**LẬP TRÌNH HỆ THỐNG– LỚP NT209.L21.ANTN**

**RE CHALLENGES 3: XBS**

| Giảng viên hướng dẫn | Phạm Văn Hậu | | ĐIỂM |
|---|---|---|---|
| Sinh viên thực hiện 1 | Trần Đức Lương | 19521815 | |

Đây là file thực thi ELF 64-bit. Thực hiện chạy thử chương trình.

```
┌──(janlele91㉿kali)-[~/Documents/RE Challenges/Release_3]
└─$ ./a.out
test
Try again!
```

Mở file a.out bằng IDA Pro thực hiện quá trình dịch ngược. Mở subview strings thì thấy xuất hiện dòng "Congrats!" chính là mục tiêu hướng đến. Truy xuất thì thấy nó nằm trong hàm main.

| | | | |
|---|---|---|---|
| .rodata:000000... | 0000000B | C | Try again! |
| .rodata:000000... | 0000000A | C | Congrats! |

Thực hiện phân tích hàm main:

```
  v11 = (char *)a;
  v12 = 0;
  do
  {
    v12 += *(_DWORD *)v11;
    v11 += 4;
  }
  while ( v11 != (char *)&a[31] );
  v13 = "Congrats!";
  if ( v12 != 1073840184 )
    goto LABEL_18;
LABEL_19:
  puts(v13);
```

Để chương trình in ra chuỗi "Congrats!" thì giá trị v12 là tổng của tất cả các phần tử trong mảng a phải bằng 107384184. Ban đầu mảng a gồm 31 phần tử được khởi tạo bằng 0. Ta sẽ đi tìm cách thay đổi giá trị trong mảng a.

Chương trình yêu cầu nhập vào 5 số tương ứng với 5 vòng lặp. Chương trình sẽ gán giá trị hex của mỗi số vừa nhập vào v4 và xử lí.

```
  do
  {
    __isoc99_scanf(&unk_2004, &v15, envp);
    v4 = v15;
    v5 = 30LL;
    v6 = 0;
    v7 = 0;
    while ( ((v4 >> v5) & 1) == 0 )
    {
LABEL_9:
      if ( v5-- == 0 )
      {
        if ( v6 )
          v15 = v4;
        goto LABEL_12;
      }
    }
    v8 = a[v5];
    if ( v8 )
    {
      ++v7;
      v4 ^= v8;
      v6 = 1;
      goto LABEL_9;
    }
    if ( v6 )
      v15 = v4;
    a[(int)v5] = v4;
LABEL_12:
    v10 = v7 <= 1;
    envp = (const char **)(unsigned int)(v7 - 1);
    if ( v10 && v3 > 1 )
    {
LABEL_18:
      v13 = "Try again!";
      goto LABEL_19;
    }
    ++v3;
  }
  while ( v3 != 5 );
```

Chú ý chương trình sẽ chỉ thực hiện in ra chuỗi sai “Try again!” kể từ số thứ 3 trở đi (do v3>1).

Với mỗi số nhập vào, chương trình sẽ đi tìm vị trí v5 là vị trí xuất hiện bit 1 lần đầu tiên tình từ trái sang của số đó.

- Với a[v5] == 0 tức là a[v5] chưa được gán nên a[v5] = v4.

- Với a[v5] != 0 tức là a[v5] vừa được gán ở số trước:  Chương trình tăng v7 lên 1 rồi tiếp tục quay lại vòng lặp tìm vị trí bit 1 xuất hiện lần đầu kể từ trái sang của v4 sau khi xor với a[v5].

Để chương trình tránh  "Try again!" v7 phải > 1 tức là phải đi vào thực thi block if(v8) ít nhất 2 lần.

Từ những phân tích trên, em đề xuất ra ý tưởng là sẽ tách 1073840184 = 1073840183 + 1. Cụ thể:

- x1 = 1073840183 sẽ là số đầu tiên và nó sẽ được gán vào a[30].
- Ở số thứ 2 ta cần nhập vào 1 số x2 cũng có vị trí bit 1 xuất hiện giống với x1 và x2 ^ x1 phải bằng 1 (khi đó v4 = 1 => v5 = 0) để gán a[0] = 1. Để ý 2 số xor nhau bằng 1 chỉ có thể là hơn kém nhau 1 đơn vị, khi đó ta chọn x2 = 1073840182.
- Bắt đầu từ số thứ 3 trở đi, chúng ta phải nhập một số x3 sao cho v7 = 2 và v4 sau khi xor 2 lần phải bằng 0 (để không thể gán giá trị mới vào mảng a nữa). Ở đây ta sẽ lấy chính số x2 để làm số x3. Khi v7 = 1: v4 = x3 ^ x1 = 1 => v5 = 0. Chương trình kiểm tra thì thấy a[0] đã tồn tại và a[0] = 1 nên tiếp tục tăng v7 = 2, khi đó v4 = v4^a[0] = 1^1 = 0 mà v5 = 0 nên chương trình nhảy tới LABEL_12 để kiểm tra thì thấy thỏa mãn nên cho phép nhập số thứ 4. Có thể thấy mảng a sau khi xử lí số thứ 3 không thay đổi.
- Tương tự số thứ 3 ta cũng sẽ nhập số thứ 4 và số thứ 5: x4 = x2 và x5 = x2.

Khi đó tổng các phần tử mảng a sẽ đảm bảo bằng 1073840183 + 1 = 1073840184. Khi đó chương trình sẽ in ra chuỗi "Congrats!".