**Báo cáo Thực hành**

**Lab 4: Kỹ thuật dịch ngược (tiếp theo)**

Nhóm 6:

Lê Minh Nhã - 20521690

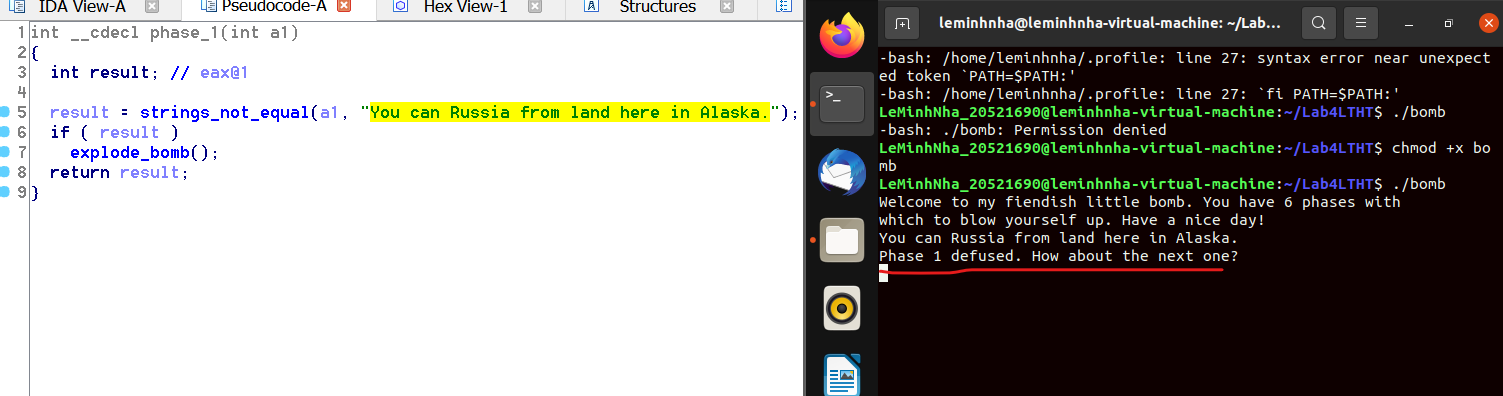
Vương Đinh Thanh Ngân - 20521649

**Phase 1:**

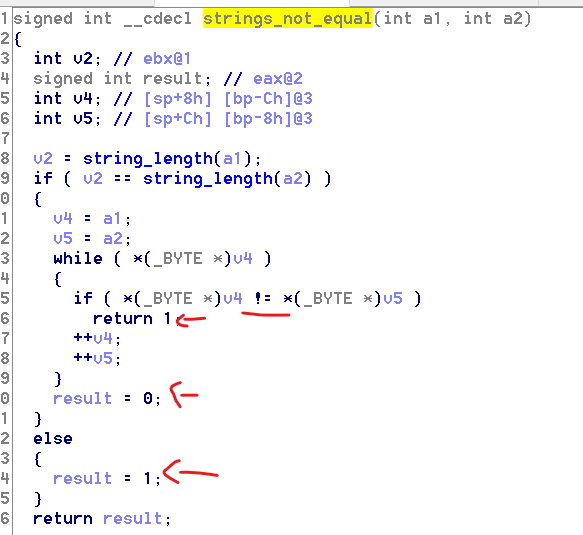
- Nhận thấy mã giả của phase 1 có một hàm string\_not\_equal(a1, “chuỗi...”) (hình 1), có thể hiểu là hàm này dùng để so sánh giá trị nhập vào a1 với chuỗi có sẵn nếu không bằng nhau trả về 1 (đúng), bằng trả về 0 (sai) (hình 2)

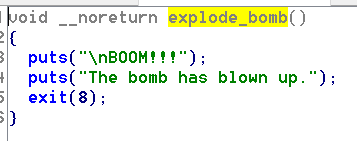
- Tại hình 1 ta thấy có hàm if(result) nếu result = 1 thì chạy hàm explode\_bomb() (quả bomb sẽ bị nổ)(hình 3) => vậy kết quả result = 0 được trả về từ hàm string\_not\_equal(a1, “chuỗi...”) và việc so sánh a1 với chuỗi được ép kiểu về int “You can Russia from land here in Alaska.” thì a1 chính là chuỗi và là đáp án cần tìm.

Khóa: You can Russia from land here in Alaska.



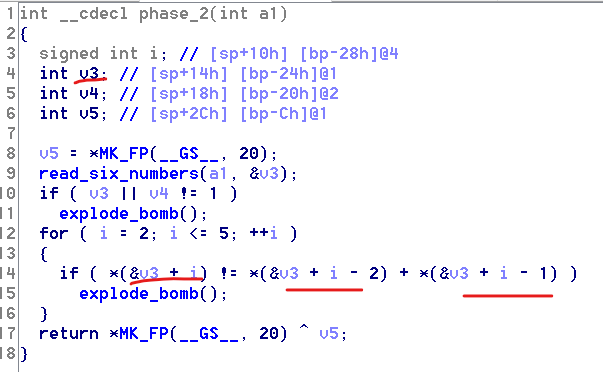
Hình 1

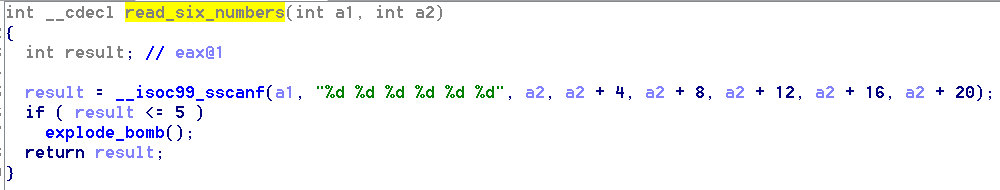
 Hình 2

hình 3

**Phase 2:**

- Kiểm tra mã giả, ta nhận thấy biến v3 có khả năng là một mảng kiểu int với 6 phần tử vì ở hàm read\_six\_numbers(a1, &v3) là so sánh 6 kí tự của v3 với a1 đầu vào ta nhập(hình 1 và 2)

 hình 1

 hình 2

- Ta thực hiện đổi v3 về kiểu mảng với 6 phần tử thì nhận thấy v4, v5 được chuyển về theo kiểu mảng của v3 (hình 3)

- Tại dòng if(v3[0] || v3[1] != 1) thì khi một trong hai hoặc cả hai điều kiện đúng tức v3[0] ==0 v3[1] != 1 đúng thì chương trình sẽ nhảy đến hàm explode\_bomb() (hình 4) thì bomb sẽ nổ

=> để làm bomb không nổ thì điều kiện if phải là cả hai cùng sai => v3[0] == 0, v3[1] == 1

Vậy ta sẽ coi như v3[0] = 0, v3[1] = 1

- Tại vòng for có i chạy trong khoảng [2;5], với điều kiện if ta lại có nếu v3[i] != v3[i-2] + v3[i-1] đúng thì bomb sẽ nổ => để bomb không nổ thì điều kiện về sai tức phải là v3[i] == v3[i-2] + v3[i-1]

- Từ v3[0], v3[1] ta tính tiếp được:

+ v3[2] = v3[0] + v3[1] = 0 + 1 = 1 (i = 2)

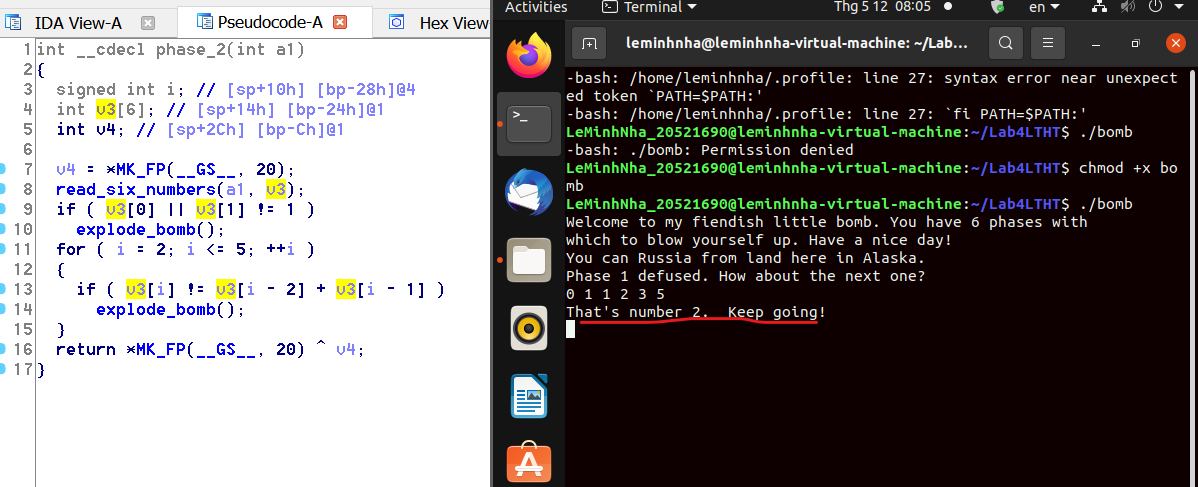
+ v3[3] = v3[1] + v3[2] = 1 + 1 = 2 (i = 3)

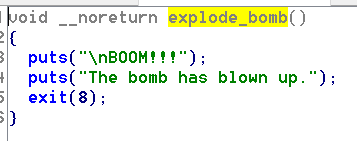
+ v3[4] = v3[3] + v3[2] = 2 + 1 = 3 (i = 4)

+ v3[5] = v3[4] + v3[3] = 3 + 2 = 5 (i = 5) => dừng vòng for

=> Vậy từ hàm read\_six\_numbers(a1, v3) thì suy ra được khóa cần tìm bằng các giá trị trong mảng v3 có => a1 = 0 1 1 2 3 5 (pass phase 1 hình 3)

Khóa : 0 1 1 2 3 5

 hình 3

 hình 4

**Phase 3:**

- Tại hình 1, ta thấy biến v6 = \_\_isoc99\_sscanf(a1, “%d %c %d”, &v4, &v2, &v5) => a1 chứa khóa nhập vào có các giá trị bằng v4 (số nguyên), v2 (ký tự char), v5 (số nguyên)

- Thấy rằng v4 là điều kiện đầu vào của switch nếu đúng thì chạy đến các case, ta xét ở dạng assembly của hàm switch thì nhận thấy v4 tại thanh ghi ebp + Var\_18 được so sánh với số 7 (hình 3), nếu v4 != 7 thì nhảy đến nhãn loc\_8048E14 ở đây gọi hàm explode\_bomb (hình 5) là vị trí bomb nổ, vậy để bomb không nổ thì v4 == 7 (nhận giá trị v4 = 7 )

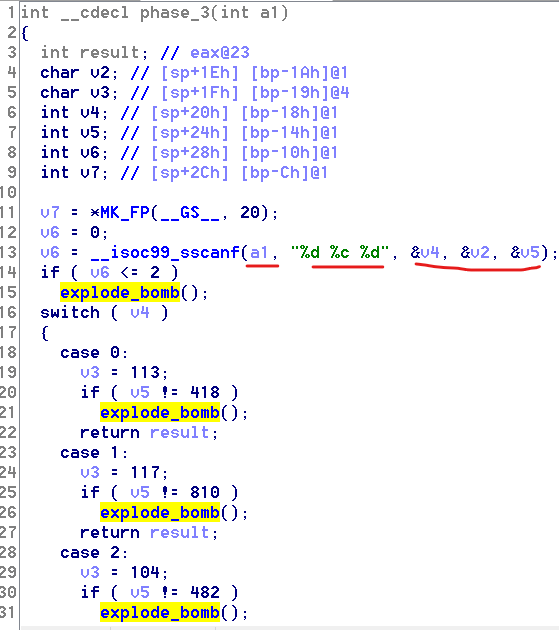
- Tại case 7:

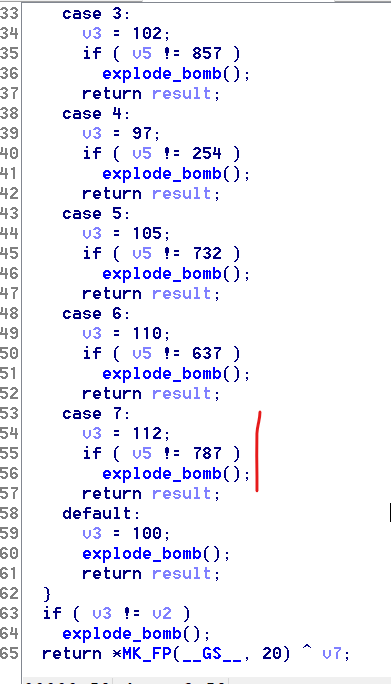
+ Switch(v4) với v4 = 7 gọi đến case có giá trị 7 (hình 2), tại hình 4 ta thấy v3 được so sánh với v2 nếu v3 != v2 thì nhảy đến hàm explore\_bomb() làm bomb nổ=> v3 == v2 thì bomb không nổ và có v2 là ký tự char mà v3 ở đây là một số kiểu decimal, nên ta chuyển v3 về char => v3 = ‘p’ (hình4)

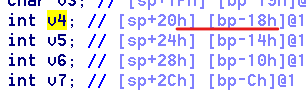
+ Tiếp điều kiện if với v5 != 787 thì nhảy đến hàm explode\_bomb() làm bomb nổ => v5 == 787 thì đúng (nhận v5 = 787)

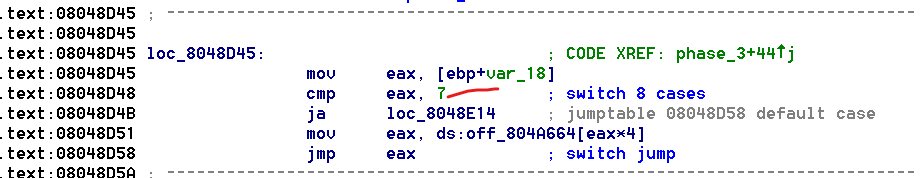
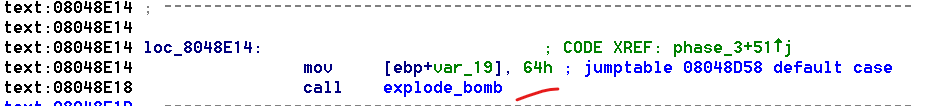
=> Vậy a1 nhận các giá trị lần lượt là v4, v2, v5 : 7 p 787 (hình 4)

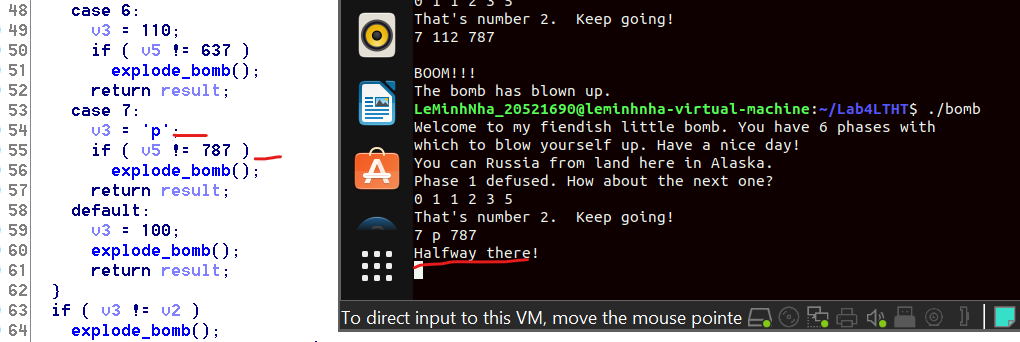
Khóa: 7 p 787

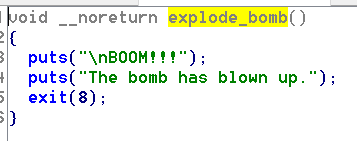
 hình 1

 hình 2



hình 3 (dạng assembly của hàm switch-case)

hình 4

hình 5

**Phase 4:**

- Từ mã giả ta thấy a1 được nhập vào được biểu diễn là 2 số nguyên và là v2, v3 (hình 1)

- Tại dòng if(v4 != 2 || v2 < 0 || v2 > 14) sẽ nhảy đến hàm explore\_bomb() (hình 1.1)bomb sẽ nổ khi điều kiện đúng vậy để tránh bomb nổ => v4 == 2, v2 > 0, v2 < 14 ( nhận v4 = 2 đúng, v2 thuộc [0;14]) (hình 1)

- Tại dòng if( v6 != v5 || v5 != v3) thì nhảy đến hàm explode\_bomb() khiến bomb nổ => để bomb không nổ thì v6 == v5, v5 == v3 mà ta có v5 =1 => v6 = v5 = 1, v3 = v5 = 1 (hình 1)

- Thấy v6 = func4(v2, 0, 14); => tìm được v2 từ đây với hàm hàm func4() (hình 2), từ mã giả hàm func4() ta viết lại chương trình với c++ thu được kết quả của v2 với 3 giá trị lần lượt là 8, 9, 11

- Từ trên ta có a1 với hai giá trị là v2, v3 thì ta có các cặp khóa lần lượt là 8 1, 9 1, 11 1 (hình 3.1, 3.2)

Khóa: 8 1, 9 1, 11 1

Code c++: tìm v2

#include <iostream>

#include <math.h>

using namespace std;

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

{

int result = 0; // 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;

}

int main()

{

int v2; // [sp+18h] [bp-20h]@

for (v2 = 0; v2 < 15; v2++)

{

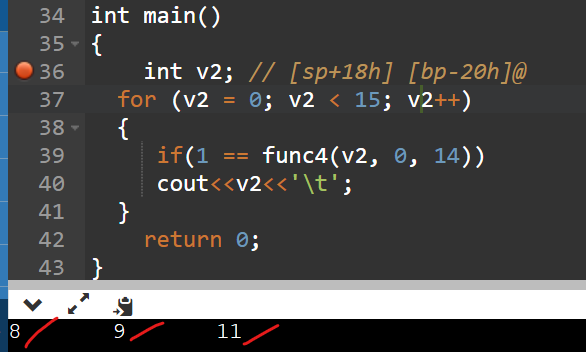
if(1 == func4(v2, 0, 14))

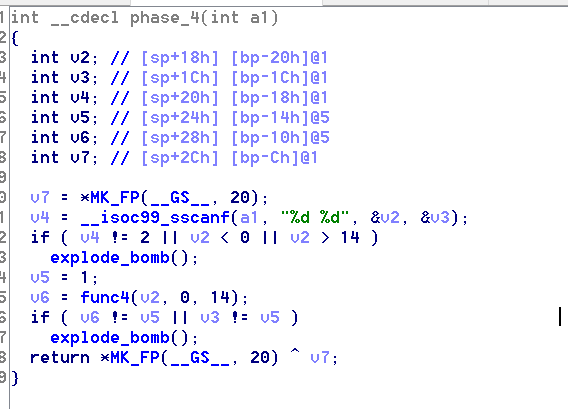
cout<<v2<<'\t';

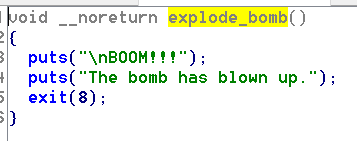
}

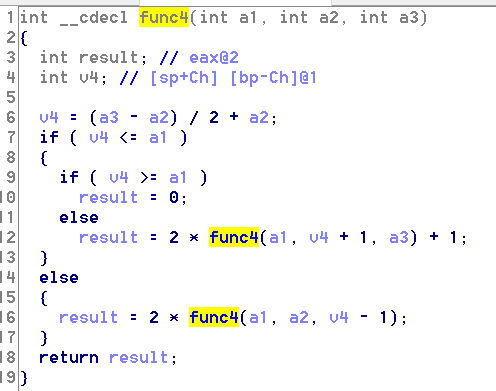
return 0;

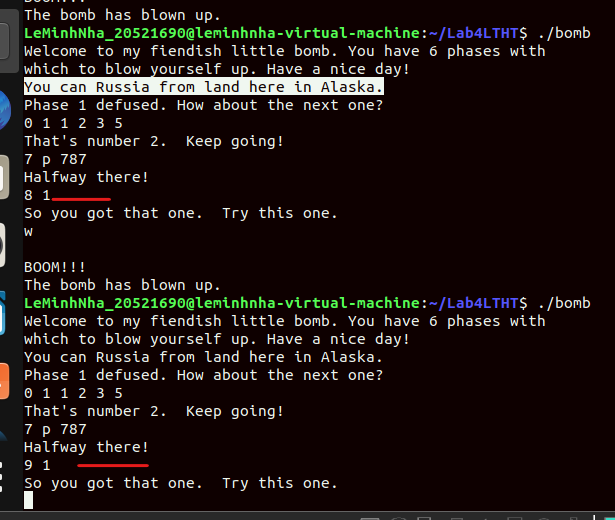
}

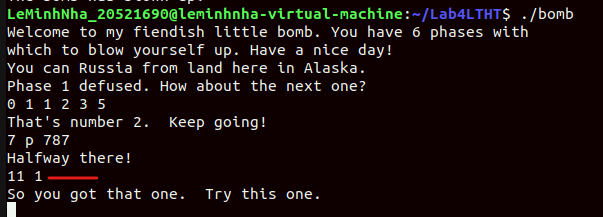
 hình 0 (kết quả chạy chương trình tìm v2)

 hình 1 (mã giả phase 4)

hình 1.1

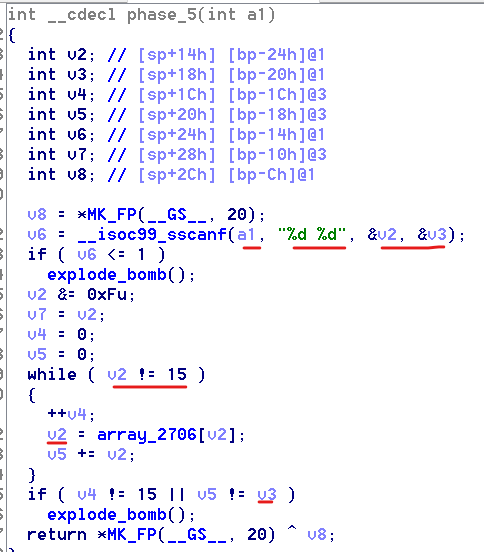
 hình 2 (mã giả hàm func4())

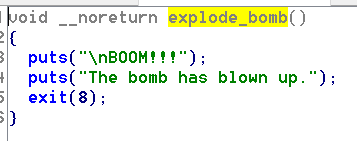
hình 3.1

hình 3.2

**Phase 5:**

- Mã giả phase 5, ta thấy a1 chứa 2 kí tự đầu vào và là khóa dạng số nguyên kí hiệu v2, v3

 hình 1

hình 1.1

- Từ vòng while ta có thể tìm được v2, v3 với v3 == v5 (và v4 == 15) thì bomb không nổ vì không đi vào vào hàm explode\_bomb() (hình 1.1)

- Khi ở đây ta cần tìm vị trí mảng làm v2 = 15 để dừng vòng while, và vị trí đó cũng là v2 cần tìm để xác định khóa, v3 = v5 = v5+v2 (lặp) => vậy thì ta kiểm tra khi mà không chắc chắn rằng vị trí v2 ban đầu vào là giá trị bao nhiêu và bắt đầu từ vị trí nào ta thực hiện chạy chương trình với nội dung là muốn liệt kê ra tổng giá trị v5 nhận được khi v2 bắt đầu random với các giá trị tồn tại trong mảng array\_2706[] cho đến khi thoát vòng while()

- Nhưng để bom không nổ thì còn chắc chắn v4 = 15 và trong vòng while() ++v4 bắt đầu từ v4 = 0 thì phải xét khi v4 = 15 là khi v2 và v5 có giá trị bao nhiêu và vị trí v2 ban đầu bắt đầu từ số nào trong khoảng [0; 15]

- Kết quả chạy chương trình (hình 2), ta thấy được vị trí ban đầu v2 = 5, tổng ta có của v5 = 115 (= v3) => a1 nhận 2 số là 5 115 là khóa (hình4)

Khóa: 5 115

Code c++:

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

for(int i = 0; i< 15; i++)

{

int v2 = i & 15;

cout<<"Vi tri ban dau cua v2: "<<i<<"\n";

int v4 = 0;

int v5 = 0;

int array\_2706[16] = {10, 2, 14, 7, 8, 12, 15, 11, 0, 4, 1, 13, 3, 9, 6, 5};

//Vị trí của mảng được xác định như hình 3.1, 3.2 với giá trị đã được đổi từ hex -> dec (thập lục phân -> thập phân)

while(v2 != 15)

{

if(v4 !=15)

{

++v4;

v2 = array\_2706[v2];

v5+=v2;

cout<<v4 <<": v2 = "<<v2<<"\t v5 = "<<v5 <<"\n";

}

}

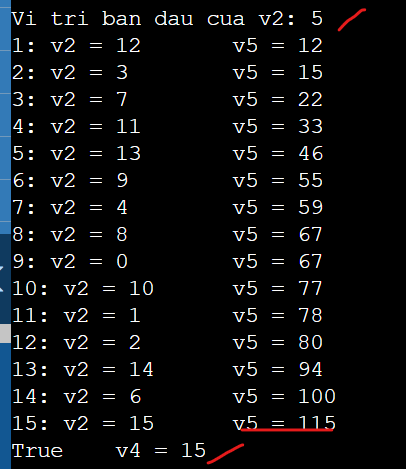
if(v4 !=15) cout<<"False\tv4 = "<<v4<<"\n";

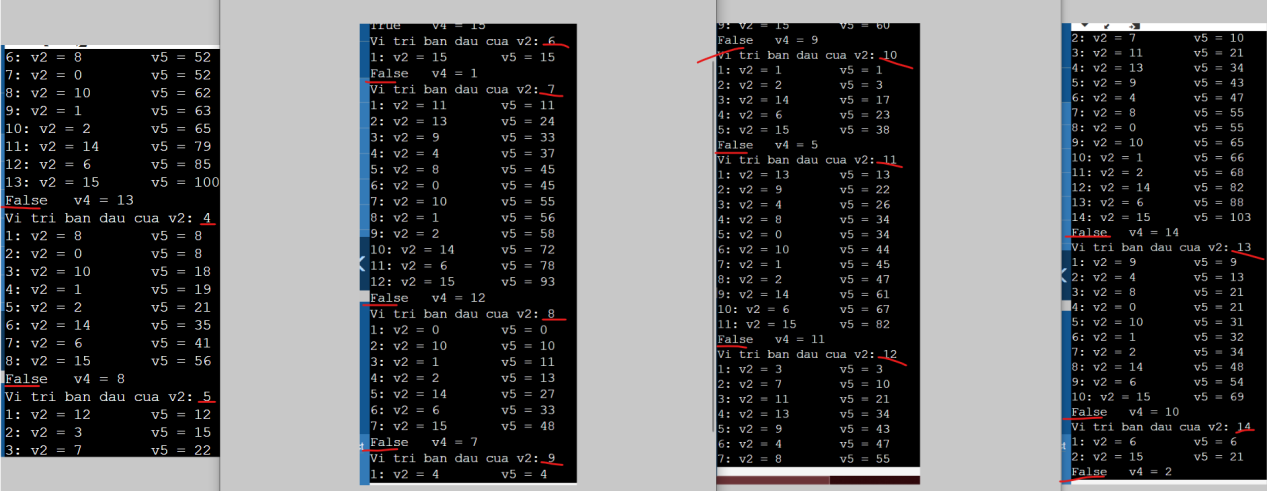
else

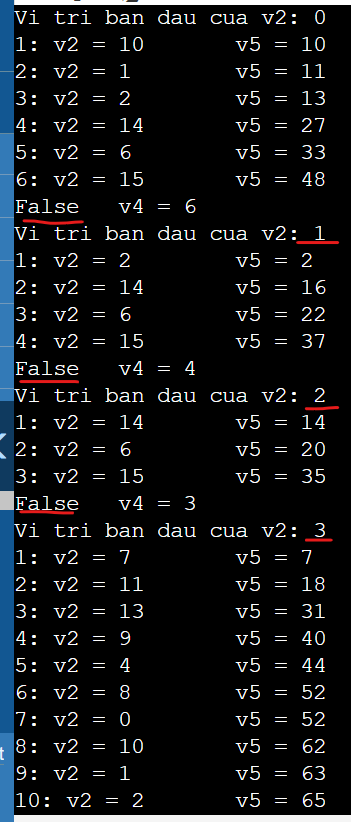
cout<<"True\tv4 = "<<v4<<"\n";}

return 0;

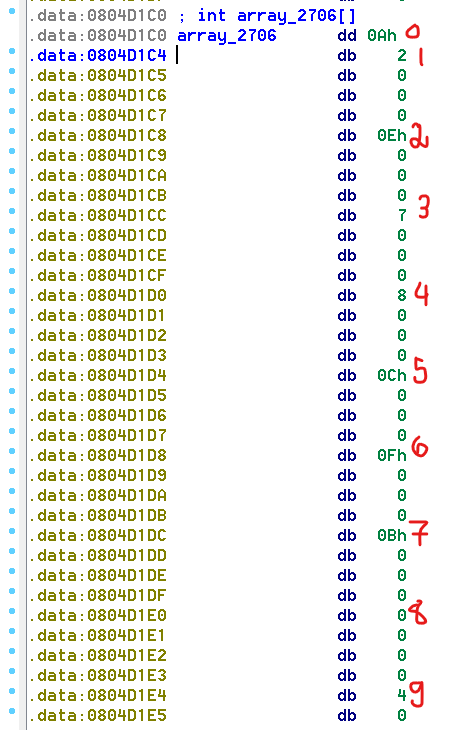
}

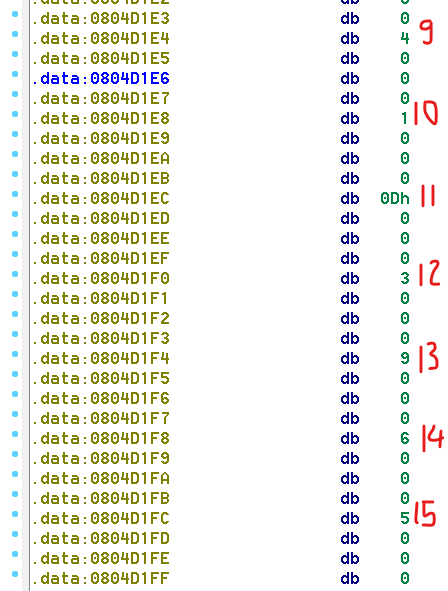
 hình 2

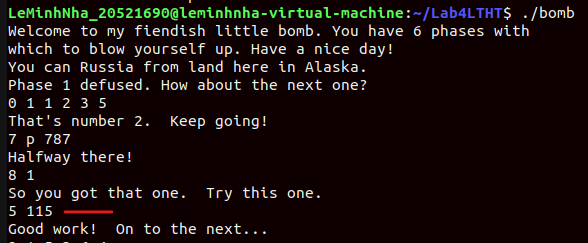
hình 2.1 kết quả chạy chương trình với vị trí v2 có v4 != 15



hình 2.2 kết quả chạy chương trình với vị trí v2 có v4 != 15

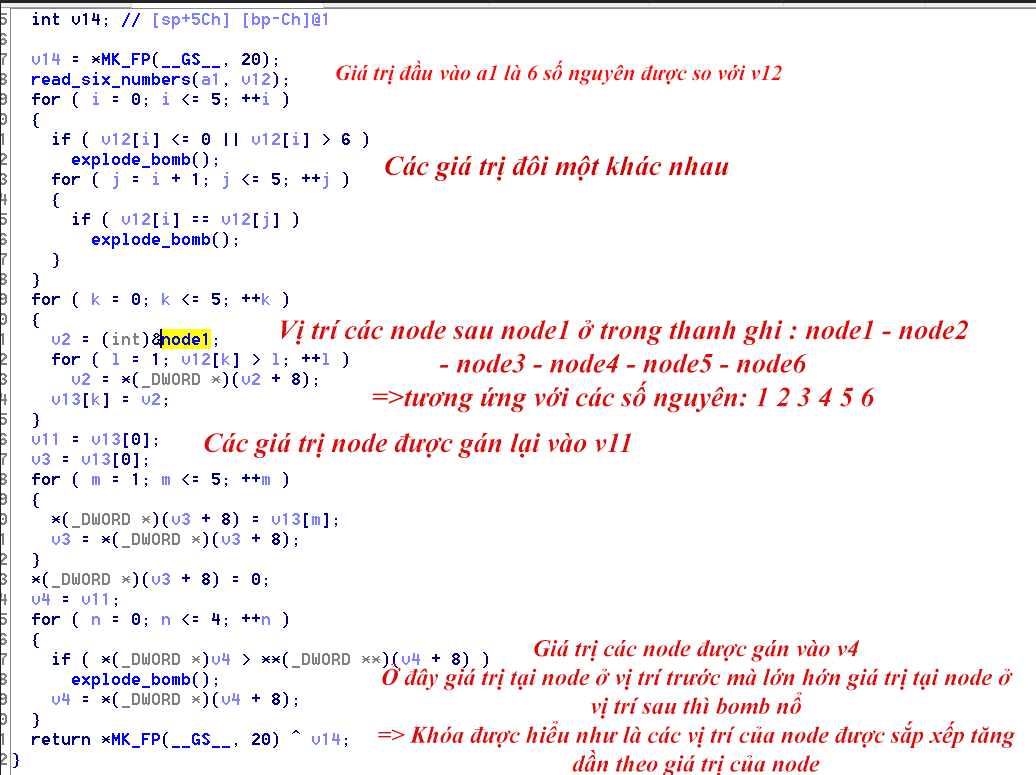
hình 3.1 (array\_2706[])

hình 3.2(array\_2706[])

 hình 4

**Phase 6:**

- Từ mã giả ta có: (hình 1)

hình 1

- Chạy file bomb trên linux với lệnh $gdb bomb để xác định giá trị của từng node ở kiểu hex, ta chuyển về hiểu dec được:(hình 2)

+(1) node1: 240

+(2) node2: 199

+(3) node3: 410

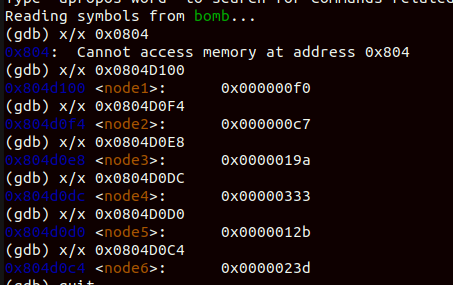
+(4) node4: 819

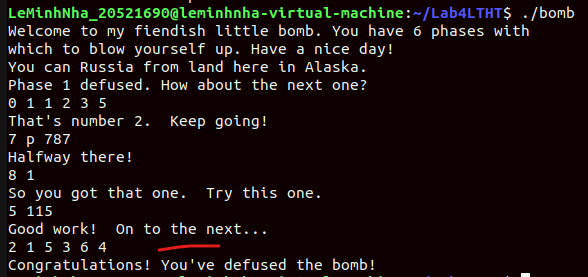
+(5) node5: 299

+(6) node6: 573

- Sắp xếp theo thứ tự tăng dần sẽ là: node2-node1-node5-node3-node6-node4 => Khóa có thể hiểu là: 2 1 5 3 6 4 (hình 3)

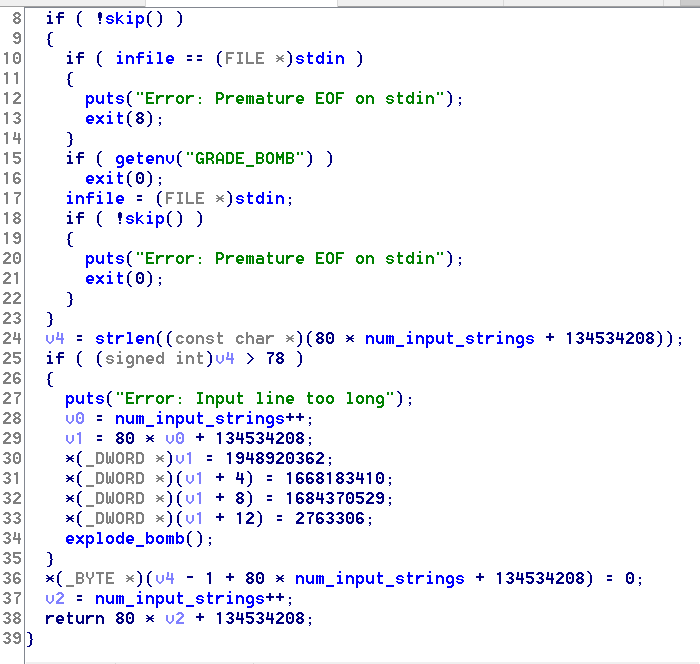
Khóa: 2 1 5 3 6 4

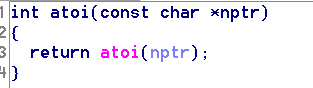
hình 2

hình 3

**Phase secret:**

- Từ mã giả ta nhận thấy được khóa của cần tìm sẽ bằng v2 từ hàm read\_line() mà atoi(nptr)

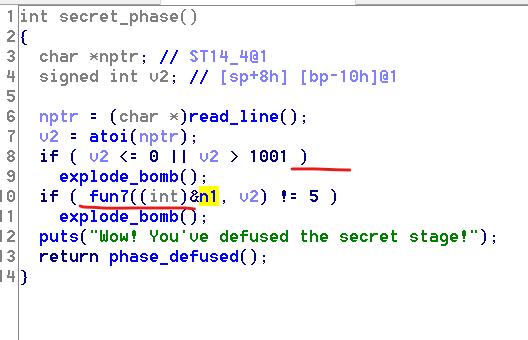
 hình: hàm read\_line()

 hình: hàm atoi()

- Để bomb không nổ thì v2 thuộc [1, 1001] hình 1

- Xác định v2 thông qua hàm fun7((int)&n1, v2) hình 1

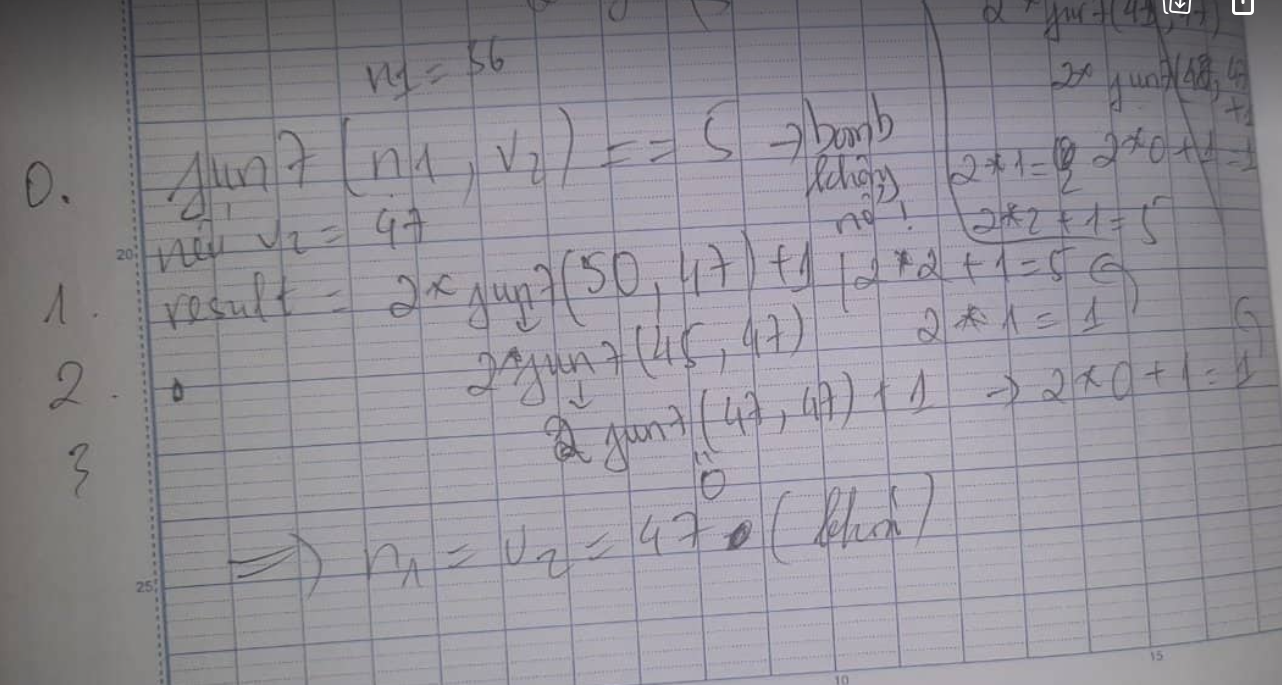
- Từ n1 và hàm fun7 ta nhận thấy các giá trị được có thể được tìm thấy ở v2 là được sắp xếp theo dạng nhị phân hình 3 và hình 3.1...n

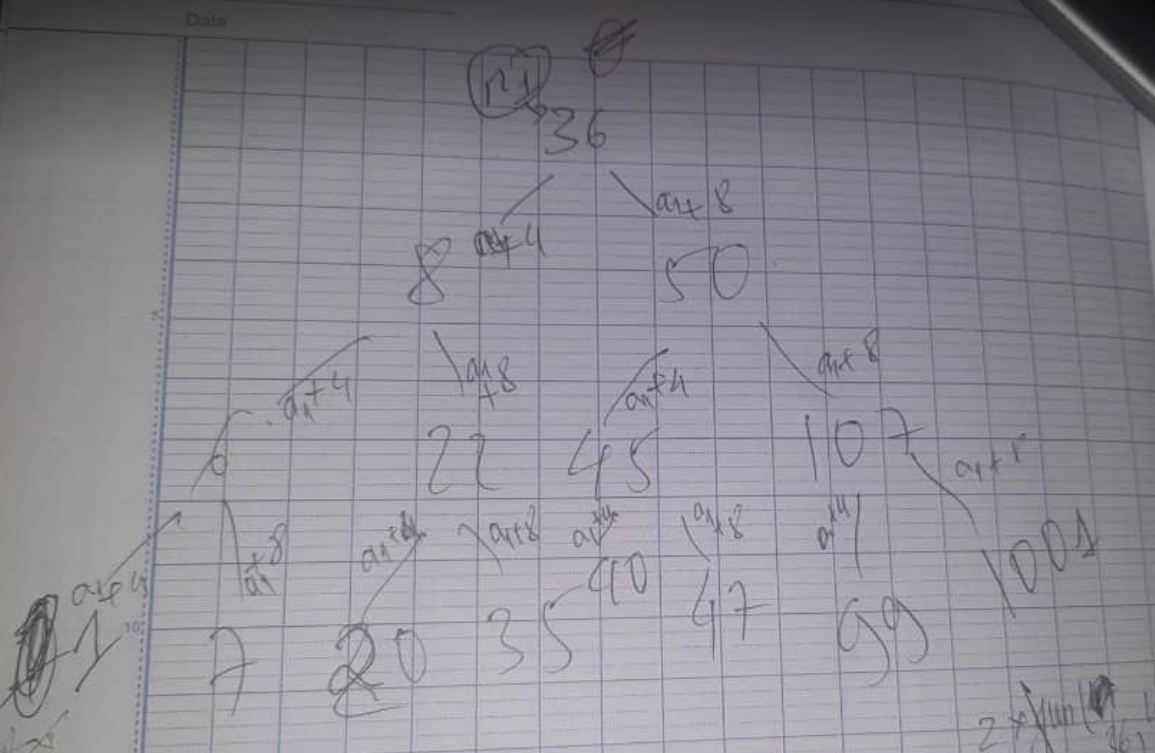
 hình 1

- Mà để bomb không nổ thì hàm fun7(...) phải trả về kết quả bằng 5

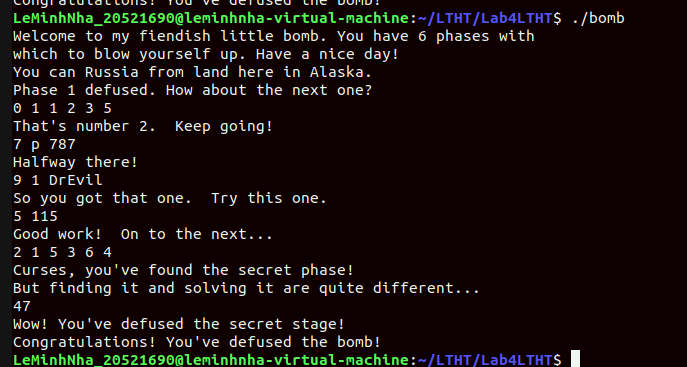
- Hàm fun7(...) là một hàm đệ quy, thì sau khi thử các trường hợp của v2 thì em tìm được giá trị v2 = 47 thì sẽ trả kết quả về cho hàm fun7(...) = 5 (hình 2) và nhập khóa thì được kết quả phá bomb phase secret (hình 0)

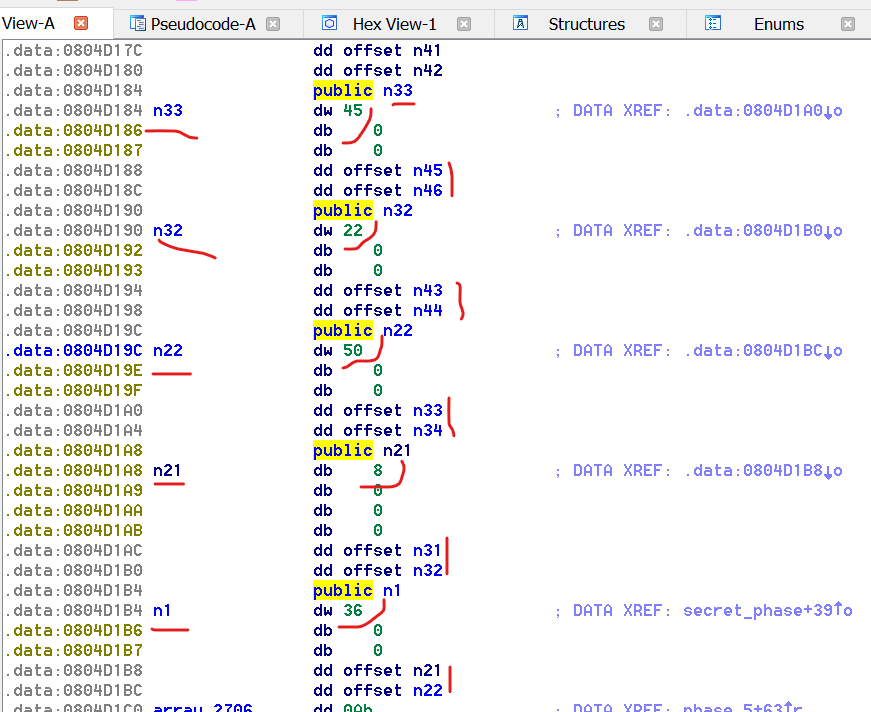
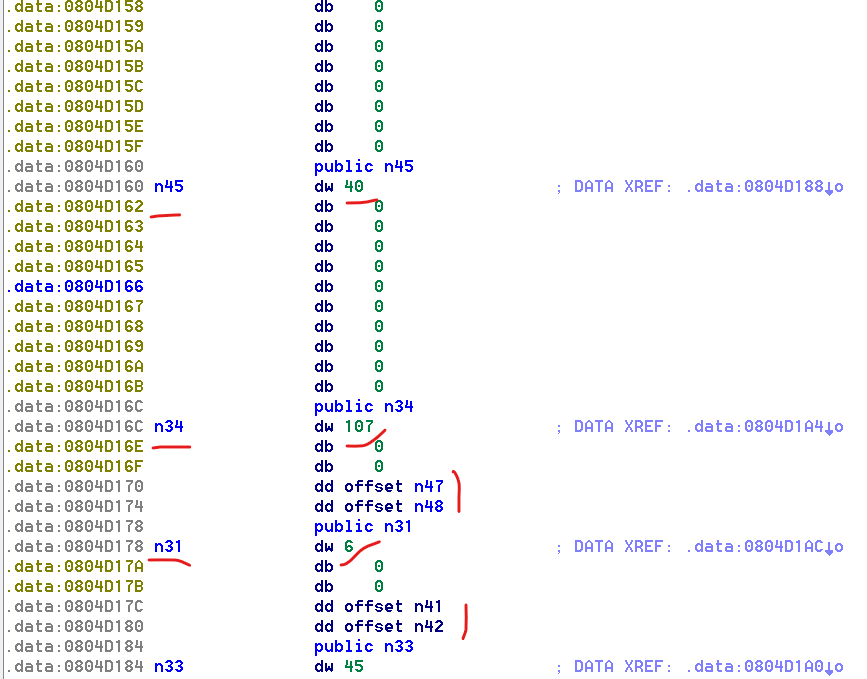
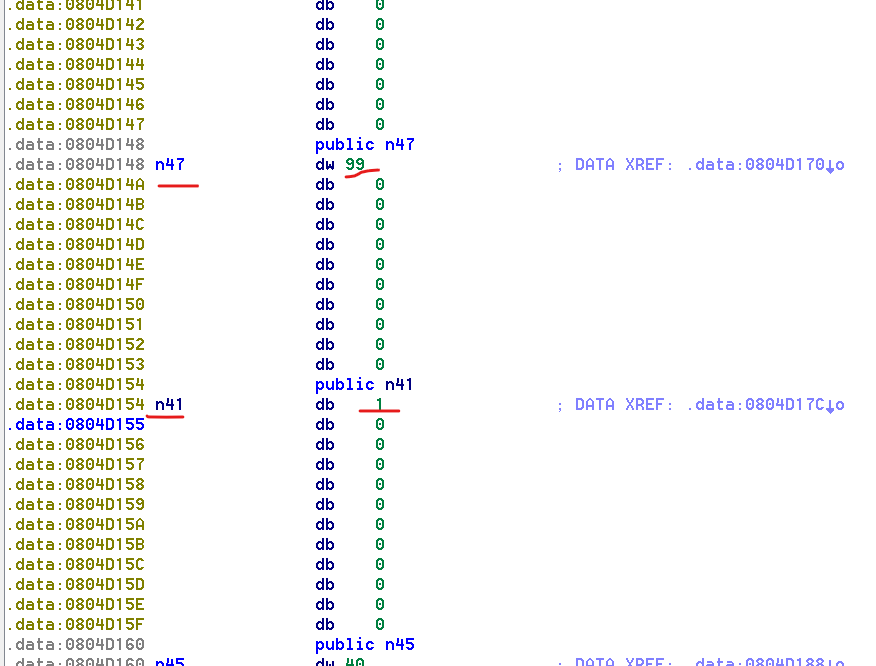
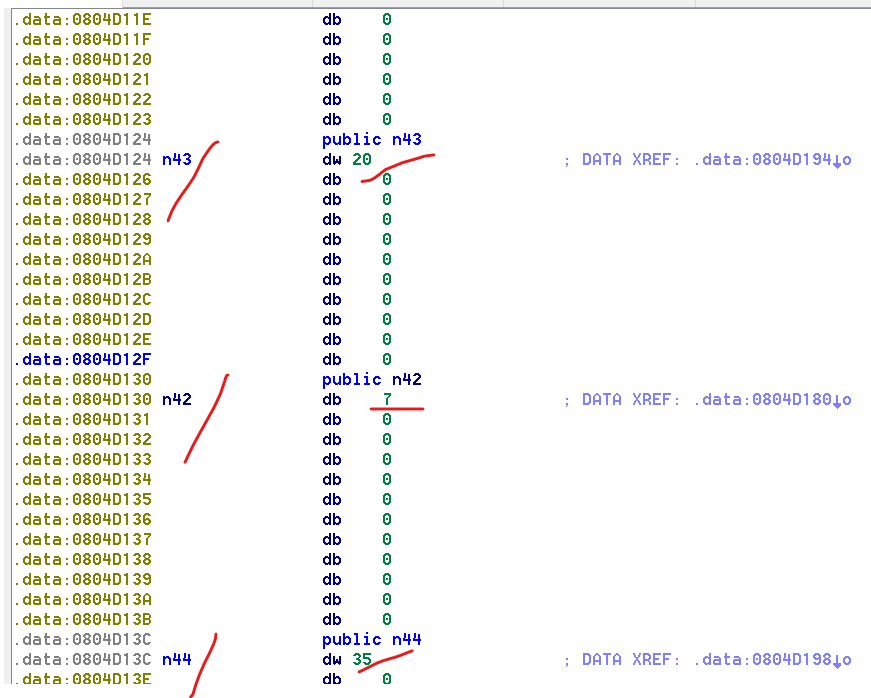
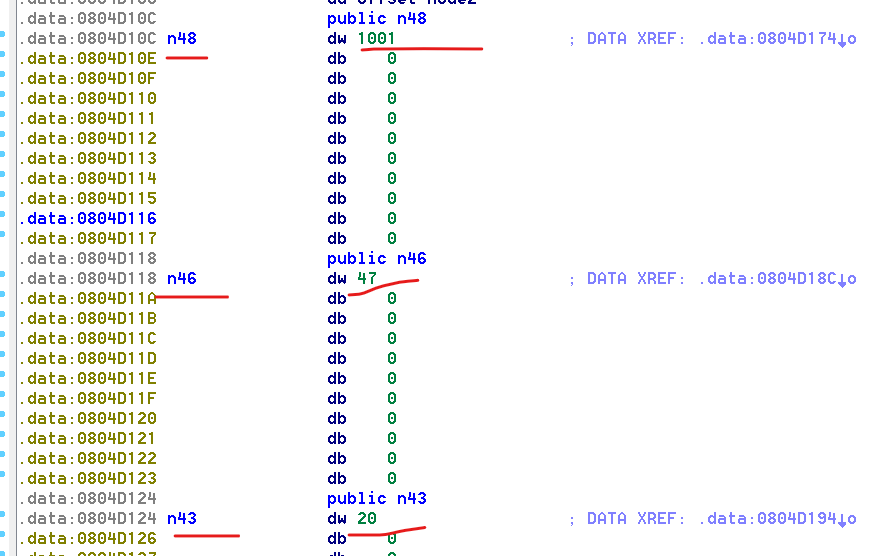
=> Khóa: 47

hình 2

 hình 3

Hoàn thành

 hình 0

 hình 3.1...n