Para resolver esta bomba, me quise centrar en dos cosas, vista la manera de resolver mi propia bomba. La primera era conseguir las claves de la bomba; y la segunda, la manera de decodificar estas para conseguir el pin y contraseña verdaderos. Además, sabiendo cómo se decodifican estos, podre alterar las contraseñas.

Una vez cargado el código ASM de la bomba, atenderemos a los siguientes fragmentos de código:

1. Obtención del password. A partir de este fragmento, podemos localizar que en la posición 0x601068 se encuentra el puntero a <password>. Si lo leemos, tenemos la contraseña. También, como se puede observar, hay una función denominada invertir, a la que accederemos más adelante. Eso puede significar que si metiera la contraseña tal y como estaba, explotaria.

```
0x4007f7 <main+91>      lea    0x20086a(%rip),%rdi        # 0x601068 <password>
0x4007fe <main+98>      callq  0x40075b <invertir>
0x400803 <main+103>     lea    0x30(%rsp),%rdi
0x400808 <main+108>     mov    $0x8,%edx
0x40080d <main+113>     lea    0x200854(%rip),%rsi        # 0x601068 <password>
```

```
(gdb) p (char*) 0x601068
$2 = 0x601068 <password> "acisum\n"
```

2. Obtención y decodificacion del pin. En este fragmento observamos que, antes de comparar el fragmento de código que he introducido para comparar, resta uno. Eso nos da a entender que el passcode codificado será el passcode original menos 1. Esto, por lo tanto, implica que habrá que sumar 1 al passcode que hay en 0x601060.

```
0x400892 <main+246>             sub    $0x1,%eax
0x400895 <main+249>             cmp    0x2007c5(%rip),%eax        # 0x601060
<passcode>
```

```
(gdb) p (int)passcode
$4 = 999
```

3. Decodificación del password. En este fragmento podemos ver la función invertir. Haciendo uso de ingeniería inversa podremos ver que , mientras que no se llegue al final, se intercambia la primera letra de la cadena por la última, y así sucesivamente.

```
0x40075b <invertir>     mov    $0x0,%eax
0x400760 <invertir+5>   movslq %eax,%rdx
0x400763 <invertir+8>   cmpb   $0xa,0x1(%rdi,%rdx,1)
0x400768 <invertir+13>  jne    0x400775 <invertir+26>
0x40076a <invertir+15>  mov    $0x0,%edx
0x40076f <invertir+20>  cmp    %edx,%eax
0x400771 <invertir+22>  jg     0x40077a <invertir+31>
0x400773 <invertir+24>  repz retq
0x400775 <invertir+26>  add    $0x1,%eax
0x400778 <invertir+29>  jmp    0x400760 <invertir+5>
0x40077a <invertir+31>  movslq %edx,%rsi
0x40077d <invertir+34>  add    %rdi,%rsi
```

```
0x400780 <invertir+37>  movzbl (%rsi),%r8d
0x400784 <invertir+41>  movslq %eax,%rcx
0x400787 <invertir+44>  add    %rdi,%rcx
0x40078a <invertir+47>  movzbl (%rcx),%r9d
0x40078e <invertir+51>  mov    %r9b,(%rsi)
0x400791 <invertir+54>  mov    %r8b,(%rcx)
0x400794 <invertir+57>  sub    $0x1,%eax
0x400797 <invertir+60>  add    $0x1,%edx
0x40079a <invertir+63>  jmp    0x40076f <invertir+20>
```

La clave “alterada” era *“acisum/n”*. Esta función lo convierte en *“musica/n”*, siendo esta la clave.

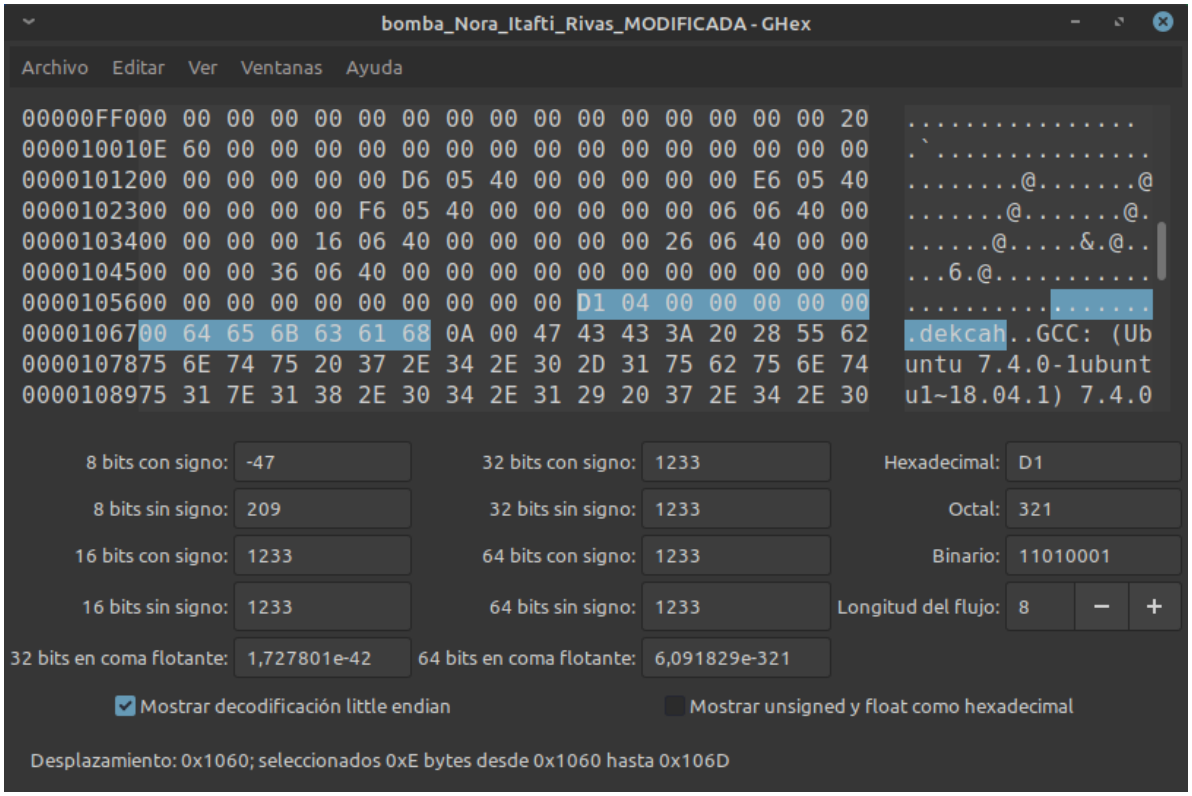
Dicho esto, las claves eran:

Contraseña : musica
PIN: 1000

La modificación de las claves se hizo con ghex. Debido a la codificación, había que introducir las claves “alteradas”.

En mi caso, introduje *“hacked/n”*, el cual se debería introducir en el programa invertido, es decir, cambiaría *“acisum/n”* por *“dekcah/n”*

Por otra parte, el pin alterado original sería 999, que se corresponde a 0x3E7, y debería buscar en little endian una cadena E7 03, la cual alteraré por el pin 1234, que se alteraría a 1233, o 0x4D1, en little endian, D1 04.



Tras las pruebas, al introducir “musica/n” o 1000 en el password/pin, explota, pero al poner “hacked/n” o 1234, se desactiva.

Las claves de la bomba alterada serían:

Contraseña mod: hacked
PIN mod : 1234