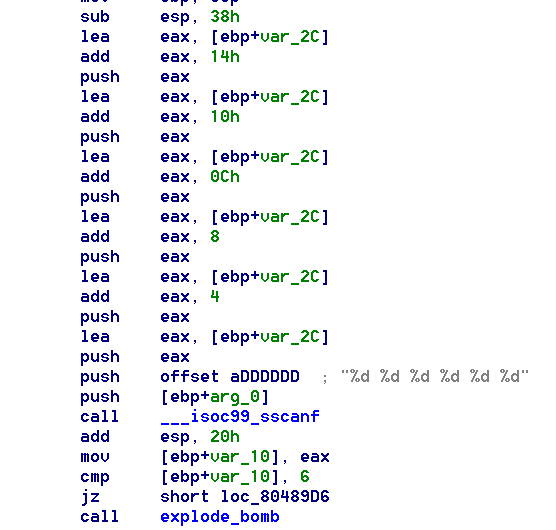
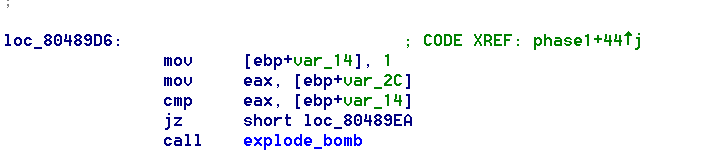
Lập Trình Hệ Thống  
Nhóm 9

22520481 – Võ Nhật Hoàng  
22520330 - Nguyễn Huỳnh Duy  
22520156 – Trần Văn Chiến

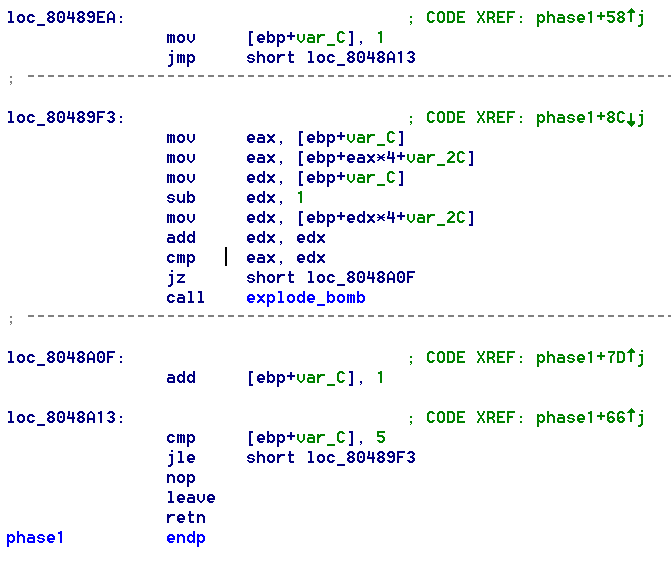
Phase 1:



- Tại phase 1, yêu cầu nhập 6 chữ số được lưu trong mảng có địa chỉ ebp + var\_2C – tạm gọi là mảng arr. Sau đó thực hiện hàm scanf để đọc dữ liệu vào. và lưu độ dài chuỗi nhập vào ở ô nhớ có địa chỉ [ebp + var\_10] – tạm gọi là A. Sau đó so sánh A với 6, nếu khác 6 thì hàm jz không thực hiện và sau đó thực hiện lệnh call explode\_bomb sẽ nổ quả bom. Vì vậy độ dài chuỗi phải bằng 6.



- Sau đó nhảy đến nhãn loc\_80489D6, thực hiện mov 1 vào ô nhớ có địa chỉ [ebp + var\_14] – tạm gọi là B. Mov arr[0] vào eax và thực hiện so sánh với B, nếu khác B(1) thì không thực hiện nhảy và bomb nổ. Vì vậy arr[0] phải bằng 1. Tức là số đầu tiên phải là 1.



- Sau khi nhảy đến nhãn loc\_80489EA thì thực hiện mov 1 vào ô nhớ [ebp + var\_C] – tạm gọi là biến C và nhảy đến nhãn loc\_8048A13.

- Tại nhãn loc\_8048A13 thực hiện so sánh C với 5, nếu C <= 5 thì nhảy đến nhãn loc\_80489F3 phía trên. Ta nhận thấy đây là cấu trúc vòng lặp với điều kiện C <= 5.

- Trong vòng lặp, thực hiện mov C vào eax, sau đó mov giá trị ô nhớ có địa chỉ [ebp + eax\*4 + var\_2C] vào eax, có nghĩa là mov arr[C] vào eax. Tương tự với edx nhưng có thực hiện thêm lệnh trừ C với 1 => edx = arr[C-1]. Tiếp tục thực hiện lệnh add edx, edx tức là edx = 2\*arr[C-1]. Sau đó so sánh eax với adx ( so sánh arr[C] với 2\*arr[C-1] ), nếu khác nhau thì bỏ qua lệnh nhảy để vào nhãn kiểu tra điều kiện vòng lặp và thực hiện hàm call explode\_bomb sẽ nổ bom. Dó đó có thể suy ra arr[C] = 2\*arr[C-1], tức sô phía sau phải bằng số số phía trước nhân 2, kết hợp điều hiện có 6 số và số đầu tiên là 1

=> đáp án là: 1 2 4 8 16 32

Phase 2:

A computer screen shot of a program

Description automatically generated

* Ban đầu gán cho thanh ghi có địa chỉ (ebp+var\_c) là 0Ch (hex) tức 12(decimal)
* Sau đó lấy giá trị của (ebp+var\_C) lưu vào eax.
* Rồi lưu vào eax giá trị của mảng QUESTIONS tại địa chỉ (QUESTIONS + eax\*4)
* Lưu eax vào ô nhớ có địa chỉ (ebp+var\_10).

Hay \*(ebp+var\_10) =\*(QUESTIONS + \*(ebp+var\_C)\*4)

* Tương tự: \*(ebp+var\_14) = \*(QA\_MAP+ \*(ebp+var\_C)\*4)
* Và \*(ebp+s2) =\*(ANSWERS + \*(ebp+var\_14)\*4)
* s2= ANSWERS[\*(QA\_MAP +12\*4)]
* push [ebp + arg\_0] đưa tham số đầu vào tạm gọi là địa chỉ của a1 (&a1) vào stack.
* Chuyển đến hàm transfer.

A screen shot of a computer program

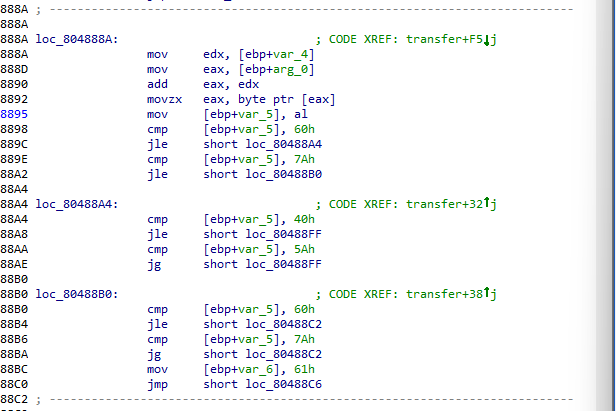
Description automatically generated

* Khởi tạo ô nhớ có địa chỉ (ebp+var\_4) là 0, gọi biến này là i.
* Khởi tạo ô nhớ có địa chỉ (ebp+var\_C) là 5,
* Nhảy đến địa chỉ loc\_8048952

A computer screen shot of a computer code

Description automatically generated

* Lưu i và a1 lần lượt vào edx, eax.
* Cộng edx vào eax, eax=a1+i
* Lấy giá trị \_BYTE tương ứng với địa chỉ (a1+i) và kiểm tra liệu byte đó có null hay không.
* Nếu null thì trả về a1, a1 có giá trị là địa chỉ đầu của mảng int. và kết thúc hàm transfer
* Còn không null thì jmp đến loc\_804888A



* Lưu giá trị của i vào edx, và lưu a1 vào eax.
* Gán cho eax là giá trị byte của ô nhớ tại địa chỉ (a1+i)
* Và lưu byte đầu tiên của eax (al) vào ô nhớ có địa chỉ (ebp+var\_5) tạm gọi là v3.
* So sánh v3 với 60h (hex) hay 96(decimal)
* Nếu v3<=96 hoặc v3>122 thì jmp đến loc\_80488A4

- mà v3<= 40h(hex)=64(dec) hoặc lớn hơn 5Ah (hex)= 90 thì nhảy đến loc\_80488FF

* nếu v3 <=7Ah(hex) =122 và v3> 96 thì nhảy đến loc\_80488B0

loc\_80488FF:

A computer code with many letters and numbers

Description automatically generated

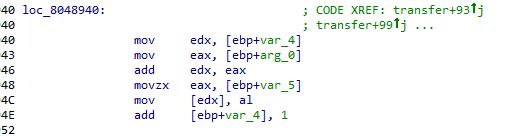
* So sánh v3 với 2Fh(hex)=47(dec)
* Nếu v3> 47 và v3 <=39h(hex)=57(dec) thì lưu giá trị biến v3 vào eax, sau đó trừ đi -30h(hex) = 48(dec) rồi cộng với ô nhớ có địa chỉ (ebp+var\_C), \*(ebp+var\_c)=5.

Hay v3-48+5

A screenshot of a computer program

Description automatically generated

* (v3-48+5) chia lấy dư cho 10
* Rồi cộng thêm 30h(hex)=48(dec).
* Lưu byte đầu tiên vào v3.
* Do đó, v3=(v3-48+5)%10+48



* Sau đó lưu v3 vào ô nhớ có địa chỉ (a1+i)
* Rồi tăng I lên 1.

A computer code with green and blue text

Description automatically generated

* Kế tiếp:
* Lưu i và a1 lần lượt vào edx, eax.
* Cộng edx vào eax, eax=a1+i
* Lấy giá trị \_BYTE tương ứng với địa chỉ (a1+i) và kiểm tra liệu byte đó có null hay không.
* Nếu null thì trả về a1, a1 có giá trị là địa chỉ đầu của mảng int.
* Còn không null thì jmp đến loc\_804888A => Vòng lặp mới.

loc\_80488B0:

A computer code with yellow text

Description automatically generated

* Nếu v3>60h(hex)=96(dec) và v3<=7Ah(hex)=122(dec) thì gán ô nhớ có địa chỉ (ebp+var\_6) với giá trị 61h(hex)=97(dec) gọi biến này là v4

và nhảy đến loc\_80488C6

* Nếu v3<=60h(hex)=96(dec) hoặc v3>7Ah(hex)=122 thì nhảy đến loc\_80488C2.

loc\_80488C2:

A screenshot of a computer code

Description automatically generated

* Ta có v4=65
* Tiếp tục đến loc\_80488C6.

loc\_80488C6:

* Ta có: edx=v3-v4;
* Và edx=v3-v4 công với ô nhớ có địa chỉ (ebp+var\_C) mà \*(ebp+var\_C)=5;
* Lưu vào ecx= edx+5=v3-v4+5;
* A screenshot of a computer code

  Description automatically generated
* Lấy ecx chia lấy dư cho 26
* Cộng với v4, sau đó lưu vào v3.

Hay v3=(v3-v4+5)%26+v4;

* Nhảy đến loc\_8048940:

A computer screen shot of a code

Description automatically generated

* Tiếp tục vòng lặp mới.
* Đến khi tại giá trị byte của ô nhớ có địa chỉ (a1+i) là null thì kết thúc hàm transfer và trả về a1 có giá trị là địa chỉ đầu của một mảng int.

A screenshot of a computer code

Description automatically generated

* Lưu giá trị trả về vào s1.
* Lưu ký tự đầu tiên của s2 vào eax, nếu byte đầu tiên của ký tự này là có giá trị là 0
* Thi nhảy đến loc\_8048A7D: call hàm explode\_bomb.
* Nếu không, s2 và s1 lần lượt được đưa vào stack và gọi đến hàm is\_equal

A screenshot of a computer program

Description automatically generated

* Nếu chuỗi s1 = s2 thì strcmp trả về 0 -> ZF đặt lên 1 -> al=1 -> trả về giá trị 1
* Và ngược lại là trả về 0.

A computer code with text

Description automatically generated

* Nếu giá trị trả về là 1 thì ZF sẽ được đặt về 0 -> nhảy đến loc\_8048A82
* Ngược lại hàm explode\_bomb được call và kết thúc.

loc\_8048A82:

A group of people in a white background

Description automatically generated with medium confidence

* Trả về true.
* Kết thúc phase 2.

Cụ thể:

Ta có s2=ANSWERS[\*(QA\_MAP+12)]=ANSWER[12]= "53/51/7551"

Nếu ở dạng decimal thì ta có mảng {53, 51, 47, 53, 49, 47, 55, 53, 53, 49}

Để thỏa đề bài thì, mảng này trước khi transfer cũng không chứa chữ cái, vì nếu chứa chữ cái thì sau transfer mảng phải chưa các phần từ 65-90 hoặc 97 -122.

Do đó, trong lúc transfer thì biểu thức v3=(v3-48+5)%10+48; luôn được áp dụng

A screenshot of a computer

Description automatically generated

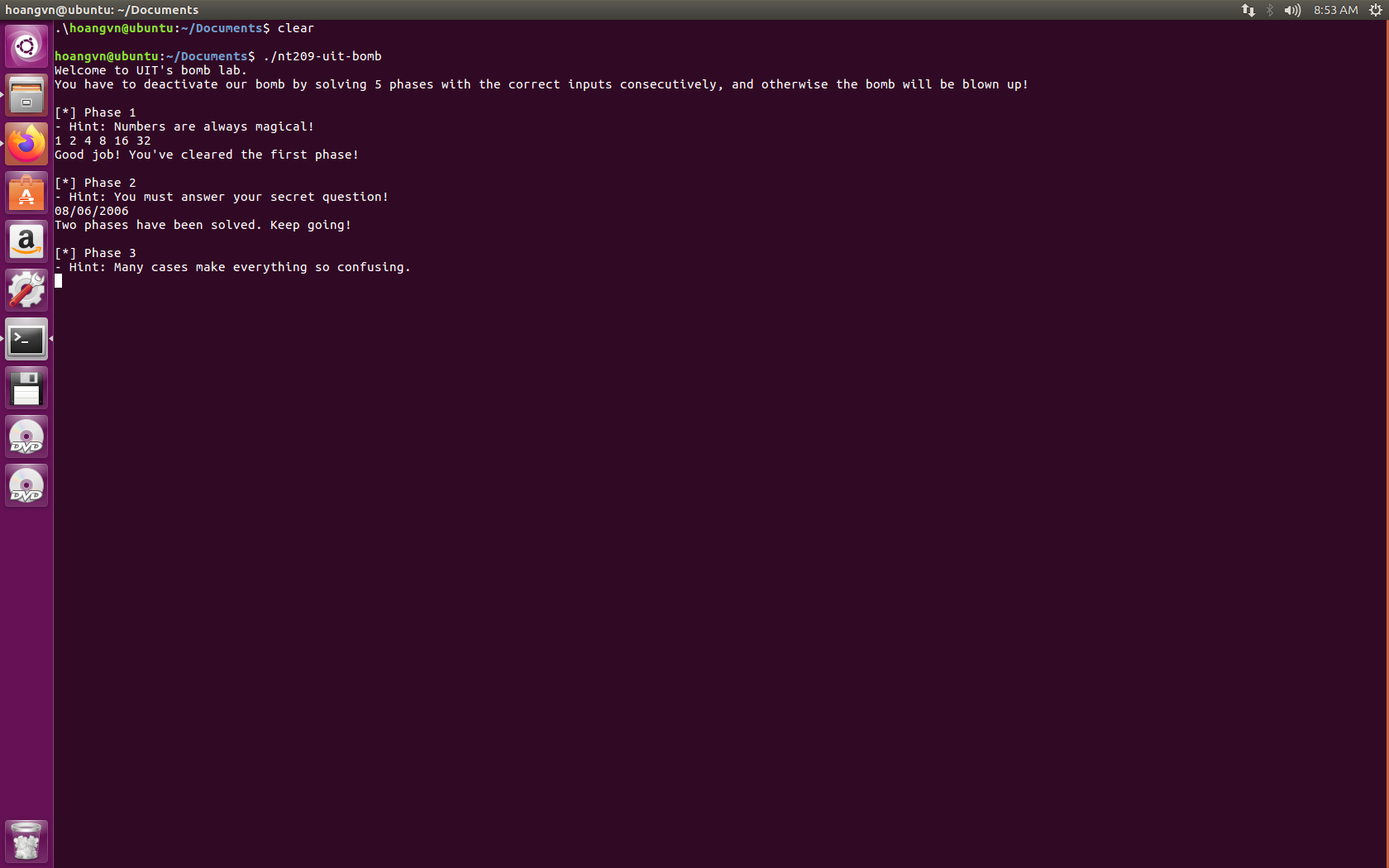
Gọi giá trị của phần tử trong mảng sau transfer là u,

Thì giá trị ban đầu sẽ là v3=(u-48) + 10\*k +43; k thuộc {0,1}

Những phần từ trước transfer <=47 hoặc >57 sẽ không bị biến đổi

Ta suy ra mảng trước khi transfer là {48, 56, 47, 48, 54, 47, 50, 48, 48, 54}

Chuyển lại thành chuỗi dạng ascii : “08/06/2006”



**Phase 4:**

* Trong hàm **main**, ta sẽ thấy **phase4** như sau:

A screenshot of a computer program

Description automatically generated

* Nhấn vào phase4 ta được chương trình như sau:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generatedA screen shot of a computer code

Description automatically generated

**\*Phân tích:**

- Dựa theo thông tin của đoạn code sau:

A screenshot of a computer program

Description automatically generated

* Đoạn code trên là in ra **“%d %d”** và dòng so sánh **[epb+var\_C]** với 2, dựa theo code C ta biết rằng việc cần làm là nhập vào 2 số nguyên. Nếu không phải 2 số thì chương trình sẽ đi đến **short loc\_8048C6C** chính là **explode\_bomb**, khi đó sẽ thất bại.
* Dựa vào những dòng tiếp theo, ta sẽ phân tích được:

A screenshot of a computer code

Description automatically generated

* Số đầu tiên nhập vào phải lớn hơn 0, bởi 3 dòng **mov, test** và **js** ở trên, nếu không thì sẽ nhảy đến **short loc\_8048C6C** chính là **explode\_bomb**, khi đó sẽ thất bại. Với 3 dòng **mov,** **cmp** và **jle** ta thấy số đầu tiên phải là 1 số <=14, vì nếu không thỏa mãn sẽ không nhảy đến **short loc\_8048C71** mà sẽ chạy xuống dòng tiếp theo, chính là **short loc\_8048C6C** chính là **explode\_bomb**, khi đó sẽ thất bại.
* Nếu số nhập vào thỏa mãn sẽ nhảy đến **short loc\_8048C71**, ta sẽ phân tích code sau:

ĐA screen shot of a computer code

Description automatically generated

* Đầu tiên gán giá trị 19 vào ô nhớ (**epb + var\_10**) tạm gọi là **v5**.
* Gán giá trị ô nhớ (**epb + var\_18**) tạm gọi là **v3** vào thanh ghi eax.
* **Push** giá trị **14, 0** và giá trị **v3** vào **stack**.
* Gọi **func4**, khi đó **func4** sẽ có 3 giá trị vào lần lượt là **v3,** **0, 14** (theo quy tắc truy xuất biến cục bộ của stack)
* Kết quả của hàm **func4** được lưu tại thanh ghi **eax,** và được gán vào ô nhớ (**epb + var\_14**) tạm gọi và **v4**, so sánh **v4** và **v5**, nếu **v4** khác **v5** (**v4 != 19**) sẽ nhảy đến **short loc\_8048C9E** chính là **explode\_bomb**, khi đó sẽ thất bại. Đồng thời so sánh (**ebp + var\_1C**) chính là số thứ 2 khi nhập vào với **v5**, nếu **v2==v5** sẽ bỏ qua bomb và đi đến kết thúc. Ngược lại, sẽ đi đến **short loc\_8048C9E** chính là **explode\_bomb**, khi đó sẽ thất bại. từ đó suy ra số thứ 2 cần nhập vào chính là số **19(1).**
* Mở hàm func4, ta được những đoạn code sau:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A computer code with blue and green text

Description automatically generated

A computer code with blue and green text

Description automatically generated

**\*Phân tích hàm func4:**

- Khi phân tích đoạn sau:

A blue lines on a white background

Description automatically generated

* Ta có thể suy ra được 3 đối số đưa vào hàm lần lượt gọi là **a1, a2, a3** được gán với ô nhớ (**epb+arg\_0), (epb+arg\_)** và **(epb+arg\_8)** biến **temp** là **(epb+var\_C).**
* Qua đoạn code sau có thể suy ra công thức thực thi:

A screenshot of a computer code

Description automatically generated

* Công thức toán tương ứng với đoạn code trên là:

**temp = (a3-a2)/2 + a2**

* So sánh **temp** với **a1:**



* Nếu **temp <= a1** sẽ thực thi hàm **short loc\_8048C08:**

A computer screen shot of a computer code

Description automatically generated

* Hàm trên so sánh **temp** với **a1**, nếu **temp >= a1** thì sẽ thực thi hàm **short loc\_8048C31**

A computer code with blue and green text

Description automatically generated

* Khi đó sẽ gán **temp** cho **eax** rồi đi đến kết thúc hàm, với **temp** được tính với công thức toán ở trên: **temp = (a3-a2)/2 + a2**
* Nếu **temp < a1** trong hàm **short loc\_8048C08:** thực thi **temp + 1**, **push** 3 giá trị **a3, temp + 1, a1** vào bộ nhớ stack và gọi hàm **func4** và cộng cho **temp,** công thức tương ứng sẽ là **result=func4(a1, temp + 1, a3) + temp**
* Nếu không thỏa mãn thực thi hàm **short loc\_8048C08**, chương trình sẽ thực thi các lệnh

A screen shot of a computer code

Description automatically generated

* Các lệnh trên thực thi **push temp – 1, a2, a1**. Sau đó gọi **func4**  với 3 đối số đã **push** rồi cộng với **temp**. Khi đó công thức tương ứng là: **result=func4(a1, a2, temp-1)+temp**
* Khi đó, tổng quan hàm **func4** sẽ như sau:

**int \_\_cdecl func4(int a1, int a2, int a3)**

**{**

**int result; // eax@2**

**int v4; // [sp+Ch] [bp-Ch]@1**

**v4 = (a3 - a2) / 2 + a2;**

**if ( v4 <= a1 )**

**{**

**if ( v4 >= a1 )**

**result = (a3 - a2) / 2 + a2;**

**else**

**result = func4(a1, v4 + 1, a3) + v4;**

**}**

**else**

**{**

**result = func4(a1, a2, v4 - 1) + v4;**

**}**

**return result;**

**}**

* Thực hiện tính toán để **result=19** khi ấy sẽ tìm được giá trị nhập đầu tiên ở đầu chương trình tạm gọi là **v3**, cũng chính là tham số đầu tiên **a1** của hàm **func4**

**\*Tính toán:**

**Dùng code sau để chạy dò kết quả với a1 từ 1 đến 14:**

**#include <iostream>**

**using namespace std;**

**int func4(int a1, int a2, int a3)**

**{**

**int result;**

**int v4;**

**v4 = (a3 - a2) / 2 + a2;**

**if ( v4 <= a1 )**

**{**

**if ( v4 >= a1 )**

**result = (a3 - a2) / 2 + a2;**

**else**

**result = func4(a1, v4 + 1, a3) + v4;**

**}**

**else**

**{**

**result = func4(a1, a2, v4 - 1) + v4;**

**}**

**return result;**

**}**

**int main()**

**{**

**for(int i=1; i<=14; i++)**

**cout << func4(i, 0, 14) << endl;**

**}**

* Ta được kết quả:

A screenshot of a white page

Description automatically generated

Khi đó **a1=4** sẽ cho ra **result=19,** Vậy số đầu tiên nhập vào là **4(2)**

* **Từ (1) và (2) suy ra input nhập vào là: 4 19**

Phase 5:

A computer code with blue and green text

Description automatically generated

- Tại phase 5 yêu cầu nhập vào một chuỗi s (push [ebp+s]). Sau đó gọi hàm \_strlen để tính giá trị chiều dài chuỗi s và lưu vào thanh ghi eax. So sánh chiều dài chuỗi với 6 (cmp). Nếu chiều dài khác 6 thì sẽ không nhảy vào thực hiện hàm explode\_bomb. Do đó suy ra chiều dài chuỗi nhập vào phải có 6 ký tự.

A computer code with green and blue text

Description automatically generated

- Sau khi nhảy tới nhãn loc\_804CC8 thì thực hiện gán giá trị 0 cho 2 biến có địa chỉ [ebp+var\_10] - tạm gọi là biến A, [ebp+var\_C] – tạm gọi là biến B. Rồi nhảy đến nhãn loc\_8048CF7.

A screenshot of a computer code

Description automatically generated

- Tại nhãn loc\_8048CF7, thực hiện lệnh so sánh biến A với 0x43, nếu kết quả khác nhau thì sẽ bỏ qua lệnh nhảy và thực hiện hàm explode\_bomb. Do đó biến A = 0x43. Quay lại lệnh cmp đầu tiên thì sẽ so sánh B với 5 nếu B <= 5 thì sẽ nhảy đến nhãn loc\_804CD8 là nhãn phía trên. Vì vậy ta có thể nhận ra đây là 1 vòng lặp với điều kiện B <=5.

A computer code with blue and green text

Description automatically generated

- Trong vòng lặp, mov B vào edx, mov s vào eax. Sau đó eax + edx gán vào eax có được giá trị của vị trí thứ B trong chuỗi s - s[B]. Sau đó and với 0xF tức là s[B] % 16 gán vào eax. Sau đó truy xuất vào mãng array\_3855 với index là eax\*4.

A screenshot of a computer

Description automatically generatedA screenshot of a computer code

Description automatically generated

Dựa vào mảng array\_3855 – tạm gọi mảng arr có thể tóm gọn trong bảng sau:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| arr[0] | arr[1] | arr[2] | arr[3] | arr[4] | arr[5] | arr[6] | arr[7] |
| 0x2 | 0xA | 0x6 | 0x1 | 0xC | 0x10 | 0x9 | 0x3 |
| r[8] | arr[9] | arr[10] | arr[11] | arr[12] | arr[13] | arr[14] | arr[15] |
| 0x4 | 0x7 | 0xE | 0x5 | 0xB | 0x8 | 0xF | 0xD |

A computer code with numbers and letters

Description automatically generated

- Sau khi lấy được giá trị arr tương ứng thi thực hiện A + arr[index] gán vào A và B + 1 gán vào B. Sau đó hết vòng lặp và thực hiện kiểm tra điều kiện của B như trên.

- Kết hợp điều kiện A = 0x43, A = A + arr[index], index = s[B] % 16 thì đáp án chúng ta nhập vào phải là 1 chuỗi 6 ký tự, sao cho tổng giá trị của arr[index] với index là số dư của mỗi ký tự với 16 phải bẳng 0x43.

=> Ta có nhiều đáp án thỏa mãn: “LLLLLD” và hoán vị khác nhau của nó, “<<<<<4” và hoán vị khác nhau của nó, “NNNNGH” và hoán vị khác nhau của nó, …

