

# BÁO CÁO THỰC HÀNH

Môn học: **Lập trình hệ thống (NT209)**

**Lab 4 – Kỹ thuật dịch ngược – Nâng cao**

GVHD: *Đỗ Thị Thu Hiền*

## 1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT209.N21.ANTT.1

STT	Họ và tên	MSSV	Email
1	Nguyễn Thị Minh Châu	21520645	21520645@gm.uit.edu.vn
2			
3			

## 2. NỘI DUNG THỰC HIỆN:<sup>1</sup>

STT	Công việc	Kết quả tự đánh giá
1	Pha 1	Hoàn thành
2	Pha 2	Hoàn thành
3	Pha 3	Hoàn thành
4	Pha 4	Hoàn thành
5	Pha 5	Hoàn thành
6	Pha 6	Hoàn thành
7	Pha bí mật (bonus)	Hoàn thành

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

---

<sup>1</sup> Ghi nội dung công việc, yêu cầu trong bài Thực hành

## BÁO CÁO CHI TIẾT

Phương pháp phân tích: Phân tích tĩnh/Remote Debug (chụp hình ảnh minh chứng)

### 1. Pha 1

```

.text:08048C4B phase_1      proc near                               ; CODE XREF: main+CD↑p
.text:08048C4B
.text:08048C4B arg_0        = dword ptr 8
.text:08048C4B
.text:08048C4B      push    ebp
.text:08048C4C      mov     ebp, esp
.text:08048C4E      sub     esp, 8
.text:08048C51      sub     esp, 8
.text:08048C54      push    offset aVerbosityLeads ; "Verbosity leads to unclear, inarticulat"...
.text:08048C59      push    [ebp+arg_0]
.text:08048C5C      call    strings_not_equal
.text:08048C61      add     esp, 10h
.text:08048C64      test    eax, eax
.text:08048C66      jz      short loc_8048C6D
.text:08048C68      call    explode_bomb
.text:08048C6D ; -----

```

Đây là đoạn mã pha 1

Đoạn code gọi hàm strings\_not\_equal => input của ta là 1 chuỗi kí tự

Ở trên có câu lệnh push offset aVerbosityLeads => đây là chuỗi có sẵn trong bộ nhớ

```

.rodata:0804A58E      align 10h
.rodata:0804A590 aVerbosityLeads db 'Verbosity leads to unclear, inarticulate things.',0
.rodata:0804A590      ; DATA XREF: phase_1+9↑o

```

Tiếp theo ta có lệnh test eax,eax => để gán cờ Zero Flag. Tiếp theo là lệnh jz => nhảy đến lệnh loc\_8048C6D nếu = 0 => bỏ qua được gọi hàm explode\_bomb => nếu so sánh 2 chuỗi giống nhau sẽ là đáp án đúng

```

7      test    eax, eax
6      jz      short loc_8048C6D
8      call    explode_bomb
D ; -----
D
D loc_8048C6D:      ; CODE XREF: phase_1+1B↑j
D      nop
E      leave
F      retn

```

Và ta có kết quả là chuỗi "Verbosity leads to unclear, inarticulate things."

```

.rodata:0804A58E      align 10h
.rodata:0804A590 aVerbosityLeads db 'Verbosity leads to unclear, inarticulate things.',0
.rodata:0804A590      ; DATA XREF: phase_1+9↑o

```

Thử lại đáp án

```

zeri@zeri:~/LHTHLab4$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Verbosity leads to unclear, inarticulate things.
Phase 1 defused. How about the next one?

```

## 2. Pha 2

```

.text:08048C70
.text:08048C70          push    ebp
.text:08048C71          mov     ebp, esp
.text:08048C73          sub     esp, 38h
.text:08048C76          mov     eax, [ebp+arg_0]
.text:08048C79          mov     [ebp+var_2C], eax
.text:08048C7C          mov     eax, large gs:14h
.text:08048C82          mov     [ebp+var_C], eax
.text:08048C85          xor     eax, eax
.text:08048C87          sub     esp, 8
.text:08048C8A          lea     eax, [ebp+var_24]
.text:08048C8D          push    eax
.text:08048C8E          push    [ebp+var_2C]
.text:08048C91          call    read_six_numbers
.text:08048C96          add     esp, 10h
.text:08048C99          mov     eax, [ebp+var_24]
.text:08048C9C          test    eax, eax
.text:08048C9E          jnz     short loc_8048CA8
.text:08048CA0          mov     eax, [ebp+var_20]
.text:08048CA3          cmp     eax, 1
.text:08048CA6          jz      short loc_8048CAD
.text:08048CA8          loc_8048CA8:
.text:08048CA8          call    explode_bomb ; CODE XREF: phase_2+2E↑j
.text:08048CAD          : -----

```

Đây là đoạn code của pha 2

Ta thấy có gọi hàm read\_six\_numbers => đọc vào 6 số input

```

.text:0804921D          push    ebp
.text:0804921E          mov     ebp, esp
.text:08049220          push    esi
.text:08049221          push    ebx
.text:08049222          sub     esp, 10h
.text:08049225          mov     eax, [ebp+arg_4]
.text:08049228          lea     esi, [eax+14h]
.text:0804922B          mov     eax, [ebp+arg_4]
.text:0804922E          lea     ebx, [eax+10h]
.text:08049231          mov     eax, [ebp+arg_4]
.text:08049234          lea     ecx, [eax+0Ch]
.text:08049237          mov     eax, [ebp+arg_4]
.text:0804923A          lea     edx, [eax+8]
.text:0804923D          mov     eax, [ebp+arg_4]
.text:08049240          add     eax, 4
.text:08049243          push    esi
.text:08049244          push    ebx
.text:08049245          push    ecx
.text:08049246          push    edx
.text:08049247          push    eax
.text:08049248          push    [ebp+arg_4]
.text:0804924B          push    offset aDDDDDD ; "%d %d %d %d %d %d"
.text:08049250          push    [ebp+arg_0]
.text:08049253          call    ___isoc99_sscanf
.text:08049258          add     esp, 20h
.text:0804925B          mov     [ebp+var_C], eax

```

Đây là đoạn mã nhập vào theo thứ tự các số

```

1 int __cdecl phase_2(int a1)
2 {
3     signed int i; // [sp+10h] [bp-28h]@4
4     int v3; // [sp+14h] [bp-24h]@1
5     int v4; // [sp+18h] [bp-20h]@2
6     int v5; // [sp+2Ch] [bp-Ch]@1
7
8     v5 = *MK_FP(__GS__, 20);
9     read_six_numbers(a1, (int)&v3);
10    if ( v3 || v4 != 1 )
11        explode_bomb();
12    for ( i = 2; i <= 5; ++i )
13    {
14        if ( *(&v3 + i) != *(&v3 + i - 2) + *(&v3 + i - 1) )
15            explode_bomb();
16    }
17    return *MK_FP(__GS__, 20) ^ v5;
18 }

```

Đây là mã giả của pha 2

Ở dòng 10, điều kiện để bomb nổ là  $v3 \neq 0$  và  $v4 \neq 1 \Rightarrow v3 = 0$  và  $v4 = 1$

Ở vòng lặp for, ta có điều kiện để bomb không nổ là  $v[3+i] = v[3+i-2] + v[3+i-1]$  (nghĩa là từ số thứ 3 thì nó sẽ bằng tổng của 2 số liền trước nó)

Vì vậy ta có :

$v3 = 0$

$v4 = 1$

$v5 = 1$

$v6 = 2$

$v7 = 3$

$v8 = 5$

Kết quả ta cần nhập vào là **0 1 1 2 3 5**

Thử lại kết quả:

```

Phase 1 defused. How about the next one?
0 1 1 2 3 5
That's number 2. Keep going!

```

### 3. Pha 3

```

.text:08048CFA
.text:08048CFA      push    ebp
.text:08048CFB      mov     ebp, esp
.text:08048CFD      sub     esp, 38h
.text:08048D00      mov     eax, [ebp+arg_0]
.text:08048D03      mov     [ebp+var_2C], eax
.text:08048D06      mov     eax, large gs:14h
.text:08048D0C      mov     [ebp+var_C], eax
.text:08048D0F      xor     eax, eax
.text:08048D11      mov     [ebp+var_14], 0
.text:08048D18      mov     [ebp+var_10], 0
.text:08048D1F      lea     eax, [ebp+var_18]
.text:08048D22      push    eax
.text:08048D23      lea     eax, [ebp+var_1C]
.text:08048D26      push    eax
.text:08048D27      push    offset aDD      ; "%d %d"
.text:08048D2C      push    [ebp+var_2C]
.text:08048D2F      call    ___isoc99_sscanf
.text:08048D34      add     esp, 10h
.text:08048D37      mov     [ebp+var_10], eax
.text:08048D3A      cmp     [ebp+var_10], 1
.text:08048D3E      jg      short loc_8048D45
.text:08048D40      call    explode_bomb
.text:08048D45

```

Đây là code của pha 3

```

1 int __cdecl phase_3(int a1)
2 {
3     int result; // eax@16
4     int v2; // [sp+1Ch] [bp-1Ch]@1
5     int v3; // [sp+20h] [bp-18h]@1
6     int v4; // [sp+24h] [bp-14h]@1
7     int v5; // [sp+28h] [bp-10h]@1
8     int v6; // [sp+2Ch] [bp-Ch]@1
9
10    v6 = *MK_FP(__GS__, 20);
11    v4 = 0;
12    v5 = 0;
13    v5 = __isoc99_sscanf(a1, "%d %d", &v2, &v3);
14    if ( v5 <= 1 )
15        explode_bomb();
16    switch ( v2 )
17    {
18        case 0:
19            v4 += 873;
20            goto LABEL_5;
21        case 1:
22    LABEL_5:
23            v4 -= 217;
24            goto LABEL_6;
25        case 2:
26    LABEL_6:
27            v4 += 118;
28            goto LABEL_7;
29        case 3:
30    LABEL_7:
31            v4 -= 799;
32            goto LABEL_8;

```

```

30 LABEL_7:
31     v4 -= 799;
32     goto LABEL_8;
33     case 4:
34 LABEL_8:
35     v4 += 799;
36     goto LABEL_9;
37     case 5:
38 LABEL_9:
39     v4 -= 799;
40     goto LABEL_10;
41     case 6:
42 LABEL_10:
43     v4 += 799;
44     break;
45     case 7:
46     break;
47     default:
48     explode_bomb();
49     return result;
50 }
51 v4 -= 799;
52 if ( v2 > 5 || v4 != v3 )
53     explode_bomb();
54 return *MK_FP(__GS__, 20) ^ v6;
55 }

```

Đây là mã giả của pha 3

Theo như dòng 13 thì chương trình gọi đến cấu trúc “%d %d” sau đó gọi hàm scanf nên input ta cần nhập là 2 số nguyên.

Các biến ban đầu được khai báo như sau: v4 = 0, v5 = 0. Cần nhập vào input (v5) là v2, v3. Điều kiện (1) bom không nổ: v5 > 1.

Hàm switch với giá trị các case là giá trị của v2

Ta thấy điều kiện để bomb không nổ (2) là v2 <= 5 và các case trong hàm switch là từ 0-7. Ta chọn v2 từ 0-5, thế v2 vào switch ta được các giá trị của v4:

0	-25
1	-898
2	-681
3	-799
4	0
5	-799

Thử kết quả với cặp 0 -25:

```
0 -25  
Halfway there!
```

Thử kết quả với cặp 1 -898:

```
1 -898  
Halfway there!
```

Thử kết quả với cặp 2 -681:

```
2 -681  
Halfway there!
```

Thử kết quả với cặp 3 -799:

```
3 -799  
Halfway there!
```

Thử kết quả với cặp 4 0:

```
4 0  
Halfway there!
```

Thử kết quả với cặp 5 -799:

```
5 -799  
Halfway there!
```

#### **4. Pha 4**



```

.text:08048E31
.text:08048E31          push    ebp
.text:08048E32          mov     ebp, esp
.text:08048E34          sub     esp, 38h
.text:08048E37          mov     eax, [ebp+arg_0]
.text:08048E3A          mov     [ebp+var_2C], eax
.text:08048E3D          mov     eax, large gs:14h
.text:08048E43          mov     [ebp+var_C], eax
.text:08048E46          xor     eax, eax
.text:08048E48          lea     eax, [ebp+var_1C]
.text:08048E4B          push    eax
.text:08048E4C          lea     eax, [ebp+var_20]
.text:08048E4F          push    eax
.text:08048E50          push    offset aDD          ; "%d %d"
.text:08048E55          push    [ebp+var_2C]
.text:08048E58          call    ___isoc99_sscanf
.text:08048E5D          add     esp, 10h
.text:08048E60          mov     [ebp+var_18], eax
.text:08048E63          cmp     [ebp+var_18], 2
.text:08048E67          jnz     short loc_8048E78
.text:08048E69          mov     eax, [ebp+var_20]
.text:08048E6C          test    eax, eax
.text:08048E6E          js      short loc_8048E78
.text:08048E70          mov     eax, [ebp+var_20]
.text:08048E73          cmp     eax, 0Eh
.text:08048E76          jle     short loc_8048E7D
.text:08048E78          loc_8048E78:                ; CODE XREF: phase_4+36↑j
.text:08048E78                ; phase_4+3D↑j
.text:08048E78          call    explode_bomb

```

Đây là code của pha 4

```

1 int __cdecl phase_4(int a1)
2 {
3     int v2; // [sp+18h] [bp-20h]@1
4     int v3; // [sp+1Ch] [bp-1Ch]@1
5     int v4; // [sp+20h] [bp-18h]@1
6     int v5; // [sp+24h] [bp-14h]@5
7     int v6; // [sp+28h] [bp-10h]@5
8     int v7; // [sp+2Ch] [bp-Ch]@1
9
10    v7 = *MK_FP(__GS__, 20);
11    v4 = ___isoc99_sscanf(a1, "%d %d", &v2, &v3);
12    if ( v4 != 2 || v2 < 0 || v2 > 14 )
13        explode_bomb();
14    v5 = 27;
15    v6 = func4(v2, 0, 14);
16    if ( v6 != v5 || v3 != v5 )
17        explode_bomb();
18    return *MK_FP(__GS__, 20) ^ v7;
19 }

```

Đây là mã giả của pha 4

Yêu cầu input cho phase4: (v4) hai số nguyên v2, v3

Điều kiện (1) để bom không nổ:  $(v4 == 2 \ \&\& \ v2 \geq 0 \ \&\& \ v2 \leq 14)$ .

Khai báo  $v5 = 27$  và giá trị  $v6$  tính theo hàm  $\text{func4}(v2, 0, 14)$  với code  $\text{func4}$  như sau:

```

1 int __cdecl func4(int a1, int a2, int a3)
2 {
3     int result; // eax@2
4     int v4; // [sp+Ch] [bp-Ch]@1
5
6     v4 = (a3 - a2) / 2 + a2;
7     if ( v4 <= a1 )
8     {
9         if ( v4 >= a1 )
10            result = (a3 - a2) / 2 + a2;
11        else
12            result = func4(a1, v4 + 1, a3) + v4;
13    }
14    else
15    {
16        result = func4(a1, a2, v4 - 1) + v4;
17    }
18    return result;
19 }

```

Trong  $\text{func4}$ : các tham số truyền vào là  $a1, a2, a3$  tương ứng với  $v2, 0, 14$ . Ở cuối, trả về  $\text{result}$  tương ứng với giá trị  $v6$ .

Ta viết chương trình tương tự đoạn mã trên

```

1  #include <iostream>
2
3  using namespace std;
4
5  int func4(int a1, int a2, int a3)
6  {
7      int result;
8      int v4;
9      v4 = (a3 - a2)/2 + a2;
10     if (v4 <= a1)
11     {
12         if (v4 >= a1) // kết hợp điều kiện => v4 thuộc [0;14]
13         {
14             result = (a3 - a2)/2 + a2;
15         }
16         else
17         {
18             result = func4(a1, v4 + 1, a3) + v4;
19         }
20     }
21     else
22     {
23         result = func4(a1, a2, v4 - 1) + v4;
24     }
25     return result;
26 }
27 int main()
28 {
29     for (int v2 = 0; v2 <= 14; v2++)
30     {
31         cout << "v2 = " << v2 << " ";
32         cout << "v6 = " << func4(v2,0,14) << endl;
33     }
34
35     return 0;
36 }

```

Giải thích dòng 12: Ở lệnh if (dòng 12) điều kiện thực thi là  $(v4 \geq a1)$  kết hợp với lệnh if (dòng 10) điều kiện if để thực thi if dòng 12 là  $(v4 \leq a1) \rightarrow v4 = a1$ . Mà  $a1 = v2 \rightarrow v4$  thuộc đoạn giá trị  $[0; 14]$  (điều kiện (1) ở giá trị  $v2$ ).

Ta chạy thì ra kết quả :

```

v2 = 0 v6 = 11
v2 = 1 v6 = 11
v2 = 2 v6 = 13
v2 = 3 v6 = 10
v2 = 4 v6 = 19
v2 = 5 v6 = 15
v2 = 6 v6 = 21
v2 = 7 v6 = 7
v2 = 8 v6 = 35
v2 = 9 v6 = 27
v2 = 10 v6 = 37
v2 = 11 v6 = 18
v2 = 12 v6 = 43
v2 = 13 v6 = 31
v2 = 14 v6 = 45

```

Điều kiện (2) để bom không nổ: (v6 == v5 && v3 == v5). Suy ra v6 = v3.

Vậy input sẽ là trường hợp 9 27

```

9 27
So you got that one. Try this one.

```

## 5. Pha 5

```

.text:00000000
.text:08048EC3
.text:08048EC4
.text:08048EC6
.text:08048EC9
.text:08048ECC
.text:08048ECF
.text:08048ED5
.text:08048ED8
.text:08048EDA
.text:08048EDD
.text:08048EDE
.text:08048EE1
.text:08048EE2
.text:08048EE7
.text:08048EEA
.text:08048EEF
.text:08048EF2
.text:08048EF5
.text:08048EF9
.text:08048EFB
.text:08048F00 ;
.text:08048F0A
        push    ebp
        mov     ebp, esp
        sub     esp, 38h
        mov     eax, [ebp+arg_0]
        mov     [ebp+var_2C], eax
        mov     eax, large gs:14h
        mov     [ebp+var_C], eax
        xor     eax, eax
        lea     eax, [ebp+var_20]
        push    eax
        lea     eax, [ebp+var_24]
        push    eax
        push    offset aDD ; "%d %d"
        push    [ebp+var_2C]
        call    ___isoc99_sscanf
        add     esp, 10h
        mov     [ebp+var_14], eax
        cmp     [ebp+var_14], 1
        jg      short loc_8048F00
        call    explode_bomb

```

Đây là code pha 5

```

1 int __cdecl phase_5(int a1)
2 {
3     int v2; // [sp+14h] [bp-24h]@1
4     int v3; // [sp+18h] [bp-20h]@1
5     int v4; // [sp+1Ch] [bp-1Ch]@3
6     int v5; // [sp+20h] [bp-18h]@3
7     int v6; // [sp+24h] [bp-14h]@1
8     int v7; // [sp+28h] [bp-10h]@3
9     int v8; // [sp+2Ch] [bp-Ch]@1
10
11     v8 = *MK_FP(__GS__, 20);
12     v6 = __isoc99_sscanf(a1, "%d %d", &v2, &v3);
13     if ( v6 <= 1 )
14         explode_bomb();
15     v2 &= 0xFu;
16     v7 = v2;
17     v4 = 0;
18     v5 = 0;
19     while ( v2 != 15 )
20     {
21         ++v4;
22         v2 = array_2705[v2];
23         v5 += v2;
24     }
25     if ( v4 != 15 || v5 != v3 )
26         explode_bomb();
27     return *MK_FP(__GS__, 20) ^ v8;
28 }

```

Đây là mã giả của pha 5

- Yêu cầu input nhập vào 2 số v2 và v3 (dòng 12 của đoạn code)
- Khởi tạo các giá trị: v4 = 0, v5 = 0;
- Điều kiện để bomb không nổ (1) là: v4=15 và v5 = v3
- Để v4 = 15, ta phải chạy 15 vòng trong while => giá trị cuối cùng của v2 trong vòng while là 15

```

.data:0804D1C0 array_2705 dd 0Ah
.data:0804D1C4 db 2
.data:0804D1C5 db 0
.data:0804D1C6 db 0
.data:0804D1C7 db 0
.data:0804D1C8 db 0Eh
.data:0804D1C9 db 0
.data:0804D1CA db 0
.data:0804D1CB db 0
.data:0804D1CC db 7
.data:0804D1CD db 0
.data:0804D1CE db 0
.data:0804D1CF db 0
.data:0804D1D0 db 8
.data:0804D1D1 db 0
.data:0804D1D2 db 0
.data:0804D1D3 db 0
.data:0804D1D4 db 0Ch
.data:0804D1D5 db 0
.data:0804D1D6 db 0
.data:0804D1D7 db 0
.data:0804D1D8 db 0Fh
.data:0804D1D9 db 0
.data:0804D1DA db 0
.data:0804D1DB db 0
.data:0804D1DC db 0Bh
.data:0804D1DD db 0
.data:0804D1DE db 0
.data:0804D1DF db 0
.data:0804D1E0 db 0

```

- Trong hàm array\_2705 ta thấy giá trị theo thứ tự như sau:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
array_2705	10	2	14	7	8	12	15	11	0	4	1	13	3	9	6	5

- Biên dịch lại hàm trên ta có:

```

1  #include <iostream>
2
3  using namespace std;
4
5  int main(){
6  int array_2705[] = {10,2 ,14, 7, 8 ,12 ,15 ,11, 0, 4, 1, 13, 3, 9, 6, 5};
7  int v2, v4 = 0, v5= 0;
8
9  for(int i = 0; i <15; i++) {
10     v4 = 0;
11     v5 = 0;
12     v2 = i;
13     cout << "v2 = " << v2;
14     while(v2 != 15) {
15         ++v4;
16         v2 = array_2705[v2];
17         v5 += v2;
18     }
19     cout << "v5 = " << v5;
20     cout << "v4 = " << v4 << "\n";
21 };
22
23 }

```

- Chạy chương trình ta được kết quả:

```

[Running] cd "C:\Users\admin\AppData
v2 = 0v5 = 48v4 = 6
v2 = 1v5 = 37v4 = 4
v2 = 2v5 = 35v4 = 3
v2 = 3v5 = 100v4 = 13
v2 = 4v5 = 56v4 = 8
v2 = 5v5 = 115v4 = 15
v2 = 6v5 = 15v4 = 1
v2 = 7v5 = 93v4 = 12
v2 = 8v5 = 48v4 = 7
v2 = 9v5 = 60v4 = 9
v2 = 10v5 = 38v4 = 5
v2 = 11v5 = 82v4 = 11
v2 = 12v5 = 103v4 = 14
v2 = 13v5 = 69v4 = 10
v2 = 14v5 = 21v4 = 2

```

- Chỉ có 1 trường hợp thỏa điều kiện để bom không nổ là cặp: v2 = 5, v5 = 115

- Kết quả chạy được:

```
5 115
Good work! On to the next...
```

## 6. Pha 6

```
.text:08048F65
.text:08048F65      push    ebp
.text:08048F66      mov     ebp, esp
.text:08048F68      sub     esp, 68h
.text:08048F6B      mov     eax, [ebp+arg_0]
.text:08048F6E      mov     [ebp+var_5C], eax
.text:08048F71      mov     eax, large gs:14h
.text:08048F77      mov     [ebp+var_C], eax
.text:08048F7A      xor     eax, eax
.text:08048F7C      mov     [ebp+var_40], offset node1
.text:08048F83      sub     esp, 8
.text:08048F86      lea     eax, [ebp+var_3C]
.text:08048F89      push    eax
.text:08048F8A      push    [ebp+var_5C]
.text:08048F8D      call    read_six_numbers
.text:08048F92      add     esp, 10h
.text:08048F95      mov     [ebp+var_48], 0
.text:08048F9C      jmp     short loc_8048FEA
.text:08048F9E      -----
```

- Đây là code của pha 6



```

1 int __cdecl phase_6(int a1)
2 {
3     int v2; // [sp+1Ch] [bp-4Ch]@13
4     int v3; // [sp+1Ch] [bp-4Ch]@18
5     int v4; // [sp+1Ch] [bp-4Ch]@21
6     signed int i; // [sp+20h] [bp-48h]@1
7     signed int k; // [sp+20h] [bp-48h]@12
8     signed int m; // [sp+20h] [bp-48h]@18
9     signed int n; // [sp+20h] [bp-48h]@21
10    int j; // [sp+24h] [bp-44h]@5
11    int l; // [sp+24h] [bp-44h]@13
12    int v11; // [sp+28h] [bp-40h]@18
13    int v12[6]; // [sp+2Ch] [bp-3Ch]@1
14    int v13[6]; // [sp+44h] [bp-24h]@16
15    int v14; // [sp+5Ch] [bp-Ch]@1
16
17    v14 = *MK_FP(__GS__, 20);
18    read_six_numbers(a1, |(int)v12);
19    for ( i = 0; i <= 5; ++i )
20    {
21        if ( v12[i] <= 0 || v12[i] > 6 )
22            explode_bomb();
23        for ( j = i + 1; j <= 5; ++j )
24        {
25            if ( v12[i] == v12[j] )
26                explode_bomb();
27        }
28    }
29    for ( k = 0; k <= 5; ++k )
30    {
31        v2 = (int)&node1;
32        for ( l = 1; v12[k] > 1; ++l )

```

```

29 for ( k = 0; k <= 5; ++k )
30 {
31     v2 = (int)&node1;
32     for ( l = 1; v12[k] > 1; ++l )
33         v2 = *(_DWORD *)(v2 + 8);
34     v13[k] = v2;
35 }
36 v11 = v13[0];
37 v3 = v13[0];
38 for ( m = 1; m <= 5; ++m )
39 {
40     *(_DWORD *)(v3 + 8) = v13[m];
41     v3 = *(_DWORD *)(v3 + 8);
42 }
43 *(_DWORD *)(v3 + 8) = 0;
44 v4 = v11;
45 for ( n = 0; n <= 4; ++n )
46 {
47     if ( *(_DWORD *)v4 > **(_DWORD **)(v4 + 8) )
48         explode_bomb();
49     v4 = *(_DWORD *)(v4 + 8);
50 }
51 return *MK_FP(__GS__, 20) ^ v14;
52 }

```

- Đây là mã giả của pha 6
- Ta thấy rằng input của bài này vẫn là 6 số nguyên. Điều kiện là các số đều thuộc đoạn [0;6] (dòng 21-22) và các số phải khác nhau (dòng 23-27)

```

.text:0004909F ; -----
.text:0804909F
.text:0804909F loc_804909F: ; CODE XREF: phase_6+16D↓j
.text:0804909F     mov     eax, [ebp+var_4C]
.text:080490A2     mov     edx, [eax]
.text:080490A4     mov     eax, [ebp+var_4C]
.text:080490A7     mov     eax, [eax+8]
.text:080490AA     mov     eax, [eax]
.text:080490AC     cmp     edx, eax
.text:080490AE     jge     short loc_80490B5
.text:080490B0     call    explode_bomb
.text:080490B5 ; -----

```

- Phân tích dòng 46-50 như trên.

Tóm tắt: Sử dụng 2 thanh ghi %eax và %edx để lưu các giá trị. Thanh ghi %eax lưu địa chỉ ở %ebp - 0x4C sau đó gán giá trị ở %eax cho %edx. Sau đó, lưu giá trị tại địa chỉ %ebp - 0x8 vào %eax; và thực hiện so sánh giá trị hiện lưu ở %eax và %edx. Nếu giá trị %edx >= giá trị %eax thì tiếp tục chương trình. Ngược lại, bom nổ

- Phân tích dòng 29-35 xét vòng lặp for, ta xem địa chỉ của cái node từ 1 đến 6 như bên dưới.

```

.data:00000000
.data:0804D0C4
.data:0804D0C4 node6
.data:0804D0C5
.data:0804D0C6
.data:0804D0C7
.data:0804D0C8
.data:0804D0C9
.data:0804D0CA
.data:0804D0CB
.data:0804D0CC
.data:0804D0CD
.data:0804D0CE
.data:0804D0CF
.data:0804D0D0
.data:0804D0D0 node5
.data:0804D0D1
.data:0804D0D2
.data:0804D0D3
.data:0804D0D4
.data:0804D0D5
.data:0804D0D6
.data:0804D0D7
.data:0804D0D8
.data:0804D0D9
.data:0804D0DA

public node6
db 7Bh ; {
db 0
db 0
db 0
db 6
db 0
db 0
db 0
db 0
db 0
db 0
db 0
db 0
db 0
db 0
public node5
db 15h
db 3
db 0
db 0
db 5
db 0
db 0
db 0
db 0
db 0C4h ; -
db 0D0h ; -
db 4

```

```

• .data:0804D0D3      db      0
• .data:0804D0D4      db      5
• .data:0804D0D5      db      0
• .data:0804D0D6      db      0
• .data:0804D0D7      db      0
• .data:0804D0D8      db 0C4h ; -
• .data:0804D0D9      db 0D0h ; -
• .data:0804D0DA      db      4
• .data:0804D0DB      db      8
• .data:0804D0DC      public node4
• .data:0804D0DC node4 db 0A7h ; 9
• .data:0804D0DD      db      0
• .data:0804D0DE      db      0
• .data:0804D0DF      db      0
• .data:0804D0E0      db      4
• .data:0804D0E1      db      0
• .data:0804D0E2      db      0
• .data:0804D0E3      db      0
• .data:0804D0E4      db 0D0h ; -
• .data:0804D0E5      db 0D0h ; -
• .data:0804D0E6      db      4
• .data:0804D0E7      db      8
• .data:0804D0E8      | public node3
• .data:0804D0E8 node3 db 78h ; x
• .data:0804D0E9      db      2
• .data:0804D0EA      db      0
• .data:0804D0EB      db      0
• .data:0804D0EC      db      3
• .data:0804D0ED      db      0
• .data:0804D0EE      db      0
• .data:0804D0EF      db      0
• .data:0804D0F0      db 0DCh ; -

```

```

.data:0804D0F2      db      4
.data:0804D0F3      db      8
.data:0804D0F4      public node2
.data:0804D0F4      node2      db      93h ; ô
.data:0804D0F5      db      1
.data:0804D0F6      db      0
.data:0804D0F7      db      0
.data:0804D0F8      db      2
.data:0804D0F9      db      0
.data:0804D0FA      db      0
.data:0804D0FB      db      0
.data:0804D0FC      db      0E8h ; F
.data:0804D0FD      db      0D0h ; -
.data:0804D0FE      db      4
.data:0804D0FF      db      8
.data:0804D100      public node1
.data:0804D100      node1      db      76h ; v          ; DATA XREF: phase_6+17↑o
.data:0804D101      db      1
.data:0804D102      db      0
.data:0804D103      db      0
.data:0804D104      db      1
.data:0804D105      db      0
.data:0804D106      db      0
.data:0804D107      db      0
.data:0804D108      db      0F4h ; (
.data:0804D109      db      0D0h ; -
.data:0804D10A      db      4
.data:0804D10B      db      8
.data:0804D10C      public n48

```

- Node trước có lưu trữ địa chỉ của các node tiếp theo, đây là 1 danh sách liên kết đơn. Tìm và lưu trữ các node trong danh sách liên kết trên vào mảng v13. Gán v11 = v13[0] và v3 = v13[0].
- Vòng lặp for thứ hai để duyệt qua tất cả các node trong danh sách liên kết
- Vòng lặp for cuối cùng để duyệt điều kiện sao cho bomb không nổ
- Ta có giá trị lưu ở các node như sau:

Node1: 0x176

Node2: 0x193

Node3: 0x278

Node4: 0x0A7

Node5: 0x315

Node6: 0x07B

- Ở đây ta có điều kiện duyệt để bomb không nổ là giá trị trong các node phải được sắp xếp theo thứ tự tăng dần. Do đó t xếp được node6 – node4 – node1 – node2 – node3 – node5
- Vậy input là 6 4 1 2 3 5

- Thử lại ta được kết quả đúng

```
zer1@zer1:~/LHTHlab4$ ./bomb input.txt
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
That's number 2. Keep going!
Halfway there!
So you got that one. Try this one.
Good work! On to the next...
Congratulations! You've defused the bomb!
zer1@zer1:~/LHTHlab4$
```

```
Open  [+]
```

```
*input.txt
~/LHTHlab4
```

```
1 Verbosity leads to unclear, inarticulate things.
2 0 1 1 2 3 5
3 4 0
4 9 27
5 5 115
6 6 4 1 2 3 55
```

## 7. Pha bí mật (Bonus)

```
.text:08049123      | push    ebp
.text:08049124      | mov     ebp, esp
.text:08049126      | sub     esp, 18h
.text:08049129      | call    read_line
.text:0804912E      | mov     [ebp+nptr], eax
.text:08049131      | sub     esp, 0Ch
.text:08049134      | push    [ebp+nptr]          ; nptr
.text:08049137      | call    _atoi
.text:0804913C      | add     esp, 10h
.text:0804913F      | mov     [ebp+var_10], eax
.text:08049142      | cmp     [ebp+var_10], 0
.text:08049146      | jle     short loc_8049151
.text:08049148      | cmp     [ebp+var_10], 3E9h
.text:0804914F      | jle     short loc_8049156
.text:08049151      | loc_8049151:                ; CODE XREF: secret_phase+23↑j
.text:08049151      | call    explode_bomb
.text:08049156
```

- Đây là code của phase bí mật
- Ta thấy sau khi nhập đúng input phase6, file bomb không nhận thêm input cho phase nào nữa, cần tìm cách để vào được secret\_phase.

```

IDA View-A  Pseudocode-C  Pseudocode-B  Pseudocode-A  Structur
. text:0804954E      jnz     short loc_80495C7
. text:08049550      sub     esp, 0Ch
. text:08049553      lea     eax, [ebp+var_5C]
. text:08049556      push    eax
. text:08049557      lea     eax, [ebp+var_64]
. text:0804955A      push    eax
. text:0804955B      lea     eax, [ebp+var_68]
. text:0804955E      push    eax
. text:0804955F      push    offset aDDS      ; "%d %d %s"
. text:08049564      push    offset unk_804D530
. text:08049569      call    ___isoc99_sscanf
. text:0804956E      add     esp, 20h
. text:08049571      mov     [ebp+var_60], eax
. text:08049574      cmp     [ebp+var_60], 3
. text:08049578      jnz     short loc_80495B7
. text:0804957A      sub     esp, 8
. text:0804957D      push    offset aDreuil   ; "DrEvil"
. text:08049582      lea     eax, [ebp+var_5C]
. text:08049585      push    eax
. text:08049586      call    strings_not_equal
. text:0804958B      add     esp, 10h
. text:0804958E      test    eax, eax
. text:08049590      jnz     short loc_80495B7
. text:08049592      sub     esp, 0Ch
. text:08049595      push    offset aCursesYouUeFou ; "Curse
. text:0804959A      call    _puts
. text:0804959F      add     esp, 10h
. text:080495A2      sub     esp, 0Ch
. text:080495A5      push    offset aButFindingItAn ; "But f
. text:080495AA      call    _puts
. text:080495AF      add     esp, 10h
. text:080495B2      call    secret_phase

```

- Thấy rằng trong hàm phase\_defused có hàm secret\_phase

```

1 int phase_defused()
2 {
3     char v1; // [sp+0h] [bp-68h]@2
4     char v2; // [sp+4h] [bp-64h]@2
5     int v3; // [sp+8h] [bp-60h]@2
6     char v4; // [sp+Ch] [bp-5Ch]@2
7     int v5; // [sp+5Ch] [bp-Ch]@1
8
9     v5 = *MK_FP(__GS__, 20);
10    if ( num_input_strings == 6 )
11    {
12        v3 = __isoc99_sscanf(&unk_804D530, "%d %d %s", &v1, &v2, &v4);
13        if ( v3 == 3 && !strings_not_equal(&v4, "DrEvil") )
14        {
15            puts("Curses, you've found the secret phase!");
16            puts("But finding it and solving it are quite different...");
17            secret_phase();
18        }
19        puts("Congratulations! You've defused the bomb!");
20    }
21    return *MK_FP(__GS__, 20) ^ v5;
22 }

```

- Xem mã giả của hàm phase\_defused ta thấy điều kiện để vào được secret\_phase là input nhập vào 2 số và 1 chuỗi. Trong đó, chuỗi phải là "DrEvil", quay lại các phase đã giải, nhận thấy phase 3, 4, 5 đều có input là 2 số nguyên. Vì vậy, có thể thêm chuỗi "DrEvil" vào input các phase này để mở secret\_phase. Kết quả kiểm tra, chỉ có nhập thêm vào input phase4 mới mở ra secret\_phase:

```

Curses, you've found the secret phase!
But finding it and solving it are quite different...
S

1 Verbosity leads to unclear, inarticulate things.
2 0 1 1 2 3 5
3 4 0
4 9 27 DrEvil
5 5 115
6 6 4 1 2 3 55

```

- Sau khi vào được pha bí mật, ta phân tích mã giả của pha



```

1 int secret_phase()
2 {
3     char *nptr; // ST14_4@1
4     signed int v2; // [sp+8h] [bp-10h]@1
5
6     nptr = (char *)read_line();
7     v2 = atoi(nptr);
8     if ( v2 <= 0 || v2 > 1001 )
9         explode_bomb();
10    if ( fun7((int)&n1, v2) != 2 )
11        explode_bomb();
12    puts("Wow! You've defused the secret stage!");
13    return phase_defused();
14 }

```

- Ta thấy điều kiện để bomb không nổ là v2 thuộc (0;1001] và fun7(&n1,v2) != 2

```

.data:0804D1B4      public n1
.data:0804D1B4 n1      db  24h ; $ ; DATA XREF: secret_phase+39↑o
.data:0804D1B5      db  0
.data:0804D1B6      db  0
.data:0804D1B7      db  0
.data:0804D1B8      db  0A8h ; ı
.data:0804D1B9      db  0D1h ; -
.data:0804D1BA      db  4
.data:0804D1BB      db  8
.data:0804D1BC      db  9Ch ; £
.data:0804D1BD      db  0D1h ; -
.data:0804D1BE      db  4
.data:0804D1BF      db  8

```

- Bấm vào n1 ta thấy đây là địa chỉ lưu trữ n1. Ngoài n1 còn có những giá trị n khác (n21, n22, n32...). Cũng như các node ở phase6, các giá trị n này cũng là danh sách liên kết với node trước lưu trữ địa chỉ node sau

```

1 int __cdecl fun7(int a1, int a2)
2 {
3     int result; // eax@2
4
5     if ( a1 )
6     {
7         if ( *(_DWORD *)a1 <= a2 )
8         {
9             if ( *(_DWORD *)a1 == a2 )
10                result = 0;
11            else
12                result = 2 * fun7(*(_DWORD *)a1 + 8, a2) + 1;
13        }
14        else
15        {
16            result = 2 * fun7(*(_DWORD *)a1 + 4, a2);
17        }
18    }
19    else
20    {
21        result = -1;
22    }
23    return result;
24 }

```

- Đây là mã giả của hàm fun7. Xét điều kiện để bom không nổ:  $\text{fun7}(\&n1, v2) \neq 2$ . Để fun7 có giá trị trả về khác 2, thử nghiệm với nhiều đáp. Ở dòng 19 – 21 điều kiện là  $(!a1)$  và dòng 11 – 12 điều kiện là  $(a1 > a2)$  để giá trị trả về  $\neq 2$ . Ở đây a1 truyền vào từ n1, với giá trị là 0x24 (=36) và a2 là v2 cần tìm (input). Vậy, điều kiện là  $v2 < 36$ . Xét danh sách liên kết chứa các giá trị n1, n22, n21... thử với các n nào có giá trị bé hơn 36. Thử xong ta được giá trị n32 = 22 là đúng. Vậy input là 22
- Thử lại đáp án

```

zeri@zeri:~/LHTHLab4$ gcc1c input.txt
zeri@zeri:~/LHTHLab4$ ./bomb input.txt
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
That's number 2. Keep going!
Halfway there!
So you got that one. Try this one.
Good work! On to the next...
Curses, you've found the secret phase!
But finding it and solving it are quite different...
Wow! You've defused the secret stage!
Congratulations! You've defused the bomb!

```

**HẾT**