BÁO CÁO THỰC HÀNH

**Môn học: Lập trình hệ thống**

**Buổi báo cáo: Lab 05**

**Tên chủ đề:** Kỹ thuật Dịch ngược (tt)

*GVHD: Đỗ Thị Thu Hiền*

*Ngày thực hiện: 1/11/2025*

**THÔNG TIN CHUNG:**

Lớp: NT209.Q12.ANTT

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Nguyễn Lê Hạ My | 23520964 | [23520964@gm.uit.edu.vn](mailto:23520964@gm.uit.edu.vn) |
| 2 | Đặng Nguyễn Gia Bảo | 24520147 | [24520147@gm.uit.edu.vn](mailto:24520147@gm.uit.edu.vn) |
| 3 | Nguyễn Vũ Đức Hạnh | 24520453 | [24520453@gm.uit.edu.vn](mailto:24520453@gm.uit.edu.vn) |

BÁO CÁO CHI TIẾT

## Phân tích và tìm passphrase cố định của basic-reverse với option 1. Báo cáo phương pháp phân tích, input tìm được và hình ảnh minh chứng chạy file.

- Kết quả passphrase cần tìm là: “Out of sight out of mind”.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

1. Kết quả passphrase của basice-reverse khi chạy với option 1.

\* Phương pháp phân tích

- Chạy file basic-reverse bằng IDA, ta chọn view code assembly và pseudo

- Nếu người dùng chọn 1, chương trình nhảy vào hàm hardCode().

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Nội dung do AI tạo ra có thể không chính xác.

1. Mã pseudocode của điều kiện khi nhập 1.

- Phân tích hàm hardCode(): char s1[1000]

🡪 chuẩn bị buffer chứa input, tạo biến s1 dùng để lưu chuỗi người dùng nhập.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, màn hình

Nội dung do AI tạo ra có thể không chính xác.

1. Chi tiết hàm hardCode().

puts("Enter the hard-coded password (option 1):");

🡪 Yêu cầu nhập password

+ Đọc chuỗi người dùng nhập: scanf("%[^\n]", s1);

à Cho phép nhập cả chuỗi dài, bao gồm cả khảong trắng (%[^\n] nghĩa là: đọc toàn bộ chuỗi đến khi gặp xuống dòng).

+ In lại chuỗi người dùng nhập: printf("Your input hard-coded password: %s\n", s1);

+ So sánh password: if ( !strcmp(s1, "Out of sight out of mind") )

* strcmp(a, b) trả về 0 nếu bằng nhau
* !strcmp(a, b) tức là (strcmp == 0), chuỗi nhập vào giống với chuỗi cho trước trong code.

- Nên để nhập đúng thì hard-coded password phải là: “Out of sight out of mind”.

Ảnh có chứa văn bản, ảnh chụp màn hình, hàng

Nội dung do AI tạo ra có thể không chính xác.

1. Mã assembly của hàm hardCode().

|  |
| --- |
| .text:08048782 call \_\_isoc99\_scanf  .text:08048788 cmp ebp+var\_14], 1 # so sánh biến u4 với 1  .text:0804878F jz hardCode # nếu bằng, nhảy vào hardCode  hàm bên dưới |

- Khi nhập “1”, chương trình chạy vào hàm hardCode().

|  |
| --- |
| .text:08048782 call \_\_isoc99\_scanf  .text:08048788 cmp ebp+var\_14], 1 # so sánh biến u4 với 1  .text:0804878F jz hardCode # nếu bằng, nhảy vào hardCode  hàm bên dưới |

|  |
| --- |
| .text:08048744 call \_getchar  .text:08048749 push offset aEnterTheHardCo ; "Enter the hard-coded password (option 1)..." # push địa chỉ chuỗi vào stack  .text:0804874E call \_puts # in chuỗi: “Enter the hard-coded password …” |

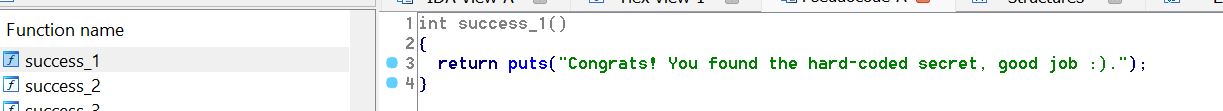
à yêu cầu nhập password.

|  |
| --- |
| .text:08048756 lea eax, [ebp+s1] # lấy địa chỉ biến s1  .text:08048759 push eax # tham số 2 của scanf  vị trí lưu input  .text:0804875C push offset asc\_8048C7E ; "%[^\n]" #tham số 1 của scanf định dạng đọc chuỗi  .text:08048761 call \_\_isoc99\_scanf # đọc chuỗi nhập vào  .text:0804876D lea eax, [ebp+s1]  .text:08048770 push eax  .text:08048775 push offset format ; "Your input hard-coded password: %s\n"  .text:0804877A call \_printf |

à In lại chuỗi người dùng nhập

|  |
| --- |
| .text:08048785 push offset s2 ; "Out of sight out of mind" # push địa chỉ chuỗi “Out of sight out of mind”  .text:0804878A lea eax, [ebp+s1] # push địa chỉ s1 (chuỗi người dùng nhập vào)  .text:0804878D push eax # gọi strcmp(s1, s2)  .text:0804878E call \_strcmp  .text:08048793 test eax, eax # kiểm tra kết quả strcmp  .text:08048795 jnz short loc\_80487A9 # nếu khác 0 🡪 password sai |

+ Hàm success\_1 in ra thông báo khi nhập đúng hardcode yêu cầu.



1. Hàm success\_1.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

1. Mã assembly của hàm success\_1

## Phân tích và tìm cặp số nguyên của basic-reverse với option 2. Báo cáo phương pháp phân tích, input tìm được và hình ảnh minh chứng chạy file.

- Kết quả cặp số nguyên basic-reverse với option 2 cần tìm là: 210 và 12.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

1. Kết quả *c*ủa basic-reverse khi chạy với option *2*.

- Nhập “2” trong hàm main sẽ nhảy tới otherhardCode()

A computer screen shot of a program code

AI-generated content may be incorrect.

1. Mã assembly của hàm otherhardCode().

- Mã giả của hàm otherhardCode()

A screenshot of a computer program

AI-generated content may be incorrect.

1. Mã giả của hàm otherhardCode().

- Trong hàm otherhardCode() sẽ gọi tới hàm funny\_func():

A screenshot of a computer code

AI-generated content may be incorrect.

1. Mã assembly của hàm funny\_func().

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, hàng

Nội dung do AI tạo ra có thể không chính xác.

1. Mã giả cùa hàm funny\_func() để tính toán cặp số.

**\* Phân tích hoạt động:**

**1. Điều kiện để gọi success\_2()**

Để đến được success\_2(), hai câu lệnh if lồng nhau phải đều là true:

- If ngoài (Dòng 15): if ( v2 == 12 )

- If trong (Dòng 19): if ( v0 == v3 )

**2. Tìm Số thứ hai (Biến v2)**

- Chương trình đọc input tại Dòng 11: \_\_isoc99\_scanf("%d %d", &v3, &v2);

+ Số đầu tiên nhập vào sẽ được lưu vào v3.

+ Số thứ hai nhập vào sẽ được lưu vào v2.

- Điều kiện if đầu tiên (Dòng 15) kiểm tra if ( v2 == 12 ).

- Đây là một phép so sánh trực tiếp. Để điều kiện này là ĐÚNG, giá trị của v2 phải chính xác là 12.

è Vì vậy, số thứ hai bắt buộc phải là 12.

**3. Tìm Số thứ nhất (Biến v3)**

- Nếu v2 = 12, chương trình tiếp tục thực thi bên trong khối if.

- Điều kiện if thứ hai (Dòng 19) là if ( v0 == v3 ).

- Để điều kiện này là ĐÚNG, giá trị của v3 (số đầu tiên bạn nhập) phải bằng giá trị của v0.

**4. Tính toán giá trị v0**

- Cần xem v0 được gán giá trị ở đâu.

- Dòng 14: Biến v4 được gán giá trị v4 = 3;

- Dòng 18: Biến v0 được gán bằng kết quả của hàm: v0 = funny\_func(12, v4);

- Vì v4 = 3, điều này có nghĩa là v0 = funny\_func(12, 3).

**5. Phân tích funny\_func(12, 3)**

Có hàm funny\_func với a1 = 12 và a2 = 3:

- Công thức (Dòng 3): return (a1 + a2 - 1) \* (a1 + a2);

- Thay số: return (12 + 3 - 1) \* (12 + 3);

- Tính toán:

+ (14) \* (15)

+ 14 \* 15 = 210

- Vậy, hàm funny\_func(12, 3) trả về giá trị 210.

- Do đó, ở Dòng 18, v0 được gán giá trị 210.

Vậy:

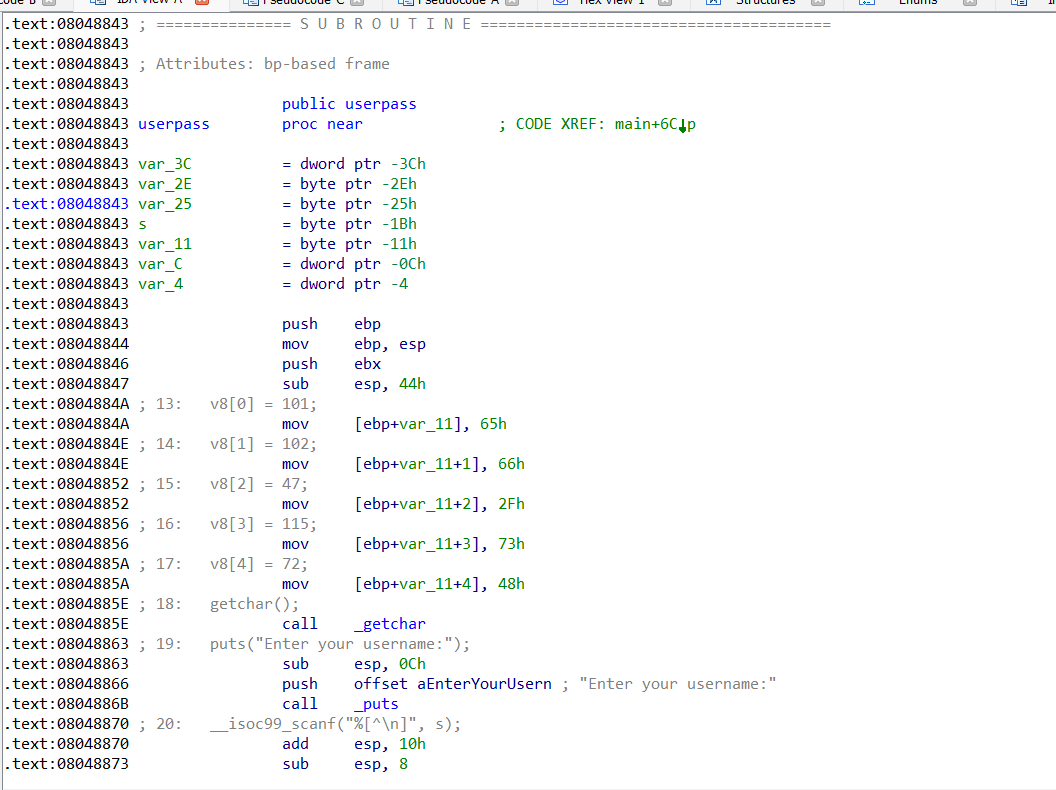
- Từ Bước 2, biết Số thứ hai (v2) phải là 12.

- Từ Bước 3 & 4, biết Số thứ nhất (v3) phải bằng v0, mà v0 = 210.

Vì vậy, để cả hai điều kiện if đều đúng và gọi được success\_2(), input cần nhập là 210 12.

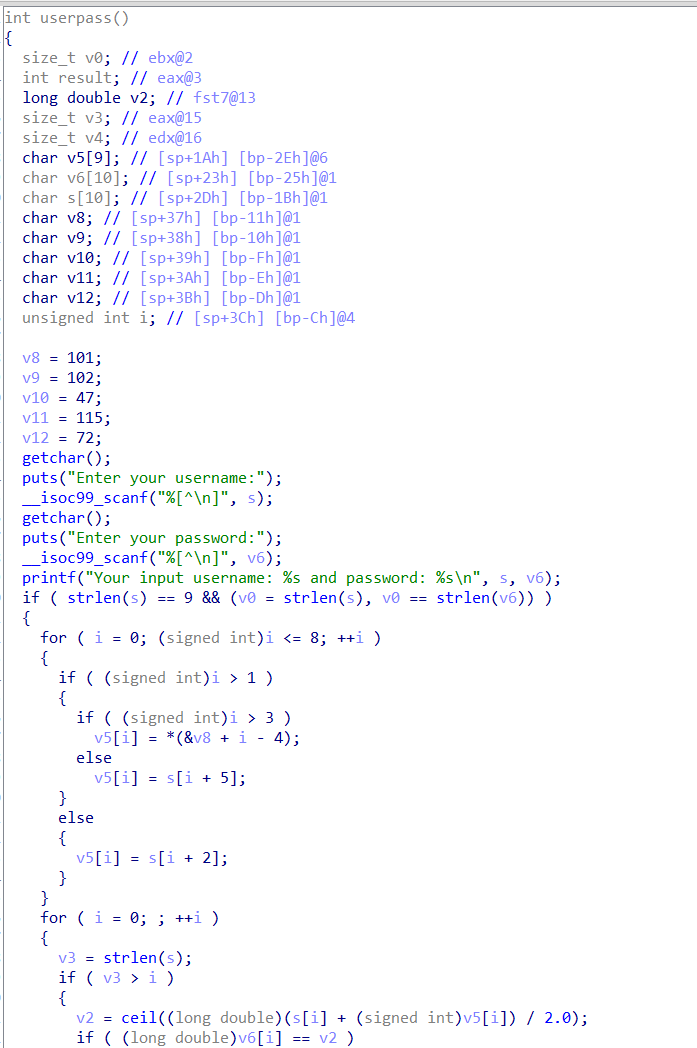
## Phân tích, tìm cặp username/password phù hợp của basic-reverse với option 3. Báo cáo phương pháp phân tích, input tìm được và hình ảnh minh chứng chạy file.

Khi nhập 3 trong hàm main thì nó nhảy tới userpass

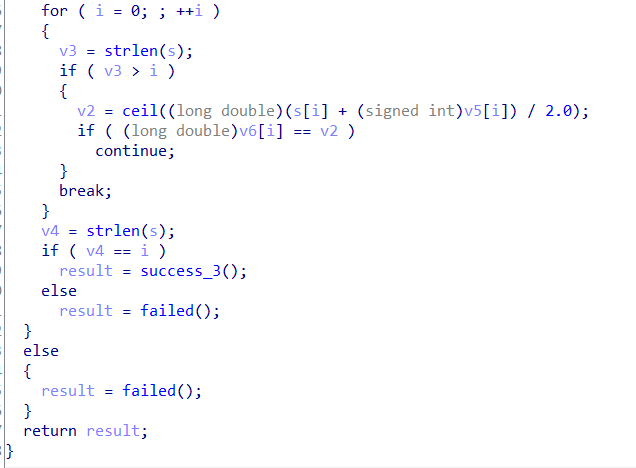


1. Mã assembly của hàm userpass.

- Hàm userpass (Mã giả):

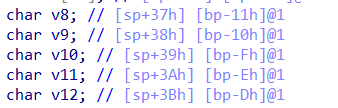


1. Mã giả của hàm userpass.

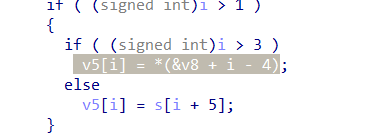


1. Mã giảCác vòng lặp thực hiện của hàm userpass.

* Có 5 biến kiểu char nằm liền kề nhau v8, v9, v10, v11,v 12 (cách 1 byte) từ địa chỉ của biến v8 (Hình 15) và các giá trị được truy xuất dựa trên địa chỉ v8 (Hình 16)



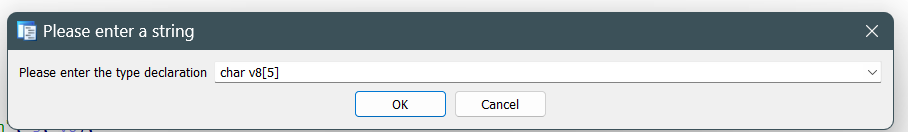
1. 5 biến kiểu char được khai báo đầu hàm userpass.
2. Địa chỉ của biến v8, v9, v10, v11, 12.



1. Giá trị được tham chiếu dựa trên địa chỉ của v8

=> Đây là mảng char 5 ký tự bắt đầu từ v8

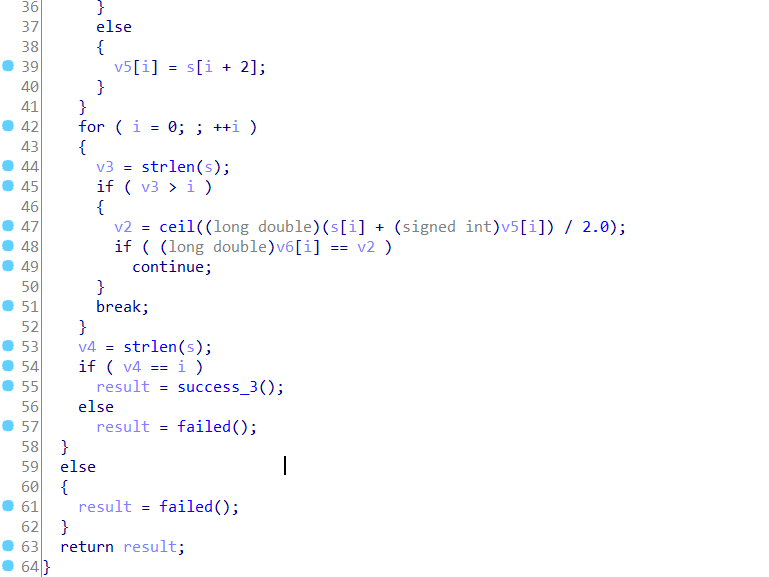
* Chuyển kiểu dữ liệu của v8 thành mảng 5 ký tự: chọn vào biến v8, chuột phải và chọn Set lvar type để đổi thành kiểu dữ liệu char v8[5]



1. Chuyển kiểu dữ liệu của v8 thành mảng bằng Set lvar type

* Ta được mã giả:



1. Mã giả mới sau khi chuyển kiểu dữ liệu v8
2. Mã giả mới sau khi chuyển kiểu dữ liệu v8

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

1. Mã giả hàm userpass.

- Sau khi chuyển: 5 biến char liền kề nhau (v8 đến v12) là một mảng 5 ký tự - v8[5], với:

+ v8[0] = 101 (ký tự 'e')

+ v8[1] = 102 (ký tự 'f')

+ v8[2] = 47 (ký tự '/')

+ v8[3] = 115 (ký tự 's')

+ v8[4] = 72 (ký tự 'H')

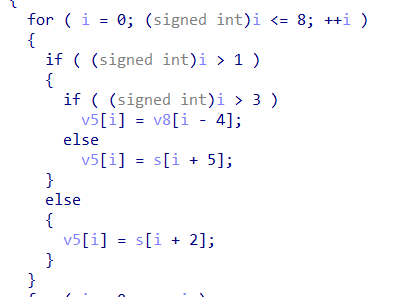
* Username được nhập vào mảng s: s = "453147964"
* Mật khẩu (chưa biết) được nhập vào mảng v6.
* Kiểm tra độ dài: if ( strlen(s) == 9 && (v0 = strlen(s), v0 == strlen(v6)) )

+ Hàm kiểm tra độ dài username (strlen(s)) có bằng 9 không.

+ strlen("453147964") là 9 => TRUE

+ Hàm kiểm tra tiếp độ dài username (gán cho v0, là 9) có bằng độ dài mật khẩu (strlen(v6)) không. Nếu bằng thì thực hiện vòng for, sai thì trả về hàm failed().

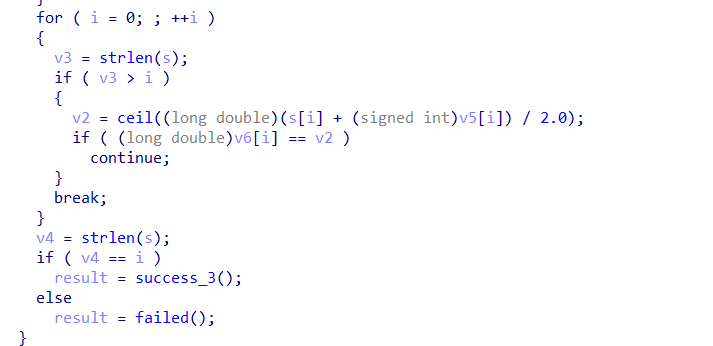
**Vòng for 1: Xây dựng mảng buffer v5**



1. Vòng for đầu tiên tạo bảng buffer v5

* Với i>1: thực hiện nhánh if
  + i >3, v5[i] nhận giá trị từ v8[0] đến v8[4] .
  + i = 2 và i = 3, v5[i] nhận giá trị từ s[7] và s[8].
* Với i=0: thực hiện nhánh else
  + v5[0] = s[2]

**Vòng for 2: Xác thực mật khẩu (Hình 3.4)**



1. Vòng for thứ hai xác thực mật khẩu nhập vào

* Kiểm tra if (v3 > i): i có còn nằm trong độ dài của chuỗi hay không.

+ Nếu đúng (chưa hết chuỗi): Thực hiện phép tính v2 = ceil((long double)(s[i] + (signed int)v5[i]) / 2.0);: công thức để tạo ra ký tự mật khẩu đúng. => v2 chứa chuỗi mật khẩu đúng

+ Nếu đã duyệt hết chuỗi: nhảy đến break để thoát vòng lặp.

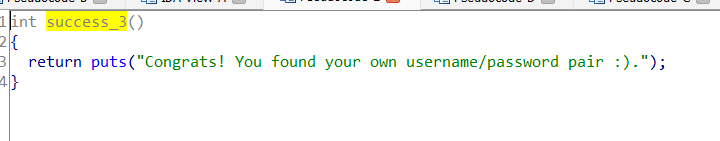
* Kiểm tra if ((long double)v6[i] == v2).

+ Nếu v6[i] (ký tự password nhập vào) khớp với v2 (mật khẩu đúng đúng), tiếp tục vòng lặp để kiểm tra ký tự tiếp theo.

+ Nếu không khớp: nhảy đến lệnh break và thoát khỏi vòng lặp.

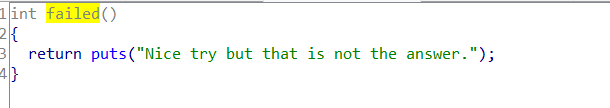
* Sau khi vòng lặp break, kiểm tra if ( v4 == i ). Với v4 là độ dài của username.

+ Nếu i = v4, vòng lặp đã chạy hết toàn bộ chuỗi mà không bị "break" do sai mật khẩu. Tất cả các ký tự đều khớp. Chương trình gọi success\_3().



1. Hàm success\_3 trả về thông báo thành công

+ Nếu i < v4, vòng lặp đã bị break giữa chừng do một ký tự không khớp. Chương trình gọi failed().



1. Hàm failed trả về thông báo khi mật khẩu sai

**Tính mật khẩu và kết quả thực thi file basic -reverse với input đã tìm được:**

Bước 1: Tính toán mảng v5 (Theo Vòng for1)

* s = "453147964"
* v8[5] = {'e', 'f', '/', 's', 'H'} (tính theo mã ASCII 101, 102, 47, 115, 72)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i | (i > 1)? | (i > 3)? | Phép toán | Kết quả v5[i] |
| 0 | Sai | - | v5[0] = s[0+2] = s[2] | 3 |
| 1 | Sai | - | v5[1] = s[1+2] = s[3] | 1 |
| 2 | Đúng | Sai | v5[2] = s[2+5] = s[7] | 6 |
| 3 | Đúng | Sai | v5[3] = s[3+5] = s[8] | 4 |
| 4 | Đúng | Đúng | v5[4] = v8[4-4] = v8[0] | e (101) |
| 5 | Đúng | Đúng | v5[5] = v8[5-4] = v8[1] | f (102) |
| 6 | Đúng | Đúng | v5[6] = v8[6-4] = v8[2] | / (47) |
| 7 | Đúng | Đúng | v5[7] = v8[7-4] = v8[3] | s (115) |
| 8 | Đúng | Đúng | v5[8] = v8[8-4] = v8[4] | H (72) |

Kết quả mảng v5: ['3', '1', '6', '4', 'e', 'f', '/', 's', 'H']

Bước 2: Tính toán mật khẩu v6 (Theo Vòng for 2)

* Dùng công thức: v6[i] = ceil( (s[i] + v5[i]) / 2.0 )

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i | s[i]  (Username) | v5[i]  (Mảng buffer) | Phép tính  (s[i] + v5[i]) / 2.0 | ceil() | Mật khẩu  v6[i] |
| 0 | s[0] = '4' (52) | v5[0] = '3' (51) | (52 + 51) / 2.0 = 51.5 | 52 | 4 |
| 1 | s[1] = '5' (53) | v5[1] = '1' (49) | (53 + 49) / 2.0 = 51.0 | 51 | 3 |
| 2 | s[2] = '3' (51) | v5[2] = '6' (54) | (51 + 54) / 2.0 = 52.5 | 53 | 5 |
| 3 | s[3] = '1' (49) | v5[3] = '4' (52) | (49 + 52) / 2.0 = 50.5 | 51 | 3 |
| 4 | s[4] = '4' (52) | v5[4] = 'e' (101) | (52 + 101) / 2.0 = 76.5 | 77 | M |
| 5 | s[5] = '7' (55) | v5[5] = 'f' (102) | (55 + 102) / 2.0 = 78.5 | 79 | O |
| 6 | s[6] = '9' (57) | v5[6] = '/' (47) | (57 + 47) / 2.0 = 52.0 | 52 | 4 |
| 7 | s[7] = '6' (54) | v5[7] = 's' (115) | (54 + 115) / 2.0 = 84.5 | 85 | U |
| 8 | s[8] = '4' (52) | v5[8] = 'H' (72) | (52 + 72) / 2.0 = 62.0 | 62 | > |

Mật khẩu đúng để gọi success\_3() với username 453147964 là: 4353MO4U>

Kết quả thực hiện:



1. Kết quả thực thi basic\_reverse option3

**HẾT**