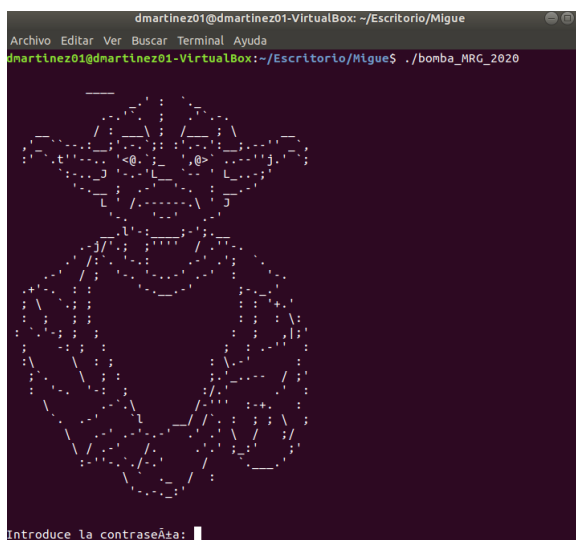


# Desactivar Bomba Digital

David Martínez Díaz

## Bomba\_MRG\_2020:

1.- En primer lugar es ejecutarlo, donde vemos que tenemos que averiguar su contraseña y pin correspondiente.

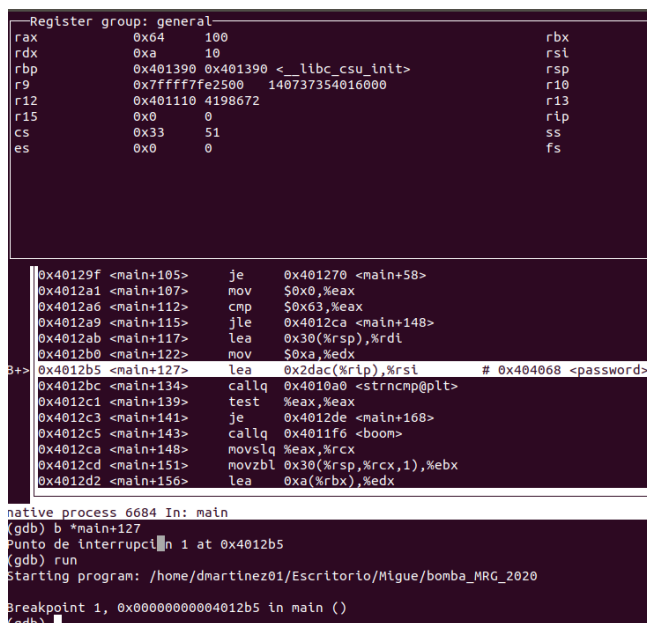


2.- Para ello debemos introducirnos en dicho ejecutable a través del debug e ir mirando los registros para comprobar y obtener las contraseñas.

→ gdb bomba

→ layout regs

Una vez estamos debugeando, hay que buscar el lugar exacto donde se comparan las contraseñas para poder continuar con el ejercicio, donde si nos damos cuenta, esta se realiza en el string compare, en la línea (\*main+127).



Por tanto, para sacar su contraseña debo ver los valores que hay en el registro %rsi, viendo así su contraseña correspondiente:

```

0x40129f <main+105>    je     0x401270 <main+58>
0x4012a1 <main+107>    mov    $0x0,%eax
0x4012a6 <main+112>    cmp    $0x63,%eax
0x4012a9 <main+115>    jle     0x4012ca <main+148>
0x4012ab <main+117>    lea     0x30(%rsp),%rdi
0x4012b0 <main+122>    mov    $0xa,%edx
0x4012b5 <main+127>    lea     0x2dac(%rip),%rsi      # 0x404068 <password>
0x4012bc <main+134>    callq  0x4010a0 <strncmp@plt>
0x4012c1 <main+139>    test   %eax,%eax
> 0x4012c3 <main+141>    je     0x4012de <main+168>
0x4012c5 <main+143>    callq  0x4011f6 <boom>
0x4012ca <main+148>    movslq %eax,%rcx
0x4012cd <main+151>    movzbl 0x30(%rsp,%rcx,1),%ebx
0x4012d2 <main+156>    lea     0xa(%rbx),%edx

native process 6684 In: main
(gdb) b *main+127
Breakpoint 1 at 0x4012b5
(gdb) run
Starting program: /home/dmartinez01/Escritorio/Migue/bomba_MRG_2020

Breakpoint 1, 0x00000000004012b5 in main ()
(gdb) ni
0x00000000004012bc in main ()
(gdb) ni
0x00000000004012c1 in main ()
(gdb) ni
0x00000000004012c3 in main ()
(gdb) x/s $rsi
0x404068 <password>:  "xy}{s-o|k\024"
(gdb)

```

Esto quiere decir que la contraseña esta encriptada, y no vamos a poder poner directamente el resultado de %rsi, para ello podemos comprobar con nuestra contraseña anterior y mirar en el registro %rdi para ver dicha comparación y calcular esa razón correspondiente para saber cuánto hay que sumarlo o restarle a esta.

Por ello si introducimos la contraseña “hola” y nos introducimos en el registro %rdi obtenemos lo siguiente:

```
(gdb) x/s $rdi
0x7fffffffde10: "ryvk\024", '\n' <repetidos 11 veces>, "\027\n\n\n\n\n\n\n\nj`\347
\001\t\211\n\n\242\350\t\t\t\211\n\n\t\277\372\n\n\n\n\n\314\n\n\n\n\n\n\n\347\0
35J\n\n\n\n\njE\350\001\t\211\n\n\n\n\n\n\n\n\n\n\232\035J\n\n\n\n\n\032\033J\n
n\n\n\nj\351\t\t\377\177"
(gdb)
```

Donde nuestra nueva contraseña encriptada seria “ryvk”, para saber la razón de encriptación simplemente basta saber que valor llega desde la primera letra de nuestra contraseña hasta la primera letra de la contraseña encriptada:

“hola” ---> “ryvk”;

“h” ---> “r”

Para sacarlo nos vamos a la tabla ASCII y vemos sus correspondientes valores:


## “h” = 104 // “r” = 114

Con esto llegamos a la conclusión de que la razón es 10, y simplemente para saber su contraseña hay que restarle 10 a cada letra.

```
"xy}{s~o|k" ---> "nosqitera";
```

Vamos a comprobarlo:

```
dmartinez01@dmartinez01-VirtualBox:~/Escritorio/Migue$ ./bomba_MRG_2020
```



```
Introduce la contraseÃ±a: nosqitera
```

```
Introduce el pin: █
```

Vemos que la contraseña era correcta pero se nos presenta otra problema, ahora nos pide un pin, veamos entonces de nuevo en el gdb, donde se encuentra este.

Como podemos ver se realiza otro string compare el cual vamos a analizar para saber si ahí se realiza la comparación entre pines. Analizando las sentencias llegamos hasta el main+276, donde se realiza una comparación entre los registros con la orden “jne”:

```

Register group: general
rax      0x1      1      rbx
rdx      0x7ffff7dcf8d0  140737351842000  rsi
rbp      0x401390  0x401390  <__libc_csu_init>  rsp
r9        0x0      0      r10
r12       0x401110  4198672  r13
r15       0x0      0      rip
cs        0x33     51      ss
es        0x0      0      fs

```

```

0x401332 <main+252>    jne    0x401345 <main+271>
0x401334 <main+254>    lea    0x1944(%rip),%rdi    # 0x402c7f
0x40133b <main+261>    mov    %0x0,%eax
0x401340 <main+266>    callq 0x4010f0 <__isoc99_scanf@plt>
0x401345 <main+271>    cmp    %0x1,%ebx
0x401348 <main+274>    jne    0x401302 <main+204>
B> 0x40134a <main+276>    mov    0x2d10(%rip),%eax    # 0x404060 <passco
0x401350 <main+282>    cmp    %eax,0xc(%rsp)
0x401354 <main+286>    je     0x40135b <main+293>
0x401356 <main+288>    callq  0x4011f6 <boom>
0x40135b <main+293>    lea    0x10(%rsp),%rdi
0x401360 <main+298>    mov    %0x0,%esi
0x401365 <main+303>    callq  0x4010c0 <gettimeofday@plt>
0x40136a <main+308>    mov    0x10(%rsp),%rax

```

```

native process 6777 In: main
(gdb) b *main+276
Punto de interrupci3n 1 at 0x40134a
(gdb) run
Starting program: /home/dmartinez01/Escriptorio/Migue/bomba_MRG_2020

Breakpoint 1, 0x000000000040134a in main ()
(gdb)

```

Donde si miramos bien las sentencias y hacemos un par de next instructions vemos como sacan un dato de la pila en la dirección de (%rip+0x2d10), y lo mete en el registro %eax, si lo mostramos vemos que:

```
Breakpoint 1, 0x000000000040134a in main ()
(gdb) ni
0x0000000000401350 in main ()
(gdb) ni
0x0000000000401354 in main ()
(gdb) print $eax
$1 = 9514
(gdb)
```

Vamos a comprobar si este pin es el correcto:



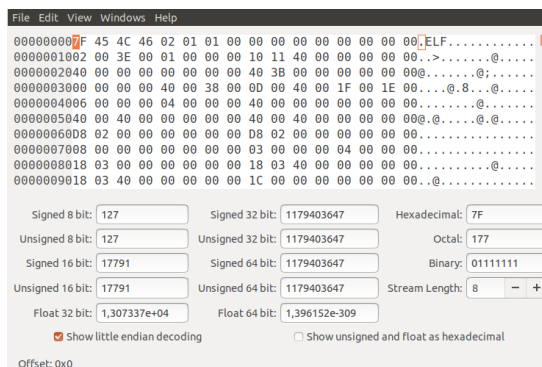
Como hemos podido comprobar el pin que habíamos obtenido era correcto y no se encontraba encriptado por lo que hemos podido desactivar la bomba. Por lo que esta sería la manera de desactivar la bomba.

### - Modificar la contraseña y pin de la bomba:

Para modificar la contraseña vamos a utilizar el comando ghex que permite ver el código de manera hexadecimal:

“ghex bomba\_MRG\_2020”

Donde nos aparecerá lo siguiente:



Una vez estamos dentro, tenemos que buscar dicha contraseña, que en mi caso se encuentra encriptada:

```
...p.@.....@.....@.....  
*%.....xy}{s~o|k.GCC: (Ubuntu 9.3.0-17ubun  
tu1~20.04) 9.3.0.....  
.....@.....8.@.....
```

Una vez que la hemos sacado vemos su Offset, simplemente si la modificamos debería cambiar (teniendo en cuenta la razón de encriptación). Por ejemplo, si yo cambio la contraseña de esta manera:

“Pajareria” → “Ztkk|o|sk”

```
dmartinez01@dmartinez01-VirtualBox:~/Escritorio/Migue$ ./bomba_MRG_2020
```



```
Introduce la contraseña: pajareria  
Introduce el pin: █
```

Vemos que si me acepta la contraseña y la hemos podido modificar con éxito. Para modificar el pin, vamos a emplear otra vez el comando ghex:

Signed 8 bit:	<input type="text" value="42"/>
Unsigned 8 bit:	<input type="text" value="42"/>
Signed 16 bit:	<input type="text" value="9514"/>
Unsigned 16 bit:	<input type="text" value="9514"/>
Float 32 bit:	<input type="text" value="1,333195e-41"/>

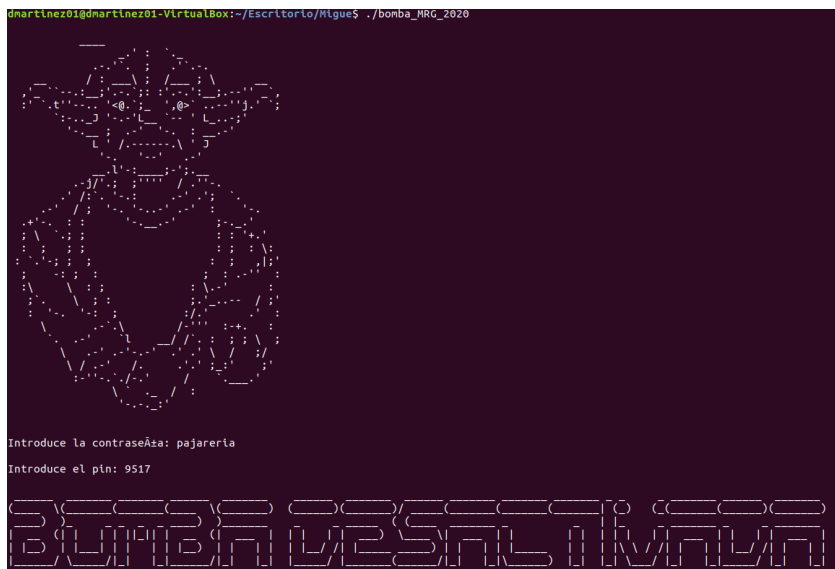
☒ Show little endian decoding

Una vez encontramos nuestro pin, probamos a cambiarlo poniendo F4 por ejemplo, quedando:

Signed 8 bit:	<input type="text" value="45"/>
Unsigned 8 bit:	<input type="text" value="45"/>
Signed 16 bit:	<input type="text" value="9517"/>
Unsigned 16 bit:	<input type="text" value="9517"/>
Float 32 bit:	<input type="text" value="1,333616e-41"/>

☒ Show little endian decoding

Por ultimo vamos a comprobar si funciona:



Y confirmamos que se ha modificado con éxito.