crackme_by_chrisK_v02

Chạy thử chương trình

```
C:\Users\BILL\Desktop\Release_3\crackme_by_chrisK_v02>.\crackme_by_chrisK_v02.exe
Enter Password: fsaf
Wrong
C:\Users\BILL\Desktop\Release_3\crackme_by_chrisK_v02>
```

Phân tích với IDA

```
14  v9 = 0;
15  printf("Enter Password: ");
16  scanf("%s", Str);
17  if ( strlen(Str) > 9 && Str[5] == 46 )
18  {
19    printf("Enter password key: ");
20    scanf("%s", Str2);
21    while ( 1 )
22    {
23      v4 = i++;
24      if ( !Str[v4] )
25        break;
26      ++v9;
27    }
28    v8 = (char *)malloc(4 * v9);
29    srand(v9);
30    for ( i = 0; i < v9; ++i )
31    {
32      v5 = &v8[4 * i];
33      *(_DWORD *)v5 = rand() % v9 + v9 / -2;
34    }
35    magic(Str, v8);
36    if ( !strcmp(mstring, Str2) )
37    {
38      printf("Nice!!");
39      result = 0;
40    }
41    else
42    {
43      printf("Wrong");
44      result = 1;
45    }
```

Bài này khá đơn giản. Chương trình yêu cầu nhập password và lưu vào chuỗi Str, sau đó để in ra được dòng chữ “Nice”, password nhập vào cần >9 kí tự, và kí tự thứ 6 là ‘.’ Sau đó chương trình yêu cầu nhập password key và lưu vào Str2. Sau đó gọi hàm strcmp(mstring, Str2). Chuỗi Str2 không thay đổi từ lúc nhập vào, còn chuỗi mstring được tạo ra từ hàm magic(Str, v8) dựa vào Str (chuỗi password).

```
1  char *__cdecl magic(int a1, int a2)
2  {
3    int v2; // eax
4    char *result; // eax
5    int i; // [esp+8h] [ebp-Ch]
6    int v5; // [esp+Ch] [ebp-8h]
7    int j; // [esp+Ch] [ebp-8h]
8
9    v5 = 0;
10   for ( i = 0; ; ++i )
11   {
12     v2 = v5++;
13     if ( !*(_BYTE *)(v2 + a1) )
14       break;
15   }
16   for ( j = 0; j < i; ++j )
17     mstring[j] += *(_BYTE *)(4 * j + a2) + *(_BYTE *)(j + a1);
18   result = &mstring[j];
19   mstring[j] = 0;
20   return result;
21 }
```

Hàm magic để tạo ra chuỗi mstring, ta có thể thấy chuỗi v8 được cấp phát vùng nhớ với độ dài (4 * v9) với v9 là độ dài của chuỗi password. Sau đó được gọi hàm random, tuy nhiên do tham số srand(v9) nên các giá trị random này không thay đổi. do đó ta có thể tính được dễ dàng và dựa vào vòng lặp để tính toán mstring[j] = password[j] + v8[j * 4].

Tuy nhiên chúng ta có một cách đơn giản hơn mà không phải tính tay, đó là debug và chặn ngay trước hàm strcmp để xem giá trị mstring

Đặt breakpoint và nhập password là “cobra.cobra”, chuỗi mstring là

```
.bss:00407080 _mstring db 66h, 6Fh, 62h, 73h, 65h, 2Ch, 5Eh, 72h, 5Eh, 74h, 5Eh, 0, 0, 0, 0, 0
```

Sử dụng python tìm chuỗi

```
>>> a = [0x66, 0x6f, 0x62, 0x73, 0x65, 0x2c, 0x5e, 0x72, 0x5e, 0x74, 0x5e, 0]
>>> print(''.join([chr(i) for i in a]))
fobse,^r^t^
```

Ta có chuỗi passwordkey cần nhập là “fobse,^r^t^”

```
C:\Users\BILL\Desktop\Release_3\crackme_by_chrisK_v02>.\crackme_by_chrisK_v02.exe
Enter Password: cobra.cobra
Enter password key: fobse,^r^t^
Nice!!
```

Password: cobra.cobra

Password Key: fobse,^r^t^