BOMBA PRÁCTICA 4

José María Ramírez González

ÍNDICE

CÓDIGO DE LA BOMBA

Para compilar el código de la bomba, tan solo necesitamos ejecutar la orden que tenemos al comienzo del archivo “bombaJoseMaria.c”

gcc -Og bombaJoseMaria.c -o bombaJoseMaria -no-pie -fno-guess-branch-probability

La contraseña original es “*AverSiAdivina”*.
El pin original es *1314*.
Una vez cambiadas las claves, la contraseña será “*hola,adios123*” y el PIN será *1315*.

A continuación se muestra el código de la bomba:

```
// gcc -Og bombaJoseMaria.c -o bombaJoseMaria -no-pie -fno-guess-branch-probability

#include <stdio.h> // para printf(), fgets(), scanf()
#include <stdlib.h> // para exit()
#include <string.h> // para strncmp()
#include <sys/time.h> // para gettimeofday(), struct timeval

#define SIZE 100
#define TLIM 7

char password[]="AverSiAdivina\n"; // contraseña
int passcode = 1314; // pin

void boom(void){                        //Imprime la explosión
    printf( "\n"
        "***************\n"
        "*** BOOM!!! ***\n"
        "***************\n"
        "\n");
    exit(-1);
}

void defused(void){             //Imprime bomba desactivada
    printf( "\n"
        "-------------------------------------------------------------------\n"
        "------- bomba desactivada -------\n"
        "-------------------------------------------------------------------\n"
        "\n");
    exit(0);
}
```

```
int main(){
	char pw[SIZE];
	int pc, n;

	struct timeval tv1,tv2; // gettimeofday() secs-usecs

	gettimeofday(&tv1,NULL);		//Tomamos el tiempo

	do printf("\nIntroduce la contraseña, tienes 7 segundos.\nContraseña: ");
	while ( fgets(pw, SIZE, stdin) == NULL );	//Obtenemos un array de char desde stdin

	if ( strncmp(pw,password,sizeof(password)) )	//Comparamos las claves
		boom();

	gettimeofday(&tv2,NULL);		//Tomamos el tiempo

	if ( tv2.tv_sec - tv1.tv_sec > TLIM )	//Si tarda mas de TLIM (7s), explota
		boom();

	do { printf("\nIntroduce el pin, tienes 7 segundos.\nPIN: ");	//Leemos el pin
		if ((n=scanf("%i",&pc))==0)
			scanf("%*s") ==1;
	} while ( n!=1 );

	if ( pc != passcode )		//Si no coinciden, explota
		boom();

	gettimeofday(&tv1,NULL);

	if ( tv1.tv_sec - tv2.tv_sec > TLIM )	//Si hemos tardado más de lo permitido, explota
		boom();

	defused();		//Se desactiva la bomba

}
```

DESACTIVAR

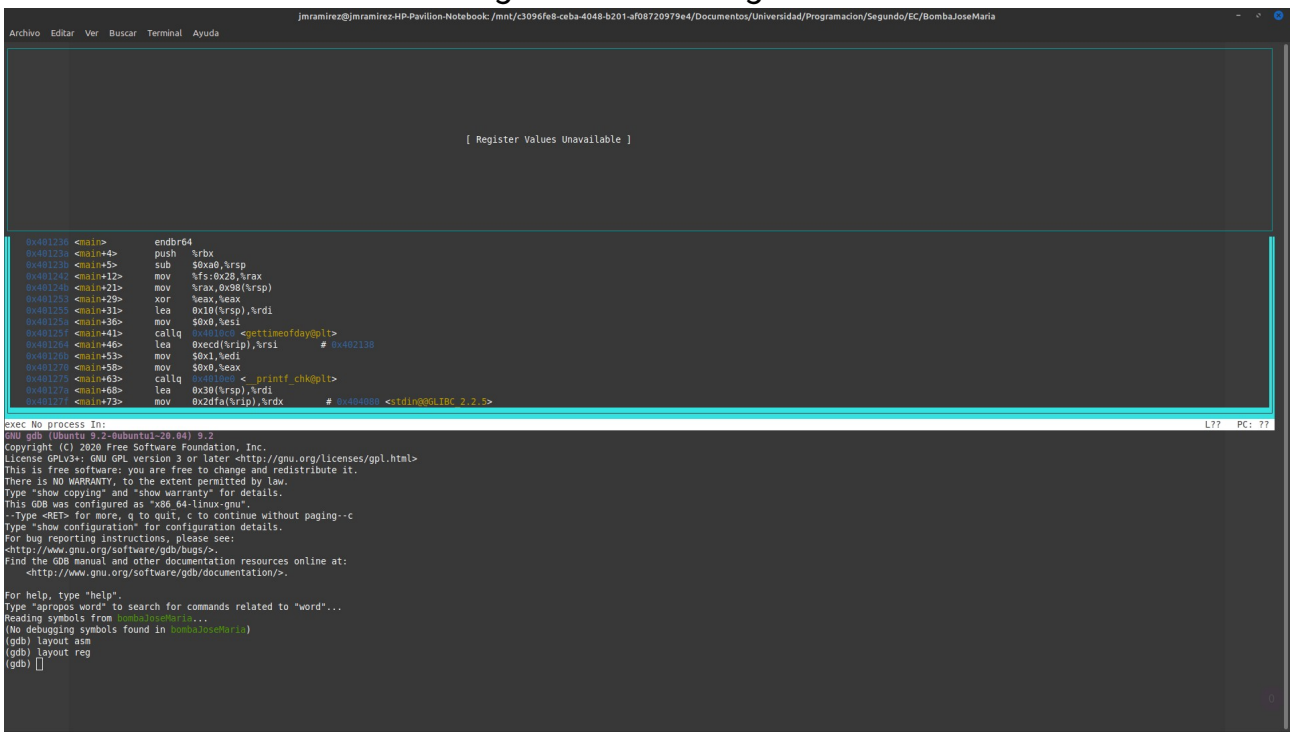
Para desactivar la bomba, primero compilamos con gcc usando el comando indicado al principio del código.
Luego abrimos gdb ejecutando la siguiente orden

gdb <nombre de la bomba>

Una vez abierto gdb, mostramos el código ensamblador y los registros de memoria

layout asm
layout reg

Ahora mismo deberíamos ver algo similar a lo siguiente



Una vez aquí, creamos un *breakpoint* en el main y comenzamos la ejecución del programa

br main
run

Ahora si navegamos con las flechas del teclado, encontraremos la variable *password*, que es la contraseña que tenemos que descifrar.
Podemos ejecutar la siguiente orden y nos mostrará la clave

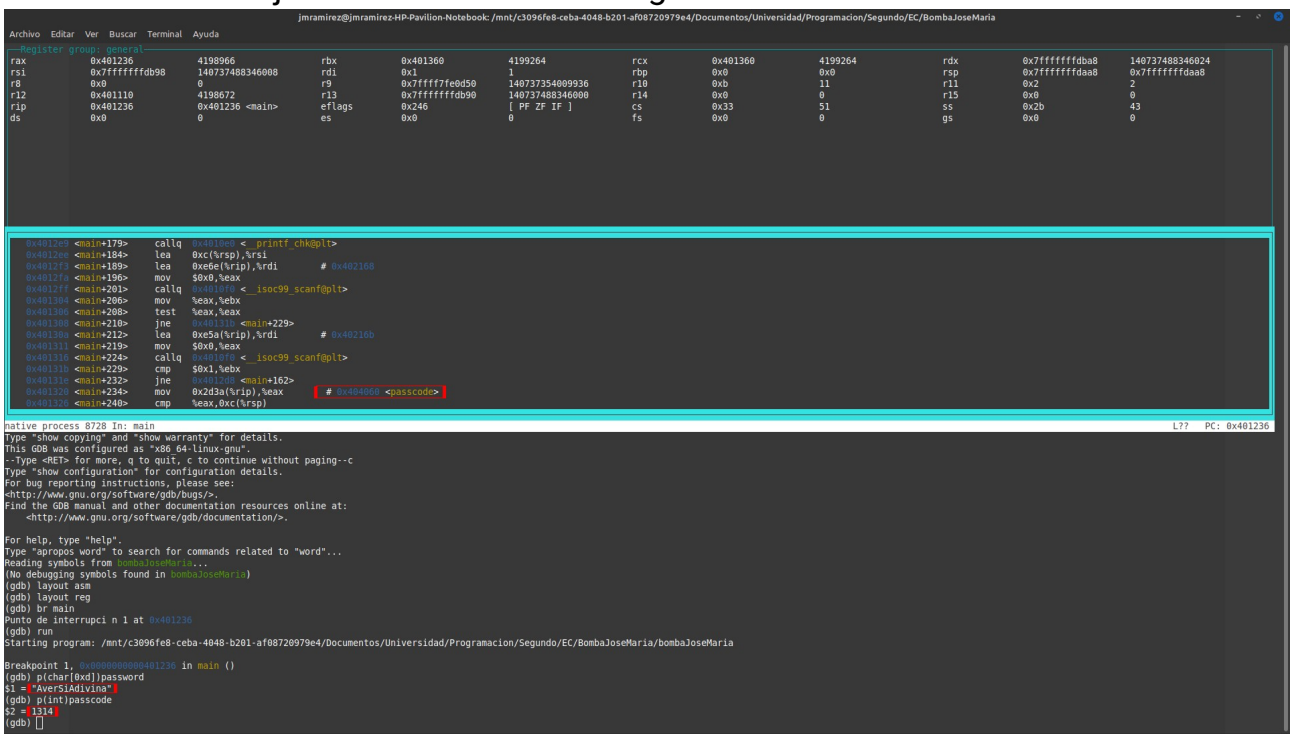
p(char[0xd])password

Y podemos hacer lo mismo con la variable *passcode*, el PIN

p(int)passcode

Si bajando al buscar la variable se nos descuadra la página, podemos volver a cuadrarla pulsando *Ctrl+L*.

Ahora mismo la ejecución se vería de la siguiente forma



Una vez realizado esto, tenemos 2 opciones, o bien nos aseguramos de meter la contraseña y el PIN rápido, ya que aún no ha empezado a contar el tiempo, o bien vamos paso a paso desactivando la comprobación del tiempo y de las claves.

Vamos a optar por la 2ª, ya que es la que incluye más complicaciones.

Si nos fijamos, en la línea *main+112* llama a *strncmp,* una función para comparar cadenas de caracteres. Vamos a crear un *breakpoint* en el *test* que hay después de esa línea para saltarnos la comprobación de la contraseña

br *main+117
cont

Cuando nos pida la contraseña, realmente podemos escribir la que nos de la gana, ya que vamos a saltarnos esa comprobación, en mi caso voy a escribir “hola”.

Ahora para saltarnos esa comprobación simplemente ponemos %eax a 0, ya que es el registro donde se realiza la comprobación

set $eax=0

Y si hacemos *ni* para pasar a la siguiente instrucción vemos que nos hemos saltado el *boom*.

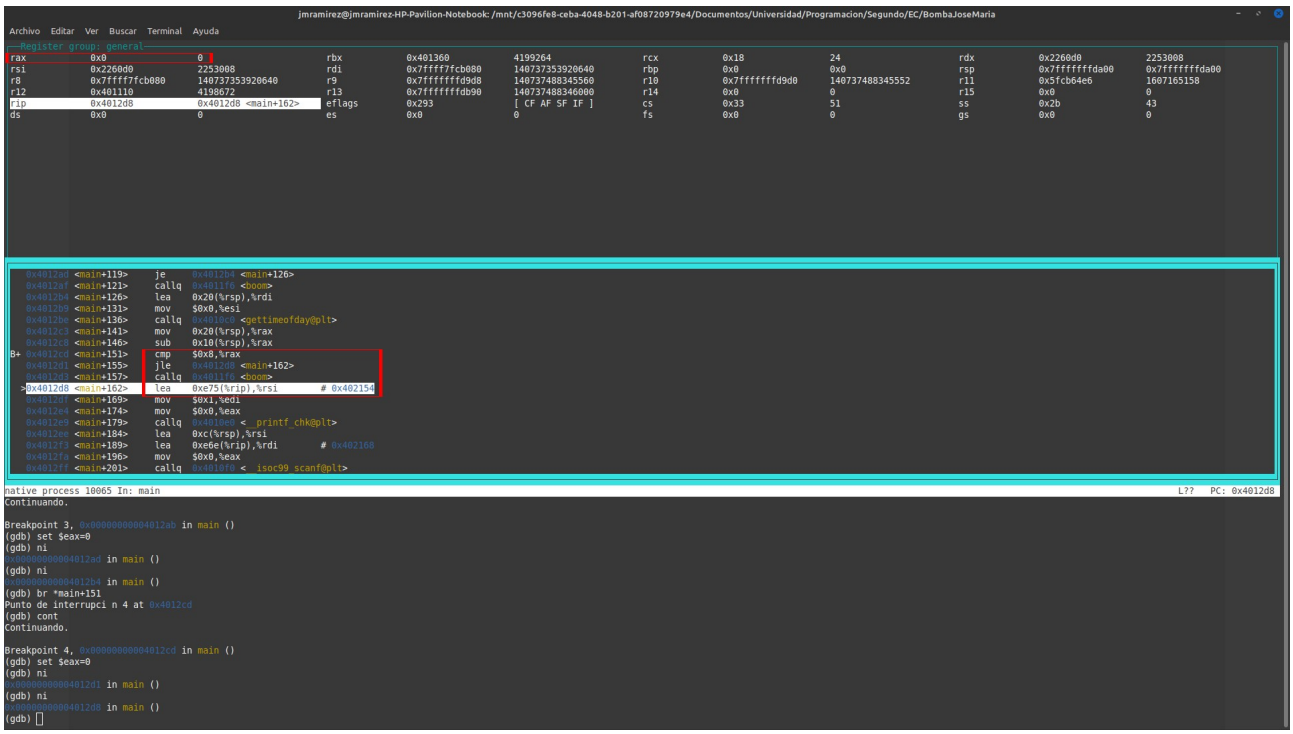
Ahora para saltarnos la comprobación del tiempo, iremos a *main+151*, donde se compara la diferencia entre los tiempos al introducir la contraseña (se realiza un cmp)

br *main+151
cont

Y para saltarnos esa comprobación volvemos a ejecutar la orden anterior

set $eax=0

Y hacemos ni para ir pasando a la siguiente instrucción hasta ver que, efectivamente, no entramos en el *boom*.



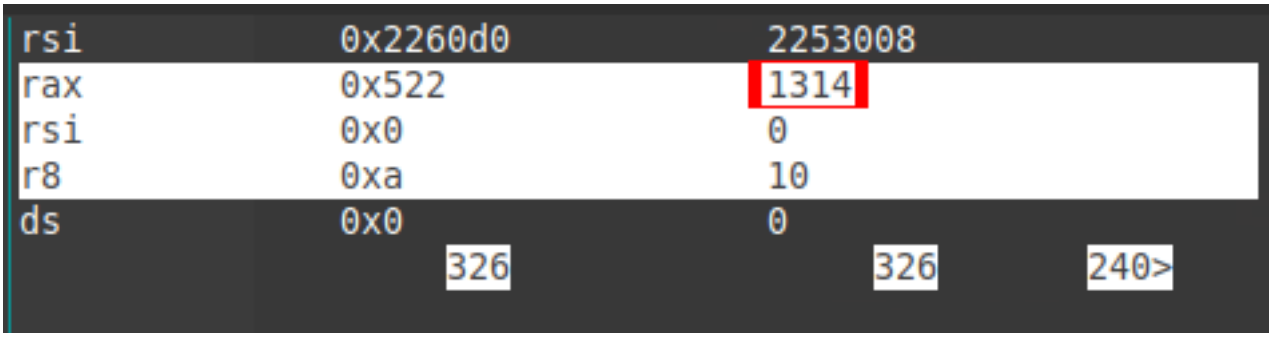
Ahora si nos fijamos en la línea *main+240*, vemos que se realiza una comparación justo después de mover la variable *passcode*, esa es la comprobación del pin introducido.

Aquí podemos o bien meter el pin (que ya sabemos que es *1314*), o bien introducir cualquier pin, el de nuestra elección, y modificar el registro sobre el que se realiza la comparación para que no nos de error.

Vamos a optar por lo segundo.
Creamos entonces un *breakpoint* en la línea *main+240*

```
br *main+240
cont
```

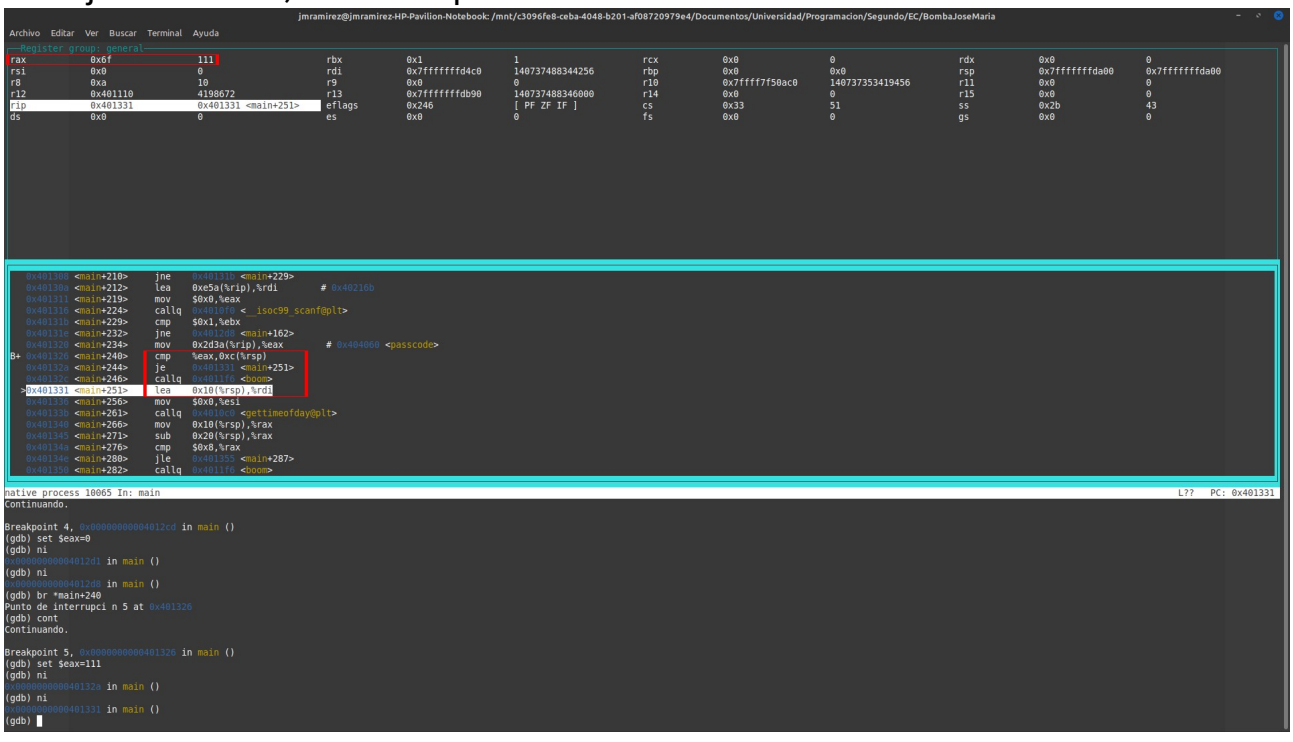
Cuando nos pida el PIN, vamos a introducir, por ejemplo, “111”.
Ahora, si nos fijamos en los registros veremos que *%rax* tiene el valor de la *passcode* (*1314*).



Tenemos que cambiar el valor del registro para que la comparación del valor salga bien. En nuestro caso, como hemos introducido “111” como PIN, tendremos que poner que *%rax* valga “111”

```
set $eax=111
```

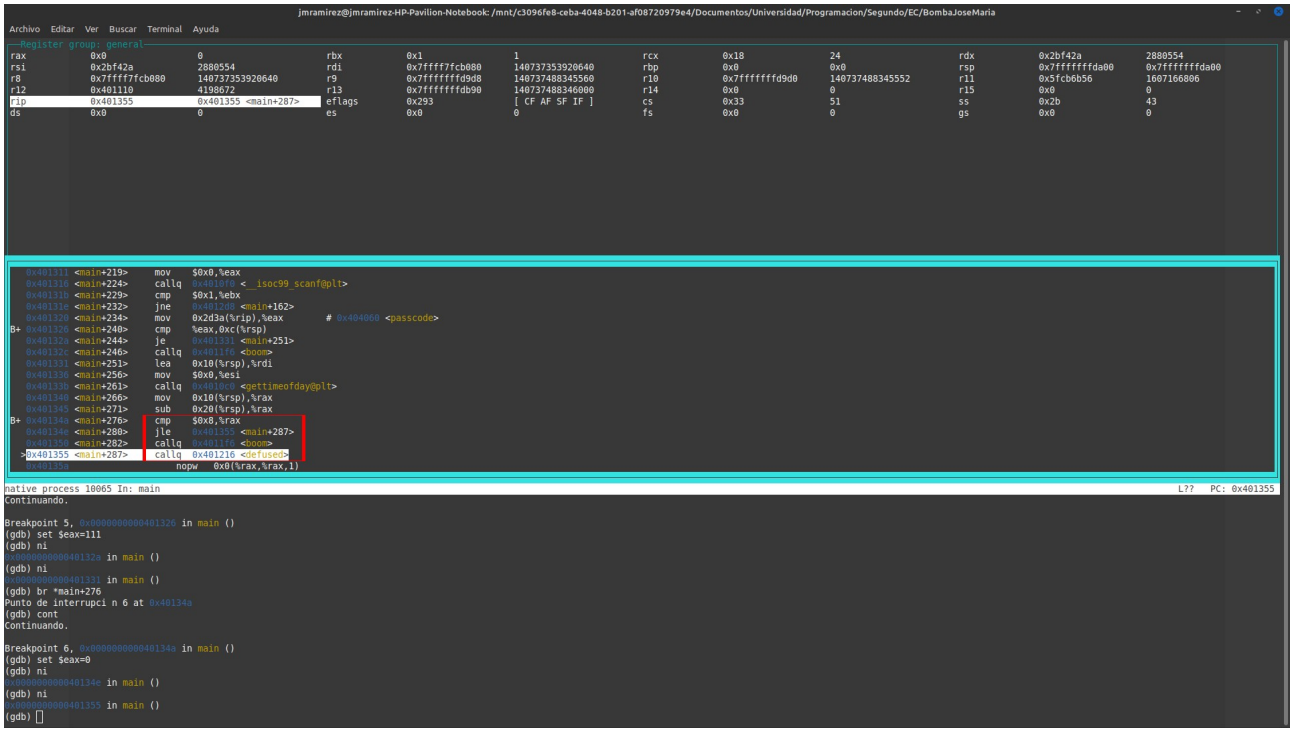
Y si ejecutamos *ni*, veremos que nos saltamos el *boom*.



Ahora tendremos que desactivar el contador del tiempo, que se realiza exactamente igual que como lo hicimos anteriormente.
Vamos a *main+276*, donde se realiza el *cmp*, y cambiamos el valor de *%rax* a 0

br *main+276
cont
set $eax=0

Ahora al ejecutar ni hasta llegar al final del programa, veremos que a bomba ha quedado desactivada.



CAMBIO DE CLAVES CON GHEX

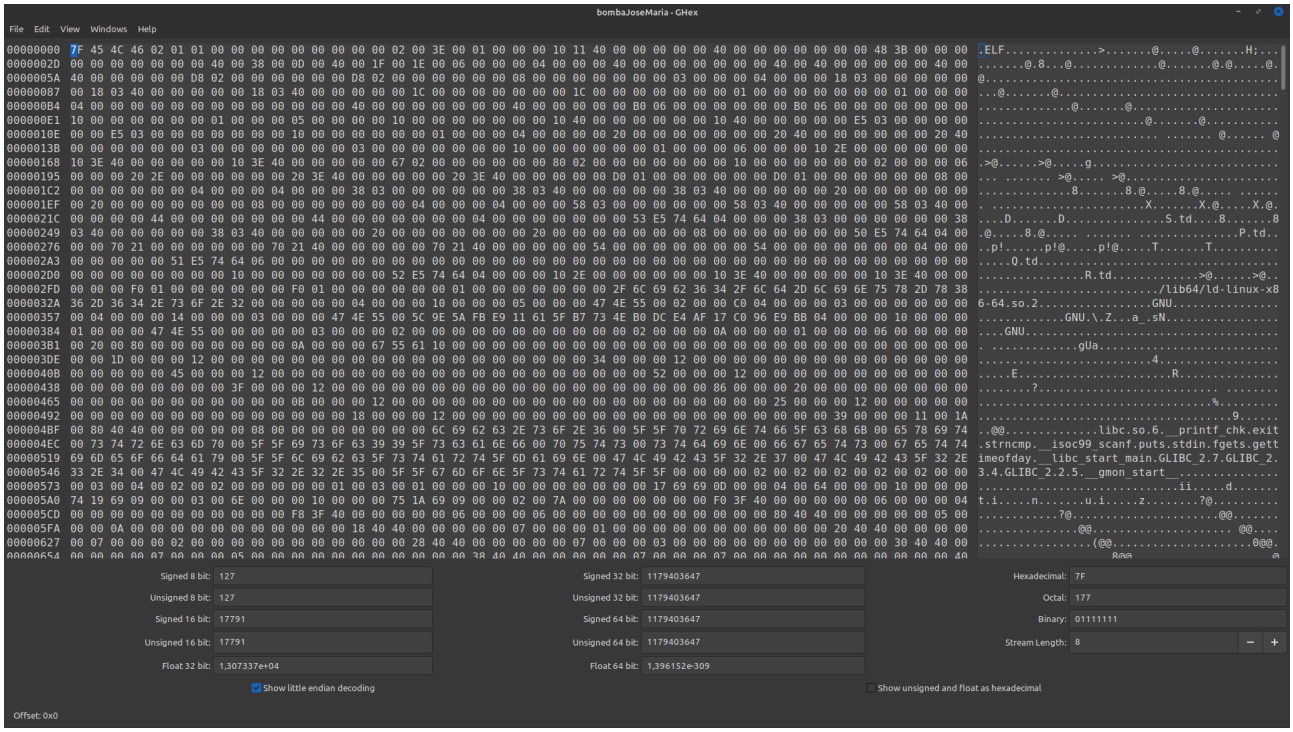
Para realizar el cambio de claves, usaremos *ghex*.
Si no tenemos este programa instalado, podemos instalarlo con la siguiente orden

sudo apt install ghex

Una vez instalado, vamos a abrir *ghex* con la siguiente orden

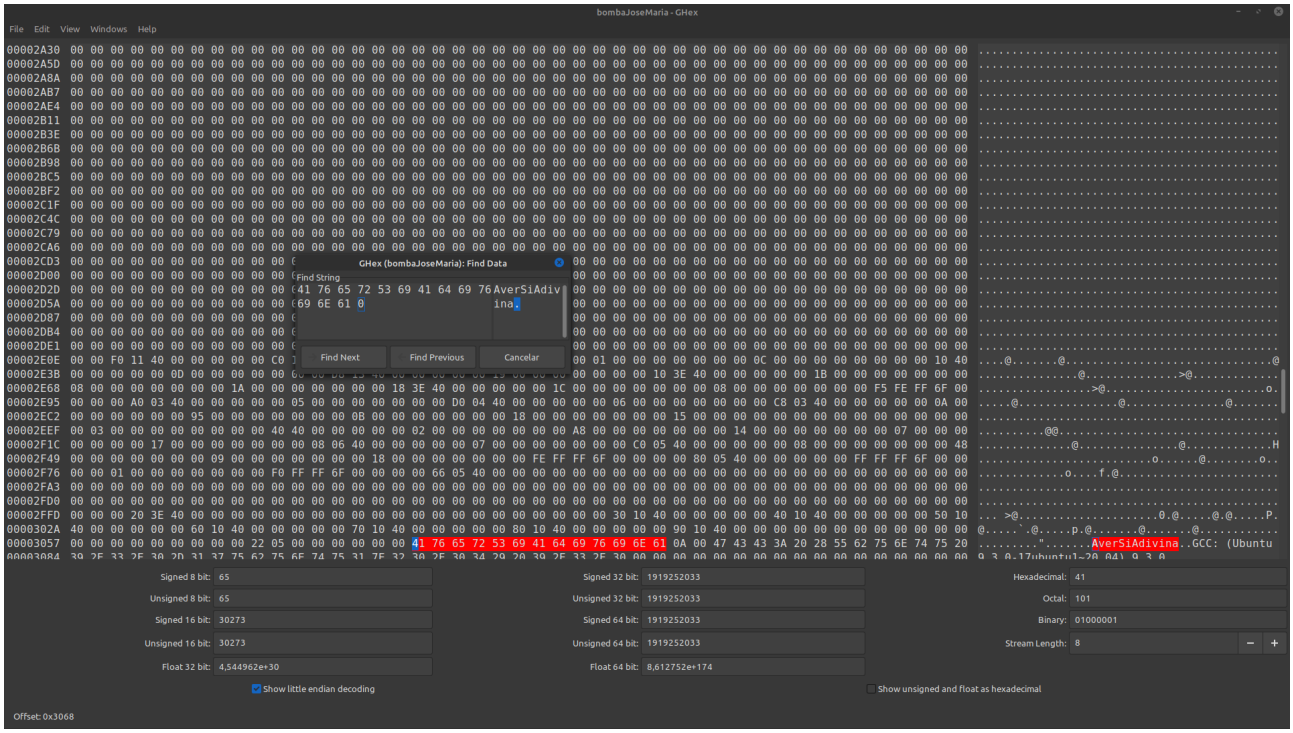
ghex <nombre de la bomba>

Y nos debería abrir el programa con una interfaz similar a esta

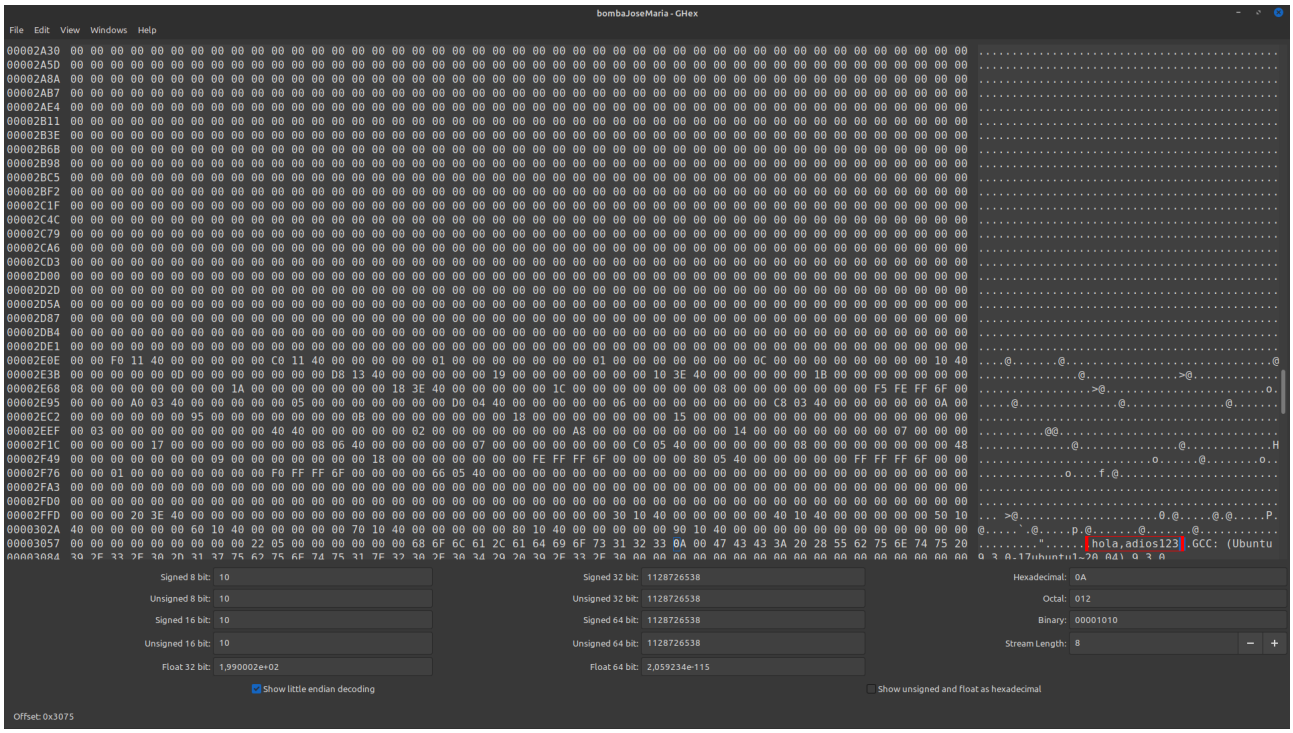


Recordemos que ya sabemos las claves por el apartado anterior (“AverSiAdivina”, “*1314*”).
Para encontrar estas claves entre tanto hexadecimal, podemos usar la función de búsqueda de *ghex*. Si pulsamos Ctrl+F se nos abre el menú de búsqueda.
Vamos a empezar con la contraseña.

Escribimos “AverSiAdivina” en el menú de búsqueda y vemos que solo tenemos una coincidencia.

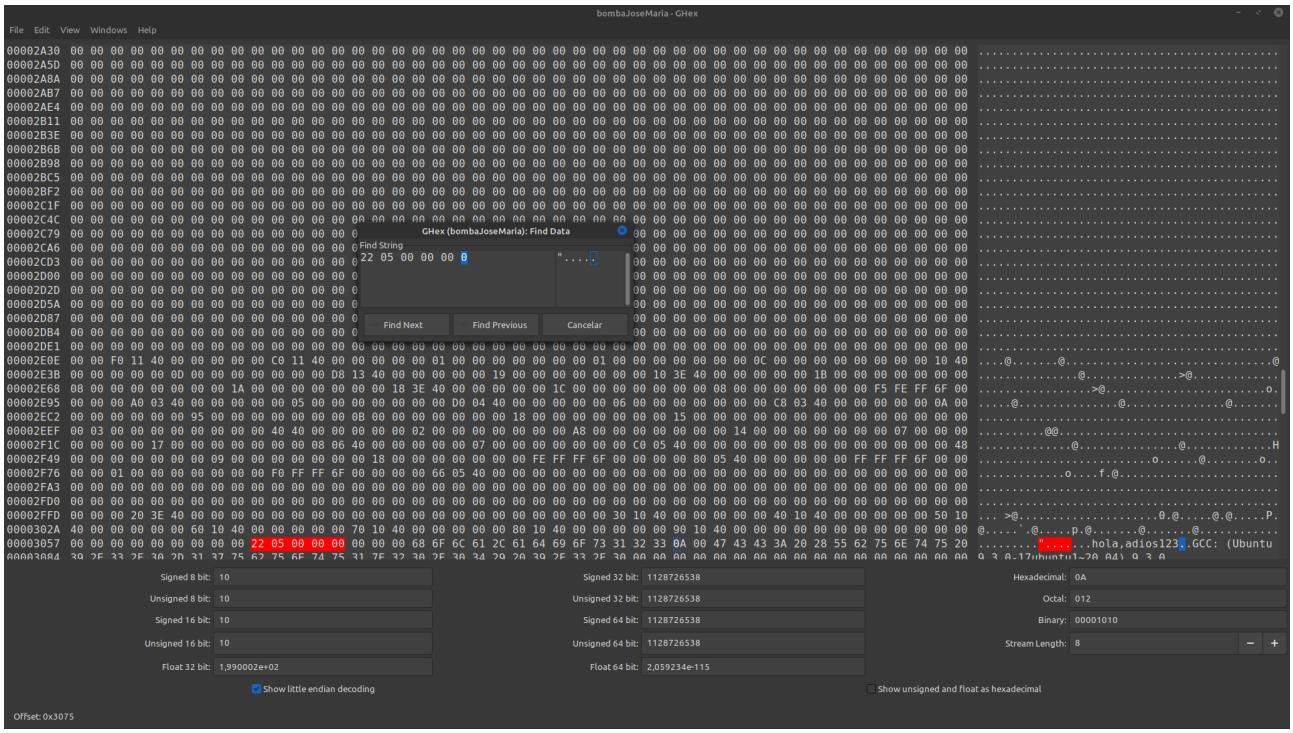


Para cambiar la contraseña, vamos a lo señalado en rojo a la derecha, clickamos en la primera letra y vamos escribiendo.
Por ejemplo podemos escribir “hola,adios123”. Después de cambiarla, guardamos con *Ctrl+S*.



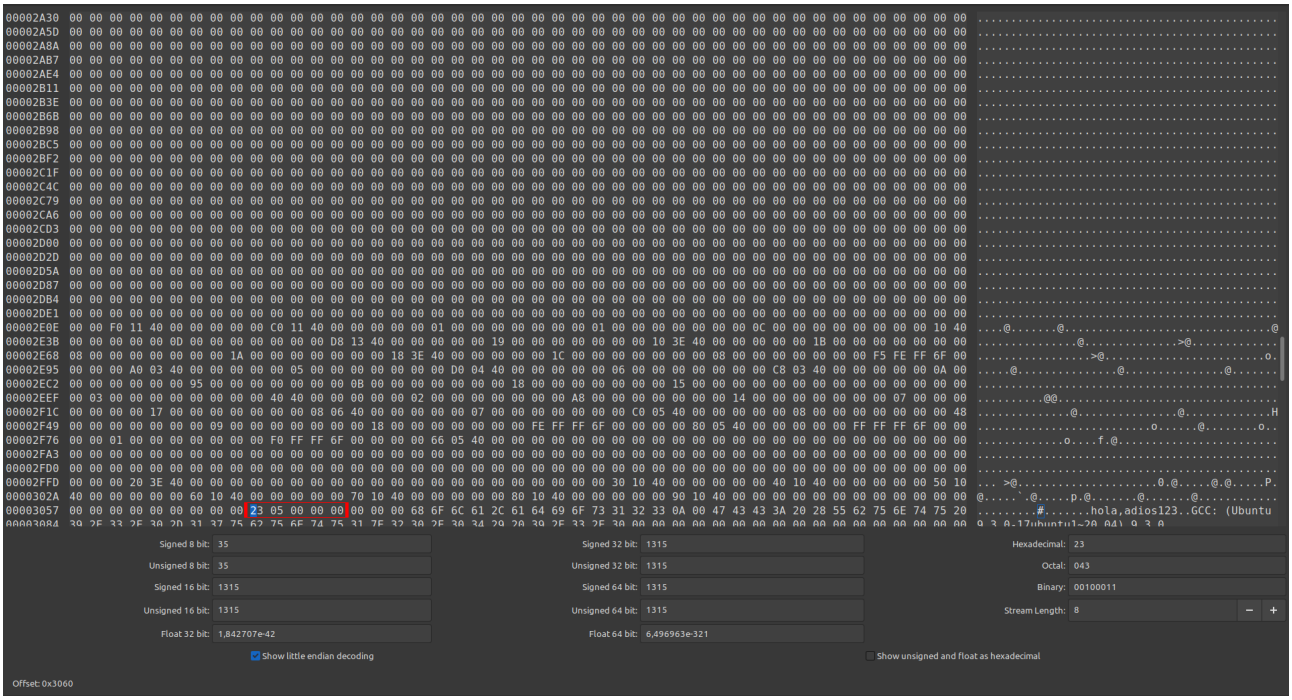
Para cambiar el PIN necesitaremos la calculadora de Ubuntu.
Pasaremos el PIN a hexadecimal con la calculadora en modo programación.
$1314_{10} \rightarrow 522_{16}$
Una vez que tenemos el valor en hexadecimal, lo buscamos en el cuadro de búsquedas de *ghex*, pero en little-endian.

Es decir 00 00 00 05 22 → 22 05 00 00 00.



Como vemos solo tenemos 1 coincidencia. Si cambiamos los dígitos (es decir, pasamos, por ejemplo, de *22 05 00 00 00* a *23 05 00 00 00*) el PIN pasará a ser el nuevo valor de esa cadena (**Recordemos, en little-endian**).

Así pues, yo voy a cambiar el PIN a *1315*.



Después de cambiar el PIN, guardamos con *Ctrl+S* y una vez que volvamos a ejecutar el programa, la nueva contraseña será “hola,adios123” y el nuevo PIN será *1315*