
.text:08048693                lea     eax, [ebp+s1]
.text:08048699                push    eax
.text:0804869A                push    offset format   ; "Your input hard-coded password: %s\n"
.text:0804869F                call    _printf
.text:080486A4                add     esp, 10h
.text:080486A7                sub     esp, 8
.text:080486AA                push    offset s2       ; "Work hard in silence. Let success make "...
.text:080486AF                lea     eax, [ebp+s1]
.text:080486B5                push    eax             ; s1
.text:080486B6                call    _strcmp
.text:080486BB                add     esp, 10h
.text:080486BE                test    eax, eax
.text:080486C0                jnz     short loc_80486C9
.text:080486C2                call    success_1
.text:080486C7                jmp     short loc_80486CE
```

```
.text:080486C9 loc_80486C9:                            ; CODE XREF: hardCode+65↑j
.text:080486C9                 call    failed
```

Đây là đoạn code của hàm xử lý hardcode của câu 1.

Input đầu vào được lưu vào địa chỉ [ebp+s1].

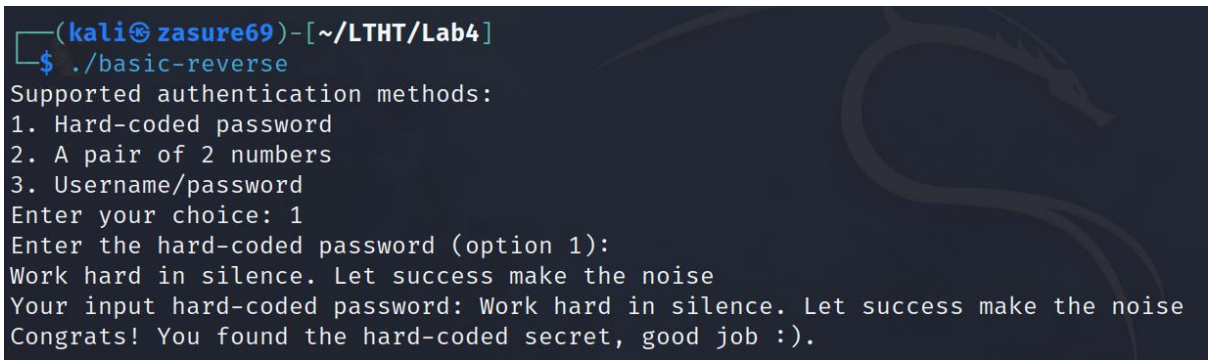
Sau đó câu lệnh push offset s2 đẩy chuỗi s2 vào stack.

Chuỗi input được đẩy vào stack và gọi hàm strcmp để so sánh.

Hàm success_1 sẽ được gọi nếu ZF = 1 (được gán bằng câu lệnh test eax, eax), ngược lại nếu ZF = 0, lệnh jnz sẽ nhảy đến nhãn loc_80486C9 và gọi hàm failed.

Để ZF = 1 → strcmp trả về 0 → 2 chuỗi so sánh phải bằng nhau → Chuỗi cần tìm là s2 = "Work hard in silence. Let success make the noise".

Kết quả thực thi:



Câu 2

Hàm xử lý của câu 2 là otherhardCode

```
push    offset aEnterYour2Numb ; "Enter your 2 numbers (separated by spac"...
call    _puts
add     esp, 10h
sub     esp, 4
lea     eax, [ebp+var_14]
push    eax
lea     eax, [ebp+var_10]
push    eax
push    offset aDD      ; "%d %d"
call    ___isoc99_scanf
add     esp, 10h
```

2 số input đầu vào sẽ được lưu ở 2 ô nhớ [ebp+var_14] và [ebp+var_10]

```
mov     [ebp+var_C], 3
mov     eax, [ebp+var_10]
cmp     eax, [ebp+var_C]
jnz     short loc_804875C
mov     edx, [ebp+var_10]
mov     eax, [ebp+var_10]
mov     eax, ds:funny_seq[eax*4]
sub     esp, 8
push    edx
push    eax
call    funny_func

loc_804875C:                            ; CODE XREF: otherhardCode+58↑j
call    failed
```

Ta thấy ô nhớ [ebp+var_C] được gán giá trị 3, và thanh ghi eax được gán giá trị của input thứ 1.

Lệnh cmp eax, [ebp+var_C] so sánh 2 giá trị input 1 và 3 nếu không bằng sẽ gọi hàm failed → **input 1 sẽ là 3**.

Hàm funny_func được gọi và truyền 2 tham số là input 1 và ds:funny_seq[3*4] (eax được gán giá trị input 1)

```
.rodata:08048A60                public funny_seq
.rodata:08048A60 funny_seq      dd 2                    ; DATA XREF: otherhardCode+60↑r
.rodata:08048A64                db    4
.rodata:08048A65                db    0
.rodata:08048A66                db    0
.rodata:08048A67                db    0
.rodata:08048A68                db    6
.rodata:08048A69                db    0
.rodata:08048A6A                db    0
.rodata:08048A6B                db    0
.rodata:08048A6C                db    8
.rodata:08048A6D                db    0
.rodata:08048A6E                db    0
.rodata:08048A6F                db    0
.rodata:08048A70                db    0
.rodata:08048A71                db    0
.rodata:08048A72                db    0
.rodata:08048A73                db    0
.rodata:08048A74                db    1
.rodata:08048A75                db    0
.rodata:08048A76                db    0
.rodata:08048A77                db    0
.rodata:08048A78                db    3
.rodata:08048A79                db    0
```

Phần tử funny_seq[3*4] = 8

```
                public funny_func
funny_func      proc near               ; CODE XREF: otherhardCode+6C↓p

var_4           = dword ptr -4
arg_0           = dword ptr  8
arg_4           = dword ptr  0Ch

                push    ebp
                mov     ebp, esp
                sub     esp, 10h
                mov     [ebp+var_4], 0
                mov     edx, [ebp+arg_0]
                mov     eax, [ebp+arg_4]
                add     eax, edx
                lea     ecx, [eax-1]
                mov     edx, [ebp+arg_0]
                mov     eax, [ebp+arg_4]
                add     eax, edx
                imul    eax, ecx
                mov     [ebp+var_4], eax
                mov     eax, [ebp+var_4]
                leave
                retn
funny_func      endp
```

Đây là nội dung hàm funny_func sẽ trả về kết quả biểu thức (arg_0 + arg_4 - 1) * (arg_0 + arg_4) mà arg_0 và arg_4 lần lượt là 8 và 3 →hàm trả về 110.

```
                mov     edx, eax
                mov     eax, [ebp+var_14]
                cmp     edx, eax
                jnz     short loc_8048755
                call    success_2
                jmp     short loc_8048761
; ---------------------------------------------------------------------------

loc_8048755:                            ; CODE XREF: otherhardCode+7B↑j
                call    failed
                jmp     short loc_8048761
; ---------------------------------------------------------------------------
```

Sau khi gọi hàm funny_func sẽ so sánh giá trị trả về từ hàm funny_func với input 2 nếu không bằng nhau sẽ gọi hàm failed → **input 2 = kết quả trả về của hàm funny_func = 110**.

Kết quả thực thi:

```
┌──(kali㉿zasure69)-[~/LTHT/Lab4]
└─$ ./basic-reverse
Supported authentication methods:
1. Hard-coded password
2. A pair of 2 numbers
3. Username/password
Enter your choice: 2
Enter your 2 numbers (separated by space) (option 2):
3 110
Your input: 3 110
Congrats! You found a secret pair of numbers :).
```

Câu 3:

Hàm xử lý của câu 3 là hàm userpass

```
push    offset aEnterYourUsern ; "Enter your username:"
call    _puts
add     esp, 10h
sub     esp, 8
lea     eax, [ebp+s]
push    eax
push    offset asc_804916A ; "%[^\n]"
call    ___isoc99_scanf
add     esp, 10h
call    _getchar
sub     esp, 0Ch
push    offset aEnterYourPassw ; "Enter your password:"
call    _puts
add     esp, 10h
sub     esp, 8
lea     eax, [ebp+var_25]
push    eax
push    offset asc_804916A ; "%[^\n]"
call    ___isoc99_scanf
```

2 input username, password sẽ được lưu lần lượt vào 2 ô nhớ [ebp+s] và [ebp+var_25].

Và username = 0080-0410.

```
add     esp, 10h
sub     esp, 0Ch
lea     eax, [ebp+s]
push    eax             ; s
call    _strlen
add     esp, 10h
cmp     eax, 9
jnz     short loc_8048821
sub     esp, 0Ch
lea     eax, [ebp+s]
push    eax             ; s
call    _strlen
add     esp, 10h
mov     ebx, eax
sub     esp, 0Ch
lea     eax, [ebp+var_25]
push    eax             ; s
call    _strlen
add     esp, 10h
cmp     ebx, eax
jz      short loc_804882B

loc_8048821:                            ; CODE XREF: userpass+97↑j
call    failed
jmp     loc_804891B
```

Đoạn lệnh này kiểm tra độ dài của username và password có độ dài bằng 9 không nếu không sẽ gọi hàm failed nếu không sẽ nhảy đến nhãn loc_804882B → password có độ dài bằng 9.

```
loc_804882B:                            ; CODE XREF: userpass+BB↑j
mov     [ebp+var_C], 0
mov     [ebp+var_C], 0
jmp     short loc_804888E
```

Gán 0 vào ô nhớ [ebp+var_C] → nhảy đến nhãn loc_804888E.

```
loc_804888E:                            ; CODE XREF: userpass+D5↑j
cmp     [ebp+var_C], 8
jle     short loc_804883B
mov     [ebp+var_C], 0
jmp     short loc_80488DC
```

Kiểm tra giá trị ô nhớ [ebp+var_C] <= 8 → nhảy đến nhãn loc_804883B

```
loc_804883B:                            ; CODE XREF: userpass+12E↓j
                cmp     [ebp+var_C], 1
                jg      short loc_8048858
                mov     eax, [ebp+var_C]
                add     eax, 2
                movzx   eax, [ebp+eax+s]
                lea     ecx, [ebp+var_2E]
                mov     edx, [ebp+var_C]
                add     edx, ecx
                mov     [edx], al
                jmp     short loc_804888A
```

Kiểm tra giá trị ô nhớ [ebp+var_C] < 1→gán kí tự thứ 3 (kí tự '8') của username vào ô nhớ [ebp+var_2E] ([ebp+var_C] = 0) →nhảy đến nhãn loc_804888A.

```
loc_804888A:                            ; CODE XREF: userpass+F2↑j
                                        ; userpass+10F↑j
                add     [ebp+var_C], 1

loc_804888E:                            ; CODE XREF: userpass+D5↑j
                cmp     [ebp+var_C], 8
                jle     short loc_804883B
                mov     [ebp+var_C], 0
                jmp     short loc_80488DC
```

Tăng giá trị ô nhớ [ebp+var_C] lên 1 và tiếp tục vòng lặp từ nhãn loc_804888E.

```
loc_804883B:                            ; CODE XREF: userpass+12E↓j
                cmp     [ebp+var_C], 1
                jg      short loc_8048858
                mov     eax, [ebp+var_C]
                add     eax, 2
                movzx   eax, [ebp+eax+s]
                lea     ecx, [ebp+var_2E]
                mov     edx, [ebp+var_C]
                add     edx, ecx
                mov     [edx], al
                jmp     short loc_804888A
```

Sau khi chạy đoạn lệnh này ô nhớ [ebp+var_2E +1] ([ebp+var_C] = 1) được gán giá trị bằng kí tự thứ 4 của username (kí tự '0')

Ở lần lặp tiếp theo, [ebp+var_C] = 2 > 1 → nhảy đến nhãn loc_8048858.

```
loc_8048858:                            ; CODE XREF: userpass+DB↑j
                cmp     [ebp+var_C], 3
                jg      short loc_8048875
                mov     eax, [ebp+var_C]
                add     eax, 5
                movzx   eax, [ebp+eax+s]
                lea     ecx, [ebp+var_2E]
                mov     edx, [ebp+var_C]
                add     edx, ecx
                mov     [edx], al
                jmp     short loc_804888A
```

Sau khi thực thi đoạn lệnh này khi [ebp+var_C] = 2 và [ebp+var_C] = 3, ô nhớ [ebp+var_2E+2] và [ebp+var_2E+3] sẽ được gán bằng ký tự thứ 8 và thứ 9 của chuỗi username (kí tự '1' và '0').

Ở lần lặp tiếp theo, [ebp+var_C] = 4 > 3 → nhảy đến nhãn loc_8048875.

```
loc_8048875:                            ; CODE XREF: userpass+F8↑j
                mov     eax, [ebp+var_C]
                sub     eax, 4
                movzx   eax, [ebp+eax+var_11]
                lea     ecx, [ebp+var_2E]
                mov     edx, [ebp+var_C]
                add     edx, ecx
                mov     [edx], al

loc_804888A:                            ; CODE XREF: userpass+F2↑j
                                        ; userpass+10F↑j
                add     [ebp+var_C], 1

loc_804888E:                            ; CODE XREF: userpass+D5↑j
                cmp     [ebp+var_C], 8
                jle     short loc_804883B
                mov     [ebp+var_C], 0
                jmp     short loc_80488DC
```

Sau khi thực hiện lệnh ô nhớ [ebp+var_2E+4] sẽ được gán bằng giá trị trong ô nhớ [ebp+var_11] ([ebp+var_C]-4 = 0). Giá trị trong ô nhớ [ebp+var_11] bằng 0x42 hay kí tự 'B'.

```
mov     [ebp+var_11], 42h
mov     [ebp+var_10], 2Fh
mov     [ebp+var_F], 43h
mov     [ebp+var_E], 46h
mov     [ebp+var_D], 59h

var_11          = byte ptr -11h
var_10          = byte ptr -10h
var_F           = byte ptr -0Fh
var_E           = byte ptr -0Eh
var_D           = byte ptr -0Dh
```

Ở các vòng lặp tiếp theo, khi [ebp+var_C] = 5, 6, 7, 8 các ô nhớ [ebp+var_2E+5], [ebp+var_2E+6] , [ebp+var_2E+7], [ebp+var_2E+8] sẽ được gán các giá trị trong các ô nhớ [ebp+var_11+1] hay [ebp+var_10], [ebp+var_11+2] hay [ebp+var_F], [ebp+var_11+3] hay [ebp+var_E], [ebp+var_11+4] hay [ebp+var_E], [ebp+var_11+5] hay [ebp+var_D] với các kí tự '/', 'C', 'F', 'Y'.

```
loc_804888E:                            ; CODE XREF: userpass+D5↑j
                cmp     [ebp+var_C], 8
                jle     short loc_804883B
                mov     [ebp+var_C], 0
                jmp     short loc_80488DC
```

Sau khi kết thúc vòng lặp [ebp+var_C] > 8, chuỗi được lưu trong bộ nhớ (tạm gọi s) là "8010B/CFY". [ebp+var_C] được gán bằng 0 và nhảy đến nhãn loc_80488DC.

```
loc_80488DC:                            ; CODE XREF: userpass+137↑j
                sub     esp, 0Ch
                lea     eax, [ebp+s]
                push    eax             ; s
                call    _strlen
                add     esp, 10h
                mov     edx, eax
                mov     eax, [ebp+var_C]
                cmp     edx, eax
                ja      short loc_804889D
                jmp     short loc_80488F7
```

Kiểm tra [ebp+var_C] = 0 < 9 → nhảy đến nhãn loc_804889D.

```
loc_804889D:                            ; CODE XREF: userpass+18E↓j
                lea     edx, [ebp+s]
                mov     eax, [ebp+var_C]
                add     eax, edx
                movzx   eax, byte ptr [eax]
                movsx   edx, al
                lea     ecx, [ebp+var_2E]
                mov     eax, [ebp+var_C]
                add     eax, ecx
                movzx   eax, byte ptr [eax]
                movsx   eax, al
                add     eax, edx
                mov     edx, eax
                shr     edx, 1Fh
                add     eax, edx
                sar     eax, 1
                mov     ecx, eax
                lea     edx, [ebp+var_25]
                mov     eax, [ebp+var_C]
                add     eax, edx
                movzx   eax, byte ptr [eax]
                movsx   eax, al
                cmp     ecx, eax
                jnz     short loc_80488F6
                add     [ebp+var_C], 1
```

Đoạn lệnh này lấy 2 kí tự thứ [ebp+var_C] ở 2 chuỗi username và chuỗi được lưu trong bộ nhớ s = “8010B/CFY” cộng lại và shift phải 1 bit. Sau đó so sánh với kí tự thứ [ebp+var_C] trong chuỗi password nếu không bằng nhau thì hàm failed sẽ được gọi như đoạn lệnh bên dưới.

```
loc_80488F6:                            ; CODE XREF: userpass+172↑j
                nop

loc_80488F7:                            ; CODE XREF: userpass+190↑j
                sub     esp, 0Ch
                lea     eax, [ebp+s]
                push    eax             ; s
                call    _strlen
                add     esp, 10h
                mov     edx, eax
                mov     eax, [ebp+var_C]
                cmp     edx, eax
                jnz     short loc_8048916
                call    success_3
                jmp     short loc_804891B
; ---------------------------------------------------------------------------

loc_8048916:                            ; CODE XREF: userpass+1A9↑j
                call    failed

loc_804891B:                            ; CODE XREF: userpass+C2↑j
                                        ; userpass+1B0↑j
                nop
                mov     ebx, [ebp+var_4]
                leave
                retn
```

→ password[i] = (username[i] + s[i]) >> 1

hay (username[i] + s[i]) / 2.

password[0] = (‘0’ + ‘8’) / 2 = ‘4’

password[1] = (‘0’ + ‘0’) / 2 = ‘0’

password[2] = (‘8’ + ‘1’) / 2 = ‘4’

password[3] = (‘0’ + ‘0’) / 2 = ‘0’

password[4] = (‘-’ + ‘B’) / 2 = ‘7’

password[5] = (‘0’ + ‘/’) / 2 = ‘/’

password[6] = (‘4’ + ‘C’) / 2 = ‘;’

password[7] = (‘1’ + ‘F’) / 2 = ‘;’

password[8] = (‘0’ + ‘Y’) / 2 = ‘D’

Sau khi áp dụng công thức ta tính ra được password = “40407/;;D”

Kết quả thực thi: