TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN – ĐHQG-HCM
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

BÁO CÁO THỰC HÀNH

Bài thực hành số 04: Tìm hiểu cơ bản về Kỹ thuật Dịch ngược

Môn học: Lập trình hệ thống

Lớp: NT209.Q13.ANTT.2

THÀNH VIÊN THỰC HIỆN (Nhóm 6):

STT	Họ và tên	MSSV
1	Huỳnh Đăng Khoa	24520815
2	Phan Hoàng Dũng	24520348
3	Phạm Ngọc Dũng	24520346

NỘI DUNG BÁO CÁO:

1. Phân tích và tìm passphrase cố định của basic-reverse với option 1.

Khi nhập số 1 vào prompt, chương trình nhảy tới hàm hardcode:

```
.text:08048964                mov     eax, [ebp+var_14]
.text:08048967                cmp     eax, 1
.text:0804896A                jnz     short loc_8048973
.text:0804896C                call    hardCode
.text:08048971                jmp     short loc_80489AB
```

Quan sát hàm hardcode:

```
.text:08048659 hardCode        proc near               ; CODE XREF: main+4E↓p
.text:08048659
.text:08048659 s1              = byte ptr -3F0h
.text:08048659
.text:08048659                 push    ebp
.text:0804865A                 mov     ebp, esp
.text:0804865C                 sub     esp, 3F8h
.text:08048662                 call    _getchar
.text:08048667                 sub     esp, 0Ch
.text:0804866A                 push    offset aEnterTheHardCo ; "Enter the hard-coded password (option 1"...
.text:0804866F                 call    _puts
.text:08048674                 add     esp, 10h
.text:08048677                 sub     esp, 8
.text:0804867A                 lea     eax, [ebp+s1]
.text:08048680                 push    eax
.text:08048681                 push    offset asc_8048B7E ; "%[^\n]"
.text:08048686                 call    ___isoc99_scanf
.text:0804868B                 add     esp, 10h
.text:0804868E                 sub     esp, 8
.text:08048691                 lea     eax, [ebp+s1]
.text:08048697                 push    eax
.text:08048698                 push    offset format   ; "Your input hard-coded password: %s\n"
.text:0804869D                 call    _printf
.text:080486A2                 add     esp, 10h
.text:080486A5                 sub     esp, 8
.text:080486A8                 push    offset s2       ; "No pains no gains"
.text:080486AD                 lea     eax, [ebp+s1]
```

```
.text:080486A8                 push    offset s2       ; "No pains no gains"
.text:080486AD                 lea     eax, [ebp+s1]
.text:080486B3                 push    eax             ; s1
.text:080486B4                 call    _strcmp
.text:080486B9                 add     esp, 10h
.text:080486BC                 test    eax, eax
.text:080486BE                 jnz     short loc_80486C7
.text:080486C0                 call    success_1
.text:080486C5                 jmp     short loc_80486CC
```

Các hàm liên quan:

_getchar : hàm in chuỗi.

_puts: hàm xuất chuỗi prompt.

__isoc99_scanf: đọc và quét dữ liệu input của người nhập.

_printf: hàm in lại chuỗi vừa nhập.

_strcmp: hàm so sánh 2 chuỗi.

success_1: hàm riêng, dùng để in thông báo thành công

```
public success_1
success_1 proc near
; __unwind {
push    ebp
mov     ebp, esp
sub     esp, 8
sub     esp, 0Ch
push    offset s        ; "Congrats! You found the hard-coded secr"...
call    _puts
add     esp, 10h
nop
leave
retn
; } // starts at 80485F5
success_1 endp
```

failed: hàm riêng, dùng khi in thông báo thất bại:

```
public failed
failed proc near
; __unwind {
push    ebp
mov     ebp, esp
sub     esp, 8
sub     esp, 0Ch
push    offset aNiceTryButThat ; "Nice try but that is not the answer."
call    _puts
add     esp, 10h
nop
leave
retn
; } // starts at 8048640
failed endp
```

Giải thích:

Đầu tiên, ta thấy có thông báo yêu cầu nhập hard-coded password:

```
.text:0804866A                  push    offset aEnterTheHardCo ; "Enter the hard-coded password (option 1"...
```

Tiếp theo, ta lại thấy có 2 chuỗi là s1 được gán trong %eax và chuỗi offset s2 được biết là "No pains no gains", cùng với đó là hàm _strcmp có tác dụng so sánh 2 chuỗi này. Kết quả của hàm sẽ trả về 0 (nếu 2 chuỗi giống nhau) và 1 (nếu 2 chuỗi khác nhau) và được lưu trong %eax.

```
.text:080486A2                  add     esp, 10h
.text:080486A5                  sub     esp, 8
.text:080486A8                  push    offset s2       ; "No pains no gains"
.text:080486AD                  lea     eax, [ebp+s1]
.text:080486B3                  push    eax             ; s1
.text:080486B4                  call    _strcmp
```

Sau đó, ta tiếp tục kiểm tra %eax (kết quả trả về của so sánh chuỗi). Nếu kết quả là 0 thì ta gọi hàm success_1 (hàm báo thành công).

```
.text:080486B9                  add     esp, 10h
.text:080486BC                  test    eax, eax
.text:080486BE                  jnz     short loc_80486C7
.text:080486C0                  call    success_1
```

Nếu kết quả của %eax là 1 thì nhảy đến short loc_80486C7 (nhãn có gọi hàm failed):

```
loc_80486C7:
                call    failed

loc_80486CC:
                nop
                leave
                retn
hardCode        endp
```

Vậy, password ta cần phải nhập là "No pains no gains".Tiến hành nhập chuỗi vào xác minh phân tích, ta thấy trả về đáp án đúng.

```
Enter your choice: 1
Enter the hard-coded password (option 1):
No pains no gains
Your input hard-coded password: No pains no gains
Congrats! You found the hard-coded secret, good job :).
```

2. Phân tích và tìm cặp số nguyên của basic-reverse với option 2.

Khi nhập số 2 vào prompt, chương trình nhảy tới hàm otherhardCode:

```
; =============== S U B R O U T I N E =======================================

; Attributes: bp-based frame

                public otherhardCode
otherhardCode   proc near               ; CODE XREF: main+5D↓p

var_18          = dword ptr -18h
var_14          = dword ptr -14h
var_10          = dword ptr -10h
var_C           = dword ptr -0Ch

                push    ebp
                mov     ebp, esp
                sub     esp, 18h
                call    _getchar
                sub     esp, 0Ch
                push    offset aEnterYour2Numb ; "Enter your 2 numbers (separated by spac"...
                call    _puts
                add     esp, 10h
                sub     esp, 4
                lea     eax, [ebp+var_18]
                push    eax
                lea     eax, [ebp+var_14]
                push    eax
                push    offset aDD      ; "%d %d"
                call    ___isoc99_scanf
                add     esp, 10h
                mov     edx, [ebp+var_18]
                mov     eax, [ebp+var_14]
                sub     esp, 4
                push    edx
                push    eax
                push    offset aYourInputDD ; "Your input: %d %d\n"
                call    _printf
                add     esp, 10h
```

```
                mov     edx, [ebp+var_18]
                mov     eax, [ebp+var_14]
                sub     esp, 4
                push    edx
                push    eax
                push    offset aYourInputDD ; "Your input: %d %d\n"
                call    _printf
                add     esp, 10h
                mov     [ebp+var_C], 9
                mov     [ebp+var_10], 0
                mov     eax, [ebp+var_14]
                cmp     eax, [ebp+var_C]
                jnz     short loc_8048759
                mov     eax, [ebp+var_14]
                sub     esp, 8
                push    eax
                push    [ebp+var_10]
                call    funny_func
                add     esp, 10h
                mov     edx, eax
                mov     eax, [ebp+var_18]
                cmp     edx, eax
                jnz     short loc_8048752
                call    success_2
                jmp     short loc_804875E
; ---------------------------------------------------------------------------

loc_8048752:                            ; CODE XREF: otherhardCode+7A↑j
                call    failed
                jmp     short loc_804875E
; ---------------------------------------------------------------------------

loc_8048759:                            ; CODE XREF: otherhardCode+5F↑j
                call    failed

loc_804875E:                            ; CODE XREF: otherhardCode+81↑j
                                        ; otherhardCode+88↑j
                nop
                leave
                retn
otherhardCode   endp
```

Ta dùng tính năng dịch của IDA để chuyển dạng hợp ngữ sang mã giả dạng C để dễ dàng phân tích:

```
 1 int otherhardCode()
 2 {
 3   int v0; // edx@2
 4   int result; // eax@3
 5   int v2; // [sp+0h] [bp-18h]@1
 6   int v3; // [sp+4h] [bp-14h]@1
 7   int v4; // [sp+8h] [bp-10h]@1
 8   int v5; // [sp+Ch] [bp-Ch]@1
 9
10   getchar();
11   puts("Enter your 2 numbers (separated by space) (option 2):");
12   __isoc99_scanf("%d %d", &v3, &v2);
13   printf("Your input: %d %d\n", v3, v2);
14   v5 = 9;
15   v4 = 0;
16   if ( v3 == 9 )
17   {
18     v0 = funny_func(v4, 9);
19     if ( v0 == v2 )
20       result = success_2();
21     else
22       result = failed();
23   }
24   else
25   {
26     result = failed();
27   }
28   return result;
29 }
```

Ta quan sát thấy được 2 lệnh if lồng có điều kiện so sánh là v3 == 9 và v0 == v2. Nếu 2 điều kiện này thỏa mãn thì sẽ gọi hàm success_2() nên v3 và v0 sẽ là 2 số nguyên cần tìm.

*Tìm v3:

Lệnh if đầu tiên so sánh v3 == 9, nếu v3 khác 9 thì chương trình sẽ gọi hàm failed(). Do đó v3 phải bằng 9 để tiếp tục xét điều kiện thứ 2 ⇒ v3 = 9.

*Tìm v0:

- Ở dòng 18, ta thấy v0 được gán bằng kết quả của hàm funny_func(v4, 9) với v4 đã được gán bằng 0.
- Tìm đến hàm funny_func(), ta thấy hàm trả về kết quả của phép tính (a1 + a2) * (a1 + a2), ứng với 2 tham số là v4, 9 thì v0 có kết quả là (0 + 9) * (0 + 9) = 81⇒ v0 = 81.

```
1 int __cdecl funny_func(int a1, int a2)
2 {
3   return (a1 + a2) * (a1 + a2);
4 }
```

Vậy 2 số cần tìm là 9 và 81.

Thực thi chương trình với cặp số nguyên tìm được, chương trình cho kết quả đúng:

```
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):
9 81
Your input: 9 81
Congrats! You found a secret pair of numbers :).
```

3. Phân tích, tìm cặp username/password phù hợp của basic-reverse với option 3.

Dựa theo lưu ý, ta lấy username bằng cách lấy 3 số cuối của mã số sinh viên:

24520**346** - Phạm Ngọc Dũng

24520**348** - Phan Hoàng Dũng

24520**815** - Huỳnh Đăng Khoa

Ta được username là **346348815**.

Khi nhập số 3 vào prompt, chương trình nhảy tới hàm userpass:

```
                public userpass
userpass        proc near               ; CODE XREF: main+6C↓p

var_2E          = byte ptr -2Eh
var_25          = byte ptr -25h
s               = byte ptr -1Bh
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
var_C           = dword ptr -0Ch
var_4           = dword ptr -4

; __unwind {
                push    ebp
                mov     ebp, esp
                push    ebx
                sub     esp, 34h
                mov     [ebp+var_11], 75h ; 'u'
                mov     [ebp+var_10], 32h ; '2'
                mov     [ebp+var_F], 5Eh ; '^'
                mov     [ebp+var_E], 23h ; '#'
                mov     [ebp+var_D], 50h ; 'P'
                call    _getchar
                sub     esp, 0Ch
                push    offset aEnterYourUsern ; "Enter your username:"
                call    _puts
                add     esp, 10h
                sub     esp, 8
                lea     eax, [ebp+s]
                push    eax
                push    offset asc_8048B7E ; "%[^\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_8048B7E ; "%[^\n]"
                call    ___isoc99_scanf
                add     esp, 10h
                sub     esp, 4
                lea     eax, [ebp+var_25]
                push    eax
                lea     eax, [ebp+s]
                push    eax
                push    offset aYourInputUsern ; "Your input username: %s and password: %"...
                call    _printf
                add     esp, 10h
                sub     esp, 0Ch
                lea     eax, [ebp+s]
                push    eax             ; s
                call    _strlen
                add     esp, 10h
                cmp     eax, 9
                jnz     short loc_804881E
                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_8048828

loc_804881E:                            ; CODE XREF: userpass+97↑j
                call    failed
                jmp     loc_8048918
; ---------------------------------------------------------------------------

loc_8048828:                            ; CODE XREF: userpass+BB↑j
                mov     [ebp+var_C], 0
                mov     [ebp+var_C], 0
                jmp     short loc_804888B
; ---------------------------------------------------------------------------

loc_8048838:                            ; CODE XREF: userpass+12E↓j
                cmp     [ebp+var_C], 1
                jg      short loc_8048855
                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_8048887
; ---------------------------------------------------------------------------

loc_8048855:                            ; CODE XREF: userpass+DB↑j
                cmp     [ebp+var_C], 3
                jg      short loc_8048872
                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_8048887
; ---------------------------------------------------------------------------

loc_8048872:                            ; 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_8048887:                            ; CODE XREF: userpass+F2↑j
                                        ; userpass+10F↑j
                add     [ebp+var_C], 1

loc_804888B:                            ; CODE XREF: userpass+D5↑j
                cmp     [ebp+var_C], 8
                jle     short loc_8048838
                mov     [ebp+var_C], 0
                jmp     short loc_80488D9
; ---------------------------------------------------------------------------

loc_804889A:                            ; 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_80488F3
                add     [ebp+var_C], 1

loc_80488D9:                            ; 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_804889A
                jmp     short loc_80488F4
; ---------------------------------------------------------------------------

loc_80488F3:                            ; CODE XREF: userpass+172↑j
                nop

loc_80488F4:                            ; 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_8048913
                call    success_3
                jmp     short loc_8048918
; ---------------------------------------------------------------------------

loc_8048913:                            ; CODE XREF: userpass+1A9↑j
                call    failed

loc_8048918:                            ; CODE XREF: userpass+C2↑j
                                        ; userpass+1B0↑j
                nop
                mov     ebx, [ebp+var_4]
                leave
                retn
; } // starts at 8048761
userpass        endp
```

Hàm này trông khá phức tạp dưới dạng hợp ngữ, chuyển sang mã giả ta được:

```
int userpass()
{
  size_t v0; // ebx
  size_t v2; // eax
  size_t v3; // edx
  char v4[9]; // [esp+Ah] [ebp-2Eh]
  char v5[10]; // [esp+13h] [ebp-25h] BYREF
  char s[10]; // [esp+1Dh] [ebp-1Bh] BYREF
  _BYTE v7[5]; // [esp+27h] [ebp-11h] BYREF
  int i; // [esp+2Ch] [ebp-Ch]

  qmemcpy(v7, "u2^#P", sizeof(v7));
  getchar();
  puts("Enter your username:");
  __isoc99_scanf("%[^\n]", s);
  getchar();
  puts("Enter your password:");
  __isoc99_scanf("%[^\n]", v5);
  printf("Your input username: %s and password: %s\n", s, v5);
  if ( strlen(s) != 9 )
    return failed();
  v0 = strlen(s);
  if ( v0 != strlen(v5) )
    return failed();
  for ( i = 0; i <= 8; ++i )
  {
    if ( i > 1 )
    {
      if ( i > 3 )
        v4[i] = v7[i - 4];
      else
        v4[i] = s[i + 5];
    }
    else
    {
      v4[i] = s[i + 2];

    }
  }
  for ( i = 0; ; ++i )
  {
    v2 = strlen(s);
    if ( v2 <= i || (s[i] + v4[i]) / 2 != v5[i] )
      break;
  }
  v3 = strlen(s);
  if ( v3 == i )
    return success_3();
  else
    return failed();
}
```

Lúc này, ta có thể thấy rõ cách hoạt động của hàm:

1. Chuẩn bị chuỗi ký tự "u2^#P", cho người dùng nhập vào username và password.

2. Lấy ký tự thứ 3, 4, 8, 9 của username và chuỗi ở trên lưu vào v4.

3. Duyệt từng chữ cái trong username và v4, lấy trung bình cộng rồi so với kết quả mong muốn ở password. Nếu trùng nhau thì in thông báo thành công.

Ở các bước (1), (3) đều có các kiểm tra lỗi: nếu username và password có độ dài khác 9 hay trung bình cộng mã ASCII ở bước 3 khác ký tự mong muốn trong password thì chuyển sang hàm failed() để in chuỗi thất bại.

Qua logic trên ta thấy, password cần cho bất kỳ username nào dài 9 ký tự là chuỗi với mỗi ký tự có giá trị: password[i] = (username[i] + v4[i]) / 2. Ta lấy logic đó, viết lại bằng C để tìm chuỗi password mong muốn:

```c
#include <stdio.h>

int
main(int argc, char *argv[])
{
	char a[10] = "346348815", b[5] = "u2^#P", c[10];

	for (int i = 0; i < 9; i++) {
		if (i > 1)
			if (i > 3)
				c[i] = b[i - 4];
			else
				c[i] = a[i + 5];
		 else
			c[i] = a[i + 2];
		c[i] = (c[i] + a[i]) / 2;
	}
	c[9] = 0;
	puts(c);
	return 0;
}
```

```
normal  pass.c                                  [c] utf-8 [unix] 9% 2:0
```

Chạy chương trình trên, ta được chuỗi password:

```
tch:~/build/tmp
$ ./a.out
4334T5K*B
tch:~/build/tmp
$
```

Thực thi chương trình với chuỗi password tìm được, chương trình cho kết quả đúng:

```
tch:~/build/tmp/sp
$ ./basic-reverse
Supported authentication methods:
1. Hard-coded password
2. A pair of 2 numbers
3. Username/password
Enter your choice: 3

Enter your username:
346348815
Enter your password:
4334T5K*B
Your input username: 346348815 and password: 4334T5K*B
Congrats! You found your own username/password pair :).
tch:~/build/tmp/sp
$
```