

## Práctica 4

## Salvador Romero Cortés

La clave original sin ningún tipo de cifrado es: tuxinatux.

El pin es: 218

Sin embargo la contraseña se almacena en el programa inicialmente como un desplazamiento -1 en el alfabeto, siendo almacenada como `stwhmzstw`. A la hora de la comprobación con la introducida por el usuario ambas tienen un desplazamiento positivo (9 para la introducida por el usuario y 10 para la que hay que descubrir).

Por lo tanto la clave cuando se compara es: `cdgrwjcdg`.

El pin no sufre cambios a lo largo de la ejecución del programa.

## Capturas del funcionamiento del programa

```

fish home/salva/zinho/EC/Practica4 Practic4
salva@salva-desktop ~/7/E/P/Practical4 (master) gcc -Og bomba_salva_r.c -o bomba_salva -no-pie -fno-guess-branch-probability
bomba_salva_r.c: In function 'desactivar':
bomba_salva_r.c:80:1: warning: unknown escape sequence: '\.'
    80 |   "\."
       |   ^~
       |   ~~~~~
bomba_salva_r.c:80:1: warning: unknown escape sequence: '\d'
bomba_salva_r.c:80:1: warning: unknown escape sequence: '\z'
bomba_salva_r.c:80:1: warning: unknown escape sequence: '\.'
bomba_salva_r.c:80:1: warning: unknown escape sequence: '\.'
--
salva@salva-desktop ~/7/E/P/Practical4 (master) ./bomba_salva
Introduce la contraseña: estanoes
+
+
+
+
I.. I I I .
.. I II .
... .. I..
III I...
IIII.....II .....
I.. ...II ..
      + I.. I.
      + I II I.
      II :BRRVV
      VB BRRRVR
      :BVIRRRRRR
      -.VRRRRRRRRRRR
      :RRRRRRRRRRRRR;
      :RRRR:-RRRRRRRRRRRV.
      :RRR:-RRRRRRRRRRRRRRR.
      :RRR: :RRRRRRRRRRRRRRRRR.
      :RRR: :RRRRRRRRRRRRRRRRRRRV
      XRB :RRRRRRRRRRRRRRRRRRRRR.
      BRV :RRRRRRRRRRRRRRRRRRRRRL
      :RRR- :RRRRRRRRRRRRRRRRRRRRRVY
      BRV :RRRRRRRRRRRRRRRRRRRRRRR.
      VBR :RRRRRRRRRRRRRRRRRRRRRRR.
      :RRR- :RRRRRRRRRRRRRRRRRRRRRRR
      :RRR- :RRRRRRRRRRRRRRRRRRRRRRR
      LNR:RRRRRRRRRRRRRRRRRRRRRRR.
      :RRRRRRRRRRRRRRRRRRRRRRRRR.
      :RRRRRRRRRRRRRRRRRRRRRVY
      :RRRRRRRRRRRRRRRV;
      --VRRRRRVV;
salva@salva-desktop ~/7/E/P/Practical4 (master)|

```

[illegible]



```
fish/home/salva/Zinfo/EC/Practicas/Practica4
x
gbf/home/salva/Zinfo/EC/Practicas/Practica4

B> 0x401290 <main>      endbr64
0x401290 <main+4>      push    rbp
0x401290 <main+5>      sub     $0xa0,%rsp
0x4012a4 <main+12>     mov     rsi,0x20,%rax
0x4012a6 <main+21>     mov     rax,0x00(%rsp)
0x4012b5 <main+29>     xor     %eax,%eax
0x4012b7 <main+31>     mov     $0xa,%esi
0x4012bc <main+36>     lea     0x20d(%rip),%rdi    # 0x404060 <pass>
0x4012c3 <main+43>     callq  0x4011f6 <cfrr>
0x4012c8 <main+48>     lea     0x10(%rsp),%rdi
0x4012cc <main+53>     mov     $0x0,%esi
0x4012d2 <main+58>     callq  0x4010c0 <gettimeofday@plt>
0x4012d7 <main+63>     lea     0xd36(%rip),%rsi    # 0x402014
0x4012de <main+78>     mov     $0x1,%edi
0x4012e3 <main+75>     mov     $0x0,%eax
0x4012e8 <main+88>     callq  0x4010e0 <_printf_chk@plt>
0x4012ee <main+95>     lea     0x30(%rsp),%rdi
0x4012f2 <main+98>     mov     0x2077(%rip),%rdx    # 0x404070 <stdin@Glibc_2.2.5>
0x4012f9 <main+97>     mov     $0x04,%esi
0x4012fe <main+102>    callq  0x401000 <getc@plt>
0x401303 <main+107>    test    %rax,%rax
0x401306 <main+112>    lea     0x30(%rsp),%rbx
0x401308 <main+117>    mov     $0x9,%esi
0x401312 <main+122>    mov     %rbx,%rdi
0x401315 <main+125>    callq  0x4011f6 <cfrr>
0x40131a <main+138>    mov     $0x9,%edx
0x40131f <main+135>    lea     0x203a(%rip),%rsi    # 0x404060 <pass>
0x401326 <main+142>    mov     %rbx,%rdi
0x401329 <main+145>    callq  0x4010a0 <strncmp@plt>
0x40132e <main+158>    test    %eax,%eax
0x401330 <main+152>    je      0x40133c <main+164>
0x401332 <main+154>    mov     $0x0,%eax
0x401337 <main+159>    callq  0x40122c <exploat>
0x40133c <main+164>    lea     0x20(%rsp),%rdi

native process 71134 in: main
0x000000000040129c in main ()
s1 = 0x404060 <pass> "stuhmztw"
(gdb) n1
0x00000000004012c3 in main ()
(gdb) n1
0x00000000004012c8 in main ()
(gdb) n1
0x00000000004012cd in main ()
0x00000000004012d2 in main ()
0x00000000004012d7 in main ()
0x00000000004012de in main ()
0x00000000004012e3 in main ()
0x00000000004012e8 in main ()
0x00000000004012f2 in main ()
0x00000000004012f9 in main ()
0x00000000004012fe in main ()
Introduce la contraseña: aaaa
0x0000000000401303 in main ()
(gdb) |
```

Vemos como después de introducir la contraseña esta se pasa como argumento a la función "cifrar".

```
fish/home/salva/Zinfo/EC/Practicas/Practica4
x
gbf/home/salva/Zinfo/EC/Practicas/Practica4

B> 0x401290 <main>      endbr64
0x401290 <main>      endbr64
0x401290 <main+4>      push    rbp
0x401290 <main+5>      sub     $0xa0,%rsp
0x4012a4 <main+12>     mov     rsi,0x20,%rax
0x4012a6 <main+21>     mov     rax,0x00(%rsp)
0x4012b5 <main+29>     xor     %eax,%eax
0x4012b7 <main+31>     mov     $0xa,%esi
0x4012bc <main+36>     lea     0x20d(%rip),%rdi    # 0x404060 <pass>
0x4012c3 <main+43>     callq  0x4011f6 <cfrr>
0x4012c8 <main+48>     lea     0x10(%rsp),%rdi
0x4012cc <main+53>     mov     $0x0,%esi
0x4012d2 <main+58>     callq  0x4010c0 <gettimeofday@plt>
0x4012d7 <main+63>     lea     0xd36(%rip),%rsi    # 0x402014
0x4012de <main+78>     mov     $0x1,%edi
0x4012e3 <main+75>     mov     $0x0,%eax
0x4012e8 <main+88>     callq  0x4010e0 <_printf_chk@plt>
0x4012ee <main+95>     lea     0x30(%rsp),%rdi
0x4012f2 <main+98>     mov     0x2077(%rip),%rdx    # 0x404070 <stdin@Glibc_2.2.5>
0x4012f9 <main+97>     mov     $0x04,%esi
0x4012fe <main+102>    callq  0x401000 <getc@plt>
0x401303 <main+107>    test    %rax,%rax
0x401306 <main+112>    lea     0x30(%rsp),%rbx
0x401308 <main+117>    mov     $0x9,%esi
0x401312 <main+122>    mov     %rbx,%rdi
0x401315 <main+125>    callq  0x4011f6 <cfrr>
0x40131a <main+138>    mov     $0x9,%edx
0x40131f <main+135>    lea     0x203a(%rip),%rsi    # 0x404060 <pass>
0x401326 <main+142>    mov     %rbx,%rdi
0x401329 <main+145>    callq  0x4010a0 <strncmp@plt>
0x40132e <main+158>    test    %eax,%eax
0x401330 <main+152>    je      0x40133c <main+164>
0x401332 <main+154>    mov     $0x0,%eax
0x401337 <main+159>    callq  0x40122c <exploat>
0x40133c <main+164>    lea     0x20(%rsp),%rdi

native process 71134 in: main
0x000000000040129c in main ()
0x00000000004012cd in main ()
0x00000000004012d2 in main ()
0x00000000004012d7 in main ()
0x00000000004012f2 in main ()
0x00000000004012f9 in main ()
0x00000000004012fe in main ()
Introduce la contraseña: aaaa
0x0000000000401303 in main ()
(gdb) n1
0x0000000000401306 in main ()
0x0000000000401308 in main ()
0x000000000040130a in main ()
0x0000000000401312 in main ()
0x0000000000401315 in main ()
(gdb) p (char *) srdi
s2 = 0x7fffffa460 <_IO_stdfile_0_lock> ""
(gdb) n1
0x0000000000401315 in main ()
(gdb) p (char *) srdi
s3 = 0x7fffff7fe570 "aaaa\n"
(gdb) |
```

A continuación, se llama a `strncmp` y por lo tanto sabemos que se va a realizar la comparación de las contraseñas. Si mostramos los registros `%rdi` y `%rsi` podremos ver los valores que se comparan.

```
00401290 <main>      endbr64
00401290 <main>      endbr64
0040129c <main+4>    push    rrbx
0040129e <main+5>    sub     $0x0,rbp
004012a0 <main+12>   mov     r15,rbx
004012a4 <main+21>   mov     rax,0x00000000
004012a5 <main+29>   xor     %eax,%eax
004012a7 <main+31>   mov     $0x0,%esi
004012ac <main+36>   lea     0x209d(rbp),rdi      # 0x404060 <pass>
004012c3 <main+43>   callq   0x4011f6 <cfrrar>
004012c5 <main+48>   lea     0x10(rbp),rdi
004012c8 <main+53>   mov     $0x0,%esi
004012d7 <main+58>   callq   0x4018c8 <gettimeofday@plt>
004012d7 <main+58>   lea     0x36(rbp),rsi      # 0x402014
004012de <main+78>   mov     $0x1,%edi
004012e3 <main+75>   mov     $0x0,%eax
004012e5 <main+80>   callq   0x4010e9 <_printf_chk@plt>
004012e8 <main+85>   lea     0x30(rbp),rdi
004012f2 <main+98>   mov     0x2077(rbp),rdx      # 0x404070 <stdin@Glibc_2.2.5>
004012f5 <main+97>   mov     $0x04,%esi
004012fe <main+102>  callq   0x4018d8 <gets@plt>
00401303 <main+107>  test    rax,rax
00401306 <main+110>  je       0x4012e7 <main+63>
00401308 <main+112>  lea     0x30(rbp),rbx
0040130e <main+117>  mov     $0x0,%esi
00401312 <main+122>  mov     rrbx,rdi
00401315 <main+125>  callq   0x4011f6 <cfrrar>
0040131a <main+138>  mov     $0x0,%edx
0040131f <main+135>  lea     0x203a(rbp),rsi      # 0x404060 <pass>
00401326 <main+142>  mov     rrbx,rdi
00401329 <main+145>  callq   0x4018a8 <strncmp@plt>
0040132e <main+150>  test    %eax,%eax
00401330 <main+152>  je       0x40133c <main+164>
00401332 <main+154>  mov     $0x0,%eax
00401337 <main+159>  callq   0x40122c <exploat>
0040133c <main+164>  lea     0x20(rbp),rdi

native process 71134 in: main
0x000000000040132e in main ()
(gdb) p (char *) $rdi
$2 = 0x7fffffff6460 <_IO_stdfile_0_lock> ""
(gdb) ni
0x0000000000401315 in main ()
(gdb) p (char *) $rdi
$3 = 0x7fffffff570 "aaaaa\n"
(gdb) ni
0x000000000040131a in main ()
(gdb) ni
0x000000000040131f in main ()
(gdb) ni
0x0000000000401326 in main ()
(gdb) ni
0x0000000000401329 in main ()
(gdb) p (char*) $rsi
$4 = 0x404060 <pass> "cdgrvjcdg"
(gdb) p (char*) $rdi
$5 = 0x7fffffff570 "jjjj\023"
(gdb) |
```

Vemos entonces que ninguna coincide con las que originalmente introdujimos ni con la que descubrimos en la posición `0x404060`.

Sin embargo, se puede notar que la contraseña `aaaa` pasa a ser `jjjj` de lo que se puede deducir que ha habido un desplazamiento de 9 posiciones en el alfabeto. De forma análoga con las contraseña original se descubre que el desplazamiento en ese caso es de 10.

Lo siguiente será saltarse la comprobación para poder continuar con la ejecución del programa. Cuando se llega a la instrucción `test` sabemos que nos llevará a la explosión de la bomba o seguirá con la ejecución normal.

Asignamos el valor inmediato 0 a `%eax` para seguir con el programa.

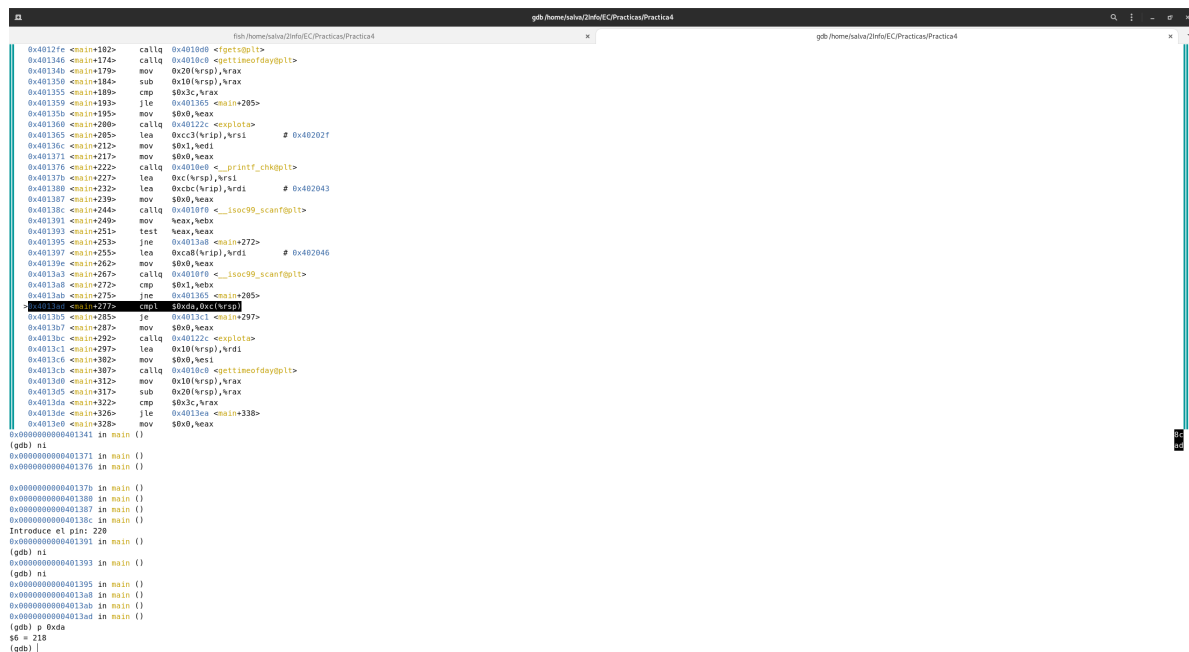
```
00401290 <main>      endbr64
00401290 <main>      endbr64
0040129c <main+4>    push    rrbx
0040129e <main+5>    sub     $0x0,rbp
004012a0 <main+12>   mov     r15,rbx
004012a4 <main+21>   mov     rax,0x00000000
004012a5 <main+29>   xor     %eax,%eax
004012a7 <main+31>   mov     $0x0,%esi
004012ac <main+36>   lea     0x209d(rbp),rdi      # 0x404060 <pass>
004012c3 <main+43>   callq   0x4011f6 <cfrrar>
004012c5 <main+48>   lea     0x10(rbp),rdi
004012c8 <main+53>   mov     $0x0,%esi
004012d7 <main+58>   callq   0x4018c8 <gettimeofday@plt>
004012d7 <main+58>   lea     0x36(rbp),rsi      # 0x402014
004012de <main+78>   mov     $0x1,%edi
004012e3 <main+75>   mov     $0x0,%eax
004012e5 <main+80>   callq   0x4010e9 <_printf_chk@plt>
004012e8 <main+85>   lea     0x30(rbp),rdi
004012f2 <main+98>   mov     0x2077(rbp),rdx      # 0x404070 <stdin@Glibc_2.2.5>
004012f5 <main+97>   mov     $0x04,%esi
004012fe <main+102>  callq   0x4018d8 <gets@plt>
00401303 <main+107>  test    rax,rax
00401306 <main+110>  je       0x4012e7 <main+63>
00401308 <main+112>  lea     0x30(rbp),rbx
0040130e <main+117>  mov     $0x0,%esi
00401312 <main+122>  mov     rrbx,rdi
00401315 <main+125>  callq   0x4011f6 <cfrrar>
0040131a <main+138>  mov     $0x0,%edx
0040131f <main+135>  lea     0x203a(rbp),rsi      # 0x404060 <pass>
00401326 <main+142>  mov     rrbx,rdi
00401329 <main+145>  callq   0x4018a8 <strncmp@plt>
0040132e <main+150>  test    %eax,%eax
00401330 <main+152>  je       0x40133c <main+164>
00401332 <main+154>  mov     $0x0,%eax
00401337 <main+159>  callq   0x40122c <exploat>
0040133c <main+164>  lea     0x20(rbp),rdi

native process 71134 in: main
0x000000000040132e in main ()
(gdb) ni
0x000000000040131f in main ()
(gdb) ni
0x0000000000401326 in main ()
(gdb) ni
0x0000000000401329 in main ()
(gdb) p (char*) $rsi
$4 = 0x404060 <pass> "cdgrvjcdg"
(gdb) p (char*) $rdi
$5 = 0x7fffffff570 "jjjj\023"
(gdb) ni
0x000000000040132e in main ()
(gdb) set $eax=0
No symbol "eax" in current context.
(gdb) set $eax=0
(gdb) ni
0x0000000000401330 in main ()
(gdb) |
```

A continuación habrá comprobaciones del tiempo que se ha tardado en completar la entrada de la contraseña. Si fuera necesario se puede poner `%rax` como 0 para poder seguir con normalidad.

Finalmente llegamos a la entrada del pin.

Aquí insertamos un valor cualquiera y seguimos con la ejecución del programa hasta `main+277` donde vemos que hay una comparación de enteros. Vemos que se compara el valor inmediato `$0xda` que es el pin de la bomba. Lo podemos mostrar en gdb para descubrir que es 218 en decimal.



```
0x40127c <main+182> callq 0x401808 <?getsgpt?>
0x401346 <main+174> callq 0x4018c0 <?gettimeofday?>
0x401346 <main+174> mov 0x20(%rsp),%rax
0x401350 <main+184> sub 0x10(%rsp),%rax
0x401355 <main+189> cmp $0x3c,%rax
0x401359 <main+193> jle 0x401365 <main+205>
0x401359 <main+193> mov $0x0,%eax
0x401360 <main+200> callq 0x40122c <?explo?>
0x401365 <main+205> lea 0xc03(%rip),%rsi # 0x40202f
0x40136c <main+212> mov $0x1,%edi
0x401371 <main+217> mov $0x0,%eax
0x401376 <main+222> callq 0x4010e0 <?printf_chk?>
0x401376 <main+222> lea 0xc(%rip),%rsi
0x401380 <main+232> lea 0xc0c(%rip),%rdi # 0x402043
0x401387 <main+239> mov $0x0,%eax
0x40138c <main+246> callq 0x4010ff <?__libc99_scanf?>
0x401391 <main+249> mov %eax,%ebx
0x401393 <main+251> test %eax,%eax
0x401395 <main+253> jne 0x4013a0 <main+272> # 0x402046
0x401397 <main+255> lea 0xca8(%rip),%rdi # 0x402046
0x40139c <main+262> mov $0x0,%eax
0x4013a1 <main+269> callq 0x4010ff <?__libc99_scanf?>
0x4013a8 <main+272> cmp $0x1,%ebx
0x4013ab <main+275> jne 0x4013b5 <main+285>
0x4013ab <main+275> jmp 0x4013c0 <?explo?>
0x4013b5 <main+285> je 0x4013c1 <main+297>
0x4013b7 <main+287> mov $0x0,%eax
0x4013bc <main+292> callq 0x40122c <?explo?>
0x4013c1 <main+297> lea 0x10(%rsp),%rdi
0x4013c6 <main+302> mov $0x0,%esi
0x4013ca <main+307> callq 0x4010c0 <?gettimeofday?>
0x4013c0 <main+312> mov 0x10(%rsp),%rax
0x4013c5 <main+317> sub 0x20(%rsp),%rax
0x4013c6 <main+322> cmp $0x3c,%rax
0x4013c6 <main+326> jle 0x4013e0 <main+338>
0x4013c0 <main+328> mov $0x0,%eax
0x0000000000000341 in main ()
(gdb) n1
0x0000000000000371 in main ()
0x0000000000000376 in main ()
0x0000000000000376 in main ()
0x0000000000000388 in main ()
0x0000000000000387 in main ()
0x000000000000038c in main ()
Introduce el pin: 228
0x0000000000000391 in main ()
(gdb) n1
0x0000000000000393 in main ()
(gdb) n1
0x0000000000000395 in main ()
0x00000000000003a8 in main ()
0x00000000000003ab in main ()
0x00000000000003ad in main ()
(gdb) p $6
$6 = 218
(gdb) |
```

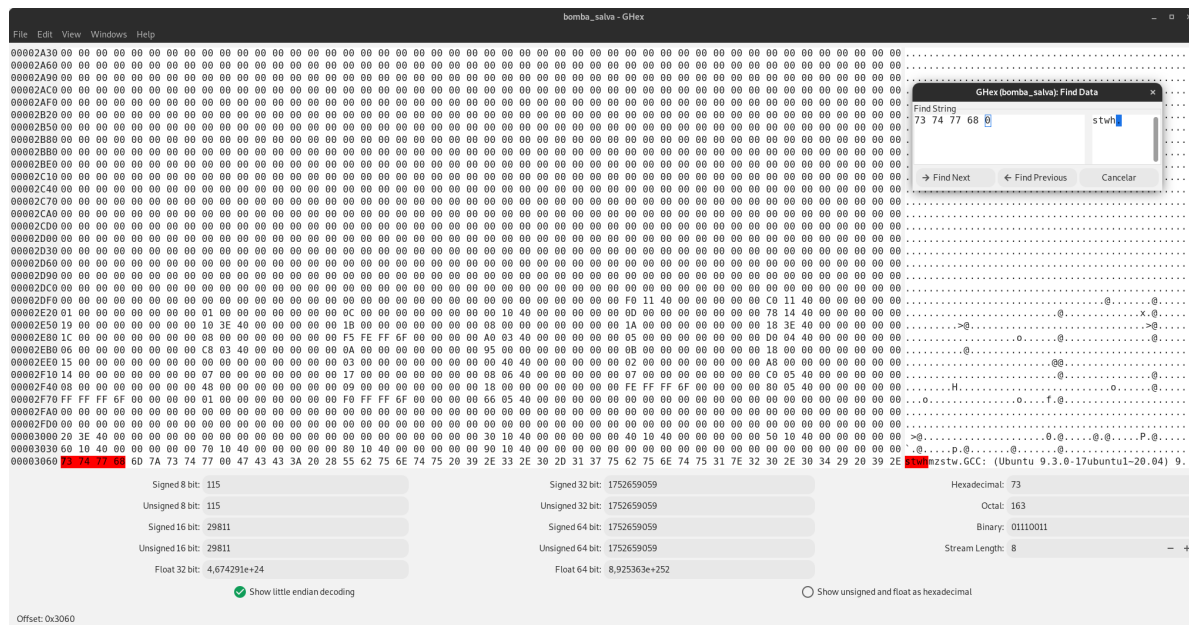
Aquí ya podemos finalizar la ejecución del programa puesto que ya hemos sacado los datos relevantes.

## Como cambiar las claves

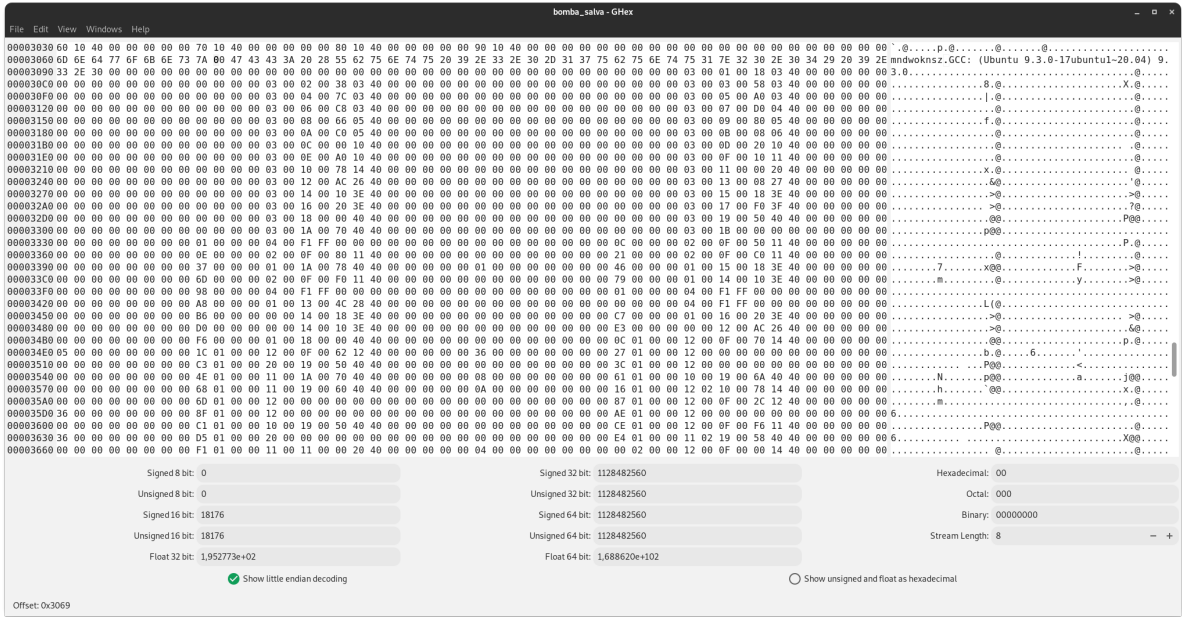
Para cambiar la clave y el pin usaremos un editor hexadecimal ( `ghex` ). Como conocemos los valores de la contraseña y del pin será relativamente sencillo modificarlos.

Para modificar la clave hay que tener en cuenta que al principio descubrimos que la contraseña original guardada en memoria y la que se comprobaba tenía un desplazamiento de 10 y que la contraseña que introducimos nosotros tiene uno de 9. Por lo tanto, tenemos que introducir una contraseña con un desplazamiento de -1 (9-10) para que al cifrarse con el desplazamiento de 10 tenga el mismo desplazamiento que el de la que insertamos por teclado.

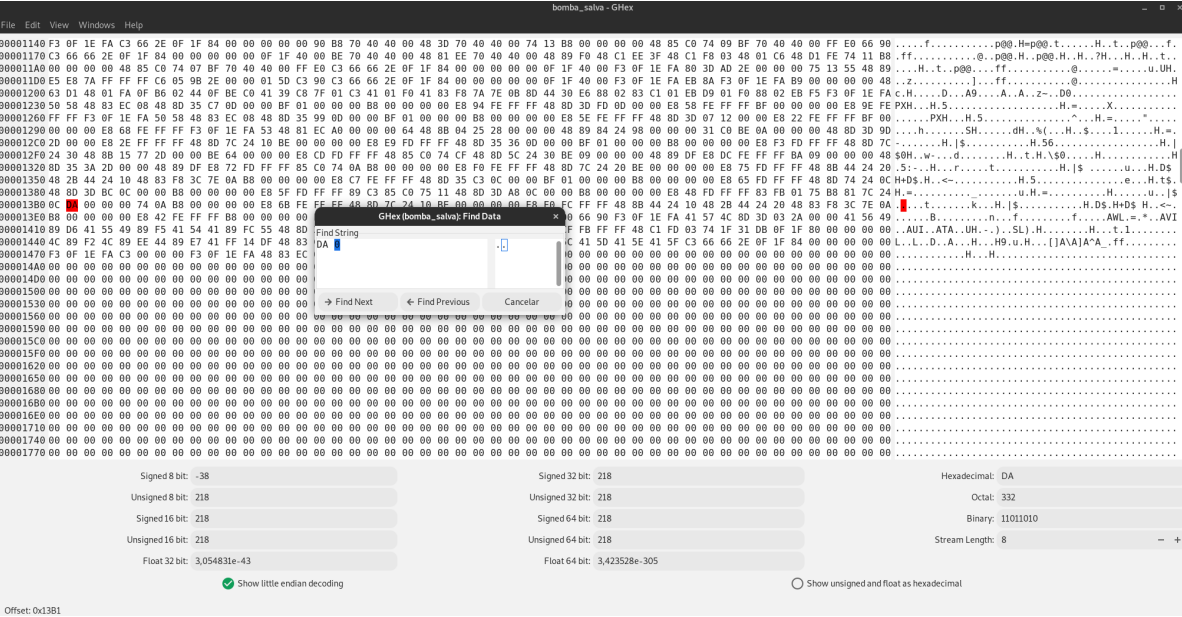
De esta manera buscamos el string y lo sustituimos en el editor.



En este ejemplo cambié la contraseña por `mndwoknszx` que es `noexplota` con un desplazamiento de -1.



Hacemos lo mismo con el valor del pin (vimos que era el valor inmediato `0xdb`). Lo buscamos y lo editamos (por ejemplo a `0xdb`, 19 en decimal).



Finalmente, guardamos el archivo, le damos permiso de ejecución y probamos su funcionamiento.

```
fish/home/salva/2Info/EC/Practicas/Practica4
salva@salva-desktop ~/2/E/P/Practica4 (master)> ./bomba_salva_parcheado
Introduce la contraseña: noexplota

Introduce el pin: 219
      _nnnn_
    dGGGGMMb ,.....
  @p~qp~~qMb | Has desactivado la bomba! |
  M|@||@) M| _;.....
  @,-----JM| -'
  JS^___/ qKL
    dZP      qKRb
    dZP      qKKb
    fZP      SMMb
    HZM      MMMM
    FqM      MMMM
  _| '      |dS''qML
  | '      | ' Zq
  _)'      )MMMMMP| '
  _'      ' _'

salva@salva-desktop ~/2/E/P/Practica4 (master)> |
```