

BÁO CÁO THỰC HÀNH LẬP TRÌNH HỆ THỐNG LAB4

Nhóm 12:

22520825 Nguyễn Đức Luân

22520661 Vũ Ngọc Quốc Khánh

22521110 Đoàn Hoàng Phúc

Ta xem cấu trúc của hàm main bằng công cụ IDA pro 32 bit:

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    int line; // eax
    int v4; // eax
    int v5; // eax
    int v6; // eax
    char *s; // [esp+0h] [ebp-Ch]

    puts(
        "Welcome to UIT's bomb lab.\n"
        "You have to deactivate our bomb by solving 5 phases with the correct inputs consecutively, and otherwise the bomb will be blown up!");
    if ( argc == 1 )
    {
        infile = (FILE *)_bss_start;
    }
    else
    {
        if ( argc != 2 )
        {
            printf("Usage: %s [<input_file>]\n", *argv);
            exit(8);
        }
        infile = fopen(argv[1], "r");
        if ( !infile )
        {
            printf("%s: Error: Couldn't open %s\n", *argv, argv[1]);
            exit(8);
        }
    }
    puts("\n[*] Phase 1\n- Hint: Numbers are always magical!");
    line = read_line();
    phase1(line);
    defuse_bomb();
    puts("\n[*] Phase 2\n- Hint: You must answer your secret question!");
    v4 = read_line();
    phase2(v4);
    defuse_bomb();
    puts("\n[*] Phase 3\n- Hint: Many cases make everything so confusing.");
    v5 = read_line();
    phase3(v5);
    puts("\n[*] Phase 4\n- Hint: Let's dig in to recursive function :)");
    v6 = read_line();
    phase4(v6);
    defuse_bomb();
    puts("\n[*] Phase 5\n-Hint: No hint is also a hint :)");
    s = (char *)read_line();
    phase5(s);
    defuse_bomb();
    return 0;
}
```

Dựa vào mô tả ban đầu khi quan sát chương trình ta thấy cần phải giải quyết hết 5 phase của chương trình để gỡ được bom. Có hai cách để nhập input vào chương trình, một là nhập từng dòng ứng với từng phase, hai là đưa toàn bộ đáp án của các phase vào một file text và nhập theo cú pháp `./nt209-uit-bomb <file_name>`.

Đầu tiên ta quan sát phase1:

```
1 int __cdecl phase1(int a1)
2 {
3     int result; // eax
4     int v2; // [esp+Ch] [ebp-2Ch] BYREF
5     char v3; // [esp+10h] [ebp-28h] BYREF
6     char v4; // [esp+14h] [ebp-24h] BYREF
7     char v5; // [esp+18h] [ebp-20h] BYREF
8     char v6; // [esp+1Ch] [ebp-1Ch] BYREF
9     char v7; // [esp+20h] [ebp-18h] BYREF
10    int v8; // [esp+24h] [ebp-14h]
11    int v9; // [esp+28h] [ebp-10h]
12    int i; // [esp+2Ch] [ebp-Ch]
13
14    v9 = __isoc99_sscanf(a1, "%d %d %d %d %d %d", &v2, &v3, &v4, &v5, &v6, &v7);
15    if ( v9 != 6 )
16        explode_bomb();
17    v8 = 15;
18    result = v2;
19    if ( v2 != 15 )
20        explode_bomb();
21    for ( i = 1; i <= 5; ++i )
22    {
23        result = *(&v2 + i);
24        if ( result != 2 * *(&v2 + i - 1) )
25            explode_bomb();
26    }
27    return result;
28 }
```

Từ đây ta thấy hàm này lưu sáu input có kiểu dữ liệu là số nguyên là v2 v3 v4 v5 v6 v7, bắt đầu kiểm tra xem v9 (đếm số lượng input nhập vào đầu tiên là số nguyên) có bằng 6 không, nếu không thỏa thì bomb nổ và kết thúc chương trình, sau đó kiểm tra điều kiện từng input. Đầu tiên kiểm tra giá trị của v2 có bằng 15 hay không, nếu thỏa tiếp tục kiểm tra giá trị của các giá trị còn lại theo quy tắc là giá trị phần tử đứng sau (tính từ v3 trở đi) phải gấp đôi giá trị của phần tử trước đó. Nếu không thì bomb sẽ nổ.

Như ta đã biết giá trị v2 phải là 15, do đó suy ra giá trị của các input còn lại là v3=30, v4=60, v5=120, v6=240, v7=480

Vậy input cần nhập ở Phase1 là: 15 30 60 120 240 480 (vì chương trình chỉ lưu sáu giá trị đầu tiên nên các giá trị sau có thể nhập tùy ý miễn là sáu giá trị đầu đúng với yêu cầu đề bài)

```
[*] Phase 1
- Hint: Numbers are always magical!
15 30 60 120 240 480 asdfahsjfa
Good job! You've cleared the first phase!
```

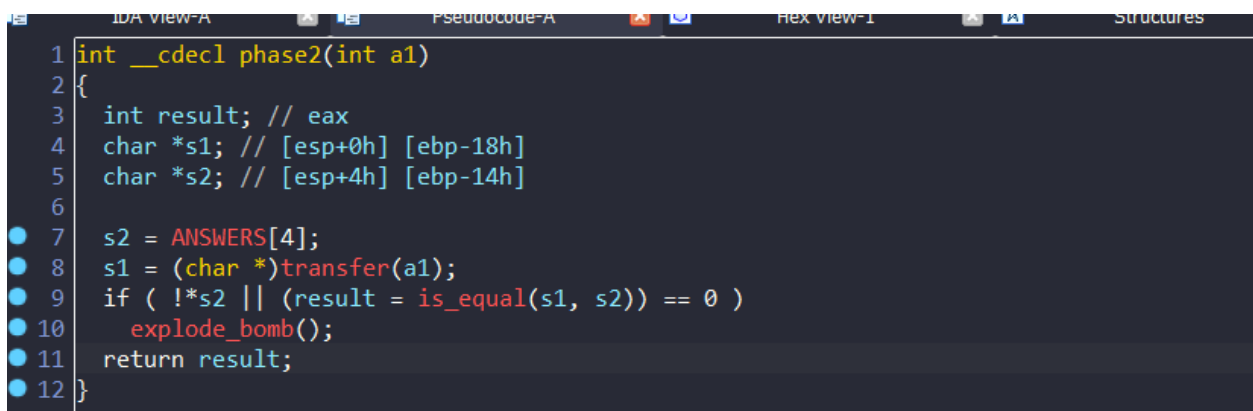
Kiểm tra kết quả phase1:

```
semloh@semloh-virtual-machine:/mnt/hgfs/D/OBJECTS/HK4/LAP TRINH HE THONG/lab/lab4$ ./nt209-uit-bomb
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!
15 30 60 120 240 480
Good job! You've cleared the first phase!

[*] Phase 2
- Hint: You must answer your secret question!
```

Tiếp theo ta phân tích Phase2:



```
1 int __cdecl phase2(int a1)
2 {
3     int result; // eax
4     char *s1; // [esp+0h] [ebp-18h]
5     char *s2; // [esp+4h] [ebp-14h]
6
7     s2 = ANSWERS[4];
8     s1 = (char *)transfer(a1);
9     if ( !*s2 || (result = is_equal(s1, s2)) == 0 )
10         explode_bomb();
11     return result;
12 }
```

Từ hình trên ta thấy phase2 trước tiên sẽ biến đổi giá trị của input nhập vào (tham số a1 chính là input), sau đó so sánh giá trị của chuỗi input sau khi biến đổi với giá trị của biến mà s2 trỏ tới. Ta có thể xác định được giá trị của biến đó thông qua giá trị của ANSWER[4]

Quan sát mảng ANSWER ta được:

•	.data:0804B11F	db	0
•	.data:0804B120	public	ANSWERS
•	.data:0804B120	ANSWERS	dd offset aSy754T76Fsys6
•	.data:0804B120		; DATA XREF: phase2+1D1r
•	.data:0804B120		; "SY754.T76.FSYS.6"
•	.data:0804B124	dd offset	aZsnajwxnydTkNs ; "Zsnajwxnyd Tk Nsktwrfynts YjhmstqtlD"
•	.data:0804B128	dd offset	aShZnyJizAs ; "sh.zny.jiz.as"
•	.data:0804B12C	dd offset	aNsktwrfyntsXjh ; "Nsktwrfynts Xjhzwnyd"
•	.data:0804B130	dd offset	aMnjsiyyZnyJizA ; "mnjsiyy@zny.jiz.as"
•	.data:0804B134	dd offset	a57382707557 ; "57382707557"
•	.data:0804B138	dd offset	aAnjysfrjxj ; "Anjysfrjxj"
•	.data:0804B13C	dd offset	aYzjxifd ; "Yzjxifd"
•	.data:0804B140	dd offset	a507579 ; "50/7579"
•	.data:0804B144	dd offset	aXjajs ; "Xjajs"
•	.data:0804B148	dd offset	aSy754ZnyGtrg ; "sy754-zny-gtrg"
•	.data:0804B14C	dd offset	aAnjysfrjxj ; "Anjysfrjxj"
•	.data:0804B150	dd offset	a53517551 ; "53/51/7551"
•	.data:0804B154	dd offset	aHmnhflt ; "Hmnhflt"
•	.data:0804B158	dd offset	aAfsRnjzVzthYzL ; "Afs Rnjz Vzth Yz Lnfr"
•	.data:0804B15C	dd offset	aGnsmIztsl ; "Gnsm Iztsl"
•	.data:0804B160	dd offset	aWzxxnf ; "Wzxxnf"
•	.data:0804B164	dd offset	aNshtwwjhyqd ; "Nshtwwjhyqd"
•	.data:0804B168	dd offset	aBnsitb ; "Bnsitb"
•	.data:0804B16C	dd offset	aYmjLwjfyBfqqTk ; "Ymj Lwjfy Bfqq Tk Hmnsf"
•	.data:0804B170	public	PHASE MSG

Ta biết được rằng mảng ANSWER là một mảng con trữ chứa giá trị là địa chỉ của các biến mà các phần tử trong mảng trữ tới. Vì vậy kích thước của mỗi phần tử trong mảng là 4 byte. Ta xác định được giá trị của ANSWER[4] tại địa chỉ $0x804B120 + 4 \times 4 = 0x804B130$ là offset aMnjsiyyZnyJizA và có giá trị là ["mnjsiyy@zny.jiz.as"](mailto:mnjsiyy@zny.jiz.as)

Sau khi có được chuỗi để so sánh thì ta sẽ quan sát hàm chuyển đổi chuỗi input đầu vào là hàm transfer(a1)

```

1 int __cdecl transfer(int a1)
2 {
3     char v2; // [esp+Ah] [ebp-6h]
4     char v3; // [esp+Bh] [ebp-5h]
5     int i; // [esp+Ch] [ebp-4h]
6
7     for ( i = 0; *(_BYTE *)(i + a1); ++i )
8     {
9         v3 = *(_BYTE *)(i + a1);
10        if ( (v3 <= 96 || v3 > 122) && (v3 <= 64 || v3 > 90) )
11        {
12            if ( v3 > 47 && v3 <= 57 )
13                v3 = (v3 - 48 + 5) % 10 + 48;
14        }
15        else
16        {
17            if ( v3 <= 96 )
18                v2 = 65;
19            else
20                v2 = 97;
21            v3 = (v3 - v2 + 5) % 26 + v2;
22        }
23        *(_BYTE *)(a1 + i) = v3;
24    }
25    return a1;
26 }

```

Từ hình trên ta phân tích được các hoạt động của hàm là nó sẽ chuyển đổi từng phần tử trong chuỗi input bằng vòng lặp cùng với các phép biến đổi bên trong để tạo ra phần tử mới dựa theo giá trị ASCII của từng phần tử.

Từ phân tích trên ta sử dụng phương pháp vét cạn để tìm input với giới hạn từ 32 đến 126 (đây là các giá trị ASCII in đc) với độ dài bằng với chuỗi cần so sánh.

```

#phase2

key = "mnjsiyy@zny.jiz.as"

def tranfer(a):
    v3= a
    if ( (v3 <= 96 or v3 > 122) and (v3 <= 64 or v3 > 90) ):
        if ( v3 > 47 and v3 <= 57 ):
            v3 = (v3 - 48 + 5) % 10 + 48

        else:
            if ( v3 <= 96 ):
                v2 = 65
            else:
                v2 = 97
            v3 = (v3 - v2 + 5) % 26 + v2
    return v3

flag=[]
for i in key :
    for j in range (32,127):
        tmp = tranfer(j)
        if(tmp == ord(i)):
            flag.append(chr(j))
            break
for i in flag :
    print(i,end="")

```

Sau khi chạy chương trình trên ta thu được input của Phase2 là : “hiendtt@uit.edu.vn”

Kiểm tra input:

```

[*] Phase 2
- Hint: You must answer your secret question!
hiendtt@uit.edu.vn
Two phases have been solved. Keep going!

[*] Phase 3
- Hint: Many cases make everything so confusing.

```

Tiếp đó ta phân tích Phase3:

```

1 int __cdecl phase3(int a1)
2 {
3     int result; // eax
4     unsigned __int8 v2; // [esp+fh] [ebp-19h] BYREF
5     int v3; // [esp+10h] [ebp-18h] BYREF
6     int v4; // [esp+14h] [ebp-14h] BYREF
7     int v5; // [esp+18h] [ebp-10h]
8     char v6; // [esp+1fh] [ebp-9h]
9
10    v5 = 0;
11    v5 = __isoc99_sscanf(a1, "%d %c %d", &v4, &v2, &v3);
12    if ( v5 <= 2 )
13        explode_bomb();
14    switch ( v4 )
15    {
16        case 0:
17            v6 = 'v';
18            if ( v3 != 835 )
19                explode_bomb();
20            return result;
21        case 1:
22            v6 = 'w';
23            if ( v3 != 861 )
24                explode_bomb();
25            return result;
26        case 2:
27            v6 = 'b';
28            if ( v3 != 107 )
29                explode_bomb();
30            return result;
31        case 3:
32            v6 = 'x';
33            if ( v3 != 551 )
34                explode_bomb();
35            return result;
36        case 4:
37            v6 = 'b';
38            if ( v3 != 854 )
39                explode_bomb();
40            return result;

```

```

        return result;
        case 5:
            v6 = 'l';
            if ( v3 != 486 )
                explode_bomb();
            return result;
        case 6:
            v6 = 'e';
            if ( v3 != 379 )
                explode_bomb();
            return result;
        case 7:
            v6 = 'v';
            if ( v3 != 764 )
                explode_bomb();
            return result;
        default:
            v6 = 'g';
            explode_bomb();
    }
    result = v2;
    if ( v6 != v2 )
        explode_bomb();
    return result;
}

```

Từ hình trên ta thấy phase3 lấy 3 input lần lượt là các biến v4 , v2 , v3 (v4,v3 là kiểu dữ liệu số nguyên int , v2 là kiểu char)sau đó kiểm tra số lượng input nhập vào thông qua biến v5. Nếu v5 <= 2 thì hàm sẽ kết thúc và nổ bomb. Biến v5 có vai trò đảm bảo số lượng input nhập vào để thực hiện các phép kiểm tra bên dưới là hàm switch case với biến v4 và với mỗi case thì sẽ kiểm tra giá trị của biến v3 và so sánh các giá trị của biến v2 với các giá trị tương ứng mỗi case.

Dựa vào đó ta có thể dễ dàng xác định giá trị của biến v4 , v2 , v3 tương ứng với mỗi case của biến v4:

V4 = 0, v2= “v” , v3= 835

```
[*] Phase 3
- Hint: Many cases make everything so confusing.
0 v 835
You've beaten another phase, that's great. What about the fourth one?
```

V4 = 1, v2="w", v3= 861

```
[*] Phase 3
- Hint: Many cases make everything so confusing.
1 w 861
You've beaten another phase, that's great. What about the fourth one?
```

V4 = 2, v2="b", v3= 107

```
[*] Phase 3
- Hint: Many cases make everything so confusing.
2 b 107
You've beaten another phase, that's great. What about the fourth one?
```

V4 = 3, v2="x", v3= 551

```
[*] Phase 3
- Hint: Many cases make everything so confusing.
3 x 551
You've beaten another phase, that's great. What about the fourth one?
```

V4 = 4, v2="b", v3= 854

```
[*] Phase 3
- Hint: Many cases make everything so confusing.
4 b 854
You've beaten another phase, that's great. What about the fourth one?
```

V4 = 5, v2="l", v3= 486

```
[*] Phase 3
- Hint: Many cases make everything so confusing.
5 l 486
You've beaten another phase, that's great. What about the fourth one?
```

V4 = 6, v2="e", v3= 379


```
[*] Phase 3
- Hint: Many cases make everything so confusing.
6 e 379
You've beaten another phase, that's great. What about the fourth one?
```

V4 = 7, v2= "v", v3= 764

```
[*] Phase 3
- Hint: Many cases make everything so confusing.
7 v 764
You've beaten another phase, that's great. What about the fourth one?

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

Do Phase4 chỉ quan tâm đến 3 input đầu nên các input có số lượng nhiều hơn vẫn chấp nhận nếu thỏa 3 input đầu tiên:

```
[*] Phase 3
- Hint: Many cases make everything so confusing.
0 v 835 ahsfkas
You've beaten another phase, that's great. What about the fourth one?
```

Tiếp theo ta kiểm tra hàm phase4:

```
1 int __cdecl phase4(int a1)
2 {
3     int result; // eax
4     int v2; // [esp+Ch] [ebp-1Ch] BYREF
5     unsigned int v3; // [esp+10h] [ebp-18h] BYREF
6     int v4; // [esp+14h] [ebp-14h]
7     int v5; // [esp+18h] [ebp-10h]
8     int v6; // [esp+1Ch] [ebp-Ch]
9
10    v6 = __isoc99_sscanf(a1, "%d %d", &v3, &v2);
11    if ( v6 != 2 || v3 >= 15 )
12        explode_bomb();
13    v5 = 5;
14    v4 = func4(v3, 0, 14);
15    if ( v4 != v5 || (result = v2, v2 != v5) )
16        explode_bomb();
17    return result;
18 }
```

Hàm này sẽ lấy input là hai số nguyên được lưu vào biến v3, v2. Sau đó hàm kiểm tra giá trị của biến v6 (là số lượng input nhập vào đầu tiên là số nguyên) và giá trị của biến v3. Nếu v6

khác 2 hoặc v3 lớn hoặc bằng 15 thì bomb sẽ nổ và kết thúc chương trình. Tiếp đó kiểm tra điều kiện giá trị của biến v4 và v2. Nếu v4 khác v5 hoặc v2 khác v5 thì bomb sẽ nổ và kết thúc chương trình. Dựa vào hàm ta dễ thấy giá trị của biến v2 là 5 và giá trị của v4 cũng là 5. Nhưng ta cần xác định được giá trị của biến v3 vì $v4 = \text{func4}(v3, 0, 14)$

Ta kiểm tra hàm $\text{func4}(v3, 0, 14)$: vì đây là một hàm đệ quy nên ta chỉ quan tâm đến kết quả trả về từ hàm này là được.

```
1 int __cdecl func4(int v3, int v0, int v14)
2 {
3     int v4; // [esp+Ch] [ebp-Ch]
4
5     v4 = (v14 - v0) / 2 + v0;
6     if ( v4 > v3 )
7         return 2 * func4(v3, v0, v4 - 1);
8     if ( v4 >= v3 )
9         return 0;
10    return 2 * func4(v3, v4 + 1, v14) + 1;
11 }
```

Từ đây ta sử dụng phương pháp vét cạn để tìm giá trị v3 với giới hạn là từ 0 đến 14 (với trường hợp số âm thì đa số đều rơi vào vòng lặp vô tận trừ giá trị -1 thì hàm trả về là 0 nên cũng sẽ bỏ qua)

```
#phase4
def re_num(v3,v0,v14):
    v4 = (v14 - v0) / 2 + v0
    if ( v4 > v3 ):
        return 2*re_num(v3, v0, v4 - 1)
    if ( v4 >= v3 ):
        return 0
    return 2*re_num(v3, v4 + 1, v14) + 1

a1=0
a2=14
for i in range (0,15):
    if(re_num(i,a1,a2)==5):
        print(i)
```

Ta tìm được giá trị của v3 là 10 và input của Phase4: 10 5

Kiểm tra input:

```
[*] Phase 4
- Hint: Let's dig in to recursive function :)
10 5
Awesome! Only one phase left!

[*] Phase 5
-Hint: No hint is also a hint :)
```

Do Phase4 chỉ lấy hai giá trị đầu tiên là số nguyên nên các giá trị nhập sau vẫn được chấp nhận nếu thỏa hai số đầu :

```
[*] Phase 4
- Hint: Let's dig in to recursive function :)
10 5 6
Awesome! Only one phase left!
```

Cuối cùng ta xử lý hàm Phase5:

```
1 size_t __cdecl phase5(char *s)
2 {
3     size_t result; // eax
4     int v2; // [esp+8h] [ebp-10h]
5     int i; // [esp+Ch] [ebp-Ch]
6
7     result = strlen(s);
8     if ( result != 6 )
9         explode_bomb();
10    v2 = 0;
11    for ( i = 0; i <= 5; ++i )
12    {
13        result = array_3854[s[i] & 0xF];
14        v2 += result;
15    }
16    if ( v2 != 46 )
17        explode_bomb();
18    return result;
19 }
```

Từ hàm này ta thấy input là một chuỗi input có 6 ký tự có kiểu char. Hàm sẽ kiểm tra độ dài của input có thỏa 6 ký tự hay không. Sau đó vào vòng lặp, mỗi vòng lặp sẽ tăng biến v2 bằng giá trị của mảng array_3854 với index được tính bằng phần tử thứ i của input and với giá trị 0xf. Kết thúc vòng lặp so sánh giá trị của v2 với 46. Nếu v2 khác 46 thì kết thúc chương trình và bomb nổ.

Ta xác định giá trị mảng array_3854[16]={ 0x2,0xa, 0x6, 0x1, 0xc, 0x10, 0x9, 0x3, 0x4, 0x7, 0xe, 0x5, 0xb, 0x8, 0xf, 0xd} thông qua phân tích:

```
.data:0804B1A0 ; int array_3854[16]
.data:0804B1A0 array_3854      dd 2                                ; DATA XREF: phase5+43↑r
.data:0804B1A4      db  0Ah
.data:0804B1A5      db  0
.data:0804B1A6      db  0
.data:0804B1A7      db  0
.data:0804B1A8      db  6
.data:0804B1A9      db  0
.data:0804B1AA      db  0
.data:0804B1AB      db  0
.data:0804B1AC      db  1
.data:0804B1AD      db  0
.data:0804B1AE      db  0
.data:0804B1AF      db  0
.data:0804B1B0      db  0Ch
.data:0804B1B1      db  0
.data:0804B1B2      db  0
.data:0804B1B3      db  0
.data:0804B1B4      db  10h
.data:0804B1B5      db  0
.data:0804B1B6      db  0
.data:0804B1B7      db  0
.data:0804B1B8      db  9
.data:0804B1B9      db  0
.data:0804B1BA      db  0
.data:0804B1BB      db  0
.data:0804B1BC      db  3
.data:0804B1BD      db  0
.data:0804B1BE      db  0
.data:0804B1BF      db  0
.data:0804B1C0      db  4
.data:0804B1C1      db  0
.data:0804B1C2      db  0
.data:0804B1C3      db  0
```

Ta sử dụng phương pháp vét cạn để tìm ra được input. Ta để ý rằng với kết quả của phép tính $s[i] \& 0xf$ sẽ có giá trị tối đa được biểu diễn bằng 4 bit nên có giá trị tối đa là 15 và tối thiểu là 0 tương ứng với các index của mảng array_3854.

Ta thực hiện phương pháp vét cạn để tìm input:

```
#phase5
def check(a1):
    data = [0x2,0xa, 0x6, 0x1, 0xc, 0x10, 0x9, 0x3, 0x4, 0x7, 0xe, 0x5, 0xb, 0x8, 0xf, 0xd]
    v3=0

    arr=[i for i in a1]
    for i in arr :
        v3+= data[ord(i)&0xF]

    if(v3==46): return True
    else: return False

result=[]
for i in range (0,1000000):
    input_num= str(i).zfill(6)

    if(check(input_num)):
        print(input_num)
        break
```

Kết quả của chương trình trên có rất nhiều tuy nhiên đều có điểm chung. Ta phân tích một giá trị có được của chương trình trên là 000445. Sau khi kết thúc vòng lặp thì giá trị $v3 = \text{array_3854}[0] + \text{array_3854}[0] + \text{array_3854}[0] + \text{array_3854}[4] + \text{array_3854}[4] + \text{array_3854}[5] = 2 + 2 + 2 + 12 + 12 + 16 = 46$ thỏa mãn điều kiện của hàm. Các kết quả còn lại nếu thực hiện theo các bước như trên thu được kết quả 46 thì cũng thỏa mãn.

Input của phase5: 000445 và các kết quả khác

Check input

```
[*] Phase 5
-Hint: No hint is also a hint :)
000445
Amazing bomb solvers, the bomb has been deactivated. Enjoy your day :))
semloh@semloh-virtual-machine:/mnt/hgfs/D/OBJECTS/HK4/LAP TRINH HE THONG/lab/lab4$
```

Còn một điểm lưu ý đó là vì phép tính $s[i] \& 0xf$ chỉ lấy 4 bit cuối của $s[i]$ nên ta có thể cộng $s[i]$ với một số là bội của 16 để tạo ra một kết quả input khác: (miễn là giá trị đó in được)

Ví dụ với $s = "000445"$ khi ta lấy giá trị ASCII của từng phần tử trong s cộng với bội 16 (ví dụ 32) thì ta sẽ thu được chuỗi mới là $s' = "PPPTTU"$. Ta kiểm tra chuỗi này với chương trình:

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

Hoặc khi cộng với 64 thì ta thu được chuỗi s=" pppttu". Ta kiểm tra chuỗi này với chương trình:

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

Do các kết quả trên chỉ dựa theo giá trị của mảng array_3854 với index từ 0 -> 9. Ta chưa xét sự có mặt của các index từ 10 đến 15. Nếu nhập thẳng trực tiếp như index thì sẽ không thỏa độ dài input nên ta chỉ có thể cộng nó với bội của 16 để tạo nên một kí tự ASCII với 4 bit cuối có giá trị từ 10 đến 15. Ta có thể quan sát ví dụ sau để hiểu rõ hơn:

Ta tách $46 = 8+8+8+8+8+6 = \text{array_3854}[13] + \text{array_3854}[13] + \text{array_3854}[13] + \text{array_3854}[13] + \text{array_3854}[13] + \text{array_3854}[2] \Rightarrow s[0]=13, s[1]=13, s[2]=13, s[3]=13, s[4]=13, s[5]=2$

Lúc này ta không thể nhập s là "13131313132" vì sẽ không thỏa điều kiện độ dài của s là 6, còn nếu chuyển sang dạng kí tự để nhập cũng không được vì đây là các kí tự không in được. Nên ta cần cộng giá trị ASCII của mỗi phần tử của s với bội của 16 cụ thể là 64 để chuyển thành kí tự nhập được:

Thì ta sẽ thu được chuỗi s' là "MMMMMB". Thử chuỗi này với chương trình:

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

Sau khi giải xong các phase ta thu được file flag.txt chứa các input của Phase1 tới Phase5:

```
15 30 60 120 240 480
hiendtt@uit.edu.vn
0 v 835
10 5
000445
```

Kiểm tra file flag.txt:

```
semloh@semloh-virtual-machine:/mnt/hgfs/D/OBJECTS/HK4/LAP TRINH HE THONG/lab/lab4$ ./nt209-uit-bomb flag.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!
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 :)
Amazing bomb solvers, the bomb has been deactivated. Enjoy your day :)
semloh@semloh-virtual-machine:/mnt/hgfs/D/OBJECTS/HK4/LAP TRINH HE THONG/lab/lab4$
```