Lab
4

BÁO CÁO BÀI THỰC HÀNH SỐ 4

BASIC REVERSE ENGINEERING

Môn học: Lập trình Hệ thống (NT209)

Nhóm	01
Sinh viên thực hiện	22521303 – Nguyễn Đức Tấn 22521501 – Phan Nguyễn Nhật Trâm 22521541 – Thái Ngọc Diễm Trinh
Thời gian thực hiện	24/04/2024

BÀI 1:

> **Yêu cầu 1.** Phân tích và tìm ***passphrase cố định*** **(option 1)** của **basic-reverse** với phương pháp chứng thực 1. Báo cáo phương pháp phân tích, input tìm được và hình ảnh minh chứng chạy file.

Ta thấy ở hàm main, khi người dùng nhập số 1 (tương ứng với phương thức đăng nhập bằng hard code), chương trình sẽ dẫn đến hàm hardCode.

```
.text:08048967                  mov     eax, [ebp+var_14]
.text:0804896A                  cmp     eax, 1
.text:0804896D                  jnz     short loc_8048976
.text:0804896F                  call    hardCode
.text:08048974                  jmp     short loc_80489AE
```

Xem xét hàm hardCode:

```
hardCode        proc near               ; CODE XREF: main+4E↓p

s1              = byte ptr -3F0h

                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_804916A ; "%[^\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       ; "Handsome is as handsome does"
                lea     eax, [ebp+s1]
                push    eax             ; s1
                call    _strcmp
                add     esp, 10h
                test    eax, eax
                jnz     short loc_80486C9
                call    success_1
                jmp     short loc_80486CE
```

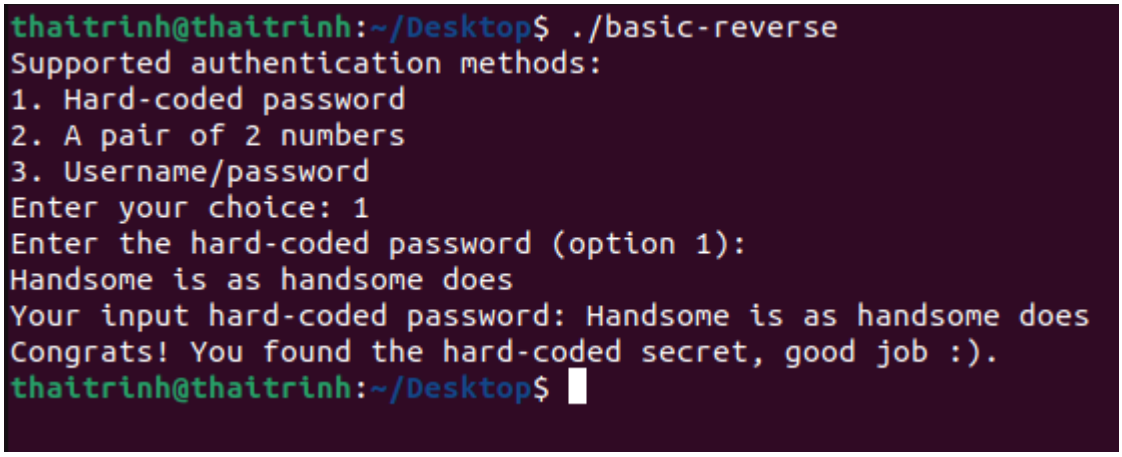
Đầu tiên có thông báo yêu cầu nhập hard-coded password.

Ta thấy, hàm **_strcmp** được truyền 2 tham số s1 (được lưu trong eax, là chuỗi mà người dùng nhập vào) và offset s2 (chuỗi "Handsome is as handsome does"). Kết quả trả về giá trị 0 (nếu 2 chuỗi giống nhau) hoặc 1 (nếu 2 chuỗi khác nhau) lưu trong eax.

Kiểm tra giá trị eax, nếu bằng 0 thì gọi **hàm success_1** (thành công). Nghĩa là, nếu chuỗi người dùng nhập bằng với chuỗi Handsome is as handsome does" thì chương trình thành công.

```
.text:080485F7
.text:080485F7                public success_1
.text:080485F7 success_1      proc near               ; CODE XREF: hardCode+67↓p
.text:080485F7                push    ebp
.text:080485F8                mov     ebp, esp
.text:080485FA                sub     esp, 8
.text:080485FD                sub     esp, 0Ch
.text:08048600                push    offset s        ; "Congrats! You found the hard-coded secr"...
.text:08048605                call    _puts
.text:0804860A                add     esp, 10h
.text:0804860D                nop
.text:0804860E                leave
.text:0804860F                retn
.text:0804860F success_1      endp
```

Kết quả khi chạy chương trình:



Vậy đáp án chính xác là:

```
1 - 1 - Handsome is as handsome does
```

BÀI 2:

> **Yêu cầu 2.** Phân tích và tìm ***cặp số nguyên*** **(option 2)** của **basic-reverse** với phương pháp chứng thực 2. Báo cáo phương pháp phân tích, input tìm được và hình ảnh minh chứng chạy file.

Ta thấy ở hàm main, khi người dùng nhập số 2 (tương ứng với phương thức thứ 2), chương trình sẽ dẫn đến hàm otherhardCode.

```
loc_8048976:                            ; CODE XREF: main+4C↑j
                mov     eax, [ebp+var_14]
                cmp     eax, 2
                jnz     short loc_8048985
                call    otherhardCode
                jmp     short loc_80489AE
```

Xét hàm otherhardCorde, đầu tiên sẽ có thông báo yêu cầu người dùng nhập và in ra con số vừa nhập:

```
.text:080486D1              push    ebp
.text:080486D2              mov     ebp, esp
.text:080486D4              sub     esp, 18h
.text:080486D7              call    _getchar
.text:080486DC              sub     esp, 0Ch
.text:080486DF              push    offset aEnterYour2Numb ; "Enter your 2 numbers (separated by spac"...
.text:080486E4              call    _puts
.text:080486E9              add     esp, 10h
.text:080486EC              sub     esp, 4
.text:080486EF              lea     eax, [ebp+var_14]
.text:080486F2              push    eax
.text:080486F3              lea     eax, [ebp+var_10]
.text:080486F6              push    eax
.text:080486F7              push    offset aDD      ; "%d %d"
.text:080486FC              call    ___isoc99_scanf
.text:08048701              add     esp, 10h
.text:08048704              mov     edx, [ebp+var_14]
.text:08048707              mov     eax, [ebp+var_10]
.text:0804870A              sub     esp, 4
.text:0804870D              push    edx
.text:0804870E              push    eax
.text:0804870F              push    offset aYourInputDD ; "Your input: %d %d\n"
.text:08048714              call    _printf
```

```
.text:08048719              add     esp, 10h
.text:0804871C              mov     [ebp+var_C], 4
.text:08048723              mov     eax, [ebp+var_10]
.text:08048726              cmp     eax, [ebp+var_C]
.text:08048729              jnz     short loc_804875C
.text:0804872B              mov     edx, [ebp+var_10]
.text:0804872E              mov     eax, [ebp+var_10]
.text:08048731              mov     eax, ds:funny_seq[eax*4]
.text:08048738              sub     esp, 8
.text:0804873B              push    edx
.text:0804873C              push    eax
.text:0804873D              call    funny_func
```

Ta có 2 số đầu vào sẽ được lưu lần lượt ở [ebp+var_10] và [ebp+var_14].

Sau đó, chương trình thực hiện so sánh số thứ nhất với 4 (lưu ở [ebp+var_C]). Nếu 2 giá trị không bằng nhau thì sẽ không thực hiện gì, nếu bằng nhau sẽ tiếp tục thực hiện hàm funny_func. **Vậy ta có thể cho rằng giá trị của số thứ nhất có thể là số 4.**

Hàm funny_func sẽ truyền 2 tham số vào là giá trị của funny_seq[eax*4] và giá trị của số thứ nhất.

Giá trị của funny_seq[eax*4]: ta có eax trước đó đang lưu giá trị của số thứ nhất (là 4). Khi đó giá trị cần tìm của funny_seq[eax*4] sẽ giá trị thứ 4 trong mảng funny_seq, hay cụ thể là số 0.

```
.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
.rodata:08048A7A                 db   0
.rodata:08048A7B                 db   0
.rodata:08048A7C                 db   5
.rodata:08048A7D                 db   0
.rodata:08048A7E                 db   0
.rodata:08048A7F                 db   0
.rodata:08048A80                 db   7
.rodata:08048A81                 db   0
.rodata:08048A82                 db   0
.rodata:08048A83                 db   0
.rodata:08048A84                 db   9
.rodata:08048A85                 db   0
```

Xét hàm **funny_func**: hàm thực hiện tính toán. Với 2 tham số truyền vào lưu ở địa chỉ [ebp+arg_0] và [ebp+arg_4], gọi tắt là a1 và a2. Đầu tiên sẽ thực hiện cộng a1+a2 và trừ đi 1: a1+a2-1, lưu giá trị này vào ecx. Tiếp tục cộng a1+a2, lưu giá trị trong ecx. Thực hiện nhân ecx và eax, lưu vào eax, hay (a1 + a2 - 1) * (a1 + a2). Kết quả lưu vào [ebp+var_4] và return kết quả.

Tham số truyền vào là giá trị của funny_seq[16] và số thứ nhất, hay (0+4-1)*(0+4) = 12. Vậy kết quả trả về là 12.

```
.text:080485AB funny_func      proc near               ; CODE XREF: otherhardCode+6C↓p
.text:080485AB
.text:080485AB var_4           = dword ptr -4
.text:080485AB arg_0           = dword ptr  8
.text:080485AB arg_4           = dword ptr  0Ch
.text:080485AB
.text:080485AB                 push    ebp
.text:080485AC                 mov     ebp, esp
.text:080485AE                 sub     esp, 10h
.text:080485B1                 mov     [ebp+var_4], 0
.text:080485B8                 mov     edx, [ebp+arg_0]
.text:080485BB                 mov     eax, [ebp+arg_4]
.text:080485BE                 add     eax, edx
.text:080485C0                 lea     ecx, [eax-1]
.text:080485C3                 mov     edx, [ebp+arg_0]
.text:080485C6                 mov     eax, [ebp+arg_4]
.text:080485C9                 add     eax, edx
.text:080485CB                 imul    eax, ecx
.text:080485CE                 mov     [ebp+var_4], eax
.text:080485D1                 mov     eax, [ebp+var_4]
.text:080485D4                 leave
.text:080485D5                 retn
.text:080485D5 funny_func      endp
```

Sau khi thực hiện hàm funny_func thì sẽ tiến hành so sánh giữa giá trị vừa tính được (12) và số thứ hai. Nếu hai giá trị không bằng nhau thì thoát, ngược lại sẽ thành công.

```
.text:08048742                 add     esp, 10h
.text:08048745                 mov     edx, eax
.text:08048747                 mov     eax, [ebp+var_14]
.text:0804874A                 cmp     edx, eax
.text:0804874C                 jnz     short loc_8048755
.text:0804874E                 call    success_2
.text:08048753                 jmp     short loc_8048761
```

Chạy thử chương trình với input là 4 và 12, ta được:

```
phannguyennhattram-22521501@Phan-Nguyen-Nhat-Tram:~/lab4_NT209$ ./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):
4 12
Your input: 4 12
Congrats! You found a secret pair of numbers :).
phannguyennhattram-22521501@Phan-Nguyen-Nhat-Tram:~/lab4_NT209$
```

Vậy đáp án chính xác là:

```
2 - 1 - 4 12
```

BÀI 3:

> **Yêu cầu 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 các thành viên trong nhóm.
>
> - Nhóm **3 sinh viên:** sắp xếp 3 MSSV theo thứ tự tăng dần rồi lấy 3 số cuối nối nhau. Ví dụ 21520013 < 22520123 < 22521021 sẽ có username là **013123021**.
> - Nhóm **2 sinh viên:** sắp xếp 2 MSSV theo thứ tự tăng dần rồi lấy 4 số cuối nối nhau bằng dấu "-". Ví dụ 22520013 < 22520123 sẽ có username là 0013-0123.
> - Nhóm có **1 sinh viên** có MSSV là 2252xxxx thì username là **2252-xxxx**.

Dựa theo lưu ý, ta tìm được username bằng việc kết hợp 3 số cuối của MSSV mỗi thành viên:

1. Nguyễn Đức Tấn – 22521303
2. Phan Nguyễn Nhật Trâm – 22521501
3. Thái Ngọc Diễm Trinh – 22521541

Nên username là: **303501541**

KIỂM TRA FUNCTION USERPASS(): #Tương ứng với bài 3

Phân tích sơ bộ mã assembly:

```
08048764 <userpass>:
 8048764:   55                      push   %ebp
 8048765:   89 e5                   mov    %esp,%ebp
 8048767:   53                      push   %ebx
 8048768:   83 ec 34                sub    $0x34,%esp
 804876b:   c6 45 ef 66             movb   $0x66,-0x11(%ebp) // array
 804876f:   c6 45 f0 60             movb   $0x60,-0x10(%ebp) // a[1]
 8048773:   c6 45 f1 6c             movb   $0x6c,-0xf(%ebp) // a[2]
 8048777:   c6 45 f2 72             movb   $0x72,-0xe(%ebp) // a[3]
 804877b:   c6 45 f3 5a             movb   $0x5a,-0xd(%ebp) // a[4]
 804877f:   e8 bc fc ff ff          call   8048440 <getchar@plt>
 8048784:   83 ec 0c                sub    $0xc,%esp
 8048787:   68 e3 91 04 08          push   $0x80491e3 // enter user name
 804878c:   e8 bf fc ff ff          call   8048450 <puts@plt>
 8048791:   83 c4 10                add    $0x10,%esp
 8048794:   83 ec 08                sub    $0x8,%esp
 8048797:   8d 45 e5                lea    -0x1b(%ebp),%eax // user
 804879a:   50                      push   %eax
 804879b:   68 6a 91 04 08          push   $0x804916a
 80487a0:   e8 eb fc ff ff          call   8048490 <__isoc99_scanf@plt>
 80487a5:   83 c4 10                add    $0x10,%esp
 80487a8:   e8 93 fc ff ff          call   8048440 <getchar@plt>
 80487ad:   83 ec 0c                sub    $0xc,%esp
 80487b0:   68 f8 91 04 08          push   $0x80491f8 // enter pass
```

```
 80487b5:   e8 96 fc ff ff          call   8048450 <puts@plt>
 80487ba:   83 c4 10                add    $0x10,%esp
 80487bd:   83 ec 08                sub    $0x8,%esp
 80487c0:   8d 45 db                lea    -0x25(%ebp),%eax // pass
 80487c3:   50                      push   %eax
 80487c4:   68 6a 91 04 08          push   $0x804916a
 80487c9:   e8 c2 fc ff ff          call   8048490 <__isoc99_scanf@plt>
 80487ce:   83 c4 10                add    $0x10,%esp
 80487d1:   83 ec 04                sub    $0x4,%esp
 80487d4:   8d 45 db                lea    -0x25(%ebp),%eax
// -0x25(%ebp): pass
// -0x1b(%ebp): user
 80487d7:   50                      push   %eax
 80487d8:   8d 45 e5                lea    -0x1b(%ebp),%eax
 80487db:   50                      push   %eax
 80487dc:   68 10 92 04 08          push   $0x8049210
 80487e1:   e8 3a fc ff ff          call   8048420 <printf@plt>
 80487e6:   83 c4 10                add    $0x10,%esp
 80487e9:   83 ec 0c                sub    $0xc,%esp
 80487ec:   8d 45 e5                lea    -0x1b(%ebp),%eax // eax = user
 80487ef:   50                      push   %eax
 80487f0:   e8 7b fc ff ff          call   8048470 <strlen@plt>
// strlen(usr) --> eax
 80487f5:   83 c4 10                add    $0x10,%esp
 80487f8:   83 f8 09                cmp    $0x9,%eax // eax ss với $9
 80487fb:   75 24                   jne    8048821 <userpass+0xbd>
     // if strlen(usr) != 9 ==> fail
 80487fd:   83 ec 0c                sub    $0xc,%esp
 8048800:   8d 45 e5                lea    -0x1b(%ebp),%eax // eax = pass
 8048803:   50                      push   %eax
 8048804:   e8 67 fc ff ff          call   8048470 <strlen@plt>
// eax = strlen(pas)
 8048809:   83 c4 10                add    $0x10,%esp
 804880c:   89 c3                   mov    %eax,%ebx
// ebx = eax = strlen(pas)
 804880e:   83 ec 0c                sub    $0xc,%esp
 8048811:   8d 45 db                lea    -0x25(%ebp),%eax
// eax = user
 8048814:   50                      push   %eax
 8048815:   e8 56 fc ff ff          call   8048470 <strlen@plt>
// eax = strlen(user)
 804881a:   83 c4 10                add    $0x10,%esp
 804881d:   39 c3                   cmp    %eax,%ebx
// strlen(user) cmp strlen(pas)
 804881f:   74 0a                   je     804882b <userpass+0xc7>
// if strlen(usr) == strlen(pas) ==> continue
// else: fail
 8048821:   e8 1c fe ff ff          call   8048642 <failed> // --> fail
 8048826:   e9 f0 00 00 00          jmp    804891b <userpass+0x1b7>
```

```
 804882b:   c7 45 f4 00 00 00 00    movl   $0x0,-0xc(%ebp) //
 8048832:   c7 45 f4 00 00 00 00    movl   $0x0,-0xc(%ebp) // clean
// -0xc(%ebp): i = 0
 8048839:   eb 53                   jmp    804888e <userpass+0x12a>
 804883b:   83 7d f4 01             cmpl   $0x1,-0xc(%ebp)
 804883f:   7f 17                   jg     8048858 <userpass+0xf4>
// if (i > 1): jmp: 0x8048858
// else:
 8048841:   8b 45 f4                mov    -0xc(%ebp),%eax
// eax = i
 8048844:   83 c0 02                add    $0x2,%eax
// eax = i + 2
 8048847:   0f b6 44 05 e5          movzbl -0x1b(%ebp,%eax,1),%eax
// eax = eax + -0x1b(ebp): eax += user
 804884c:   8d 4d d2                lea    -0x2e(%ebp),%ecx
 804884f:   8b 55 f4                mov    -0xc(%ebp),%edx
 8048852:   01 ca                   add    %ecx,%edx
 8048854:   88 02                   mov    %al,(%edx)
 8048856:   eb 32                   jmp    804888a <userpass+0x126>
 8048858:   83 7d f4 03             cmpl   $0x3,-0xc(%ebp)
 804885c:   7f 17                   jg     8048875 <userpass+0x111>
 804885e:   8b 45 f4                mov    -0xc(%ebp),%eax
 8048861:   83 c0 05                add    $0x5,%eax
 8048864:   0f b6 44 05 e5          movzbl -0x1b(%ebp,%eax,1),%eax
 8048869:   8d 4d d2                lea    -0x2e(%ebp),%ecx
 804886c:   8b 55 f4                mov    -0xc(%ebp),%edx
 804886f:   01 ca                   add    %ecx,%edx
 8048871:   88 02                   mov    %al,(%edx)
 8048873:   eb 15                   jmp    804888a <userpass+0x126>
 8048875:   8b 45 f4                mov    -0xc(%ebp),%eax
 8048878:   83 e8 04                sub    $0x4,%eax
 804887b:   0f b6 44 05 ef          movzbl -0x11(%ebp,%eax,1),%eax
 8048880:   8d 4d d2                lea    -0x2e(%ebp),%ecx
 8048883:   8b 55 f4                mov    -0xc(%ebp),%edx
 8048886:   01 ca                   add    %ecx,%edx
 8048888:   88 02                   mov    %al,(%edx)
 804888a:   83 45 f4 01             addl   $0x1,-0xc(%ebp)
 804888e:   83 7d f4 08             cmpl   $0x8,-0xc(%ebp) //  $8 … $0
 8048892:   7e a7                   jle    804883b <userpass+0xd7>
// loop: if (i <= 8): jump 0x804883b
Else:
 8048894:   c7 45 f4 00 00 00 00    movl   $0x0,-0xc(%ebp)
 804889b:   eb 3f                   jmp    80488dc <userpass+0x178>
 804889d:   8d 55 e5                lea    -0x1b(%ebp),%edx
 80488a0:   8b 45 f4                mov    -0xc(%ebp),%eax
 80488a3:   01 d0                   add    %edx,%eax
 80488a5:   0f b6 00                movzbl (%eax),%eax
 80488a8:   0f be d0                movsbl %al,%edx
 80488ab:   8d 4d d2                lea    -0x2e(%ebp),%ecx
```

```
80488ae:   8b 45 f4             mov    -0xc(%ebp),%eax
80488b1:   01 c8                add    %ecx,%eax
80488b3:   0f b6 00             movzbl (%eax),%eax
80488b6:   0f be c0             movsbl %al,%eax
80488b9:   01 d0                add    %edx,%eax
80488bb:   89 c2                mov    %eax,%edx
80488bd:   c1 ea 1f             shr    $0x1f,%edx
80488c0:   01 d0                add    %edx,%eax
80488c2:   d1 f8                sar    %eax
80488c4:   89 c1                mov    %eax,%ecx
80488c6:   8d 55 db             lea    -0x25(%ebp),%edx
80488c9:   8b 45 f4             mov    -0xc(%ebp),%eax
80488cc:   01 d0                add    %edx,%eax
80488ce:   0f b6 00             movzbl (%eax),%eax
80488d1:   0f be c0             movsbl %al,%eax
80488d4:   39 c1                cmp    %eax,%ecx
80488d6:   75 1e                jne    80488f6 <userpass+0x192>
80488d8:   83 45 f4 01          addl   $0x1,-0xc(%ebp)
80488dc:   83 ec 0c             sub    $0xc,%esp
80488df:   8d 45 e5             lea    -0x1b(%ebp),%eax
80488e2:   50                   push   %eax
80488e3:   e8 88 fb ff ff       call   8048470 <strlen@plt>
80488e8:   83 c4 10             add    $0x10,%esp
80488eb:   89 c2                mov    %eax,%edx
80488ed:   8b 45 f4             mov    -0xc(%ebp),%eax
80488f0:   39 c2                cmp    %eax,%edx
80488f2:   77 a9                ja     804889d <userpass+0x139>
80488f4:   eb 01                jmp    80488f7 <userpass+0x193>
80488f6:   90                   nop
80488f7:   83 ec 0c             sub    $0xc,%esp
80488fa:   8d 45 e5             lea    -0x1b(%ebp),%eax
80488fd:   50                   push   %eax
80488fe:   e8 6d fb ff ff       call   8048470 <strlen@plt>
8048903:   83 c4 10             add    $0x10,%esp
8048906:   89 c2                mov    %eax,%edx
8048908:   8b 45 f4             mov    -0xc(%ebp),%eax
804890b:   39 c2                cmp    %eax,%edx
804890d:   75 07                jne    8048916 <userpass+0x1b2>
804890f:   e8 15 fd ff ff       call   8048629 <success_3>
8048914:   eb 05                jmp    804891b <userpass+0x1b7>
8048916:   e8 27 fd ff ff       call   8048642 <failed>
804891b:   90                   nop // delay 1 chu kỳ
804891c:   8b 5d fc             mov    -0x4(%ebp),%ebx
804891f:   c9                   leave
8048920:   c3                   ret
```

Mã assembly khá phức tạp, ta sử dụng chức năng chuyển sang mã giả của IDAPro để xem xét code:

Mã giả:

IDA View-A | Pseudocode-A | Hex View-1 | Structures

```
1  int userpass()
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
9    char s[10]; // [sp+1Dh] [bp-1Bh]@1
0    char v7[5]; // [sp+27h] [bp-11h]@1
1    unsigned int i; // [sp+2Ch] [bp-Ch]@4
2
3    v7[0] = 102;
4    v7[1] = 96;
5    v7[2] = 108;
6    v7[3] = 114;
7    v7[4] = 90;
8    getchar();
9    puts("Enter your username:");
0    __isoc99_scanf("%[^\n]", s);
1    getchar();
2    puts("Enter your password:");
3    __isoc99_scanf("%[^\n]", v5);
4    printf("Your input username: %s and password: %s\n", s, v5);
5    if ( strlen(s) == 9 && (v0 = strlen(s), v0 == strlen(v5)) )
6    {
7      for ( i = 0; (signed int)i <= 8; ++i )
8      {
9        if ( (signed int)i > 1 )
0        {
1          if ( (signed int)i > 3 )
2            v4[i] = v7[i - 4];
3          else
4            v4[i] = s[i + 5];
5        }
6        else
0000076B userpass:25
```

```
23  __isoc99_scanf("%[^\n]", v5);
24  printf("Your input username: %s and password: %s\n", s, v5);
25  if ( strlen(s) == 9 && (v0 = strlen(s), v0 == strlen(v5)) )
26  {
27    for ( i = 0; (signed int)i <= 8; ++i )
28    {
29      if ( (signed int)i > 1 )
30      {
31        if ( (signed int)i > 3 )
32          v4[i] = v7[i - 4];
33        else
34          v4[i] = s[i + 5];
35      }
36      else
37      {
38        v4[i] = s[i + 2];
39      }
40    }
41    for ( i = 0; ; ++i )
42    {
43      v2 = strlen(s);
44      if ( v2 <= i || (s[i] + v4[i]) / 2 != v5[i] )
45        break;
46    }
47    v3 = strlen(s);
48    if ( v3 == i )
49      result = success_3();
50    else
51      result = failed();
52  }
53  else
54  {
55    result = failed();
56  }
57  return result;
58}
000008E3 userpass:33
```

Có thể thấy username và password người dùng nhập vào được lưu ở biến s và v5. Ở

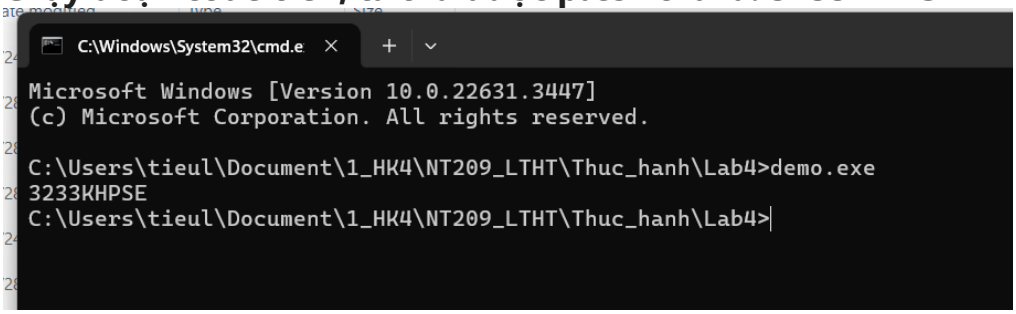
Dòng 41 đến 45 của mã giả, ta thấy bên cạnh điều kiện để duyệt hết các phần tử trong mảng (v2 <= i) thì còn có điều kiện kiểm tra (s[i] + v4[i]) / 2 != v5[i]) với v5 là password do người dùng nhập vào. Nếu điều kiện trên đúng thì input nhập vào sai. Suy ra chuỗi cần tìm là v5 được tạo ra bằng cách: v5[i] = s[i] + v4[i]) / 2.

Code đoạn mã giả trên bằng C, ta thu được code:

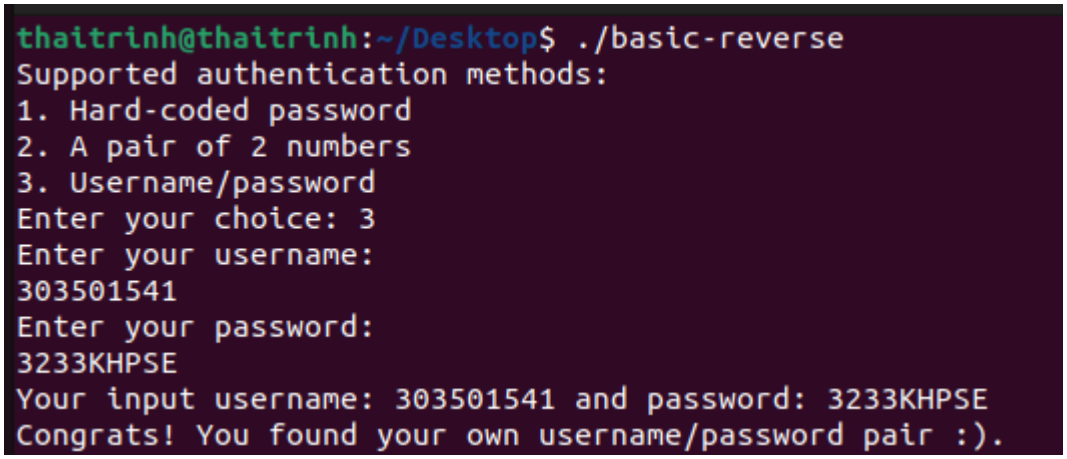
```
#include <iostream>
#include <string.h>
using namespace std;

int main(){
    size_t v0; // ebx@2
    int result; // eax@3
    size_t v2; // eax@15
    size_t v3; // edx@16
```

```
    char v4[9]; // [sp+Ah] [bp-2Eh]@6
    char v5[10]; // [sp+13h] [bp-25h]@1

    char v7[5]; // [sp+27h] [bp-11h]@1
    unsigned int i; // [sp+2Ch] [bp-Ch]@4

    string s = "303501541";

    v7[0] = 102;
    v7[1] = 96;
    v7[2] = 108;
    v7[3] = 114;
    v7[4] = 90;

    for ( i = 0; (signed int)i <= 8; ++i )
    {
      if ( (signed int)i > 1 )
      {
        if ( (signed int)i > 3 )
          v4[i] = v7[i - 4];
        else
          v4[i] = s[i + 5];
      }
      else
      {
        v4[i] = s[i + 2];
      }
    }
    for ( i = 0; ; ++i )
    {
      v2 = 9;
      v5[i] = (s[i] + v4[i]) / 2;
      if ( v2 <= i )
        break;
    }
    cout << v5;
    return 0;
}
```

Chạy đoạn code trên, ta thu được password là: 3233KHPSE



Hình ảnh thực thi file basic-reverse:



Vậy đáp án chính xác là:

```
3 - 1 - 303501541 3233KHPSE
```