Curso C Reversing Solución Ejercicio 12 (by asOlOt)

Como es menester, abrimos con el IDA el ejemplo que Ricardo nos ha preparado, en este ejemplo se trata de trabajar con punteros (la verdadera piedra de toque del Lenguaje C).

```
:004012B0
                            call
                                        chkstk
:004012B5
                            call
                                        main
                                     eax, 40A00000h
:004012BA
                            mnu
                                     [ebp+var_4], eax
:004012BF
                            mnu
:004012C2
                                     [esp+28h+var_28], 4
                            MOV
:00401209
                            call
                                     malloc
                                     [ebp+var_8], eax
:004012CE
                            mov
                                     edx, [ebp+var_8]
:004012D1
                            mov
:004012D4
                            mov
                                     eax, 40D66666h
:004012D9
                            mov
                                     [edx], eax
                                     [esp+28h+var_28], 4
:004012DB
                            mov
:004012E2
                            call
                                     malloc
:004012E7
                            MOV
                                     [ebp+var_C], eax
                                     edx, [ebp+var_C]
:004012EA
                            mov
                                     eax, [ebp+var_8]
:004012ED
                            mov
:004012F0
                            f1d
                                     dword ptr [eax]
:004012F2
                            fadd
                                     [ebp+var 4]
:004012F5
                            fstp
                                     dword ptr [edx]
:004012F7
                            mov
                                     eax, [ebp+var_C]
```

Lo primero que vemos, que vamos a trabajar en la misma función Main, y que vamos a trabajar con la FPU (ya nos advierte Ricardo que vamos a trabajar con Floats).

Para saber a qué números floats corresponden podemos correr el programa y ver en la FPU que valores cogen, veremos que son: 5.0 y 6.7

Visto los valores, podemos renombrar las variables, Tambien notamos que reservamos memoria con el malloc, para almacenar un total de 4 bytes.

```
004012B5
                           call
                                       main
004012BA
                                    eax, 5.0
                           mov
004012BF
                                    [ebp+Num1], eax
                           MOV
00401202
                                    [esp+28h+var_28], 4
                           MOV
00401209
                           call
                                    malloc
004012CE
                                    [ebp+datos0], eax
                           MOV
004012D1
                                    edx, [ebp+datos0]
                           MOV
004012D4
                           mov
                                    eax, 6.7
004012D9
                           MOV
                                    [edx], eax
                                    [esp+28h+var_28], 4
004012DB
                           MOV
004012E2
                           call
                                    malloc
                                    [ebp+datos1], eax
004012E7
                           mov
004012EA
                           mov
                                    edx, [ebp+datos1]
004012ED
                           mov
                                    eax, [ebp+datos0]
                                    dword ptr [eax]
004012F0
                           f1d
004012F2
                           fadd
                                    [ebp+Num1]
004012F5
                           fstp
                                    dword ptr [edx]
```

Pasando a C este pequeño trozo quedaría a asi:

```
main()
{
     float Num1;
     float* datos0;
     float* datos1;

num1 = 5.0;
     datos0 = (float *) malloc(sizeof(float));

     *datos0 = 6.7;
     datos1 = (float *) malloc(sizeof(float));

     *datos1 = *datos0 + num1;
}
```

Solamente hemos asignado al campo datos0 el valor de 6.7, y luego lo hemos sumado al número 5.0 guardando el resultado en la memoria reservada apuntada por datos1.

Luego vemos como sigue el código, en este caso imprime el resultado de la suma:

```
104012FC
                                   [esp+28h+var_24]
                          fstp
                                   [esp+28h+var_28], offset aElValorPrefija ; "El valor p
10401300
                          mov
                                   printf
10401307
                          call
1040130C
                          mov
                                   [esp+28h+var_28], offset unk_40302C
10401313
                          call
                                   printf
                                   eax, [ebp+Num1]
dword ptr [esp+28h+var_24], eax
10401318
                          1ea
1040131B
                          Mov
                                   [esp+28h+var_28], offset unk_40305B
1040131F
10401326
                          call
1040132B
                          mov
                                   [esp+28h+var 28], offset unk 40305E
10401332
                          call
10401337
                          mov
                                   eax, [ebp+datos0]
1040133A
                                   dword ptr [esp+28h+var_24], eax
                          mov
                                   [esp+28h+var_28], offset unk_40305B
1040133E
                          mov
10401345
                          call
                                   scanf
                                   edx. [ebp+datos1]
1040134A
```

La cadena que se imprimirá la vemos en la siguiente captura:

```
เยษษยงยยย "";ury ษยงยยยม
:00403000 aElValorPrefija db 'El valor prefijado para la suma era %4.2f',0Ah,0
                                                         ; DATA XREF: _main+70fo
:00403000
:0040302B
                             align 4
:0040302C unk 40302C
                                 41h ; A
                                                        ; DATA XREF: main+7CTo
                             db
:0040302D
                             db
                                 68h ;
                                        h
:0040302E
                             db
                                 6Fh ; o
: 0040302F
                                 72h ;
                             db
                                        r
:00403030
                             db
                                 61h; a
:00403031
                             db
                                 20h
: 00403032
                                 65h ; e
                             dh
                                 73h ; s
:00403033
                             db
:00403034
                                 20h
                             db
:00403035
                             db
                                 74h : t
```

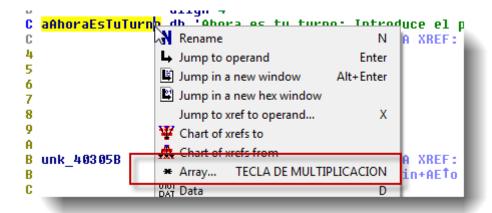
La vemos claramente "El valor prefijado.....", lo vemos bien pero en los siguiente printf no ocurre lo mismo, por ejemplo en lA dirección 04032C, no vemos la cadena alineada correctamente, esto lo solucionamos clickando el botón derecho encima de la dirección deseada y vemos la cadena formada por los bytes que siguen, si clickamos se nos presenta de una forma más legible:

```
---9"
                                                                  POTA XI
ı:0040302C <mark>unk 4030</mark>₽₽
                         Rename
                                                                  Ν
i:0040302D
1:0040302E
                         Jump to operand
                                                               Enter
1:0040302F
                         Jump in a new window
                                                            Alt+Enter
1:00403030
                         Jump in a new hex window
1:00403031
                            Jump to xref to operand...
                                                                  Χ
1:00403032
1:00403033
                         Chart of xrefs to
1:00403034
                         🤼 Chart of xrefs from
1:00403035
                         0101 Byte 41h
1:00403036
                         0101 Word 6841h
1:00403037
1:00403038
                        0101
Double word 726F6841h
1:00403039
                         "s" "Ahora es tu turno: Introduce el primer "...
i:0040303A
                            Alignment
1:0040303B
```

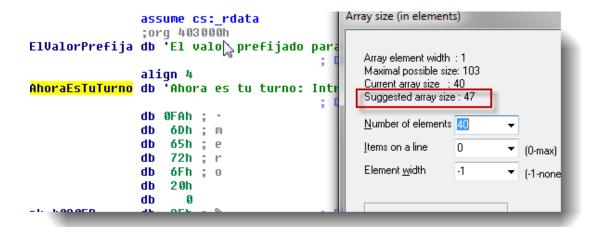
Y nos queda:

```
004Q302B
                           align 4
004 kg 02C
                          db 'Ahora es tu turno: Introduce el primer n'
         aAhoraEsTuTurno
0040302C
                                                     ; DATA XREF: _main+7Cfo
00403054
                           db OFAh ; -
00403055
                           db
                               6Dh ; m
00403056
                           db
                               65h
                                     е
00403057
                               72h
                           db
                                   ; r
00403058
                           db
                               6Fh ; o
00403059
                           db
                               20h
0040305A
                           db
                                 0
```

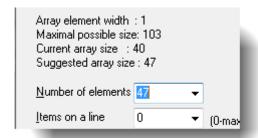
Opps, hay un carácter (0FAh), que no lo reconoce como parte de la frase, pero sabemos que si lo es, de hecho se han quedado 7 bytes fuera de la frase (incluido el null terminated). Para arreglar esto clickamos encima de la dirección y seleccionamos el array:



Ya nos dice que el tamaño aconsejable son 47 bytes:



Como el tamaño actual son 40 bytes y nos faltaban 7 por incluir, hacemos caso a lo que nos sugiere y le metemos 47 bytes de tamaño:



```
Y nos queda:
```

```
:00403\Delta B align 4
:0040302C aAhoraEsTuTurno db 'Ahora es tu turno: Introduce el primer n·mero ',0
:0040302C ; DATA XREF: _main+7C\Delta o
```

Hacemos lo mismo con las cadenas que vemos que le siguen:

```
เอนองขอย ;บry นองยอยเ
เอนองออด aElValorPrefija db 'El valor prefijado para la suma era %4.2f',0Ah,0
10403000
                                                         ; DATA XREF: main+701o
1040302B
1848382C aAhoraEsTuTurno db "Ahora es tu turno: Introduce el primer n-mero ',8
                                                         ; DATA XREF: _main+7C<sup>†</sup>o
; DATA XREF: _main+8F<sup>†</sup>o
040302C
1040305B aF
                            db '%f',0
1040305B
                                                           -main+AE↑o
.040305E aIntroduceElSeg db 'Introduce el segundo n∙mero ',0 ; DATA XREF: _main+9B†o
                            db 'Ahora la suma es %4.2f',0Ah,0 ; DATA XREF: _main+D0fo
1040307B aAhoraLaSumaEs4
0403093
                            alion 10h
```

Para la cadena %f, nos colocamos encima y apretamos la tecla 'a', y ya se nos queda en el formato correcto.

Ahora nos es más fácil seguir el código anterior:

```
004012FC
                          fstp
                                   [esp+28h+var_24]
00401300
                          mov
                                   [esp+28h+var_28], offset aElValorPrefija ; "El valor
00401307
                          call
0040130C
                                   [esp+28h+var_28], offset aAhoraEsTuTurno ; "Ahora es
                          mov
00401313
                                   printf
                          call
00401314
                          1ea
                                   eax, [ebp+Num1]
                                   dword ptr [esp+28h+var_24], eax
0040131R
                          mov
                                                                 ; "%f"
0040131F
                          mov
                                   [esp+28h+var_28], offset aF
00401326
                          call
0040132B
                                   [esp+28h+var 28], offset aIntroduceElSeg; "Introduc
                          mov
00401332
                          ca11
                                   printf
00401337
                                   eax, [ebp+datos0]
                          mnu
0040133A
                                   dword ptr [esp+28h+var_24], eax
                          mnv
                                   [esp+28h+var_28], offset aF ; "%f"
0040133E
                          mov
00401345
                          call
0040134A
                                   edx, [ebp+datos1]
                          mov
                                   eax, [ebp+datos0]
dword ptr [eax]
0040134D
                          MOV
00401350
                          f1d
00401352
                                   [ebp+Num1]
                          fadd
00401355
                          fstp
                                   dword ptr [edx]
```

Su correspondencia en C:

```
printf(''El valor prefijado para la suma era %4.2f\n'', *datos1 );
printf(''Ahora es tu turno: Introduce el primer n·mero '');
scanf(''%f'', &num1);
printf(''Introduce el segundo n·mero '');
scanf(''%f'', datos0);
*datos1= *datos0 + num1;
```

(Para introducir el caractar 250 (0fah), teclamos el numero 250 con la tecla Alt apretada, cuando soltemos la tecla Alt, tendremos tipeado el carácter ·).

Nos pide que introduzcamos los dos números y la suma de ambos la deja en la memoria apuntada en datos1.

Ya se finaliza el código, sacando la suma actual por la pantalla, y liberando la memoria reservada.

```
0401355
                                   dword ptr [edx]
                          fstp
                                   eax, [ebp+datos1]
0401357
                          mov
040135A
                                   dword ptr [eax]
                          f1d
040135C
                          fstp
                                   [esp+28h+var_24]
0401360
                                   [esp+28h+var_28], offset aAhoraLaSumaEs4 ; "Aho
                          mov
0401367
                          call
                                   printf
040136C
                                   eax, [ebp+datos0]
                          mov
040136F
                                   [esp+28h+var_28], eax
                          mov
0401372
                          call
0401377
                                   eax, [ebp+datos1]
                          mov
040137A
                                   [esp+28h+var_28], eax
                          mov
040137D
                          call
                                   free
0401382
                                   getchar
                          call
0401387
                          call
                                   getchar
040138C
                          call
                                   getchar
0401391
                          leave
0401392
                          retn
0401392 <u>mai</u>n
                          endp
A): 01202
```

```
Su correspondencia en C:
```

```
printf("Ahora la suma es %4.2f\n", *datos1);
       free(datos0);
       free(datos1);
       getchar();
        getchar();
        getchar();
Pues nada la función completa seria:
main()
       float num1;
       float* datos0;
        float* datos1;
        num1 = 5.0;
        datos0 = (float *)malloc(sizeof(float));
        *datos0 = 6.7;
        datos1 = (float *)malloc(sizeof(float));
        *datos1 = *datos0 + num1;
       printf("El valor prefijado para la suma era %4.2f\n", *datos1);
        printf("Ahora es tu turno: Introduce el primer n·mero ");
       scanf("%f", &num1);
       printf("Introduce el segundo n·mero ");
        scanf("%f", datos0);
        *datos1 = *datos0 + num1;
       printf("Ahora la suma es %4.2f\n", *datos1);
       free(datos0);
       free(datos1);
       getchar();
        getchar();
        getchar();
}
```

| identical | 401270 | onexit | 401270 | onexit |
|-----------|--------|--------------------------|--------|--------------------------|
| identical | 401280 | do sili init | 401280 | do sili init |
| identical | 401290 | _main | 401290 | _main |
| identical | 4013a0 | pei386_runtime_relocator | 4013a0 | pei386_runtime_relocator |
| identical | 4013d0 | _fpreset | 4013d0 | _fpreset |
| identical | 4013e0 | do_global_dtors | 4013e0 | do_global_dtors |
| identical | 401420 | do_global_ctors | 401420 | do_global_ctors |

Pues esta vez hemos tenido suerte.

Hasta la próxima.

asOlOt, 2010 CLS