REPORTE DE EXAMEN

Inmediatamente descargado el archivo con “file” revisé el tipo de archivo y al ser un gzip lo descomprimí. De nuevo volví a revisar el archivo extraído esta vez llamado shell_mod2 y esta vez como resultado era el binario, pero el archivo ELF no se reconocía. Al intentar ejecutarlo tampoco funcionó.

```
lilium@debian:~/Documents/ExamenVulnes$ file SHELLow
SHELLow: gzip compressed data, last modified: Fri Mar 31 19:11:39 2017, from Unix
lilium@debian:~/Documents/ExamenVulnes$ tar -xzvf SHELLow
shell_mod2
lilium@debian:~/Documents/ExamenVulnes$ ls
shell_mod2  SHELLow
lilium@debian:~/Documents/ExamenVulnes$ file shell_mod2
shell_mod2: ELF, unknown class 113
```

Al analizarlo rápidamente con Strings se mostraban varias cadenas incluyendo una extra, más tarde supe que estaba contenida en el header de ELF y con Vim lo eliminé.

```
lilium@debian:~/Documents/ExamenVulnes$ strings shell_mod2
ELFquitaestoparaquefuncioneelprograma
/lib64/ld-linux-x86-64.so.2
libc.so.6
puts
__libc_start_main
__gmon_start__
GLIBC_2.2.5
UH-H
fffff.
Baia, baH
ia ... sH
i que haH
s llegadH
```

Una vez quitada esa cadena, al volver a hacer el file del binario se podía ver más información y de hecho ya se podía ejecutar.

```
lilium@debian:~/Documents/ExamenVulnes$ file shell_mod2
shell_mod2: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, inte
rpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=4ab82577a85ffa88
94e95109fb63bdd2f199903f, not stripped
```

Al ejecutarlo parecía mantenerse a la espera de algo y al poner una entrada no ocurría nada.

```
lilium@debian:~/Documents/ExamenVulnes$ ./shell_mod2
Baia, baia ... si que has llegado lejos
It's time to crackme Miss/Mr Reverse Enginner ;)
```

Al listar los puertos que se ecnontraban escuchando se mostraba el puerto 39321, abierto por el binario.

```
lilium@debian:~/Documents$ lsof -i -P | grep LISTEN
shell_mod 25520 lilium    3u  IPv4 530229      0t0  TCP *:39321 (LISTEN)
```

Al listar el contenido de la sección de .data aparecía una cadena escrita de forma extraña

“87654-32109-87654-321DRO-WSSAP”

```
lilium@debian:~/Documents/ExamenVulnes$ readelf --hex-dump=24 shell_mod2

Hex dump of section '.data':
  0x006009c0 00000000 00000000 00000000 00000000 ................
  0x006009d0 00000000 00000000 00000000 00000000 ................
  0x006009e0 00000000 00000000 00000000 00000000 ................
  0x006009f0 00000000 00000000 00000000 00000000 ................
  0x00600a00 38373635 342d3332 3130392d 38373635 87654-32109-8765
  0x00600a10 342d3332 3144524f 2d575353 41500000 4-321DRO-WSSAP..
  0x00600a20 5348454c 4c6f7720 77617320 68657265 SHELLow was here
  0x00600a30 203a5000 00000000 00000000 00000000  :P............
  0x00600a40 ba7c35ee cfdadcd9 7424f45e 31c9b13a .|5.....t$.^1..:
```

Esa cadena, junto con la cadena revertida fueron enviadas con netcat pero causó un error.

```
lilium@debian:~/Documents$ nc localhost 39321
87654-32109-87654-321DRO-WSSAP
lilium@debian:~/Documents$ nc localhost 39321
PASSW-ORD123-45678-90123-45678
lilium@debian:~/Documents$
Baia, baia ... si que has llegado lejos
It's time to crackme Miss/Mr Reverse Enginner ;)
Segmentation fault
lilium@debian:~/Documents/ExamenVulnes$ ./shell_mod2
Baia, baia ... si que has llegado lejos
It's time to crackme Miss/Mr Reverse Enginner ;)
Segmentation fault
```

Al seguir analizando el binario, con radare2 se mostraban las siguientes funciones

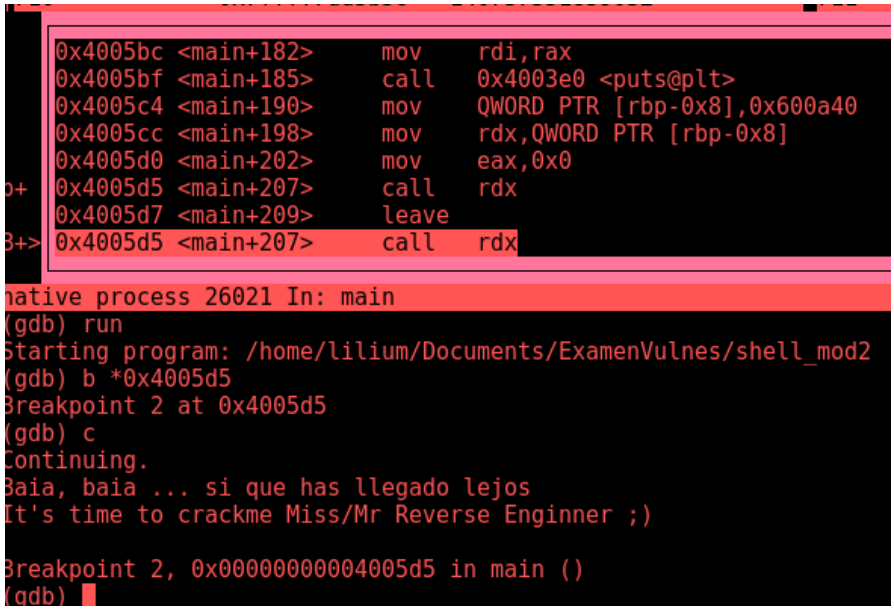
```
-[ functions ]---------------
(a) add     (x)xrefs     (q)quit
(r) rename  (c)calls     (g)go
(d) delete  (v)variables (?)help
 * 0x00400410   41 entry0
   0x00400506  211 sym.main
   0x004003e0   32 sym.imp.puts
   0x00400440   41 sym.deregister_tm_clones
   0x00400480   53 sym.register_tm_clones
   0x004004c0   28 sym.__do_global_dtors_aux
   0x004004e0   35 sym.frame_dummy
   0x00400650    2 sym.__libc_csu_fini
   0x00400654    9 sym._fini
   0x004005e0  101 sym.__libc_csu_init
   0x004003a8   26 sym._init
   0x00400400   48 loc.imp.__gmon_start__
   0x004003f0   48 sym.imp.__libc_start_main
```

De igual manera se mostraba la estructura del main de la siguiente manera

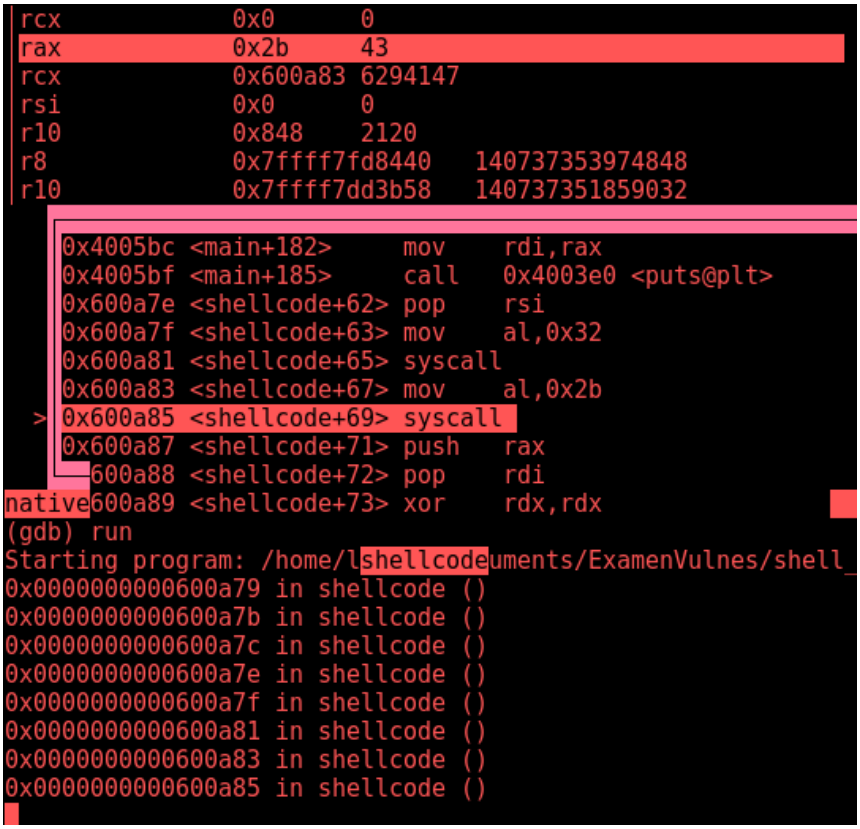
En este se puede ver que manda a llamar 3 funciones en particular. 2 puts y con call rdx, lo cual implica que manda a llamar lo que se encuentra dentro de rdx en esa parte de la ejecución.

Los puts solo son para mostrar las líneas que aparecen en cuanto se ejecuta el programa, por lo cual, la llamada importante es la que hace a rdx.

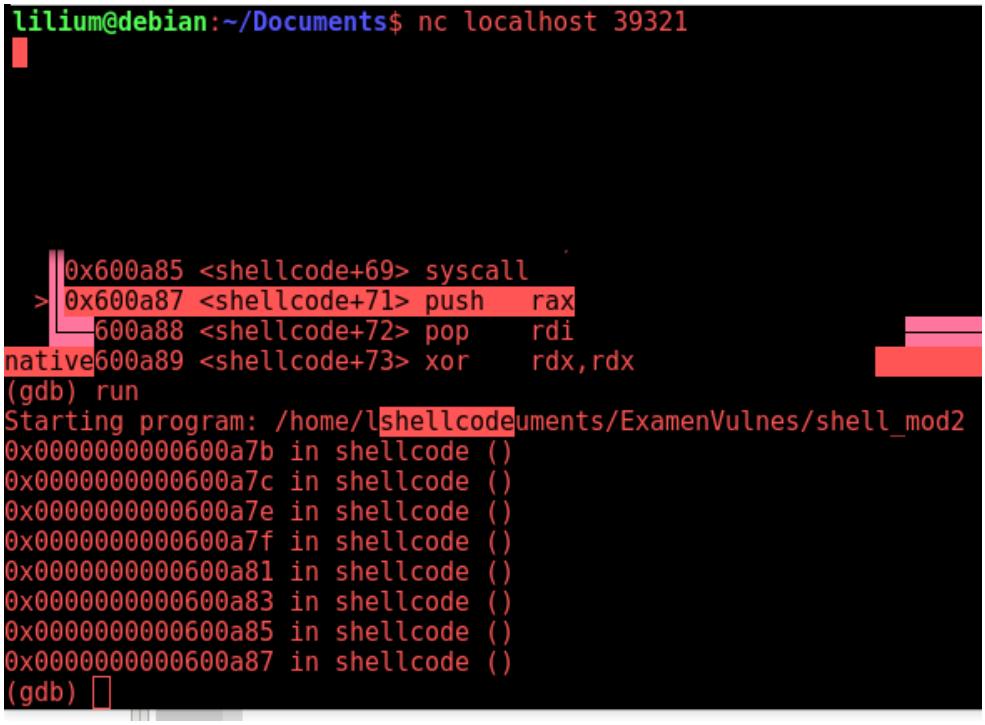
Después con gdb puse un breake en main y en la llamada a rdx para poder ver qué era lo que realizaba.



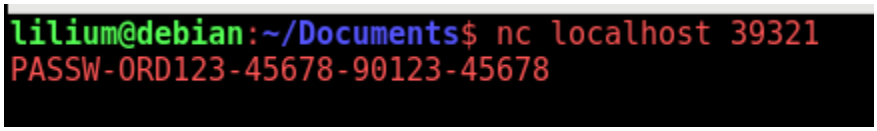
Una vez adentro, aparecían varias llamadas al sistema, pero en particular hubo una que dejaba a gdb sin poder ejecutarse hasta que se realizara una conexión a través de netcat.



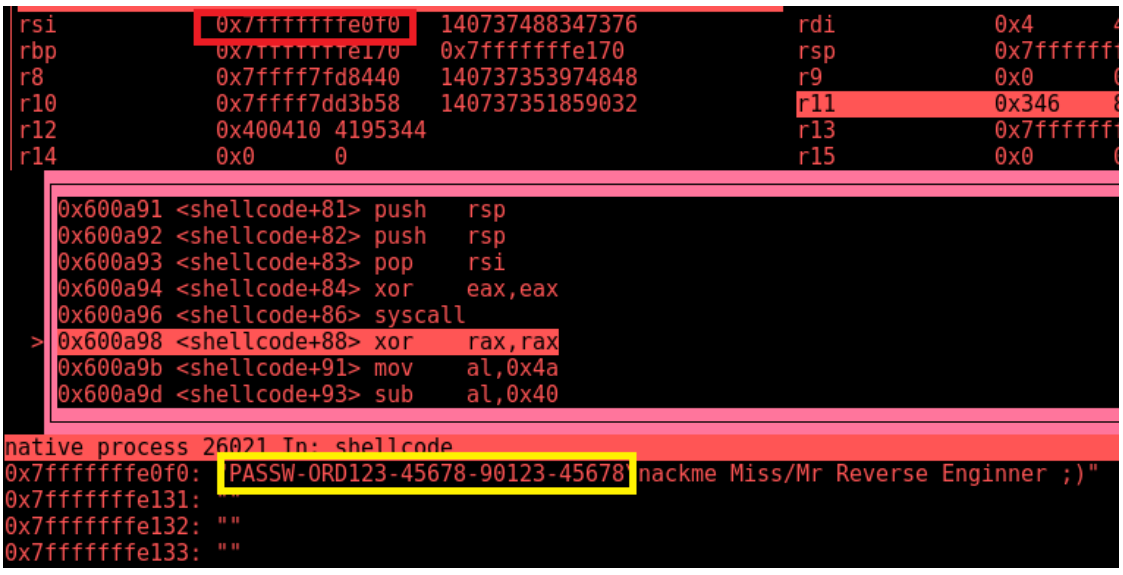
Una vez realizada la conexión ya permitía seguir ejecutando.



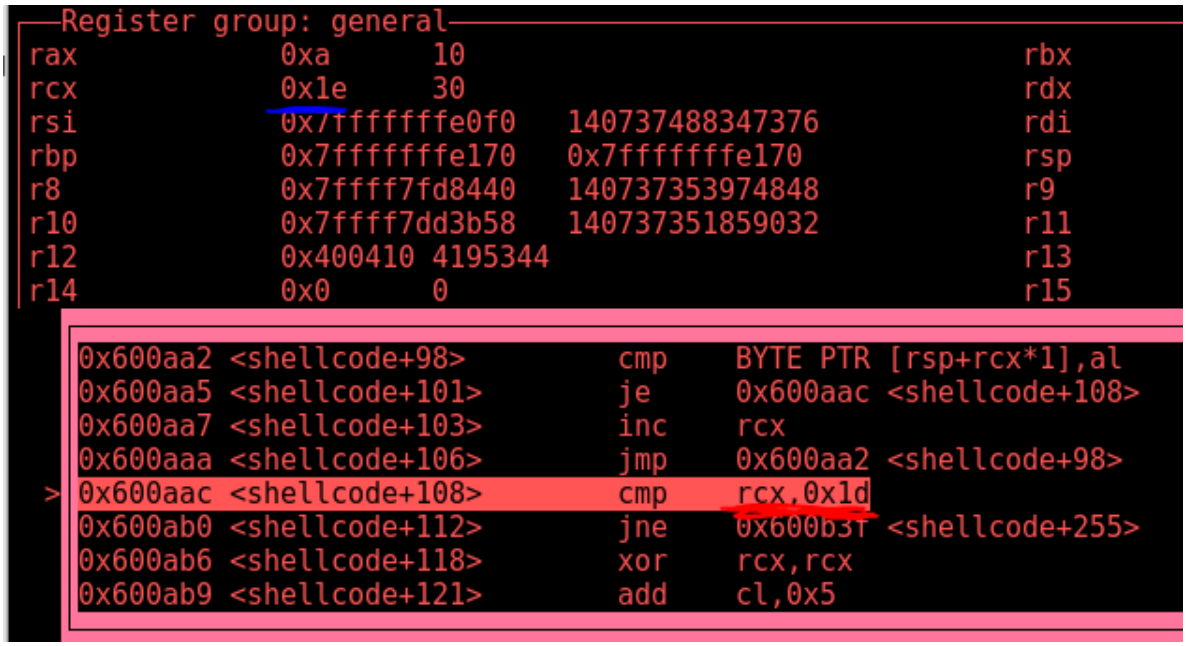
De nuevo en un punto esperaba la entrada de algo y en ese momento envié la cadena que me aparecía para ver cómo la estaba analizando.



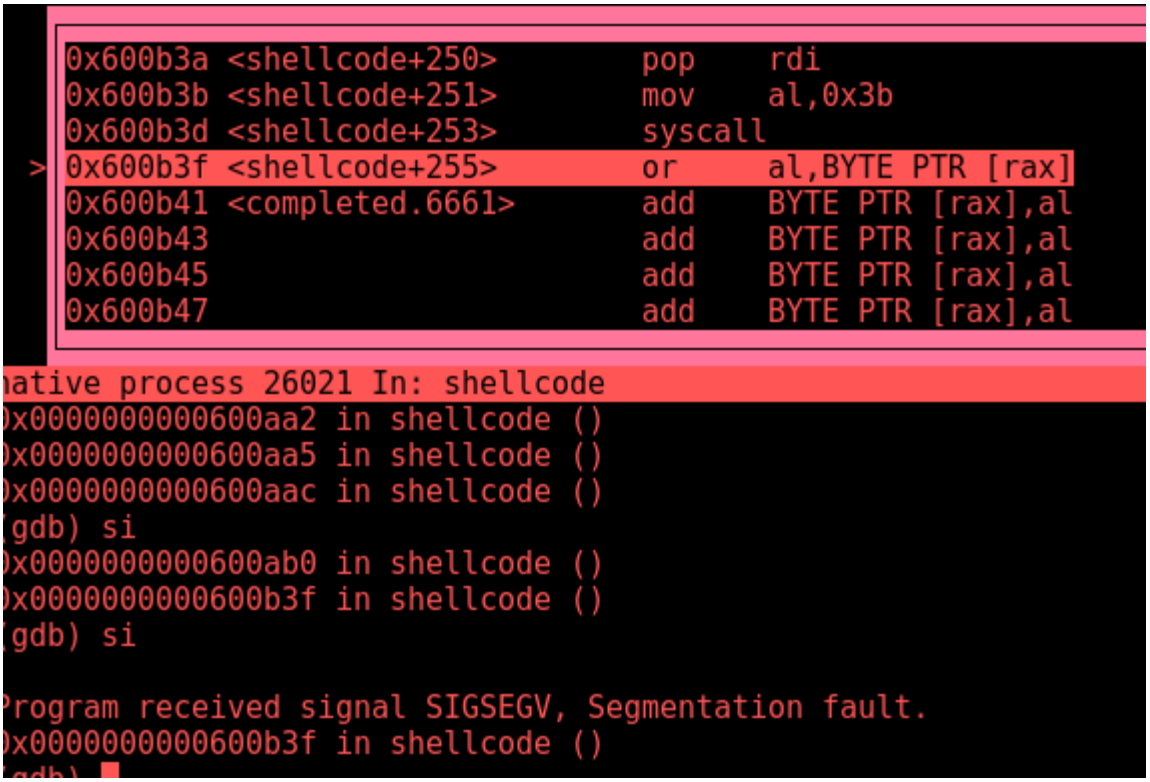
Una vez dentro vi que los registros cambiaron y había uno que apuntaba directamente a donde se encontraba la cadena que recién había enviado.



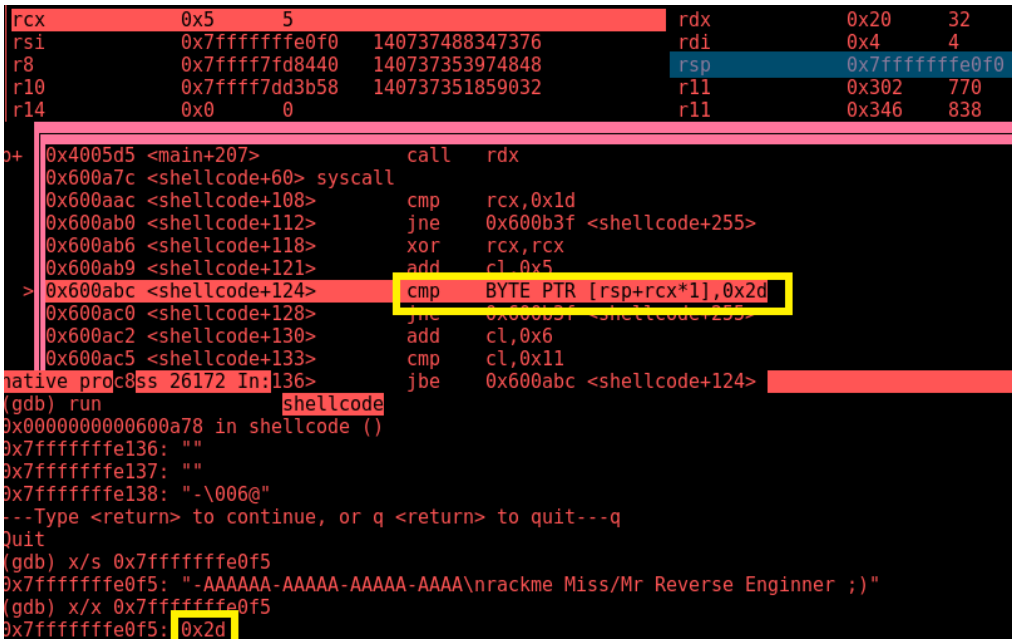
Siguiendo en la ejecución me aparecía un loop que se encontraba contando la cantidad de caracteres que habían sido recibidos, una vez terminado, hacía una comparación de la cantidad de caracteres enviados (En este caso 30) con 29.



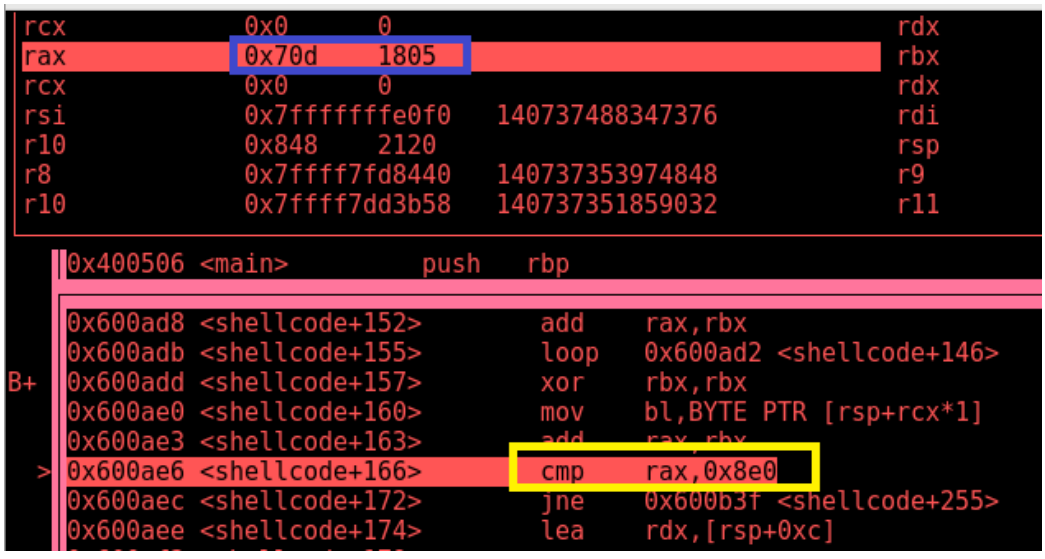
Al no cumplir con esta condición enviaba a shellcode+255 y terminaba la ejecución con “segmentation default”, lo cual indicaba que solo permitía 29 caracteres de entrada.



De nuevo envié una cadena esta vez con 29 caracteres y me permitió continuar con la ejecución, esta vez realizando una comparación del tipo de carácter, donde checaba que hubieran guiones en posiciones específicas de la cadena (cada 5 caracteres)



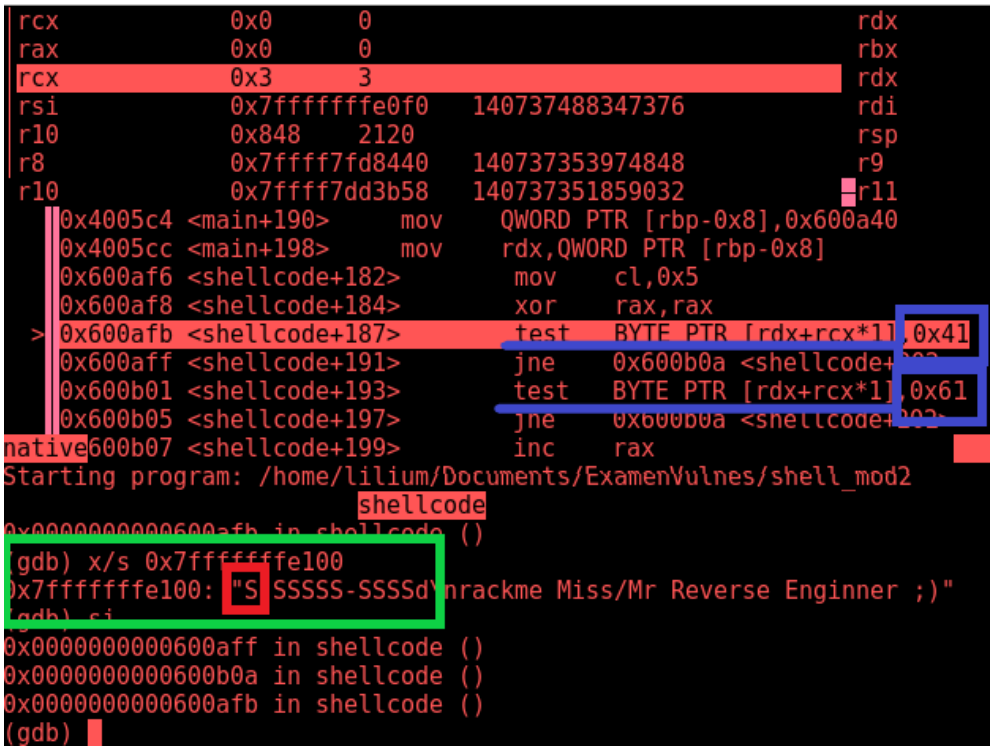
Una vez pasada esa parte, de nuevo me aparecía una comparación, pero esta vez sumaba el valor decimal del asccii de cada carácter que se enviaba y se podía notar que debía ser un total de 8e0, o en decimal 2272.



En la siguiente ejecución proporcioné una cadena que cumplía las condiciones anteriores y además sumaba 2272 en ascii, que era la siguiente

SSSSS-SSSSS-SSSSS-SSSSd

A partir de esa, entró a la última condición que verificaba que se encontrara una “A” en al menos las ultimas 2 secciones que eran separadas por los guiones de la cadena.



En particular, aquí tuve un problema pues a veces permitía seguir a la cadena y a veces no, le pasara A o a, o no le pasara nada.

Tomando en cuenta todas las condiciones anteriores armé un serial que ocupara mi nombre:

diana-tadeo-AAAAA-AAAAA-AUUUX

Al probarlo me devolvió una shell.