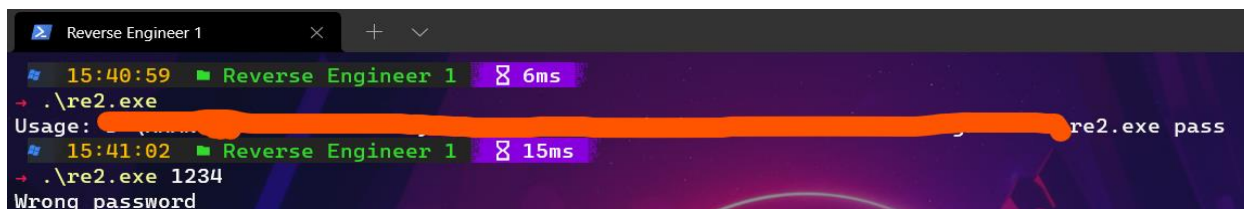


Re2.exe

Ta có 1 file PE (Window) **re2.exe**, chạy thử xem sao :



```
Reverse Engineer 1
15:40:59 Reverse Engineer 1 6ms
.\re2.exe
Usage: re2.exe pass
15:41:02 Reverse Engineer 1 15ms
.\re2.exe 1234
Wrong password
```

Tìm chuỗi **"Wrong Password"** để tìm xem chỗ xử lý ra chuỗi này. Vào **_View -> Open subviews -> Strings** (Shift + F12). _

```
.rdata:00404060 ; const char Buffer[]
.rdata:00404060 Buffer db 'Wrong password',0 ; DATA XREF: sub_401726:loc_4017AA↑o
.rdata:0040406F align 10h
```

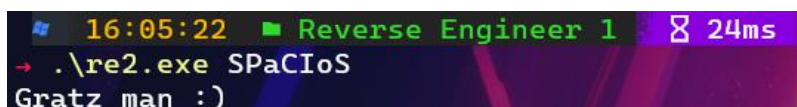
Ta thấy chuỗi này được lưu tại địa chỉ **0x00404060** tại segment **.rdata**. Tìm tất cả hàm có liên quan đến chuỗi này. *Chuột phải -> List cross references to (Ctrl + X) -> OK*. Ta thấy có duy nhất hàm **sub_401726** sử dụng chuỗi này. Phân tích hàm dưới dạng *pseudo-code*:

```
1 int __cdecl sub_401726(_BYTE *a1, int a2)
2 {
3     if ( a2 == 7 && *a1 == 83 && a1[1] == 80 && a1[2] == 97 && a1[3] == 67 && a1[4] == 73 && a1[5] == 111 && a1[6] == 83 )
4     {
5         printf("Gratz man :)");
6         exit(0);
7     }
8     return puts("Wrong password");
9 }
```

Lưu ý: Sử dụng thao tác *Chuột phải -> Jump to xref* lên tất cả các hàm ta sẽ biết được luồng gọi hàm như sau: **start() -> sub_401190() -> sub_4017B8() -> sub_401726** với tham số truyền vào **lớn hơn 1**. Còn các hàm khác thực hiện công việc như cấp phát vùng nhớ, kiểm tra số lượng tham số, ... Ta chỉ quan tâm đến hàm chính ở trên.

a2 là length của chuỗi nhập vào, **a1** là number user nhập. Nếu **ascii_code** của các ký tự tại các vị trí **0, 1, 2, 3, 4, 5, 6** thỏa: **83(S), 80(P), 97(a), 67(C), 73(I), 111(o), 83(S)** --> **"SPaCIoS"**, thì kết quả trả về là **"Gratz man :)"** thành công.

Kiểm tra lại:



```
Reverse Engineer 1
16:05:22 Reverse Engineer 1 24ms
.\re2.exe SPaCIoS
Gratz man :)
```

Password: **SPaCIoS**