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.ANTN.1

STT	Họ và tên	MSSV	Email
1	Lưu Gia Huy	21520916	21520916@gm.uit.edu.vn
2	Nguyễn Văn Khang Kim	21520314	21520314@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

STT	Công việc	Kết quả tự đánh giá
1	Pha 1	10
2	Pha 2	10
3	Pha 3	10
4	Pha 4	10
5	Pha 5	10
6	Pha 6	10
7	Pha bí mật (bonus)	10

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

[1] 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

```
result = strings_not_equal(a1, "The moon unit will be divided into two divisions.");
if ( result )
  explode_bomb();
return result;
```

Strings_not_equal() hàm này làm việc như cái tên của nó, ở đây không có 1 sự lừa lọc nào

```
1 int __cdecl strings_not_equal(_BYTE *a1, _BYTE *a2)
2 {
3   int v2; // ebx
4   _BYTE *v4; // [esp+8h] [ebp-Ch]
5   _BYTE *v5; // [esp+Ch] [ebp-8h]
6
7   v2 = string_length(a1);
8   if ( v2 != string_length(a2) )
9     return 1;
10  v4 = a1;
11  v5 = a2;
12  while ( *v4 )
13  {
14    if ( *v4 != *v5 )
15      return 1;
16    ++v4;
17    ++v5;
18  }
19  return 0;
20 }
```

Đầu tiên là kiểm tra xem độ dài của 2 chuỗi có bằng nhau không, nếu khác nhau sẽ return 1, duyệt qua từng kí tự của 2 chuỗi ở vị trí tương ứng để so sánh với nhau, nếu có tồn tại kí tự ở vị trí tương ứng nhưng lại khác nhau sẽ return 1, nếu mọi thứ đều trùng nhau thì sẽ return 0.

Ở phase 1 nếu check input không trùng với chuỗi "The moon unit will be divided into two divisions." thì biến result nhận kết quả là 1, và vòng if để check sẽ làm bom nổ

=> input đầu vào của phase 1 sẽ là: The moon unit will be divided into two divisions.

Passed

```
huy-21520916@huy21520916-virtual-machine:~/Downloads$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
The moon unit will be divided into two divisions.
Phase 1 defused. How about the next one?
```

2. Pha 2

```
unsigned int __cdecl phase_2(int input)
{
  int i; // [esp+10h] [ebp-28h]
  int v3[6]; // [esp+14h] [ebp-24h] BYREF
  unsigned int v4; // [esp+2Ch] [ebp-Ch]

  v4 = __readgsdword(20u);
  read_six_numbers(input, (int)v3);
  if ( v3[0] != 1 )
    explode_bomb();
  for ( i = 1; i <= 5; ++i )
  {
    if ( v3[i] != 2 * v3[i - 1] )
      explode_bomb();
  }
  return __readgsdword(20u) ^ v4;
}
```

Chúng ta xem liệu read_six_numbers() có làm đúng như những gì tên nó mô tả:

```
int __cdecl read_six_numbers(int a1, int a2)
{
  int result; // eax

  result = __isoc99_sscanf(a1, "%d %d %d %d %d %d", a2, a2 + 4, a2 + 8, a2 + 12, a2 + 16, a2 + 20);
  if ( result <= 5 )
    explode_bomb();
  return result;
}
```

Thât may mắn, ta vẫn không dính phải một cú lừa. Hàm này sẽ lấy 6 số nguyên đầu vào và lưu vào mảng a1,

Ở hàm chính phase_2 của chúng ta thì 6 số nguyên đầu vào sẽ được lưu vào mảng v3

Nếu v3[0] != 1 thì sẽ làm nổ bom

Và 1 vòng lặp for để xét xem nếu v3[i] mà khác 2 lần v3[i-1] thì bom nổ, có nghĩa là ngoại trừ số đầu tiên thì các số sau sẽ gấp đôi số trước nó

Theo đó ta có input ít nhất phải thỏa mãn:

1 2 4 8 16 32

Phía sau cái tiêu chí nhất định phải thỏa mãn trên ta có thể thêm các kí tự khác, nhưng nếu quá tham lam thì đây chính là kết cục:

```
huy-21520916@huy21520916-virtual-machine:~/Downloads$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
The moon unit will be divided into two divisions.
Phase 1 defused. How about the next one?
1 2 4 8 16 32 aaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbnnnnnnnaaaaaaaaaaaaaaaaaannnn
nnnnnnnaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Error: Input line too long

BOOM!!!
The bomb has blown up.
huy-21520916@huy21520916-virtual-machine:~/Downloads$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
The moon unit will be divided into two divisions.
Phase 1 defused. How about the next one?
1 2 4 8 16 32 aaaaaaaaaaaaaaaaaaaaaaaaa
That's number 2.  Keep going!
```

Ra đây là lý do ta bị ràng buộc:

```
1  int read_line()
2  {
3    int v0; // edx
4    int v1; // edx
5    signed int v3; // [esp+Ch] [ebp-Ch]
6
7    if ( !skip() )
8    {
9      if ( infile == (FILE *)stdin )
10     {
11       puts("Error: Premature EOF on stdin");
12       exit(8);
13     }
14     if ( getenv("GRADE_BOMB") )
15       exit(0);
16     infile = (FILE *)stdin;
17     if ( !skip() )
18     {
19       puts("Error: Premature EOF on stdin");
20       exit(0);
21     }
22   }
23   v3 = strlen((const char *)(80 * num_input_strings + 134534208));
24   if ( v3 > 78 )
25   {
26     puts("Error: Input line too long");
27     v0 = num_input_strings++;
28     strcpy((char *)(80 * v0 + 134534208), "***truncated***");
29     explode_bomb();
30   }
31   *(_BYTE *)(v3 - 1 + 80 * num_input_strings + 134534208) = 0;
32   v1 = num_input_strings++;
33   return 80 * v1 + 134534208;
34 }
```

Nhưng cuối cùng ta vẫn pass được phase2 với

Nhiều input thỏa mãn:

Passed

```
1 2 4 8 16 32 aaaaaaaaaaaaaaaaaaaaaaaaa
That's number 2.  Keep going!
```

```
1 2 4 8 16 32
That's number 2.  Keep going!
```

Miễn là đoạn đầu của input phải là: 1 2 4 8 16 32

Và phải thỏa hàm read_line() (có chiều dài nhỏ hơn 78)

3. Pha 3

```
1 unsigned int __cdecl phase_3(int a1)
2 {
3   int num1; // [esp+1Ch] [ebp-1Ch] BYREF
4   int num2; // [esp+20h] [ebp-18h] BYREF
5   int v4; // [esp+24h] [ebp-14h]
6   int input; // [esp+28h] [ebp-10h]
7   unsigned int v6; // [esp+2Ch] [ebp-Ch]
8
9   v6 = __readgsdword(0x14u);
10  v4 = 0;
11  input = 0;
12  input = __isoc99_sscanf(a1, "%d %d", &num1, &num2);
13  if ( input <= 1 )
14    explode_bomb();
15  switch ( num1 )
16  {
17    case 0:
18      v4 += 827;
19      goto LABEL_5;
20    case 1:
21 LABEL_5:
22      v4 -= 968;
23      goto LABEL_6;
24    case 2:
25 LABEL_6:
26      v4 += 673;
27      goto LABEL_7;
28    case 3:
29 LABEL_7:
30      v4 -= 568;
31      goto LABEL_8;
32    case 4:
33 LABEL_8:
34      v4 += 568;
35      goto LABEL_9;
36    case 5:
```

```
37 LABEL_9:
38       v4 -= 568;
39       goto LABEL_10;
40     case 6:
41 LABEL_10:
42       v4 += 568;
43       break;
44     case 7:
45       break;
46     default:
47       explode_bomb();
48   }
49   v4 -= 568;
50   if ( num1 > 5 || v4 != num2 )
51     explode_bomb();
52   return __readgsdword(0x14u) ^ v6;
53 }
```

Phase3 này sẽ nhận 2 số từ input đầu vào, nếu như không nhận đủ 2 số nguyên thì bom nổ

Nếu số đầu tiên(num1) nhận vô mà không thuộc khoảng [0;5] thì bom sẽ nổ

Nếu sau 1 hồi tính toán, kiểm tra mà kết quả nhận được (v4) khác num2(tức số thứ 2 nhập vô) thì bom nổ

Do đó ta sẽ tính và kiểm tra từng trường hợp của num1 sẽ suy ra được num2 tương ứng

Và đó sẽ là tất cả các case cho input thỏa mãn phase3 này:

num1 = 0 => num2 = -36

num1 = 1 => num2 = -36 - 827 =-863

num1 = 2 => num2 = -36 - 827 + 968 =105

num1 = 3 => num2 = -36 - 827 + 968 - 673 =-568

num1 = 4 => num2 = -36 - 827 + 968 - 673 + 568 =0

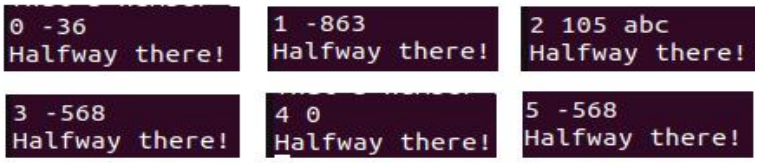
num1 = 5 => num2 = -36 - 827 + 968 - 673 + 568 - 568=-568

Vậy ta sẽ giải thích -36 từ đâu mà ra?

v4 = -36 = 827 - 968 + 673 - 568 + 568 - 568 + 568 - 568 (ứng với num1=0)

Khi mà num1 nhận các giá trị khác như 1,2,3,4,5 thì v4 sẽ thực hiện tính toán để bỏ qua các phép toán ở case đó, do đó nên ta mới có được các giá trị num2 như trên:

Passed:



Tương tự như phase2 thì ta có thể thêm các kí tự khác phía sau 2 số nguyên thỏa mãn như trên, miễn độ dài không vượt quá 78, thỏa các ràng buộc của hàm read_line()

4. Pha 4

```
1 unsigned int __cdecl phase_4(int a1)
2 {
3   int num1; // [esp+18h] [ebp-20h] BYREF
4   int num2; // [esp+1Ch] [ebp-1Ch] BYREF
5   int v4; // [esp+20h] [ebp-18h]
6   int v5; // [esp+24h] [ebp-14h]
7   int return_func4; // [esp+28h] [ebp-10h]
8   unsigned int v7; // [esp+2Ch] [ebp-Ch]
9
10  v7 = __readgsdword(0x14u);
11  v4 = __isoc99_sscanf(a1, "%d %d", &num1, &num2);
12  if ( v4 != 2 || num1 < 0 || num1 > 14 )
13    explode_bomb();
14  v5 = 2;
15  return_func4 = func4(num1, 0, 14);
16  if ( return_func4 != v5 || num2 != v5 )
17    explode_bomb();
18  return __readgsdword(0x14u) ^ v7;
19 }
```

Phase4 tương tự vẫn nhận 2 input đầu vào là số nguyên, điều này có nghĩa là các kí tự khác vẫn được thêm vào phía sau 2 số nguyên, miễn độ dài chuỗi nhập vào không vượt quá 78, thỏa mãn các ràng buộc của hàm read_line()

Trong phase4 này, nếu không nhận đủ đúng 2 số nguyên, hoặc số đầu tiên không thuộc khoảng [0;14] thì bom sẽ nổ

Sau khi đưa vô hàm func4 tính toán mà kết quả trả về khác 2 hoặc số thứ 2 khác 2 thì bom sẽ nổ

Điều này giúp ta xác định số thứ 2 (tức num2) bằng 2

Việc của ta giờ đây là xác định num1 sẽ là bao nhiêu trong khoảng [0;14] để đưa vô hàm func4 sẽ return là 2.

Trước đó ta cần xem func4 làm gì:

```
int __cdecl func4(int num1, int num_0, int num_14)
{
  int v4; // [esp+Ch] [ebp-Ch]

  v4 = (num_14 - num_0) / 2 + num_0;
  if ( v4 > num1 )
    return 2 * func4(num1, num_0, v4 - 1);
  if ( v4 >= num1 )
    return 0;
  return 2 * func4(num1, v4 + 1, num_14) + 1;
}
```

Đây là 1 hàm đệ qui, và ta có mã khai thác như sau:

```
def func4(num1, num_0, num_14):
    v4 =0
    v4=(num_14-num_0)/2 + num_0
    if(v4 > num1):
        return 2 * func4(num1, num_0, v4-1)
    if(v4>= num1):
        return 0;
    return 2*func4(num1, v4+1, num_14 )+1

for i in range(0,15):
    if(func4(i,0,14)==2):
        print("num1:",i, "num2: 2")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS D:\Code\CTF> python -u "d:\Code\CTF\bomb1.py"
num1: 4 num2: 2
num1: 5 num2: 2
```

Passed

```
4 2
So you got that one.  Try this one.
```

```
5 2 abc
So you got that one.  Try this one.
```

Các kí tự khác vẫn được thêm vào phía sau 2 số nguyên, miễn độ dài chuỗi nhập vào không vượt quá 78, thỏa mãn các ràng buộc của hàm read_line()

5. Pha 5

```
1 unsigned int __cdecl phase_5(int a1)
2 {
3   int num1; // [esp+14h] [ebp-24h] BYREF
4   int num2; // [esp+18h] [ebp-20h] BYREF
5   int v4; // [esp+1Ch] [ebp-1Ch]
6   int v5; // [esp+20h] [ebp-18h]
7   int v6; // [esp+24h] [ebp-14h]
8   int v7; // [esp+28h] [ebp-10h]
9   unsigned int v8; // [esp+2Ch] [ebp-Ch]
10
11  v8 = __readgsdword(0x14u);
12  v6 = __isoc99_sscanf(a1, "%d %d", &num1, &num2);
13  if ( v6 <= 1 )
14    explode_bomb();
15  num1 &= 0xFu;
16  v7 = num1;
17  v4 = 0;
18  v5 = 0;
19  while ( num1 != 15 )
20  {
21    ++v4;
22    num1 = array_2705[num1];
23    v5 += num1;
24  }
25  if ( v4 != 15 || v5 != num2 )
26    explode_bomb();
27  return __readgsdword(0x14u) ^ v8;
28 }
```

Ở phase 5 tương tự vẫn nhận 2 input đầu vào là số nguyên, điều này có nghĩa là các kí tự khác vẫn được thêm vào phía sau 2 số nguyên, miễn độ dài chuỗi nhập vào không vượt quá 78, thỏa mãn các ràng buộc của hàm read_line()

Nếu đầu vào không đủ 2 số nguyên bom sẽ nổ

num1 (tức là số đầu tiên được nhập) sẽ được giữ lại 4 bits cuối, có nghĩa là chỉ lấy 4 bits cuối của num1 để gán lại cho nó => num1 khi này sẽ thuộc khoảng [0;15]

Ta có 1 vòng lặp while, sau khi tính toán xong xuôi thì ta sẽ check nếu lặp không đủ 15 lần, và kết quả tính toán trong vòng lặp không bằng num2(tức là số thứ 2 được nhập) thì bom nổ

Do đó, ta phải pass chỗ này bằng cách, lặp đủ 15 lần, và kết quả đó bằng num2

Đi dô vòng while, ta thấy có 1 mảng array_2705 (maybe 2705 là 1 con số đặc biệt nào đó, hoặc không)

Ta có array_2705:

```
.data:0804D1C0 array_2705      db 0Ah                  ; DATA XREF: phase_5+63↑r
.data:0804D1C0                 db 0,0,0,2,0,0,0,0Eh,0,0,0,7,0,0,0,8,0,0,0,0Ch,0,0,0,0Fh,0,0,0,0Bh,0,0
.data:0804D1C0                 db 0,0,0,0,0,4,0,0,0,1,0,0,0,0Dh,0,0,0,3,0,0,0,9,0,0,0,6,0,0,0,5,0,0,0
```

Yah, cuối cùng ta có mã khai thác như sau:

```
26  #array_2705= [0 , 2, E,  7, 8,  C,  F,  B, 0, 4, 1, D,  3, 9, 6, 5]
27  array_2705 = [10, 2, 14, 7, 8, 12, 15, 11, 0, 4, 1, 13, 3, 9, 6, 5]
28
29  def func5(num1):
30      v4 = 0
31      v5 = 0
32      while num1 != 15:
33          v4 += 1
34          num1 = array_2705[num1]
35          v5 += num1
36      return v4, v5
37
38  for i in range(15):
39      v4, v5 = func5(i)
40      if v4 == 15:
41          print(i, v5)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS D:\Code\CTF> python -u "d:\Code\CTF\bomb1.py"
5 115
```

5 115

Ở đây chưa là tất cả, bởi ta hãy nhớ khi nãy num1 đã bị giới hạn và chỉ lấy 4 bits cuối.

Do đó 1 cách tổng quát, ngoại trừ việc có thể thêm các kí tự ở cuối sao cho thỏa mãn các ràng buộc của hàm read_line() thì:

num1 là 1 số nguyên có 4 bits cuối = **0101** (ví dụ num1 = **149** (1001 0101))

num2 bắt buộc phải bằng **115**

Passed

```
5 115
Good work!  On to the next...
213 115
Good work!  On to the next...
149 115 abc
Good work!  On to the next...
```

6. Pha 6

(1) Ở trong phase 6, chương trình sẽ lấy vào sáu số trong khoảng từ 1 đến 6 à không trùng lặp,6 số được vào mảng v12 theo thứ tự.

(2) Mảng v13[i] dùng để lưu địa chỉ của các node thứ v12[i] .

(3) Tạo một danh sách liên kết mới với thứ tự của node đầu tiên đến node cuối cùng giống như vị trí của các node có vị trí được lưu trong mảng v13. (ví dụ mảng v12 = [1,3,5,6,2,4] thì v3 sẽ là 1 danh sách liên kết node1 -> node3 -> node5 -> node6 -> node2 -> node4

(4)Duyệt qua tất cả các node trong danh sách liên kết mới tạo (v3) và so sánh giá trị của node hiện tại và node kế tiếp, nếu giá trị của node hiện tại lớn hơn giá trị của node tiếp theo thì bom sẽ nổ vì vậy nên các node cần phải sắp xếp theo vị trí theo giá trị tăng dần.

(5)Giá trị của các node tương ứng là:

node1= 0x33

node2= 0x3E1

node3=0xC8

node4=0x2BF

node5=0x306

node6=0xAA

(6)Sắp xếp các node theo thứ tự tăng dần, ta được input là: 1 6 3 4 5 2

```
17  v14 = *MK_FP(__GS__, 20);
18  read_six_numbers(a1, 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] > l; ++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}
```

1
2
3
4

5

```
.data:0804D0C4                  public node6
.data:0804D0C4 node6            db 0AAh ; ª
.data:0804D0C5                  db    0
.data:0804D0C6                  db    0
.data:0804D0C7                  db    0
.data:0804D0C8                  db    6
.data:0804D0C9                  db    0
.data:0804D0CA                  db    0
.data:0804D0CB                  db    0
.data:0804D0CC                  db    0
.data:0804D0CD                  db    0
.data:0804D0CE                  db    0
.data:0804D0CF                  db    0
.data:0804D0D0                  public node5
.data:0804D0D0 node5            db    6
.data:0804D0D1                  db    3
.data:0804D0D2                  db    0
.data:0804D0D3                  db    0
.data:0804D0D4                  db    5
.data:0804D0D5                  db    0
.data:0804D0D6                  db    0
.data:0804D0D7                  db    0
.data:0804D0D8                  db 0C4h ; Ä
.data:0804D0D9                  db 0D0h ; Đ
.data:0804D0DA                  db    4
.data:0804D0DB                  db    8
```

```
.data:0804D0DC                  public node4
.data:0804D0DC node4            db 0BFh ; ¿
.data:0804D0DD                  db    2
.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 ; Đ
.data:0804D0E5                  db 0D0h ; Đ
.data:0804D0E6                  db    4
.data:0804D0E7                  db    8
.data:0804D0E8                  public node3
.data:0804D0E8 node3            db 0C8h ; È
.data:0804D0E9                  db    0
.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 ; Ü
.data:0804D0F1                  db 0D0h ; Đ
.data:0804D0F2                  db    4
.data:0804D0F3                  db    8
```

```
.data:0804D0F4                  public node2
.data:0804D0F4 node2            db 0E1h ; á
.data:0804D0F5                  db    3
.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 ; è
.data:0804D0FD                  db 0D0h ; Đ
.data:0804D0FE                  db    4
.data:0804D0FF                  db    8
.data:0804D100                  public node1
.data:0804D100 node1            db  33h ; 3
.data:0804D101                  db    0
.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 ; ô
.data:0804D109                  db 0D0h ; Đ
.data:0804D10A                  db    4
.data:0804D10B                  db    8
```

```
khangkim@khangkim-VirtualBox:~/Downloads$ ./bomb input
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...
1 6 3 4 5 2
Congratulations! You've defused the bomb!
khangkim@khangkim-VirtualBox:~/Downloads$
```

6

7. Pha bí mật (Bonus)

Ở trong hàm phase_defused() ta có thể thấy được hàm này dùng để mở khóa phase secret.

(1)Điều kiện để xuất hiện phase secret đó là hoàn thành 6 phase đầu và phải thêm chuỗi "DrEvil" vào input của phase 4.

(2)Input của phase secret là 1 số nằm trong khoảng từ 1 đến 1001.

(3)Để gỡ bom thì hàm fun7() với đối số thứ nhất là n1 (node root) và đối số thứ hai là v2(input) có giá trị trả về phải bằng 1 .

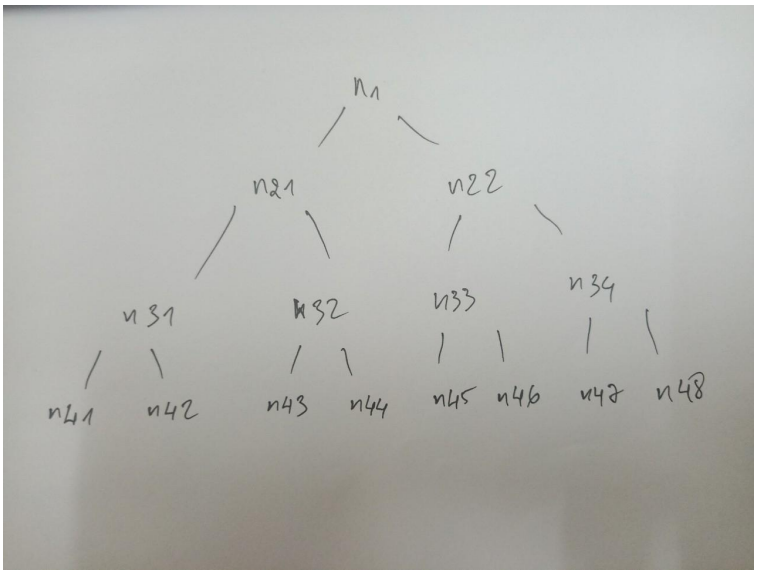
(4)Trong hàm fun7() nó sẽ duyệt qua các node và tính toán giá trị của result sau đó trả về result, nếu node hiện tại là null sẽ trả về -1, nếu node hiện tại có giá trị bằng v2 thì sẽ trả về 0, còn lớn hơn v2 thì sẽ nhảy đến node bên trái, nhỏ hơn v2 thì result tăng thêm 1 và nhảy đến node bên phải. (trái, phải ở đây được giả định bởi +4, +8)

Cụ thể như sau:

```
.data:0804D19C n22              db  32h ; 2
.data:0804D19D                  db    0
.data:0804D19E                  db    0
.data:0804D19F                  db    0
.data:0804D1A0                  db  84h
.data:0804D1A1                  db 0D1h
.data:0804D1A2                  db    4
.data:0804D1A3                  db    8
.data:0804D1A4                  db  6Ch ; l
.data:0804D1A5                  db 0D1h
.data:0804D1A6                  db    4
.data:0804D1A7                  db    8
.data:0804D1A8                  public n21
.data:0804D1A8 n21              db    8
.data:0804D1A9                  db    0
.data:0804D1AA                  db    0
.data:0804D1AB                  db    0
.data:0804D1AC                  db  78h ; x
.data:0804D1AD                  db 0D1h
.data:0804D1AE                  db    4
.data:0804D1AF                  db    8
.data:0804D1B0                  db  90h
.data:0804D1B1                  db 0D1h
.data:0804D1B2                  db    4
.data:0804D1B3                  db    8
.data:0804D1B4                  public n1
.data:0804D1B4 ; _DWORD n1[3]
.data:0804D1B4 n1               dd 24h, 804D1A8h, 804D19Ch
```

Dựa vào các địa chỉ trỏ đến node kế tiếp được khoanh đỏ (còn nhiều hơn trong ảnh được cap, như là các node n31, n32, n33, n34, n41, n42,...,n47, n48

Ta có cây biểu diễn các node như sau:



Vậy để hàm trả về kết quả là 1 thì nó phải nhảy về node bên phải 1 lần và giá trị bằng node đó .

```
1 int __cdecl fun7(_DWORD *arr3, int input)
2 {
3   if ( !arr3 )
4     return -1;
5   if ( *arr3 > input )
6     return 2 * fun7((_DWORD *)arr3[1], input);
7   if ( *arr3 == input )
8     return 0;
9   return 2 * fun7((_DWORD *)arr3[2], input) + 1;
10 }
```

Tức là đầu tiên ta cần vào được dòng return cuối cùng, sau đó ở hàm fun7 lần tiếp theo ta buộc phải return 0 để đạt được kết quả 1 như mong muốn, do đó input của ta phải lớn hơn giá trị của node n1 ở hàm fun7 lần đầu tiên, và bằng giá trị của node kế tiếp ở hàm fun7 lần tiếp theo cũng là lần cuối (return 0)

Muốn được vậy là phải check xem arr3[2] chứa gì ?

Thật vậy arr3[2] chứa : địa chỉ của node n22

```
.data:0804D19C                  public n22
.data:0804D19C n22              db 32h ; 2

.data:0804D1B4 n1               dd 24h, 804D1A8h, 804D19Ch
```

Giá trị của node n22 vừa hay là **32** thỏa mãn yêu cầu lớn hơn giá trị node n1 là **24** vậy nên để fun7 return về 1 thì input được nhập phải là giá trị của n22 = 0x32 = **50** (hệ 10)

(5)Ta tìm được v2 = n22 = 0x32 = 50.

```
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 }
```

1

Open | input ~/Downloads

```
1 The moon unit will be divided into two divisions.
2 1 2 4 8 16 32
3 0 -36
4 4 2 DrEvil
5 5 115
6 1 6 3 4 5 2
```

Thử thêm vào input của các phase 3,4,5 chuỗi “DrEvil” và ta phát hiện khi thêm vào phase 4 thì nó sẽ mở cánh cổng phase secret cho ta

```
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) != 1 )
11     explode_bomb();
12   puts("Wow! You've defused the secret stage!");
13   return phase_defused();
14 }
```

2

3

```
.data:0804D19C                  public n22
.data:0804D19C n22              db  32h ; 2
.data:0804D19D                  db    0
.data:0804D19E                  db    0
.data:0804D19F                  db    0
.data:0804D1A0                  db  84h ; „
.data:0804D1A1                  db 0D1h ; Ñ
.data:0804D1A2                  db    4
.data:0804D1A3                  db    8
.data:0804D1A4                  db  6Ch ; l
.data:0804D1A5                  db 0D1h ; Ñ
.data:0804D1A6                  db    4
.data:0804D1A7                  db    8
.data:0804D1A8                  public n21
.data:0804D1A8 n21              db    8
.data:0804D1A9                  db    0
.data:0804D1AA                  db    0
.data:0804D1AB                  db    0
.data:0804D1AC                  db  78h ; x
.data:0804D1AD                  db 0D1h ; Ñ
.data:0804D1AE                  db    4
.data:0804D1AF                  db    8
.data:0804D1B0                  db  90h ;
.data:0804D1B1                  db 0D1h ; Ñ
.data:0804D1B2                  db    4
.data:0804D1B3                  db    8
.data:0804D1B4                  public n1
.data:0804D1B4 n1               db  24h ; $
.data:0804D1B5                  db    0
.data:0804D1B6                  db    0
.data:0804D1B7                  db    0
.data:0804D1B8                  db 0A8h ; ¨
.data:0804D1B9                  db 0D1h ; Ñ
.data:0804D1BA                  db    4
.data:0804D1BB                  db    8
.data:0804D1BC                  db  9Ch ; œ
.data:0804D1BD                  db 0D1h ; Ñ
.data:0804D1BE                  db    4
.data:0804D1BF                  db    8
000041B1 0804D1B1: .data:0804D1B1
```

```
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 }
```

4

5

Passed

```
khangkim@khangkim-VirtualBox:~/Downloads$ ./bomb input
Welcome to my fiendish little bomb. You have 6 phases
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...
50
Wow! You've defused the secret stage!
Congratulations! You've defused the bomb!
khangkim@khangkim-VirtualBox:~/Downloads$
```

YÊU CẦU CHUNG

Báo cáo:

- File **.PDF**.
- Đặt tên theo định dạng: **[Mã lớp]-Lab4_NhomX_MSSV1-MSSV2.pdf** (trong đó X là số thứ tự nhóm, MSSV gồm đầy đủ MSSV của tất cả các thành viên thực hiện bài thực hành).

  Ví dụ: [NT209.N21.ANTN.1]-Lab4_Nhom2_21520001-21520013.pdf.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá:

- Hoàn thành tốt yêu cầu được giao.
- Có nội dung mở rộng, ứng dụng.

Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.

HẾT