re1.bin

Xem file format bằng **file** trên Kali Linux:

```
┌──(virus㉿virus)-[~/Desktop]
└─$ file re1.bin
re1.bin: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, in
terpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.9, not stripped
```

Vậy ta sẽ thực thi trên Linux để kiểm thử. Giờ thì dịch ngược với IDA, xem hàm main của chúng ta trong *pseudo-code* :

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3   char *s1; // [esp+1Ch] [ebp-Ch]
4   char *s1a; // [esp+1Ch] [ebp-Ch]
5
6   puts("############################################################");
7   puts("##        Bienvennue dans ce challenge de cracking        ##");
8   puts("############################################################\n");
9   printf("Veuillez entrer le mot de passe : ");
10  s1a = (char *)getString(s1);
11  if ( !strcmp(s1a, "123456789") )
12    printf("Bien joue, vous pouvez valider l'epreuve avec le pass : %s!\n", "123456789");
13  else
14    puts("Dommage, essaye encore une fois.");
15  return 0;
16 }
```

- 2 dòng đầu khai báo 2 biến con trỏ char **s1**, và **s1a**

- 4 dòng tiếp để in menu tên chương trình ra màn hình

- Sau đó biến **s1a** sẽ được xử lý với hàm **getString()** với đối số truyền vào là **s1** (Mạnh dạn đoán đây là hàm lấy ký tự người dùng nhập vào 🙃). Cứ vào hàm **getString()** cho chắc, vì đây không phải hàm thư việc C:

```
_BYTE *__cdecl getString()
{
  int *v0; // eax
  size_t v1; // eax
  int *v2; // eax
  int v4; // [esp+20h] [ebp-8h]
  _BYTE *ptra; // [esp+30h] [ebp+8h]

  v4 = 0;
```

Hàm getString()

+ Ta thấy hàm định nghĩa không cần đối số, vậy việc truyền biến **s1** vào không có ý nghĩa gì. Ban đầu hàm khai báo 1 số biến. Biến **v4** lưu index của string, giá trị ban đầu chuỗi rỗng nên **v4 = 0**

+ Sau đó, 1 bước chính là cấp phát động bộ nhớ cho dữ liệu nhập vào với hàm **malloc()** với size bộ nhớ là **2u**, ptra sẽ là chuỗi string được cấp phát. Sau đó kiểm tra cấp phát vùng nhớ nếu khả thi, nếu bộ nhớ RAM máy tính hết thì sẽ in lỗi.

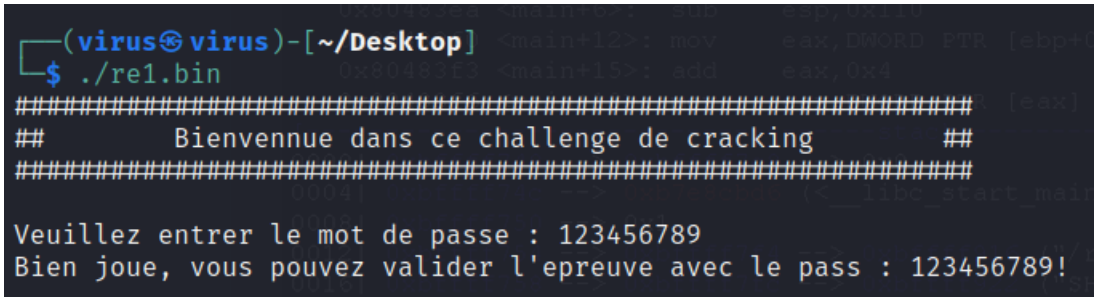
```
ptra = malloc(2u);
if ( !ptra )
{
  v0 = __errno_location();
  printError((int)"Allocating memory", *v0);
}
```

+ 1 bước chính khác còn lại là cho người dùng nhập vào bằng hàm **getchar()** với vòng lặp vô hạn **while(1),** hàm này sẽ **cast** ký tự ta nhập vào thành số, và chỉ lấy duy nhất 1 ký tự. Nếu ký tự nhập vào có ascii code bằng 10 (tức là xuống dòng) thì vòng lặp while **break** và **return ptra** (trả về kết quả nhập) và được **cast** lại sang char đưa vào biến **s1a**, sau đó gán **v1 = v4 + 2** và được cấp phát với size mới **v1**, tăng index **v4** lên 1 để đến với ký tự tiếp theo. Sau đó lại tiếp tục thực hiện tương tự như công đoạn trên: Cấp phát vùng nhớ nếu khả thi

```
while ( 1 )
{
  ptra[v4] = getchar();
  if ( ptra[v4] == 10 )
    break;
  v1 = v4 + 2;
  ++v4;
  ptra = realloc(ptra, v1);
  if ( !ptra )
  {
    v2 = __errno_location();
    printError((int)"Reallocating memory", *v2);
  }
}
ptra[v4] = 0;
return ptra;
```

Giải thích dài dòng cho vui vậy thôi, hàm **getString()** đơn giản là lấy input user (đúng như ta đoán :v). Sau đó nếu input này bằng "**123456789**" thì in ra dòng chữ **"Bien joue, vous pouvez valider l'epreuve avec le pass"** tức là password đúng.

Kiểm nghiệm lại:



Password: **123456789**