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

BÁO CÁO THỰC HÀNH LAB 3

Môn học: Lập trình hệ thống

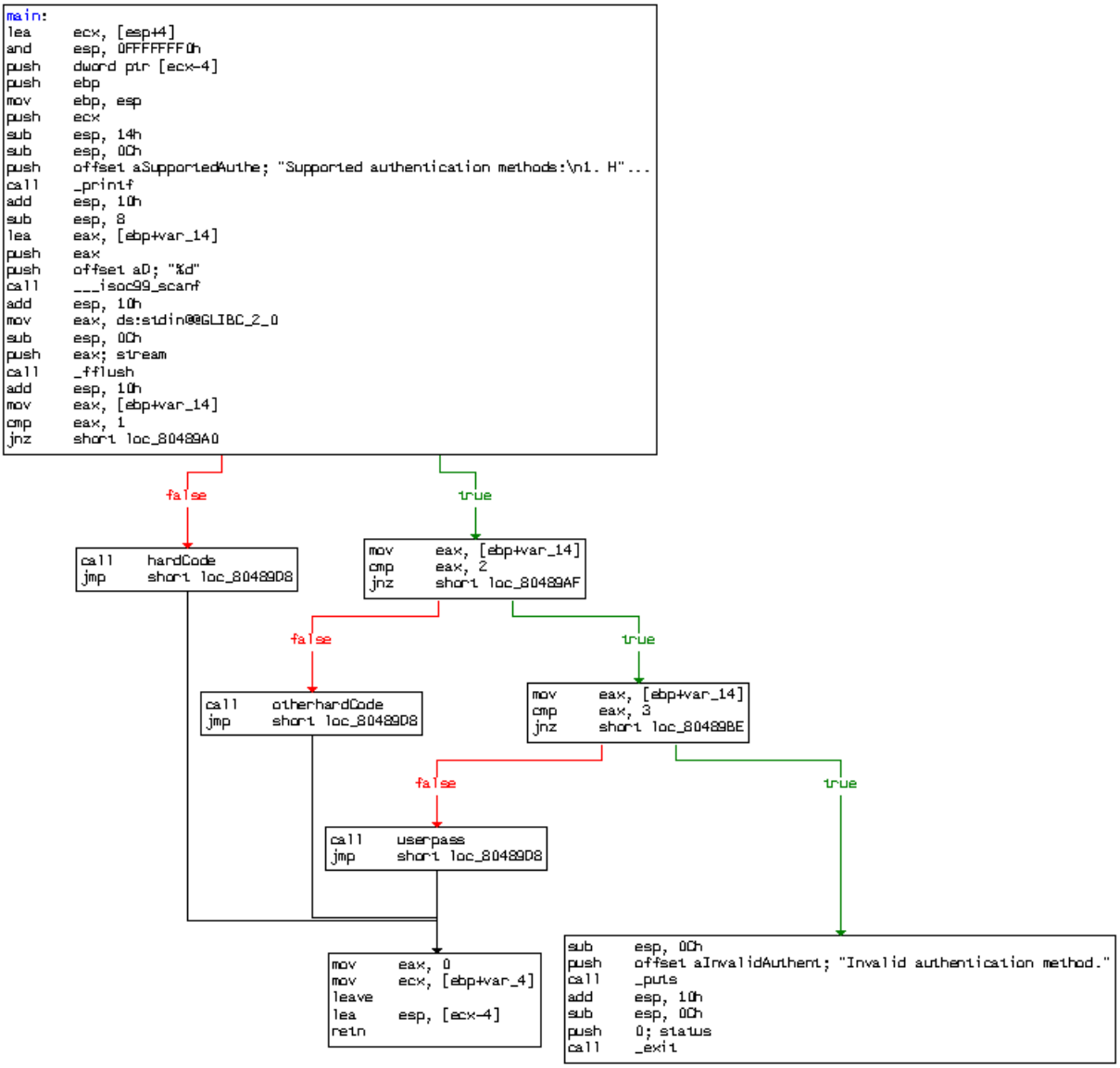
Nhóm: 2

THÀNH VIÊN THỰC HIỆN:

STT	Họ và tên	MSSV
1	Nguyễn Văn Tài	19520250
2	Trần Hoàng Khang	19521671

BÁO CÁO CHI TIẾT

Dùng IDA cho phiên bản x86 (đây là file ELF 32-bit). Ta xem *flowchart* chính của chương trình .



Chương trình sử dụng 3 hàm chính (3 options) tương ứng với 3 challenges cần solve:

- **hardCode()**
- **otherhardCode()**
- **userpass()**

Ví dụ:

```
┌──(virus㉿virus)-[~/Desktop]
└─$ ./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):
hello
Your input hard-coded password: hello
Nice try but that is not the answer.
```

Yêu cầu 2.1. Phân tích và tìm **`passphrase cố định` (option 1)** của **basic-reverse** với phương pháp chứng thực 1. Báo cáo phương pháp và input tìm được.

Xem xét hàm **hardCode().** Vì bài này easy nên mình nên phân tích dưới *mã assembly*:

```
; Attributes: bp-based frame

public hardCode
hardCode proc near

s1= byte ptr -3F0h

; __unwind {
push    ebp
mov     ebp, esp
sub     esp, 3F8h
call    _getchar
sub     esp, 0Ch
push    offset aEnterTheHardCo ; "Enter the hard-coded password (option 1"...
call    _puts
add     esp, 10h
sub     esp, 8
lea     eax, [ebp+s1]
push    eax
push    offset asc_804923E ; "%[^\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
sub     esp, 8
push    offset s2       ; "The best way to predict the future is t"...
lea     eax, [ebp+s1]
push    eax             ; s1
call    _strcmp
add     esp, 10h
test    eax, eax
jnz     short loc_80486FE
```

Hàm thực hiện tạo một biến s1 và được cấp phát. Ban đầu chương trình gọi hàm **getchar()** có lẽ để lấy ký tự xuống dòng "\n" trước đó. Sau đó dùng hàm thư viện C và chuẩn bị một số tham số cho hàm _puts và in dòng chữ "*Enter the hard-coded password (option 1):*" được lưu trữ trong biến **aEnterTheHardCo**

Tiếp tục, dùng **___isoc99_scanf** để cho người dùng nhập input. Dùng hàm **_printf** để in ra thông báo "*Your input hard-coded password: %s*" được lưu trong biến format và thay thế chuỗi "%s" với password chúng ta nhập tại s1.

Cuối cùng đến đoạn chương trình chính:

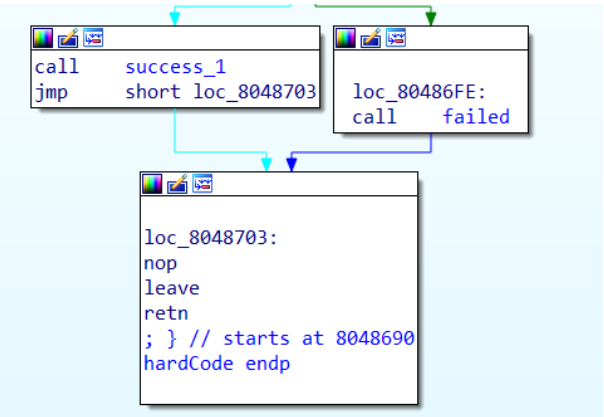
```
add     esp, 10h
sub     esp, 8
push    offset s2       ; "The best way to predict the future is t"...
lea     eax, [ebp+s1]
push    eax             ; s1
call    _strcmp
```

```
add     esp, 10h
test    eax, eax
jnz     short loc_80486FE
```

Để ý kỹ 2 dòng push các tham số vào s1 (password ta nhập), s2 (chuỗi cần so sánh). Chúng ta có thể thấy chuỗi được lưu tại biến s2 -> *The best way to predict the future is to create it*

```
.rodata:08049268 s2              db 'The best way to predict the future is to create it',0
.rodata:08049268                                        ; DATA XREF: hardCode+4F↑o
```

Sau đó kết quả trả về ở **eax**, nếu **eax = 0** thì 2 chuỗi bằng nhau thì thực hiện tiếp đến hàm success_1 và in ra thông điệp báo thành công, còn nếu không bằng thì nhảy đến **short loc_80486FE** và thông báo không phê duyệt



Vậy password là : **The best way to predict the future is to create it**

Thực nghiệm lại chương trình:

```
┌──(virus㉿virus)-[~/Desktop]
└─$ ./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):
The best way to predict the future is to create it
Your input hard-coded password: The best way to predict the fu
ture is to create it
Congrats! You found the hard-coded secret, good job :).
Hand in this to your instructor as a proof:
"Stay home for the safety of yourself and others."
```

Done 😊 !!!

>> Cheating way (dùng mã giả để phân tích cho nhanh :>)

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    int v4[4]; // [esp+4h] [ebp-14h] BYREF
4
5    printf(
6      "Supported authentication methods:\n"
7      "1. Hard-coded password\n"
8      "2. Another hard-coded password\n"
9      "3. Username/password\n"
10     "Enter your choice: ");
11   __isoc99_scanf("%d", v4);
12   fflush(stdin);
13   if ( v4[0] == 1 )
14   {
15     hardCode();
16   }
17   else if ( v4[0] == 2 )
18   {
19     otherhardCode();
20   }
21   else
22   {
23     if ( v4[0] != 3 )
24     {
25       puts("Invalid authentication method.");
26       exit(0);
27     }
28     userpass();
29   }
30   return 0;
31 }
```

Yêu cầu 2.2. Phân tích và tìm **`passphrase cố định` (option 2)** của **basic-reverse** với phương pháp chứng thực 2. Báo cáo phương pháp và input tìm được.

Xem xét hàm **otherhardCode(),** bài này dễ nên ta cũng nên đọc dưới mã assembly

```
public otherhardCode
otherhardCode proc near

s1= byte ptr -3F8h
s2= dword ptr -10h
var_C= dword ptr -0Ch

; __unwind {
push    ebp
mov     ebp, esp
sub     esp, 3F8h
call    _getchar
sub     esp, 0Ch
push    offset aEnterTheHardCo_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_804923E ; "%[^\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], 13h
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
add     esp, 10h
test    eax, eax
jnz     short loc_8048786
```

Chương trình thực thi tương tự lúc ban đầu, dùng hàm **_getchar()** (loại bỏ khoảng trắng) , **_puts()** (in ra thông điệp), **__isoc99_scanf**(cho người dùng nhập pass), **_printf()**(in ra password vừa nhập).

Giờ vào "món chính"

```
add     esp, 10h
mov     [ebp+var_C], 13h
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
```

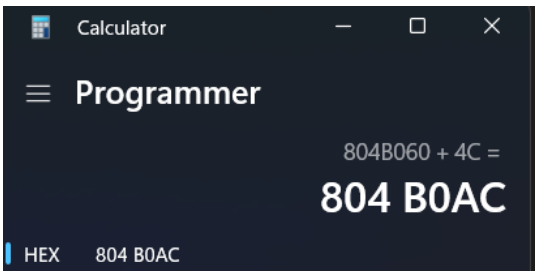
Chương trình sẽ đưa giá trị *13 (hex)* vào ô nhớ có địa chỉ *ebp + var_C* , đưa tiếp giá trị này vào **eax**.

Đưa giá trị tại địa chỉ của *WHAT_THAT + eax *4* vào **eax**, đưa giá trị này vào s2. Sau đó so sánh chuỗi user nhập vào (s1) và s2. Vậy chúng ta cần tìm ra s2 lưu giá trị gì ?

Quay lại đoạn đưa giá trị tại *WHAT_THAT + eax *4* vào s2 (một cách trung gian, thông qua **eax**), ta vào xem địa chỉ tại **WHAT_THAT**

```
.data:0804B060 WHAT_THAT       dd offset aYouScratchMyBa
.data:0804B060                                         ; 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"...
.data:0804B074                 dd offset aHandsomeIsAsHa ; "Handsome is as handsome does"
.data:0804B078                 dd offset aNeverOfferToTe ; "Never offer to teach fish to swim"
.data:0804B07C                 dd offset aToTryToRunBefo ; "To try to run before the one can walk"
.data:0804B080                 dd offset aNobodyHasEverS ; "Nobody has ever shed tears without seei"...
.data:0804B084                 dd offset aYouGetWhatYouP ; "You get what you pay for"
.data:0804B088                 dd offset aAsStrongAsAHor ; "As strong as a horse"
.data:0804B08C                 dd offset aAllRoadsLeadTo ; "All roads lead to Rome"
.data:0804B090                 dd offset aGoodWineNeedsN ; "Good wine needs no bush"
.data:0804B094                 dd offset aDiamondCutsDia ; "Diamond cuts diamond"
.data:0804B098                 dd offset aSpareTheRodAnd ; "Spare the rod and spoil the child"
.data:0804B09C                 dd offset aSpeakOneWayAnd ; "Speak one way and act another"
.data:0804B0A0                 dd offset aDonTJudgeABook ; "Don't judge a book by its cover"
.data:0804B0A4                 dd offset aItSNoUseBeatin ; "It's no use beating around the bush"
.data:0804B0A8                 dd offset aManProposesGod ; "Man proposes God deposes"
.data:0804B0AC                 dd offset aOutOfSightOutO ; "Out of sight out of mind"
```

Ta thấy ở đây **WHAT_THAT** giống như một mảng, ban đầu ta có đưa giá trị *13(hex)* vào **eax**, lấy *eax * 4 = 4C* và lấy giá trị này cộng với địa chỉ gốc của **WHAT_THAT**



Địa chỉ chứa giá trị cần tìm là **0804B0AC**. Xem thử ở đây có gì thú vị nào

```
.data:0804B090                 dd offset aGoodWineNeedsN ; "Good wine needs no bush"
.data:0804B094                 dd offset aDiamondCutsDia ; "Diamond cuts diamond"
.data:0804B098                 dd offset aSpareTheRodAnd ; "Spare the rod and spoil the child"
.data:0804B09C                 dd offset aSpeakOneWayAnd ; "Speak one way and act another"
.data:0804B0A0                 dd offset aDonTJudgeABook ; "Don't judge a book by its cover"
.data:0804B0A4                 dd offset aItSNoUseBeatin ; "It's no use beating around the bush"
.data:0804B0A8                 dd offset aManProposesGod ; "Man proposes God deposes"
.data:0804B0AC                 dd offset aOutOfSightOutO ; "Out of sight out of mind"
.data:0804B0B0                 dd offset aEastOrWestHome ; "East or West home is best"
.data:0804B0B4                 dd offset aSoManyMenSoMan ; "So many men, so many minds"
```

Ố ồ ô, vậy đây là chuỗi mà chương trình sẽ so sánh với password ta nhập. Đoán xem mật khẩu này có "working" không. Chạy thực nghiệm:

```
┌──(virus㉿virus)-[~/Desktop]
└─$ ./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):
Out of sight out of mind
Your input hard-coded password: Out of sight out of mind
Congrats! You defeated a harder level of finding hard-coded se
cret :).
Hand in this to your instructor as a proof:
"Stay positive during the COVID-19 pandemic."
```

Okay, cứ tự tin - ảo giác - quyết thắng

Password là : **Out of sight out of mind**

>> Cheating way (dùng mã giả để phân tích cho nhanh :>)

```
1  int otherhardCode()
2  {
3    int result; // eax
4    char s1[1000]; // [esp+0h] [ebp-3F8h] BYREF
5    char *s2; // [esp+3E8h] [ebp-10h]
6    int v3; // [esp+3ECh] [ebp-Ch]
7
8    getchar();
9    puts("Enter the hard-coded password (option 2):");
10   __isoc99_scanf("%[^\n]", s1);
11   printf("Your input hard-coded password: %s\n", s1);
12   v3 = 19;
13   s2 = *(&WHAT_THAT + 19);
14   if ( !strcmp(s1, s2) )
15     result = success_2();
16   else
17     result = failed();
18   return result;
19 }
```

Yêu cầu 2.3. Phân tích, tìm ***username/password*** phù hợp của **basic-reverse** với phương pháp chứng thực 3. Báo cáo phương pháp và input tìm được.
Lưu ý bắt buộc: ***username*** được tạo từ MSSV của **2 sinh viên**, lấy 4 số cuối nối nhau bằng dấu "-". Ví dụ 20520123 và 20521021 sẽ có username là **0123-1021**. Nhóm có **1 sinh viên** có MSSV là 2052xxxx thì username là **2052-xxxx**

Vào hàm **userpass(),** vì hàm này hơi "khoai" nên ta quyết định xem mã giả (pseudo-code) cho lực:

Đoạn đầu là khai báo các biến. Tiếp theo cấp vùng nhớ và giá trị "*&5p~D*" cho biến v7 và thực hiện tương tự công đoạn lấy dữ liệu đầu vào và hiển thị ra với 2 trường username và password: *getchar() -> __isoc99_scanf -> getchar() -> puts() -> __isoc99_scanf -> printf.* Xong rồi check thử xem độ dài password đầu vào của username : **len(username)== 9?** và xem **len(password) == 9?**

```
12
13  qmemcpy(v7, "&5p~D", sizeof(v7));
14  getchar();
15  puts("Enter your username:");
16  __isoc99_scanf("%[^\n]", s);
17  getchar();
18  puts("Enter your password:");
19  __isoc99_scanf("%[^\n]", v5);
20  printf("Your input username: %s and password: %s\n", s, v5);
21  if ( strlen(s) != 9 )
22    return failed();
23  v0 = strlen(s);
24  if ( v0 != strlen(v5) )
25    return failed();
```

Sau đó chương trình tạo một biến v4 mới được hình thành như sau:

```
26  for ( i = 0; i <= 8; ++i )
27  {
28    if ( i > 1 )
29    {
30      if ( i > 3 )
31        v4[i] = v7[i - 4];
32      else
33        v4[i] = s[i + 5];
34    }
35    else
36    {
37      v4[i] = s[i + 2];
38    }
39  }
```

Cho *s = "0250-1671"* (MSSV1: 1952250 && MSSV2: 19521671). Chạy **i -> [0; 8]** (9 lần):

2 ký tự đầu của v4 là ký tự tại **vị trí 2,3 của s**

2 ký tự tiếp của v4 là ký tự tại **vị trí 7,8 của s**

4 ký tự còn lại của v4 là chuỗi **v7 "&5p~D"**

Kết quả : **v4 = "5071&5p~D"**

Tiếp đến là phần hình thành mật khẩu để đối chiếu:

```
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();
return result;
```

Biến *v2=strlen(s)* (8) , dùng để làm giới hạn dừng vòng lặp, nếu thỏa một trong những điều kiện tại

```
If (v2 <= I || (s[i] + v4[i] / 2 != v5[i] )
```

thì vòng lặp dừng. Sau đó xét *v3 == i* hay không, nếu có thì **success_3()** được gọi và thành công.

Vậy vấn đề là làm thế nào để cho **i = v3 = 8**. Tức là cho i chạy hết vòng lặp cho đến khi ***v2 <= i*** (vì i++ tăng dần). Để làm được điều này ta phải luôn để cho điều kiện ***s[i] + v4[i] == v5[i]*** thì vòng lặp sẽ không bị break.

Phép tính cộng trên từng char (ký tự) được chuyển sang decimal rồi tính tổng như bình thường, sau đó chia cho 2 và được convert về lại char để so sánh. Ta có thể dùng chương trình python để mô phỏng và lấy password.

```
# studentID1 = "19520250"
# studentID2 = "19521671"

# v5 = str(user_input)

s = "0250-1671" # username (studentID1 + "-" + studentID2)

v7 = "&5p~D"

v4 = "5071&5p~D" # s[2:4] + s[7:9] + v7

v2 = 8 # len(studentID1)

# (s[i] + v4[i]) / 2 != v5[i] ?

username = s # s
key = v4 # v4

password = ""

for i in range(9):
    password += chr((ord(username[i]) + ord(key[i])) // 2)

print(password)
```

Chạy để tìm kết quả ouput (chính là password cần tìm)

```
09:19:51 ML_DL_Self_learning
→python .\solve.py
2160)3SZ:
```

Chạy thực nghiệm:

```
┌──(virus㉿virus)-[~/Desktop]
└─$ ./basic-reverse
Supported authentication methods:
1. Hard-coded password
2. Another hard-coded password
3. Username/password
Enter your choice: 3
Enter your username:
0250-1671
Enter your password:
2160)3SZ:
Your input username: 0250-1671 and password: 2160)3SZ:
Congrats! You found your own username/password pair. Nice work to rec
eive the my message.
Hand in this to your instructor as a proof:
"Vietnam can win over SARS-CoV-2."
```

Password là: **2160)3SZ:**