



Lab 5

BÁO CÁO BÀI THỰC HÀNH SỐ 5

Kỹ thuật dịch ngược (tt)

Reverse Engineering (cont)

Môn học: Lập trình hệ thống

Giảng viên hướng dẫn	ThS. Đỗ Thị Hương Lan
Sinh viên thực hiện	Nguyễn Hồ Nhật Khoa (22520677) Lê Quốc Ngô (22520951) Trần Tiến Nhật (22521030)
Mức độ hoàn thành	Hoàn thành
Thời gian thực hiện	24/5/2024
Tự chấm điểm	10/10

Phase 1

```

int __cdecl phase1(int a1)
{
    int result; // eax@3
    int v2[6]; // [sp+Ch] [bp-2Ch]@1
    int v3; // [sp+24h] [bp-14h]@3
    int v4; // [sp+28h] [bp-10h]@1
    int i; // [sp+2Ch] [bp-Ch]@5

    v4 = __isoc99_sscanf(a1, "%d %d %d %d %d %d", v2, &v2[1], &v2[2], &v2[3], &v2[4], &v2[5]);
    if ( v4 != 6 )
        explode_bomb();
    v3 = 3;
    result = v2[0];
    if ( v2[0] <= 3 )
        explode_bomb();
    for ( i = 1; i <= 5; ++i )
    {
        result = v2[i];
        if ( result != 2 * i + v2[i - 1] )
            explode_bomb();
    }
    return result;
}

```

- Hàm **phase1** nhận một tham số a1.
- Mảng v2 với 6 phần tử được khai báo để lưu trữ dãy 6 số nguyên.
- Hàm **sscanf** đọc 6 số nguyên từ chuỗi đầu vào a1 và lưu trữ chúng trong mảng v2.
- Nếu không đọc được đúng sáu số nguyên, hàm **explode_bomb()** sẽ được gọi.
- Giá trị đầu tiên của mảng v2 được gán cho result. Nếu giá trị này không lớn hơn 3, hàm **explode_bomb()** sẽ được gọi.
- Vòng lặp kiểm tra từng giá trị trong mảng v2 từ vị trí thứ 1 đến vị trí thứ 5. Với mỗi giá trị v2[i], nếu nó không bằng $2 * i + v2[i - 1]$, hàm **explode_bomb()** sẽ được gọi.

Tóm lại, Để hàm **phase1** không gọi hàm **explode_bomb()**, chuỗi đầu vào phải thỏa mãn các điều kiện sau:

- Chuỗi chứa chính xác sáu số nguyên.
- Số nguyên đầu tiên phải lớn hơn 3.
- Các số nguyên tiếp theo phải thỏa mãn công thức:

$$v2[i] = 2 * i + v2[i - 1], \text{ với } i \text{ từ } 1 \text{ đến } 5.$$

Để tìm dãy số thỏa mãn các điều trên, chúng ta có thể viết ra công thức tổng quát cho các phần tử của dãy số:

1. $v2[0] = a$, với $a > 3$
2. $v2[1] = 2 * 1 + v2[0] = 2 + a$
3. $v2[2] = 2 * 2 + v2[1] = 4 + (2 + a) = 6 + a$
4. $v2[3] = 2 * 3 + v2[2] = 6 + (6 + a) = 12 + a$
5. $v2[4] = 2 * 4 + v2[3] = 8 + (12 + a) = 20 + a$
6. $v2[5] = 2 * 5 + v2[4] = 10 + (20 + a) = 30 + a$

Vậy dãy số sẽ có dạng: a, a+2, a+6, a+12, a+20, a+30.

Một số dãy số đúng:

- Với $a = 4$, $v2 = 4, 6, 10, 16, 24, 34$
- Với $a = 5$, $v2 = 5, 7, 11, 17, 25, 35$
- ...

Như vậy, bất kỳ dãy số nào có dạng trên và với $a > 3$ đều là đáp án đúng. Điều này cho thấy bài toán có vô số đáp án đúng, tùy thuộc vào giá trị của a .

Kết quả thực thi:

```
tnh47@Nhat:~/UIT/Ky4/LTHT$ ./nt209-uit-bomb
Welcome to UIT's bomb lab.
You have to deactivate our bomb by solving 5 phases
omb will be blown up!

[*] Phase 1
- Hint: Numbers are always magical!
4 6 10 16 24 34
Good job! You've cleared the first phase!

[*] Phase 1
- Hint: Numbers are always magical!
5 7 11 17 25 35
Good job! You've cleared the first phase!
```

Phase 2:

```
char __cdecl phase2(int a1)
{
    char *v1; // ST28_4@1
    int v2; // eax@2
    char *s1; // [sp+Ch] [bp-1Ch]@1
    char *s2; // [sp+10h] [bp-18h]@1

    v1 = QUESTIONS[4];
    s2 = ANSWERS[*( &QA_MAP + 4 )];
    s1 = (char *)transfer(a1);
    if ( !*s2 || (LOBYTE(v2) = is_equal(s1, s2), !v2) )
        explode_bomb();
    return v2;
}
```

- Hàm **phase2** nhận một tham số $a1$.
- $v1$ trỏ tới câu hỏi thứ 5 trong mảng **QUESTIONS**.
- $s2$ trỏ tới câu trả lời tương ứng trong mảng **ANSWERS**, được xác định bởi giá trị $*(\&QA_MAP + 4)$ tương đương với việc truy cập giá trị tại vị trí thứ 5 trong mảng **QA_MAP**.

Lab 5: Reverse Engineering (cont)

```
.data:0804B0E0      public QA_MAP
.data:0804B0E0 QA_MAP dd 0                ; DATA XREF: phase2+1D↑r
.data:0804B0E4      db 1
.data:0804B0E5      db 0
.data:0804B0E6      db 0
.data:0804B0E7      db 0
.data:0804B0E8      db 2
.data:0804B0E9      db 0
.data:0804B0EA      db 0
.data:0804B0EB      db 0
.data:0804B0EC      db 3
.data:0804B0ED      db 0
.data:0804B0EE      db 0
.data:0804B0EF      db 0
.data:0804B0F0      db 4
.data:0804B0F1      db 0
.data:0804B0F2      db 0
.data:0804B0F3      db 0
.data:0804B0F4      db 5
.data:0804B0F5      db 0
.data:0804B0F6      db 0
.data:0804B0F7      db 0
.data:0804B0F8      db 6
.data:0804B0F9      db 0
.data:0804B0FA      db 0
.data:0804B0FB      db 0
.data:0804B0FC      db 7
.data:0804B0FD      db 0
```

- Phần tử thứ 5 trong mảng **QA_MAP** có giá trị bằng 4. Vậy s2 sẽ trở tới câu trả lời tương ứng trong mảng **ANSWERS** là **ANSWERS[4] = “Dpukvd”**.

```
.data:0804B160      public ANSWERS
.data:0804B160 ANSWERS dd offset aIpuoKbvun    ; DATA XREF: phase2+2A↑r
.data:0804B160      ; "Ipuo Kbvun"
.data:0804B164      dd offset aIhunrvr      ; "Ihunrvr"
.data:0804B168      dd offset aPumvythapvuZlj ; "Pumvythapvu Zljbyapaf"
.data:0804B16C      dd offset aZwypun        ; "Zwypun"
.data:0804B170      dd offset aDpukvd        ; "Dpukvd"
.data:0804B174      dd offset aZpunhwvyl     ; "Zpunhwvyl"
.data:0804B178      dd offset aCplauhtlzl    ; "Cplauhtlzl"
.data:0804B17C      dd offset a729791        ; "72/9791"
.data:0804B180      dd offset aIpuoKbvun     ; "Ipuo Kbvun"
.data:0804B184      dd offset aZlclu         ; "Zlclu"
.data:0804B188      dd offset aPujvyyljasf    ; "Pujvyyljasf"
.data:0804B18C      dd offset aCplauhtlzl    ; "Cplauhtlzl"
.data:0804B190      dd offset a79504929779    ; "79504929779"
.data:0804B194      dd offset aUa976BpaIvti    ; "ua976-bpa-ivti"
.data:0804B198      dd offset aJopjhvn        ; "Jopjhvn"
.data:0804B19C      dd offset aHyapmpjphsPual    ; "Hyapmpjphs Pualsspnlujl"
.data:0804B1A0      dd offset aPzhilssh       ; "Pzhilssh"
.data:0804B1A4      dd offset aBupclyzpafVmPu    ; "Bupclyzpaf Vm Pumvythapvu Aljouvsvnf"
.data:0804B1A8      dd offset aQvsspill       ; "Qvsspill"
.data:0804B1AC      dd offset aUj_bpa_lkb_cu    ; "uj.bpa.lkb.cu"
.data:0804B1B0      dd offset aYbzzph         ; "Ybzzph"
.data:0804B1B4      dd offset a75739773       ; "75/73/9773"
.data:0804B1B8      dd offset aHuahyjapjh     ; "Huahyjapjh"
.data:0804B1BC      dd offset aAolNylhaDhssVm    ; "Aol Nylha Dhss Vm Jopuh"
.data:0804B1C0      dd offset aYlnpzalyz       ; "Ylnpzalyz"
.data:0804B1C4      dd offset aTlyjbyf        ; "Tlyjbyf"
.data:0804B1C8      dd offset aNt_bpa_lkb_cu    ; "nt.bpa.lkb.cu"
.data:0804B1CC      dd offset aChuTplbXbvjAbN    ; "Chu Tplb Xbvj Ab Npht"
.data:0804B1D0      dd offset aPualyula        ; "Pualyula"
```

```
s1 = (char *)transfer(a1);
```

- Hàm **transfer** được gọi với tham số a1 (chuỗi đầu vào) và trả về chuỗi đã được chuyển đổi, lưu vào s1.

```
int __cdecl transfer(int a1)
{
    char v2; // [sp+Ah] [bp-6h]@8
    char v3; // [sp+Bh] [bp-5h]@2
    int i; // [sp+Ch] [bp-4h]@1

    for ( i = 0; *(_BYTE *)(i + a1); ++i )
    {
        v3 = *(_BYTE *)(i + a1);
        if ( (v3 <= 96 || v3 > 122) && (v3 <= 64 || v3 > 90) )
        {
            if ( v3 > 47 && v3 <= 57 )
                v3 = (v3 - 48 + 7) % 10 + 48;
        }
        else
        {
            if ( v3 <= 96 || v3 > 122 )
                v2 = 65;
            else
                v2 = 97;
            v3 = (v3 - v2 + 7) % 26 + v2;
        }
        *(_BYTE *)(a1 + i) = v3;
    }
    return a1;
}
```

```
if ( !*s2 || (LOBYTE(v2) = is_equal(s1, s2), !v2) )
    explode_bomb();
```

- Kiểm tra xem s2 có rỗng hay không và gọi hàm **is_equal(s1, s2)** để so sánh s1 và s2. Nếu không bằng nhau, v2 sẽ là 0 và **explode_bomb()** sẽ được gọi.

Tóm lại, để giải **phase2**, cần nhập đầu vào đúng để chuyển đổi thành một chuỗi khớp với **ANSWERS[4]** là “Dpukvd”.

Phân tích hàm **transfer**:

```
for ( i = 0; *(_BYTE *)(i + a1); ++i )
{
    v3 = *(_BYTE *)(i + a1);
```

- Vòng lặp lấy từng ký tự trong chuỗi a1 rồi gán vào v3

```

if ( (v3 <= 96 || v3 > 122) && (v3 <= 64 || v3 > 90) )
{
    if ( v3 > 47 && v3 <= 57 )
        v3 = (v3 - 48 + 7) % 10 + 48;
}

```

- Kiểm tra nếu ký tự là số ('0' đến '9'):
 - Mã hóa số bằng cách: $v3 = (v3 - 48 + 7) \% 10 + 48$.
 - Phép toán này dịch chuyển ký tự số đi 7 vị trí, và sau đó lấy phần dư khi chia cho 10 để đảm bảo nằm trong khoảng 0 - 9.

```

else
{
    if ( v3 <= 96 || v3 > 122 )
        v2 = 65;
    else
        v2 = 97;
    v3 = (v3 - v2 + 7) % 26 + v2;
}
*( _BYTE *) (a1 + i) = v3;

```

- Kiểm tra nếu ký tự là chữ cái hoa ('A' đến 'Z') hay thường ('a' đến 'z'):
 - Thiết lập v2 là giá trị ASCII của 'A' hoặc 'a'.
 - Mã hóa chữ cái bằng cách: $v3 = (v3 - v2 + 7) \% 26 + v2$.
 - Phép toán này dịch chuyển ký tự chữ đi 7 vị trí trong bảng chữ cái, và sau đó lấy phần dư khi chia cho 26 để đảm bảo nằm trong khoảng A-Z hoặc a-z.

Như vậy, để tìm chuỗi đầu vào cần dịch ngược lại 7 vị trí ứng với mỗi ký tự của chuỗi “Dpukvd”, đây là code Python thực hiện việc đó:

```

1  def reverse_transfer(s):
2      result = []
3      for char in s:
4          if 'a' <= char <= 'z':
5              new_char = chr((ord(char) - ord('a') - 7 + 26) % 26 + ord('a'))
6          elif 'A' <= char <= 'Z':
7              new_char = chr((ord(char) - ord('A') - 7 + 26) % 26 + ord('A'))
8          elif '0' <= char <= '9':
9              new_char = chr((ord(char) - ord('0') - 7 + 10) % 10 + ord('0'))
10         else:
11             new_char = char
12         result.append(new_char)
13     return ''.join(result)
14
15     output = "Dpukvd"
16     input_string = reverse_transfer(output)
17     print(f"{input_string}")

```

Kết quả:

```

PS D:\Code\Python> python -u "d:\Code\Python\lab5-phase2"
Window

```

Kết quả thực thi:

```

tnh47@Nhat: ~/UIT/Ky4/LTH1$ ./nt209-uit-bomb input.txt
Welcome to UIT's bomb lab.
You have to deactivate our bomb by solving 5 phases with the correct inputs consecutively, and otherwise the bomb will be blown up!

[*] Phase 1
- Hint: Numbers are always magical!
Good job! You've cleared the first phase!

[*] Phase 2
- Hint: You must answer your secret question! Find your own answer or decrypt the answer saved in this file.
Window
Two phases have been solved. Keep going!

```

Phase 3:

```

int __cdecl phase3(int a1)
{
    int result; // eax@21
    unsigned __int8 v2; // [sp+fh] [bp-19h]@1
    int v3; // [sp+10h] [bp-18h]@1
    int v4; // [sp+14h] [bp-14h]@1
    int v5; // [sp+18h] [bp-10h]@1
    char v6; // [sp+1fh] [bp-9h]@4

    v5 = 0;
    v5 = __isoc99_sscanf(a1, "%d %c %d", &v4, &v2, &v3);
    if ( v5 <= 2 )
        explode_bomb();
    switch ( v4 )
    {
        case 0:
            v6 = 'd';
            if ( v3 != 482 )
                explode_bomb();
            return result;
        case 1:
            v6 = 105;
            if ( v3 != 793 )
                explode_bomb();
            return result;
        case 2:
            v6 = 119;
            if ( v3 != 592 )
                explode_bomb();
            return result;
        case 3:
            v6 = 97;
            if ( v3 != 898 )
                explode_bomb();
            return result;
        case 4:
            v6 = 119;
            if ( v3 != 947 )
                explode_bomb();
            return result;
        case 5:
            v6 = 113;
            if ( v3 != 573 )
                explode_bomb();
            return result;
        case 6:
            v6 = 118;
            if ( v3 != 542 )
                explode_bomb();
            return result;
        case 7:
            v6 = 118;
            if ( v3 != 628 )
                explode_bomb();
            return result;
        default:
            v6 = 101;
            explode_bomb();
            return result;
    }
    result = v2;
    if ( v6 != v2 )
        explode_bomb();
    return result;
}

```

Lab 5: Reverse Engineering (cont)

- Hàm **phase3** nhận một tham số a1
 - Gán giá trị 0 cho biến v5.
 - Sử dụng hàm sscanf để quét dữ liệu từ tham số đầu vào a1 theo định dạng "%d %c %d", lưu vào v4, v2, v3.
 - Kiểm tra nếu số lượng biến đã đọc từ a1 ít hơn hoặc bằng 2 hàm **explode_bomb()** sẽ được gọi.
 - Sử dụng một câu lệnh switch-case để kiểm tra giá trị của v4:
 - Nếu v4 không thuộc các giá trị từ 0 đến 7, sẽ gọi hàm **explode_bomb()** và kết thúc.
 - Nếu v4 thuộc các giá trị từ 0 đến 7, hàm tiếp tục kiểm tra giá trị của v3 tương ứng với từng giá trị của v4. Nếu v3 không đúng, hàm **explode_bomb()** sẽ được gọi và kết thúc.
 - Sau khi kiểm tra xong, hàm tiếp tục kiểm tra xem giá trị của v6 có bằng v2 hay không. Nếu không, cũng sẽ gọi hàm **explode_bomb** và kết thúc.
- ⇒ Có tất cả 8 bộ số thỏa mãn (8 trường hợp của switch case):

v4	v2	v3
0	d	482
1	i	793
2	w	592
3	a	898
4	w	947
5	q	573
6	v	542
7	v	628

Kết quả thực thi:

```
tnh47@Nhat: ~/UIT/Ky4/LTH1 x + v
tnh47@Nhat:~/UIT/Ky4/LTH1$ ./nt209-uit-bomb input.txt
Welcome to UIT's bomb lab.
You have to deactivate our bomb by solving 5 phases with the correct inputs consecutively, and otherwise the bomb will be blown up!

[*] Phase 1
- Hint: Numbers are always magical!
Good job! You've cleared the first phase!

[*] Phase 2
- Hint: You must answer your secret question! Find your own answer or decrypt the answer saved in this file.
Two phases have been solved. Keep going!

[*] Phase 3
- Hint: Many cases make everything so confusing.
0 d 482
You've beaten another phase, that's great. What about the fourth one?

[*] Phase 4
- Hint: Let's dig in to recursive function :)
█

[*] Phase 3
- Hint: Many cases make everything so confusing.
1 i 793
You've beaten another phase, that's great. What about the fourth one?
```


Phase 4:

```

int __cdecl phase4(int a1)
{
    int result; // eax@6
    int v2; // [sp+Ch] [bp-1Ch]@1
    int v3; // [sp+10h] [bp-18h]@1
    int v4; // [sp+14h] [bp-14h]@5
    int v5; // [sp+18h] [bp-10h]@5
    int v6; // [sp+1Ch] [bp-Ch]@1

    v6 = __isoc99_sscanf(a1, "%d %d", &v3, &v2);
    if ( v6 != 2 || v3 < 0 || v3 > 14 )
        explode_bomb();
    v5 = 4;
    v4 = func4(v3, 0, 14);
    if ( v4 != v5 || (result = v2, v2 != v5) )
        explode_bomb();
    return result;
}

```

- Hàm **phase4** nhận một tham số đầu vào a1, sau đó sử dụng hàm **sscanf** để phân tích chuỗi ký tự được truyền thành hai số nguyên v3 và v2. Sau đó, hàm kiểm tra các điều kiện:

- Nếu hàm **sscanf** không phân tích được hai số nguyên hoặc v3 không nằm trong khoảng từ 0 đến 14, hàm **explode_bomb()** sẽ được gọi.
- Tiếp theo, hàm gọi hàm **func4** để thực hiện một loạt các tính toán. Nếu kết quả trả về từ **func4** không bằng 4 hoặc v2 không bằng 4, hàm **explode_bomb()** sẽ được gọi.

⇒ Giá trị của v2 là 4.

```

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 = 0;
        else
            result = 2 * func4(a1, v4 + 1, a3) + 1;
    }
    else
    {
        result = 2 * func4(a1, a2, v4 - 1);
    }
    return result;
}

```

- Hàm **func4** được gọi trong **phase4** để thực hiện các phép toán:
 - Tính toán v4 bằng cách lấy trung bình của a2 và a3.

Lab 5: Reverse Engineering (cont)

- Nếu $v4$ bằng $a1$, trả về 0.
 - Nếu $v4$ lớn hơn $a1$, gọi đệ quy hàm **func4** trên nửa phải của mảng.
 - Nếu $v4$ nhỏ hơn $a1$, gọi đệ quy hàm **func4** trên nửa trái của mảng.
- Trong trường hợp này, hàm **func4** được gọi với $a2 = 0$ và $a3 = 14$, tức là tìm kiếm được thực hiện trên một mảng từ 0 đến 14. Sau đây là code Python để tìm giá trị của $a1$ sao cho kết quả trả về là 4:

```
1  def func4(a1, a2, a3):
2      v4 = (a3 - a2) // 2 + a2
3      if v4 <= a1:
4          if v4 >= a1:
5              return 0
6          else:
7              return 2 * func4(a1, v4 + 1, a3) + 1
8      else:
9          return 2 * func4(a1, a2, v4 - 1)
10
11 def find_a1():
12     for a1 in range(15):
13         if func4(a1, 0, 14) == 4:
14             return a1
15
16 a1 = find_a1()
17 print("func4(a1, 0, 14) = 4 khi a1 =", a1)
18
```

```
● PS D:\Code\Python> python -u "d:\Code\Python\lab5-phase4"
func4(a1, 0, 14) = 4 khi a1 = 2
```

⇒ Với $v3$ vừa tìm được là 2 và $v2$ có giá trị cố định là 4 nên kết quả tìm được là: [2, 4].

Kết quả thực thi:

```
tnh47@Nhat: ~/UIT/Ky4/LTH1 x + v
tnh47@Nhat:~/UIT/Ky4/LTH1$ ./nt209-uit-bomb input.txt
Welcome to UIT's bomb lab.
You have to deactivate our bomb by solving 5 phases with the correct inputs consecutively, and otherwise the bomb will be blown up!

[*] Phase 1
- Hint: Numbers are always magical!
Good job! You've cleared the first phase!

[*] Phase 2
- Hint: You must answer your secret question! Find your own answer or decrypt the answer saved in this file.
Two phases have been solved. Keep going!

[*] Phase 3
- Hint: Many cases make everything so confusing.
You've beaten another phase, that's great. What about the fourth one?

[*] Phase 4
- Hint: Let's dig in to recursive function :)
2 4
Awesome! Only one phase left!
```

Phase 5:

```

int __cdecl phase5(int a1)
{
    int result; // eax@7
    int v2; // [sp+8h] [bp-20h]@1
    int v3; // [sp+Ch] [bp-1Ch]@1
    int v4; // [sp+10h] [bp-18h]@3
    int v5; // [sp+14h] [bp-14h]@1
    int v6; // [sp+18h] [bp-10h]@3
    int v7; // [sp+1Ch] [bp-Ch]@3

    v5 = __isoc99_sscanf(a1, "%d %d", &v3, &v2);
    if ( v5 <= 1 )
        explode_bomb();
    v3 &= 15u;
    v4 = v3;
    v7 = 0;
    v6 = 0;
    while ( v3 != 15 )
    {
        ++v7;
        v3 = array_3855[v3];
        v6 += v3;
    }
    if ( v7 != 9 || (result = v2, v6 != v2) )
        explode_bomb();
    return result;
}

```

- Hàm **phase5** nhận một tham số a1, chuyển đổi định dạng chuỗi thành hai giá trị nguyên v3 và v2 bằng cách sử dụng hàm **sscanf**. Nếu số lượng input <= 1 thì hàm **explode_bomb()** sẽ được gọi.
- v3 được đưa về dạng số nguyên không âm trong khoảng từ 0 đến 15 bằng cách thực hiện phép AND với 15.
- Khởi tạo v4 bằng v3.
- Khởi tạo v7 và v6 bằng 0.
- Bắt đầu một vòng lặp while v3 khác 15:
 - Tăng v7 lên 1 đơn vị.
 - Cập nhật v3 bằng phần tử thứ v3 của mảng array_3855.
 - Thêm v3 vào v6.
- Kiểm tra nếu v7 không bằng 9 hoặc nếu tổng các giá trị của v3 trong quá trình lặp không bằng v2, thì kích nổ.
- Điều kiện để bomb không nổ là v7 = 9 và v6 = v2. Vậy ta phải làm sao để vòng lặp này thực hiện được 9 lần để v4 có thể cộng dồn lên 9 và tính giá trị v6 sau 15 lần lặp đó để tìm v2 (input thứ 2). Đầu tiên kiểm tra các giá trị trong array_3855

Lab 5: Reverse Engineering (cont)

```
.data:0804B200 ; int array_3855[]  
.data:0804B200 array_3855 dd 10 ; DATA XREF: phase5+52↑r  
.data:0804B204 db 2  
.data:0804B205 db 0  
.data:0804B206 db 0  
.data:0804B207 db 0  
.data:0804B208 db 14  
.data:0804B209 db 0  
.data:0804B20A db 0  
.data:0804B20B db 0  
.data:0804B20C db 7  
.data:0804B20D db 0  
.data:0804B20E db 0  
.data:0804B20F db 0  
.data:0804B210 db 8  
.data:0804B211 db 0  
.data:0804B212 db 0  
.data:0804B213 db 0  
.data:0804B214 db 12  
.data:0804B215 db 0  
.data:0804B216 db 0  
.data:0804B217 db 0  
.data:0804B218 db 15  
.data:0804B219 db 0  
.data:0804B21A db 0  
.data:0804B21B db 0  
.data:0804B21C db 11  
.data:0804B21D db 0  
.data:0804B21E db 0  
.data:0804B21F db 0  
.data:0804B220 db 0  
.data:0804B221 db 0  
.data:0804B222 db 0  
.data:0804B223 db 0  
.data:0804B224 db 4  
.data:0804B225 db 0  
.data:0804B226 db 0  
.data:0804B227 db 0  
.data:0804B228 db 1  
.data:0804B229 db 0  
.data:0804B22A db 0  
.data:0804B22B db 0  
.data:0804B22C db 13  
.data:0804B22D db 0  
.data:0804B22E db 0  
.data:0804B22F db 0  
.data:0804B230 db 3  
.data:0804B231 db 0  
.data:0804B232 db 0  
.data:0804B233 db 0  
.data:0804B234 db 9  
.data:0804B235 db 0  
.data:0804B236 db 0  
.data:0804B237 db 0  
.data:0804B238 db 6  
.data:0804B239 db 0  
.data:0804B23A db 0  
.data:0804B23B db 0  
.data:0804B23C db 5  
.data:0804B23D db 0  
.data:0804B23E db 0  
.data:0804B23F db 0  
.data:0804B23F _data ends  
.data:0804B23F
```

- Địa chỉ gốc của array_3855 là ở 0x0804B200. Ta biết array_3855 có 16 phần tử, giá trị mỗi phần tử sẽ cách nhau 4 byte.
- array_3855 = [10, 2, 14, 7, 8, 12, 15, 11, 0, 4, 1, 13, 3, 9, 6, 5].

- Sau đây là code Python để tìm ra các cặp số thỏa điều kiện:

```

1  arr = [10, 2, 14, 7, 8, 12, 15, 11, 0, 4, 1, 13, 3, 9, 6, 5]
2
3  for i in range (0,100):
4      v3 = i & 15
5      count = 0
6      sum = 0
7      while v3 != 15:
8          v3 = arr[v3]
9          count += 1
10         sum += v3
11     if count == 9:
12         print (f"v3: {i} - v6: {count} - v7: {sum}")
13

```

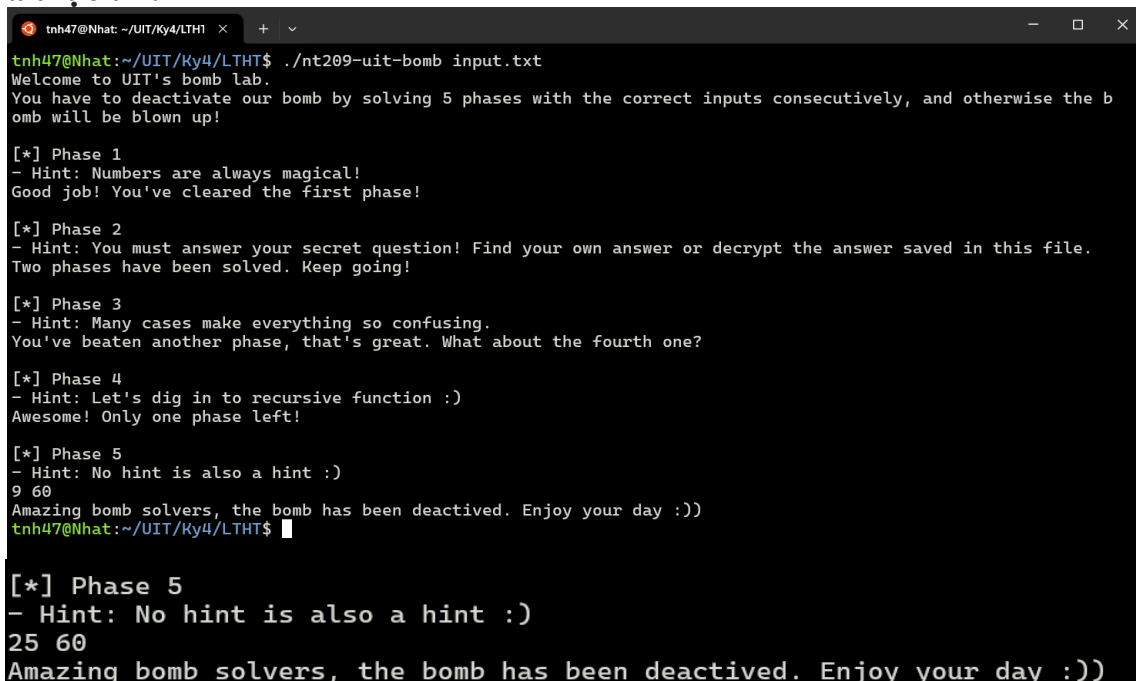
```

PS D:\Code\Python> python -u "d:\Code\Python\lab5-phase5"
v3: 9 - v6: 9 - v7: 60
v3: 25 - v6: 9 - v7: 60
v3: 41 - v6: 9 - v7: 60
v3: 57 - v6: 9 - v7: 60
v3: 73 - v6: 9 - v7: 60
v3: 89 - v6: 9 - v7: 60
PS D:\Code\Python>

```

Kết quả chạy đoạn script cho thấy v7 của ta có giá trị cố định là 60, và ta sẽ có vô số v3 thỏa mãn điều kiện của chương trình. Vì $v3 \text{ AND } 15$ cũng giống như $v3 \% 15$, nên v3 có lớn đến đâu thì cũng không thành vấn đề.

Kết quả thực thi:



```

tnh47@Nhat: ~/UIT/Ky4/LTH1
tnh47@Nhat:~/UIT/Ky4/LTH1$ ./nt209-uit-bomb input.txt
Welcome to UIT's bomb lab.
You have to deactivate our bomb by solving 5 phases with the correct inputs consecutively, and otherwise the bomb will be blown up!

[*] Phase 1
- Hint: Numbers are always magical!
Good job! You've cleared the first phase!

[*] Phase 2
- Hint: You must answer your secret question! Find your own answer or decrypt the answer saved in this file.
Two phases have been solved. Keep going!

[*] Phase 3
- Hint: Many cases make everything so confusing.
You've beaten another phase, that's great. What about the fourth one?

[*] Phase 4
- Hint: Let's dig in to recursive function :)
Awesome! Only one phase left!

[*] Phase 5
- Hint: No hint is also a hint :)
9 60
Amazing bomb solvers, the bomb has been deactivated. Enjoy your day :)
tnh47@Nhat:~/UIT/Ky4/LTH1$

[*] Phase 5
- Hint: No hint is also a hint :)
25 60
Amazing bomb solvers, the bomb has been deactivated. Enjoy your day :)

```