

# 19: Práctica 1/4

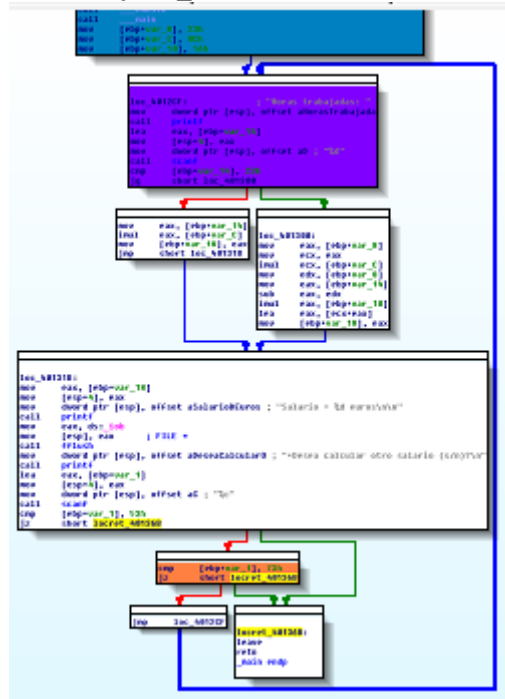
## Curso C y reversing Crackslatinos

Por sisco 0

Nos enfrentamos a una práctica, debemos reversar el código. ¿Lo conseguiremos? ¡Sí!  
Pues vamos a por ello, seguro que será interesante.

### Cargando en IDA

Cargamos la aplicación *practica1.exe* en el IDA y observamos que aparece la siguiente estructura en la vista IDA-View A bajo el `_main`:



Vista de estructura del `_main`

Observamos que se trata de un bucle, ya que la línea azul tan grande que vemos a la derecha está señalada en *negrita*, a consciencia, vamos, que se han dejado ahí el *alpino*.

Pues bien, la caja azul inicializa algunas variables, la caja violeta es el principio del bucle y la caja naranja hace la comprobación para volver arriba o salir del bucle.

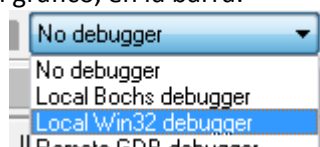
Observamos que la caja del bucle que hace la comparación está situada al final, como diría Robert Pattinson: **WAHT?** <http://www.youtube.com/watch?v=2G2GTOedk1o>

☺ Pues bien, se trata de un ciclo *do-while*, de los que disparan y luego preguntan, al final.

Pondremos un *breakpoint* en la línea:

```
.text:004012BA      mov     [ebp+var_8], 23h
```

Que está situada en la caja azul, más o menos por donde dejo ver de la imagen, para ello hacemos clic en la línea y pulsamos *F2*, si no cambia de color es porque debemos primero seleccionar el *debugger* arriba del gráfico, en la barra:



Seleccionando debugger

Una vez puesto el *breakpoint* pulsamos *F9* y ¡a correr! (El programa, nosotros seguimos sentados).

## Inicializando variables, cajita azul

```
.text:004012BA mov [ebp+var_8], 23h
.text:004012C1 mov [ebp+var_C], 0Ch
.text:004012C8 mov [ebp+var_10], 16h
```

El programa parece inicializar 3 variables con diferentes valores al inicio del programa, antes de entrar en el bucle, estas variables son *var\_8*, *var\_C* y *var\_10*, por ahora no se me ocurre qué nombre ponerles, ¿Qué nombre les ponemos?

Hacemos una panorámica seleccionando por ejemplo *var\_8* con el ratón (Como si estuviésemos seleccionando texto, pero sin el **como**, vamos, seleccionando el texto), observamos que tratará esta variable para mandarla a *ecx* en alguna parte del programa y multiplicarla por otro valor, parece ser un coeficiente que se aplica, vale, entonces **renombramos** *var\_8* y le pondremos de nombre *coef\_mul1*, para ello le hacemos clic y pulsamos *N* en el teclado.

De nuevo vemos lo mismo con *var\_C*, la **renombramos** por *coef\_mul2*.

Vaya, con *var\_10* tenemos lo mismo, pues bien, a renombrar igual, la **renombramos** por *coef\_mul3*.

Observamos qué valores tienen, para ver un valor en *decimal* simplemente pulsamos sobre este y pulsamos la tecla *H*, para volver a la normalidad (*hexadecimal*), pulsamos *Q*.

Var	Hex	Dec
coef_mul1	23h	35
coef_mul2	0Ch	12
coef_mul3	16h	22

Valores en Hex y Decimal

Pues vale, ya tendremos algo en nuestro código como:

```
int coef_mul1=35,coef_mul2=12,coef_mul3=22;
```

## Comenzando el do-while

```
.text:004012CF mov dword ptr [esp], offset aHorasTrabajada
.text:004012D6 call printf
```

Nuestro primer movimiento dentro del bucle es realizar una impresión por pantalla del *string* "Horas trabajadas: ".

```
.text:004012DB lea eax, [ebp+var_14]
.text:004012DE mov [esp+4], eax
.text:004012E2 mov dword ptr [esp], offset aD ; "%d"
.text:004012E9 call scanf
```

En este paso movemos a la pila la dirección de un buffer, y llamamos a *scanf*, podemos renombrar *var\_14* por otra cosa, ¿Pero por qué? son las horas trabajadas, así que vamos a **renombrar** *var\_14* por *horas\_trab*.

Es algo que nos pedirá con *scanf*. Así que por ello mismo carga su dirección con *lea*, para cargar el puntero.

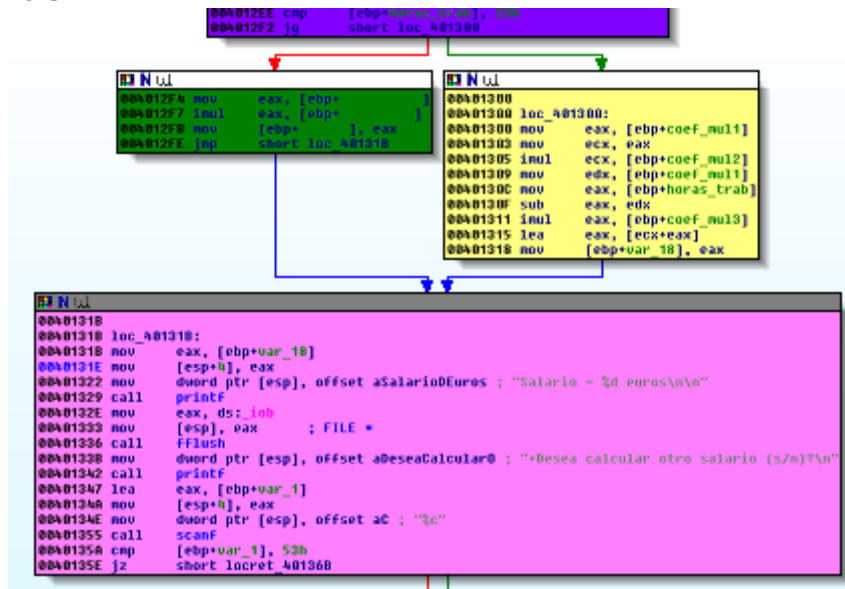
```
.text:004012EE cmp [ebp+horas_trab], 23h
.text:004012F2 jg short loc_4013
```

Ahora compara *horas\_trab* con el número *hexadecimal* 23h, que, si hacemos clic en él y pulsamos *H* es igual a 35, pulsamos *Q* de nuevo para dejarlo en *hexadecimal*.

Pues bien, por ahora nos queda algo como:

```
printf("Horas trabajadas: ");
scanf("%d",&horas_trab);
if(horas_trab > 35) {
```

## Coloreando



Verde, amarillo y rosa

Coloreo esas cajas para hacer mención a ellas en este momento, observamos que iremos a la caja verde si *horas\_trab* es menor o igual que 35, y que iremos a la caja amarilla en caso contrario.

### Menor o igual, cajita verde

```
.text:004012F4 mov     eax, [ebp+horas_trab]
.text:004012F7 imul    eax, [ebp+coef_mul2]
.text:004012FB mov     [ebp+var_18], eax
.text:004012FE jmp     short loc_40131B
```

Lo que hacemos en este caso es mover a *eax* las *horas\_trab*, multiplicamos esto por *coef\_mul2*, que recordamos que es igual a 12, al menos así lo inicializamos.

Más tarde movemos el resultado a *var\_18* y saltamos a la caja rosa.

¿Qué es *var\_18*? ¿Por qué otro nombre lo renombramos?

Lo seleccionamos y vemos que se hace referencia a él dentro de la caja roja, justo antes de imprimir salario, pues entonces lo que haremos será **renombrar** *var\_18* por *salario\_cal*, por ser el salario calculado.

Nos ha quedado algo así:

```
else {
    salario_cal=horas_trab*coef_mul2;
}
```

Forma parte del *else* en este momento ya que es lo contrario que pedía el *if*.

### Mayor, cajita amarilla

Vamos a mirar la caja amarilla antes de irnos a la rosa, ya que es lo que está dentro del *if*.

```
.text:00401300 mov     eax, [ebp+coef_mul1]
.text:00401303 mov     ecx, eax
.text:00401305 imul    ecx, [ebp+coef_mul2]
```

Aquí movemos *coef\_mul1* a *eax*, más tarde a *ecx* y lo multiplicamos *coef\_mul2*, el resultado se queda en *ecx*. Realmente estamos multiplicando 35 horas por lo que se cobra en una hora normal.

```
.text:00401309 mov     edx, [ebp+coef_mul1]
.text:0040130C mov     eax, [ebp+horas_trab]
.text:0040130F sub     eax, edx
```

Movemos *coef\_mul1* a *edx*, además movemos *horas\_trab* a *eax*, y realizamos la resta entre estos registros, es decir *horas\_trab-coef\_mul1*, que casualmente ahí teníamos un 35, por lo que lo que hacemos es decir, ¿Cuántas horas trabajadas has echado de más? (Suponiendo que trabaja 35 horas por semana a una media de 7 horas por día de los 5 días en los que trabaja y cobra por semana, esto es solo para liaros, ¡saltaos este paréntesis!).

```
.text:00401311 imul    eax, [ebp+coef_mul3]
```

Pues bien, ahora multiplicamos estas horas sobrantes por *coef\_mul3*, ahora lo entiendo, esto es lo que se cobra por cada hora de más.

```
.text:00401315 lea     eax, [ecx+eax]
```

```
.text:00401318 mov     [ebp+salario_cal], eax
```

Movemos a *eax* la suma de ambos, es decir, lo que cobramos por las primeras 35 horas y lo que cobramos extra.

Y más tarde esta cifra la movemos a *salario\_cal*.

Nos queda algo así al final:

```
//Dentro del if
```

```
salario_cal=coef_mul1*coef_mul2+(horas_trab-coef_mul1)*coef_mul3;
```

## Final del bucle, caja rosa

Es la más grande, pero no hay por qué asustarse.

```
.text:0040131B mov     eax, [ebp+salario_cal]
```

```
.text:0040131E mov     [esp+4], eax
```

```
.text:00401322 mov     dword ptr [esp], offset aSalarioDEuros ; "Salario = %d euros\n\n"
```

```
.text:00401329 call    printf
```

Movemos a la pila *salario\_cal* y lo imprimimos dentro del *string* como decimal con el *printf*.

```
.text:0040132E mov     eax, ds:_iob
```

```
.text:00401333 mov     [esp], eax ; FILE *
```

```
.text:00401336 call    fflush
```

Hacemos una llamada a *fflush*, un comando para vaciar el *buffer*, en este caso será el de entrada del teclado (Supongo, ya se verá).

```
.text:0040133B mov     dword ptr [esp], offset aDeseaCalcularO ; "+Desea calcular otro salario (s/n)?\n"
```

```
.text:00401342 call    printf
```

Nos pregunta con *printf* si deseamos calcular otro salario.

```
.text:00401347 lea     eax, [ebp+var_1]
```

```
.text:0040134A mov     [esp+4], eax
```

```
.text:0040134E mov     dword ptr [esp], offset aC ; "%c"
```

```
.text:00401355 call    scanf
```

Nos pide un carácter con *scanf*, lo guardará en la dirección de *var\_1*, así que **renombramos** *var\_1* por *si\_o\_no*.

```
.text:0040135A cmp     [ebp+si_o_no], 53h
```

```
.text:0040135E jz      short locret_40136B
```

Compara el carácter que hemos introducido con *53h* ¿Qué carácter será? Pues hacemos clic derecho y observamos que es la *S*.

```
.text:00401360 cmp     [ebp+si_o_no], 73h
```

```
.text:00401364 jz      short locret_40136B
```

Lo mismo con la *s* minúscula.

```
.text:00401366 jmp     loc_4012CF
```

Si no ha sido ni una *s* ni una *S* entonces llegamos hasta aquí, que es un salto a la cabeza del bucle.

Entonces, ¿por qué saltamos a la cabeza del bucle si hemos metido algo que no es una *s* ni una *S*? ¡Pues está claro! Para que no seamos listillos y pensemos un poco.

Pues bien, eso es todo, nos queda algo así para esta caja:

```
printf("Salario = %d euros\n\n",salario_cal);
fflush(stdin);
printf("+Desea calcular otro salario (s/n)?\n");
scanf("%c",&si_o_no);
if(si_o_no=='S') break;
if(si_o_no=='s') break;
```

```
} while(1); //Si señores (Y la señora de la lista) parece ser algo infinito.
```

No hago referencia a la última parte del *leave* y *retn*, pero observamos que no devuelve ningún número, por lo que el *main* será de tipo *void*.

## Creando el código fuente

Pues bien, ya lo tenemos casi todo, o eso pensamos, así que creamos el código fuente, yo usaré *Dev-C++*.

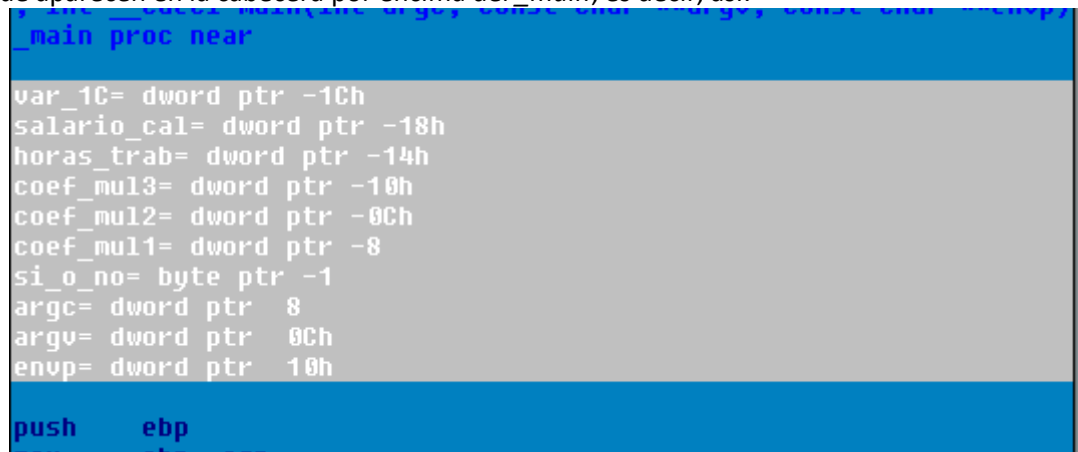
Tras algunas pruebas he visto que he tenido que modificar algunas líneas, como por ejemplo cambiar la condición del *if*. La lógica no estaba mal, simplemente es que al final queda algo como:

```
if (horas_trab<=35)
{
    //Caja verde
}
else {
    //Caja amarilla
}
```

Con el *fflush* hemos acertado, aunque la condición final queda como:

```
if(si_o_no=='S' || si_o_no=='s') break;
```

Recaltar que la inicialización y declaración de las variables se hace en el mismo orden con el que aparecen en la cabecera por encima del *\_main*, es decir, así:



```
var_1C= dword ptr -1Ch
salario_cal= dword ptr -18h
horas_trab= dword ptr -14h
coef_mul3= dword ptr -10h
coef_mul2= dword ptr -0Ch
coef_mul1= dword ptr -8
si_o_no= byte ptr -1
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h

push    ebp
mov     ebp, esp
```

*Detalle en IDA*

Pues yo lo realizo usando el mismo orden, de abajo a arriba.  
Código fuente en la siguiente página.

```

/* Ejercicio 19 Practica 1 */
#include <stdio.h>
void main(void)
{
    char si_o_no;
    int coef_mul1=35,coef_mul2=12,coef_mul3=22;
    int horas_trab,salario_cal;

    do {
        //Caja violeta
        printf("Horas trabajadas: ");
        scanf("%d",&horas_trab);
        if (horas_trab<=35)
        {
            //Caja verde
            salario_cal=horas_trab*coef_mul2;
        }
        else {
            //Caja amarilla
            salario_cal=coef_mul1*coef_mul2+(horas_trab-coef_mul1)*coef_mul3;
        }
        //Caja rosa
        printf("Salario = %d euros\n\n",salario_cal);
        fflush(stdin);
        printf("+Desea calcular otro salario (s/n)?\n");
        scanf("%c",&si_o_no);
        if(si_o_no=='S' || si_o_no=='s') break;
    } while(1);
}

```

## Turbodiff

Usamos el *plugin* de *turbodiff* de *IDA* para ver si hay alguna diferencia.

Explico el proceso, tal y como lo hice en otros tutes.

- Abrimos un *IDA* con el fichero original *practica1.exe*
  - o *Edit* → *Plugins* → *Turbodiff*
    - *Take info from this idb*
      - OK
  - o Botón de guardar (*El del diskette de arriba a la izquierda*)
- Abrimos otro *IDA* con el fichero nuevo que hemos compilado, en mi caso *practica1\_\_sisco\_0.exe*
  - o *Edit* → *Plugins* → *Turbodiff*
    - *Take info from this idb*
      - OK
  - o Botón de guardar (*El del diskette de arriba a la izquierda*)
  - o *Edit* → *Plugins* → *Turbodiff*
    - *Compare with...*
      - Doble clic en el fichero referente a nuestro original, es decir, *Practica1.idb*
        - o OK

Y observamos cómo nos aparece todo *identical*, esto quiere decir que son completamente iguales.

identical	401200	__uu_spl_init	401200	__uu_spl_init
identical	401290	_main	401290	_main
identical	401370	__pei386_runtime_relocator	401370	__pei386_runtime_relocator
identical	4013a0	_fpreset	4013a0	_fpreset
identical	4013b0	__do_global_dtors	4013b0	__do_global_dtors
identical	4013f0	do_global_ctors	4013f0	do_global_ctors