Reversando Ejercicio 12

Curso C y reversing Crackslatinos

Por sisco 0

Programas necesarios:

IDA Pro+Hex Rays, Dev C++

Cargamos el programa en *IDA Pro*.

Más tarde pulsamos *F5* (Modo pseudocódigo), así tenemos una bonita vista del código. Pero volvamos al modo anterior, en el caso de que tengamos alguna duda podríamos volver a este.

Primeramente renombraremos las variables que vemos:

Nombre	Nuevo Nombre
var_4	numero
var_8	puntero1
var_C	puntero2

Estos nombres no han sido elegidos al azar, tras una vista rápida del código he comprendido cuáles eran punteros y cuáles no lo eran.

Recordar que para renombrar, simplemente hacer click sobre el nombre y pulsar *n*.

Otras variables como *var_10* no han sido modificadas ya que no las usaremos, es código del compilador situando antes de la posición de memoria virtual *0x004012BA*.

Estudio del código en *IDA*, valor prefijado

Cargando *numero*

```
.text:004012BA        mov    eax, 40A00000h
.text:004012BF        mov    [ebp+numero], eax
```

En esta parte del código lo que hacemos es mover al registro *eax* un valor hexadecimal para más tarde cargarlo dentro de nuestra variable *numero*.

Por ahora no sabemos si *numero* es de tipo *float* o de tipo *int*. (Aunque más tarde lo veremos que es de tipo *float*).

Creando *puntero1*

```
.text:004012C2        mov    dword ptr [esp], 4 ; size_t
.text:004012C9        call   malloc
.text:004012CE        mov    [ebp+puntero1], eax
```

En este paso llamamos a *malloc* dándole como parámetro 4, este es el número de *bytes* que queremos reservar en memoria, más tarde *malloc* deja en *eax* la dirección que apunta al espacio reservado.

Como podemos ver movemos a *puntero1* esta misma dirección.

Cargando algo en el espacio apuntado por *puntero1*

```
.text:004012D1        mov    edx, [ebp+puntero1]
.text:004012D4        mov    eax, 40D66666h
.text:004012D9        mov    [edx], eax
```

Ahora cargamos en *edx* la dirección a la que apunta *puntero1*, mientras que en el registro *eax* introducimos la constante hexadecimal *0x40D66666*.

Cargamos en el espacio apuntado por *edx* el valor de *eax*, realmente estamos haciendo una asignación al espacio apuntado por *puntero1*.

Creando puntero2

```
.text:004012DB          mov     dword ptr [esp], 4 ; size_t
.text:004012E2          call    malloc
.text:004012E7          mov     [ebp+puntero2], eax
```

De nuevo, al igual que con puntero 1, reservamos un espacio de 4 bytes y asignamos su posición al puntero *puntero2*.

Algunas operaciones con *coma flotante*

```
.text:004012EA          mov     edx, [ebp+puntero2]
.text:004012ED          mov     eax, [ebp+puntero1]
.text:004012F0          fld     dword ptr [eax]
.text:004012F2          fadd    [ebp+numero]
.text:004012F5          fstp    dword ptr [edx]
```

Lo primero que hacemos es mover a *edx puntero2*, en la siguiente instrucción hacemos lo propio con *puntero1* y *eax*.

Más tarde lo que hacemos es cargar en la pila de operaciones con coma flotante el valor apuntado por *eax*, es decir, el valor apuntado por *puntero1*. Es como hacer un *push* al valor apuntado por *puntero1*.

Lo próximo es *fadd*, que es una suma de coma flotante, lo que hacemos es sumar el valor de *numero*, por lo tanto, podemos tomar a *numero* como algo de tipo *float*.

Tenemos el nemónico *fstp*, que lo que hará será un *pop*, este lo que hará será guardar el resultado en el valor apuntado por *edx*, es decir, el valor apuntado por *puntero2*.

Guardando *puntero2* en la pila en forma de *coma flotante*

```
.text:004012F7          mov     eax, [ebp+puntero2]
.text:004012FA          fld     dword ptr [eax]
.text:004012FC          fstp    qword ptr [esp+4]
```

Tenemos que movemos a *eax puntero2*, mientras que cargamos el contenido de *puntero2* en la pila de *coma flotante* y lo descargamos en la pila que estamos acostumbrados a usar, en la posición *esp+4*, es decir, la segunda posición de la pila.

Quizás para usarlo más adelante ☺.

Imprimiendo el valor de *puntero2* con mensaje

```
.text:00401300          mov     dword ptr [esp], offset aElValorPrefija ; "El valor prefijad"...
.text:00401307          call    printf
```

Como podemos ver, ahora movemos a *esp* el apuntador a la cadena de caracteres *El valor prefijado…* que más tarde se le llamará con *printf*, este mismo requiere que se le pase una variable para darle valor a *%4.2f*, nosotros le daremos el valor que anteriormente metimos en la pila ya que este es el segundo parámetro de nuestra función.

Es decir, le estamos pasando el valor de *puntero2*.

Resumen de la zona pre-introducción de datos

Primero deberemos realizar una conversión de *Hexadecimal* a *float* de los valores que hemos visto hardcodeados.

Para ello utilizo el siguiente código fuente que encontré en un foro:

```
#include <stdio.h>
int main(void) {
```

```
    union {
       int i;
       float f;
    } switcheroo;
    switcheroo.i = 0x40A00000;
    printf("0x%X %f\n", switcheroo.i, switcheroo.f);
    switcheroo.i = 0x40D66666;
    printf("0x%X %f\n", switcheroo.i, switcheroo.f);
    return 0;
}
```

Observo que los valores son *5.0* y *6.7* ☺

Puedo comenzar a programar:

```
#include <stdio.h>
int main(void)
{
    float numero=5.0;
    float *puntero1=malloc(4);
    *puntero1=6.7;
    float *puntero2=malloc(4);
    *puntero2=*puntero1+numero;
    printf("El valor prefijado para la suma era %4.2f\n",*puntero2);
    return 0;
}
```

Parece que todo va parejo con respecto al programa original, vamos por buen camino, ahora comenzaremos con la pedida de datos.

Estudio del código en *IDA*, pedida de datos

Modificando *numero*

```
.text:0040130C        mov    dword ptr [esp], offset aAhoraEsTuTurno ; "Ahora es tu "...
.text:00401313        call   printf
.text:00401318        lea    eax, [ebp+numero]
.text:0040131B        mov    [esp+4], eax
.text:0040131F        mov    dword ptr [esp], offset asc_40305B ; "%f"
.text:00401326        call   scanf
```

En la primera instrucción movemos la cadena al *stack* e invocamos a *printf* en la segunda.

Más tarde podemos ver como cargamos la dirección de *numero* en *eax*.

Después lo pasamos al *stack*.

Movemos al *stack* la dirección del string de descripción del dato que vamos a meter.

Llamamos a *scanf*, es decir, vamos a introducir por teclado un valor de *coma flotante* que se va a almacenar en la variable *numero*, esa donde antes teníamos el *5.0* .

Modificando valor de lo apuntado por *puntero1*

```
.text:0040132B        mov    dword ptr [esp], offset aIntroduceElSeg ; "Introduce "…
.text:00401332        call   printf
.text:00401337        mov    eax, [ebp+puntero1]
.text:0040133A        mov    [esp+4], eax
.text:0040133E        mov    dword ptr [esp], offset asc_40305B ; "%f"
.text:00401345        call   scanf
```

En *0x0040132B* imprimimos un *string* (Podemos ver qué *string* es completamente si hacemos doble click sobre *aIntroduceElSeg*, debemos tener en cuenta que las tildes no se tratan correctamente).

Más tarde llamamos a *printf* para imprimir este *string*.
Más tarde cargamos en el registro *eax* el contenido de *puntero1*, que se trata de una dirección de memoria, por lo tanto, ya tenemos la dirección de memoria que más tarde necesitará el *scanf*.
Esto lo pasamos a la pila.
Además también pasamos a la pila el descriptor de *string*, vamos a introducir algo de tipo float, por lo tanto se pasa una dirección de memoria donde está contenida la cadena *%f*.
Llamamos a *scanf* y esperamos que el usuario introduzca los datos, es decir, tendremos en la dirección apuntada por *puntero1* el segundo número tipo *coma flotante*.

Realizando la suma

```
.text:0040134A        mov    edx, [ebp+puntero2]
.text:0040134D        mov    eax, [ebp+puntero1]
.text:00401350        fld    dword ptr [eax]
.text:00401352        fadd   [ebp+numero]
.text:00401355        fstp   dword ptr [edx]
```

En estos pasos movemos a *edx puntero2*.
Movemos a *eax puntero1*.
Más tarde cargamos en la pila de manejo de números en *coma flotante* el contenido de la dirección apuntada por *eax*, es decir, apuntada por *puntero1*.
Añadimos a este número el contenido de *numero*.
El resultado de esta suma lo sacamos de la pila de manejo de *coma flotante* y lo almacenamos en la dirección apuntada por *edx*, es decir, la dirección apuntada por *puntero2*.

Imprimiendo resultado

```
.text:00401357        mov    eax, [ebp+puntero2]
.text:0040135A        fld    dword ptr [eax]
.text:0040135C        fstp   qword ptr [esp+4]
.text:00401360        mov    dword ptr [esp], offset aAhoraLaSumaEs4 ; "Ahora "…
.text:00401367        call   printf
```

En la primera instrucción movemos a *eax puntero2*.
Luego cargamos en la pila de manejo de *coma flotante* el contenido de lo apuntado por *eax*, es decir, el contenido de lo apuntado por *puntero2*.
Más tarde descargamos esta cifra de la pila de manejo de *coma flotante* para cargarlo en la pila *stack* para pasarlo más tarde a la función *printf*.
Movemos a la primera posición del *stack* la cadena con el identificador *%4.2f* para imprimir la variable.
Llamamos a *printf*.
De esta forma hemos imprimido el valor tipo *coma flotante* de lo contenido en *puntero2*.

Liberando espacio

```
.text:0040136C        mov    eax, [ebp+puntero1]
.text:0040136F        mov    [esp], eax     ; void *
.text:00401372        call   free
.text:00401377        mov    eax, [ebp+puntero2]
.text:0040137A        mov    [esp], eax     ; void *
.text:0040137D        call   free
```

Aquí lo que hacemos con *puntero1* y con *puntero2* luego es moverlos a la pila, más tarde cargamos estas direcciones que contienen y que es donde apuntan cada uno de ellos dentro de *esp* y llamamos a *free*, estamos liberando el espacio que al principio del programa habíamos ocupado con *malloc*.

Esperando a los tres toques

```
.text:00401382        call   getchar
.text:00401387        call   getchar
.text:0040138C        call   getchar
.text:00401391        leave
.text:00401392        retn
```

Llamamos 3 veces a *getchar* y hacemos *leave* y *retn*.

Implementación completa del programa

Ya podemos añadir la parte que nos restaba para acabar nuestro código fuente.

```
#include <stdio.h>
int main(void)
{
   float numero=5.0;
   float *puntero1=malloc(4);
   *puntero1=6.7;
   float *puntero2=malloc(4);
   *puntero2=*puntero1+numero;
   printf("El valor prefijado para la suma era %4.2f\n",*puntero2);

   printf("Ahora es tu turno: Introduce el primer número ");
   scanf("%f",&numero);
   printf("Introduce el segundo numero ");
   scanf("%f",puntero1);
   *puntero2=*puntero1+numero;
   printf("Ahora la suma es %4.2f\n",*puntero2);

   free(puntero1);
   free(puntero2);
   getchar();
   getchar();
   getchar();
   return;
}
```