BÁO CÁO THỰC HÀNH

Môn học: Lập trình hệ thống (NT209)

Lab 3 – Kỹ thuật dịch ngược – Cơ bản

GVHD: Đỗ Thị Thu Hiền

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT209.N21.ANTN.1

STT	Họ và tên	MSSV	Email
1	Phạm Nguyễn Hải Anh	21520586	21520586@gm.uit.edu.vn
2	Nguyễn Nhật Quân	21522497	21522497@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

STT	Công việc	Kết quả tự đánh giá
1	Yêu cầu 2.1	
2	Yêu cầu 2.2	9.5
3	Yêu cầu 2.3	9.5

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

[1] Ghi nội dung công việc, yêu cầu trong bài Thực hành

BÁO CÁO CHI TIẾT

1. Yêu cầu 2.1

+ Phân tích bằng **mã assembly**

```
lea     eax, [ebp-3F0h]
push    eax
push    offset asc_804922A ; "%[^\n]"
call    ___isoc99_scanf
add     esp, 10h
sub     esp, 8
lea     eax, [ebp-3F0h]
push    eax
```

So `scanf("%[^\n]", s);` will read all characters until you reach `\n` (or `EOF`) and put them in `s`.
It is a common idiom to read a whole line in C.

Hàm scanf sẽ được nhận 2 tham số, vị trí để lưu giá trị và kiểu định dạng. Sau khi bấm enter để hoàn thành nhập chuỗi, chương trình sẽ lấy địa chỉ lưu chuỗi và push vào stack

```
sub     esp, 8
push    offset s2       ; "Be so good they can't ignore you"
lea     eax, [ebp-3F0h]
push    eax             ; s1
call    _strcmp
```

Sau đó hàm strcmp nhận 2 tham số lần lượt là địa chỉ chuỗi s1 và địa chỉ chuỗi s2, và thực hiện so sánh

```
test    eax, eax
jnz     short loc_80486FE
call    success_1
jmp     short loc_8048703
```

Kết quả trả về sẽ lưu vào thanh ghi eax. Nếu giống nhau sẽ eax = 0, sau đó test thực hiện phép AND, 0&0 = 0 => thanh ghi ZF = 1 => jnz không thực hiện nhảy => gọi hàm success. Hoặc thất bại nếu ngược lại.

```
.text:080486DF                  push    offset s2       ; "Don't stop until you're proud"
.text:080486E4                  lea     eax, [ebp+s1]
.text:080486EA                  push    eax             ; s1
.text:080486EB                  call    _strcmp
```

Lấy chuỗi từ [ebp+s1] lưu vào eax, rồi push eax. Trước đó: push chuỗi từ s2

```
.text:080486DC                  sub     esp, 8
.text:080486DF                  push    offset s2       ; "Don't stop until you're proud"
```

rồi gọi _strcmp để so sánh

```
.text:080486F5                  jnz     short loc_80486FE
.text:080486F7                  call    success_1
```

Nếu s1 = s2, nhảy tới success_1

+ Input tìm được: Be so good they can't ignore you

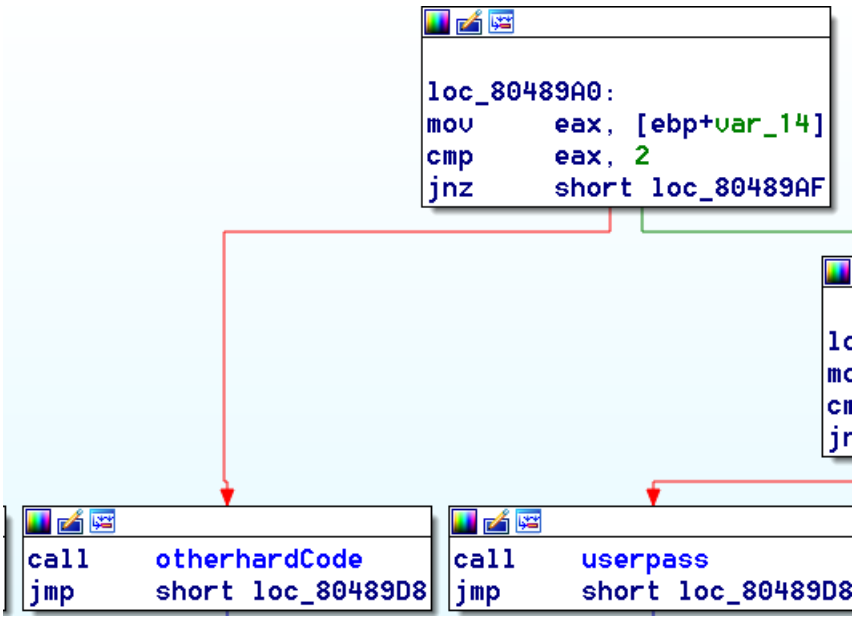
+ Thực thi file:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ ./basic-reverse
Supported authentication methods:
1. Hard-coded password
2. Another hard-coded password
3. Username/password
Enter your choice: 1
Enter the hard-coded password (option 1):
Be so good they can't ignore you
Your input hard-coded password: Be so good they can't ignore you
Congrats! You found the hard-coded secret, good job :).
Hand in this to your instructor as a proof:
"Learning never exhausts the mind."
```

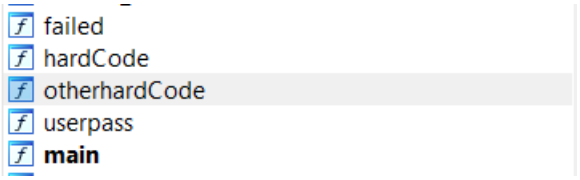
2. Yêu cầu 2.2

+ Phân tích bằng **mã assembly**

Tại hàm main, ta để ý ở loc_80489A0: nếu input là 2 thì sẽ nhảy tới otherhardCode



Ta xem mã assembly ở phần otherhandCode



Ta thấy bài này cũng nhập input và gọi hàm strcmp tương tự với bài 1:

```
push    offset aEnterTheHard_0 ; "Enter the hard-coded password (option 2"...
call    _puts
add     esp, 10h
sub     esp, 8
lea     eax, [ebp+s1]
push    eax
push    offset asc_804922A ; "%[^\n]"
call    ___isoc99_scanf
add     esp, 10h
sub     esp, 8
lea     eax, [ebp+s1]
push    eax
push    offset format   ; "Your input hard-coded password: %s\n"
call    _printf
add     esp, 10h
mov     [ebp+var_C], 2
mov     eax, [ebp+var_C]
mov     eax, WHAT_THAT[eax*4]
mov     [ebp+s2], eax
sub     esp, 8
push    [ebp+s2]        ; s2
```

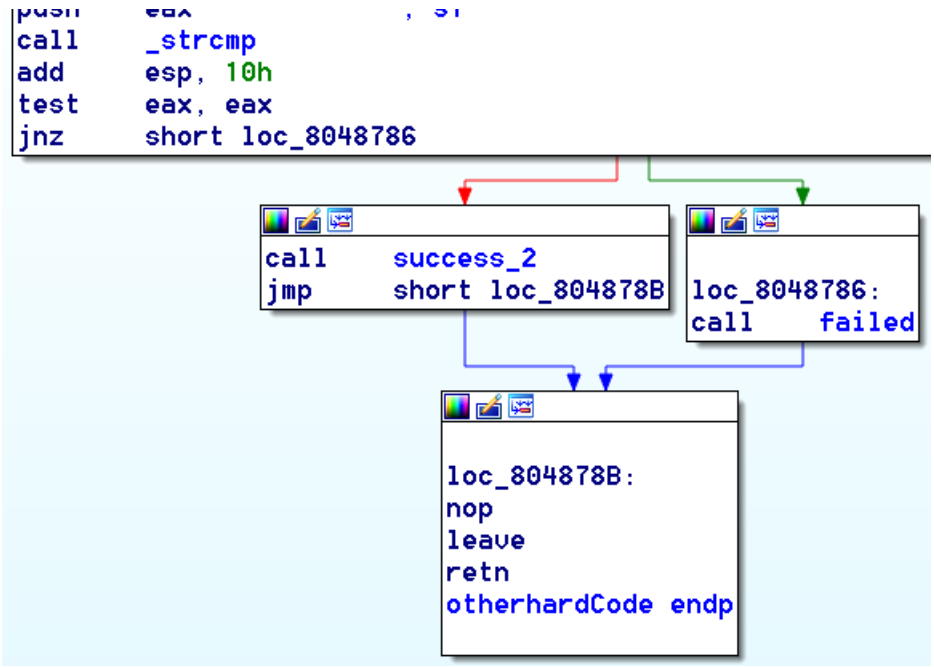
Hàm isoc99_scanf được gọi với 2 tham số từ 2 lệnh push trước đó là push eax và push offset asc_804922A. offset này là tương tự với truyền option "%[^\n]" cho việc nhập chuỗi, tham số từ eax sẽ là nơi lưu input nhập từ bàn phím. Input nhập từ bàn phím sẽ lưu vào [ebp+s1], là input thứ nhất của strcmp nhờ vào lệnh lea eax, [ebp+s1] và push eax:

```
add     esp, 10h
mov     [ebp+var_C], 2
mov     eax, [ebp+var_C]
mov     eax, WHAT_THAT[eax*4]
mov     [ebp+s2], eax
sub     esp, 8
push    [ebp+s2]        ; s2
lea     eax, [ebp+s1]
push    eax             ; s1
call    _strcmp
```

input thứ 2 là giá trị của ebp + s2. Vậy giá trị của ebp+s2 là password cần tìm. Trước đó, ebp+s2 được truyền giá trị từ eax, eax lúc đó có giá trị là WHAT_THAT[eax*4] = WHAT_THAT [value(ebp+var_C) * 4] = WHAT_THAT [2*4]. Theo gợi ý, WHAT_THAT là 1 mảng các chuỗi cách nhau 4 byte. Vậy chuỗi cần tìm là WHAT_THAT[2].

```
.data:0804B060                 public WHAT_THAT
.data:0804B060 WHAT_THAT       dd offset aYouScratchMyBa ; DATA XREF: otherhardCode+56↑r
.data:0804B060                                           ; "You scratch my back and I'll scratch yo"...
.data:0804B064                 dd offset aNewOneInOldOne ; "New one in, old one out"
.data:0804B068                 dd offset aItTooLateToLoc ; "It' too late to lock the stable when th"...
.data:0804B06C                 dd offset aWithAgeComesWi ; "With age comes wisdom"
.data:0804B070                 dd offset aNothingIsMoreP ; "Nothing is more precious than independe"
```

2 chuỗi so sánh với nhau bằng hàm _strcmp.



Nếu giống nhau, strcmp sẽ trả về eax = 1; khi đó test sẽ gán ZF = AND (1, 1) = 1, dẫn tới việc lệnh jnz không thực thi và chương trình sẽ thẳng tới lệnh call success_2

+ Input cần tìm: It' too late to lock the stable when the horse is stolen.

+ Thực thi file:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ ./basic-reverse
Supported authentication methods:
1. Hard-coded password
2. Another hard-coded password
3. Username/password
Enter your choice: 2
Enter the hard-coded password (option 2):
It' too late to lock the stable when the horse is stolen
Your input hard-coded password: It' too late to lock the stable when the hors
e is stolen
Good job! You defeated a harder level of finding hard-coded secret :).
Hand in this to your instructor as a proof:
"Don't let what you cannot do interfere what you can do."
```

3. Yêu cầu 2.3

+ Phân tích mã giả:

Ta thấy v7, v8, v9, v10, v11 là các ô nhớ liên tiếp nhau

```
 7  char v4[9]; // [sp+Ah] [bp-2Eh]@6
 8  char v5[10]; // [sp+13h] [bp-25h]@1
 9  char s[10]; // [sp+1Dh] [bp-1Bh]@1
10  char v7; // [sp+27h] [bp-11h]@1
11  char v8; // [sp+28h] [bp-10h]@1
12  char v9; // [sp+29h] [bp-Fh]@1
13  char v10; // [sp+2Ah] [bp-Eh]@1
14  char v11; // [sp+2Bh] [bp-Dh]@1
15  unsigned int i; // [sp+2Ch] [bp-Ch]@4
16
17  v7 = 93;
18  v8 = 33;
19  v9 = 126;
20  v10 = 104;
21  v11 = 53;
```

Từ đó v7 là 1 mảng 5 phần tử với các phần từ có giá trị từ dòng 17 đến 21 trong hình trên.

username, password nhập từ bàn phím sẽ lần lượt được lưu vào s, v5.

```
23  puts("Enter your username:");
24  __isoc99_scanf("%[^\n]", s);
25  getchar();
26  puts("Enter your password:");
27  __isoc99_scanf("%[^\n]", v5);
```

Phân tích hàm if:

```
29  if ( strlen(s) == 9 && (v0 = strlen(s), v0 == strlen(v5)) )
30  {
31    for ( i = 0; (signed int)i <= 8; ++i )
32    {
33      if ( (signed int)i > 1 )
34      {
35        if ( (signed int)i > 3 )
36          v4[i] = *(&v7 + i - 4);
37        else
38          v4[i] = s[i + 5];
39      }
40      else
41      {
42        v4[i] = s[i + 2];
43      }
44    }
45    for ( i = 0; ; ++i )
46    {
47      v2 = strlen(s);
48      if ( v2 <= i || (s[i] + v4[i]) / 2 != v5[i] )
49        break;
50    }
51    v3 = strlen(s);
52    if ( v3 == i )
53      result = success_3();
54    else
55      result = failed();
56  }
57  else
58  {
59    result = failed();
60  }
```

Từ điều kiện dòng 29, nếu s và v5 có chiều dài khác 9 thì sẽ trả về failed(). Nếu bằng 9:

Đầu tiên sẽ đi vào vòng for như hình dưới:

```
31    for ( i = 0; (signed int)i <= 8; ++i )
32    {
33      if ( (signed int)i > 1 )
34      {
35        if ( (signed int)i > 3 )
36          v4[i] = *(&v7 + i - 4);
37        else
38          v4[i] = s[i + 5];
39      }
40      else
41      {
42        v4[i] = s[i + 2];
43      }
```

Chương trình đang gán giá trị của 9 phần tử mảng v4. Ta biết v4 là 1 chuỗi 9 kí tự

```
2 {
3   size_t v0; // ebx@2
4   int result; // eax@3
5   size_t v2; // eax@15
6   size_t v3; // edx@16
7   char v4[9]; // [sp+Ah] [bp-2Eh]@6
8   char v5[10]; // [sp+13h] [bp-25h]@1
```

logic gán giá trị cho v4:

v4[0] = s[2] và v4[1] = s[3]

v4[2] = s[7] và v4[3] = s[8]

từ phần tử thứ 4 trở đi:

v4[4] = v7[0] = 93. Nhưng vì v4 là chuỗi kí tự nên v7[0] sẽ được chuyển về kí tự tương ứng trong ascii là ‘]’ (không tính dấu nháy đơn).

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

v4[5] = char (33)

v4[6] = char (126)

v4[7] = char (104)

v4[8] = char (53)

Chuỗi v4 và v7 là chuỗi phụ, dùng để tính ra password tương ứng, vì còn phải qua 1 vòng for và 1 lệnh if:

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

Tại lệnh for trên, v2 là chiều dài của username. Từng kí tự của v5 (password từ bàn phím) được so sánh với s[i] + v4[i]. Từ đây ta hiểu 2 điều: username phải có 9 kí tự, nếu không sẽ dẫn tới if (v3==i) sai và thực thi dòng result=failed(); phần tử thứ k của password sẽ được tính từ tổng của v4[k] và s[k] chia 2, tất nhiên chúng được tính dưới dạng integer rồi chuyển thành kí tự ascii tương ứng. Chẳng hạn, nếu s[4] = 3 và v4[3] = 2, thì (s[4] + v4[3]) / 2 = 5/2 = 2.5, v5[4] sẽ lấy phần nguyên là 2 rồi chuyển qua char(2).

Một chương trình C được viết dựa trên mã giả của bài 3, để đưa ra giá trị của các biến trong mã giả. Chạy chương trình để tìm ra v5.

```
Enter your username:
0586-2497
Enter your password:
4586E)YP6
Your input username: 0586-2497 and password: 4586E)YP6
v7: ]!~h5
i: 0
i: 1
i: 2
i: 3
i: 4
i: 5
i: 6
i: 7
i: 8
Chuoi~ v4: 8697]!~h54586E)YP6
4586E)YP6ki tu db s, v, /2: · 4 ·
v3: 9
Ket qua la`: 1
```

v5 = 4586E)YP6

Chương trình trả về 1 nếu password nhập từ bàn phím là password được sinh ra từ username.

Password cần tìm: 4586E)YP6

Thực thi file:

```
└─$ ./basic-reverse
Supported authentication methods:
1. Hard-coded password
2. Another hard-coded password
3. Username/password
Enter your choice: 3
Enter your username:
0586-2497
Enter your password:
4586E)YP6
Your input username: 0586-2497 and password: 4586E)YP6
Awesome! You found your own username/password pair. Nice work.
Hand in this to your instructor as a proof:
"It always seems impossible until it's done."
```

YÊU CẦU CHUNG

Báo cáo:

- File **.PDF**.
- Đặt tên theo định dạng: **[Mã lớp]-Lab3_NhomX_MSSV1-MSSV2.pdf** (trong đó X là số thứ tự nhóm, MSSV gồm đầy đủ MSSV của tất cả các thành viên thực hiện bài thực hành).

  Ví dụ: [NT209.N21.ANTN.1]-Lab3_Nhom2_21520001-21520013.pdf.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá:

- Hoàn thành tốt yêu cầu được giao.
- Có nội dung mở rộng, ứng dụng.

Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.

HẾT