Enviamelo

El participante recibirá un binario con el que tiene que interactuar para sacar el flag. El binario necesitará recibir 2 argumentos por la línea de comandos. Para ello, el participante deberá realizar ingeniería inversa sobre el binario y descubrir cuáles son los condicionantes que deben cumplir los argumentos para que se muestre el flag por pantalla. A continuación, describimos dichas condiciones:

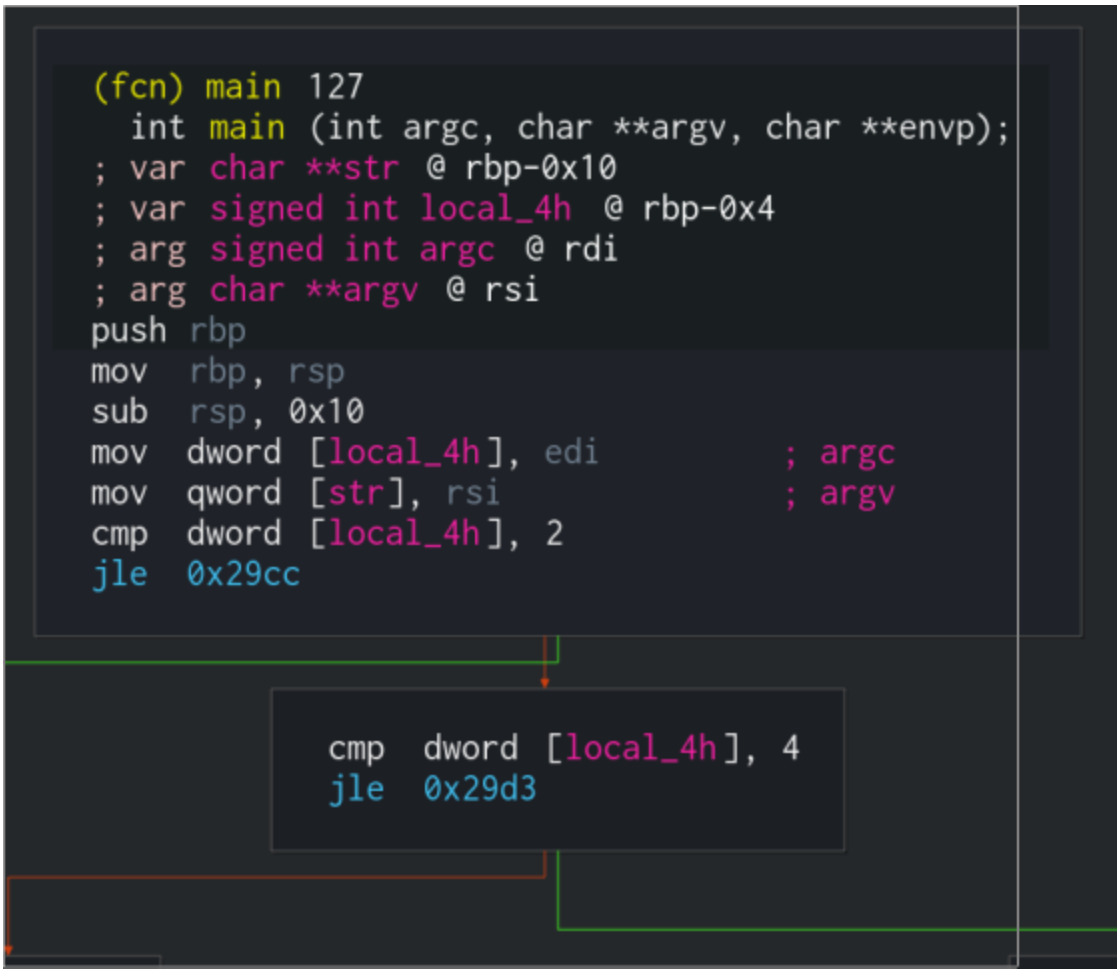
El primer argumento comprueba el número de archivos y carpetas que contiene el directorio donde se está ejecutando el programa, por lo que dependiendo de donde se ejecute su valor será distinto.

En el segundo argumento se deberá introducir una ruta de un fichero en concreto. Al comprobar los condicionantes, el participante verá que se le solicita escribir un PATH en concreto y además comprueba que el fichero exista. Para que el argumento se dé por válido, el participante deberá crear ese fichero en la ruta específica donde se le indica.

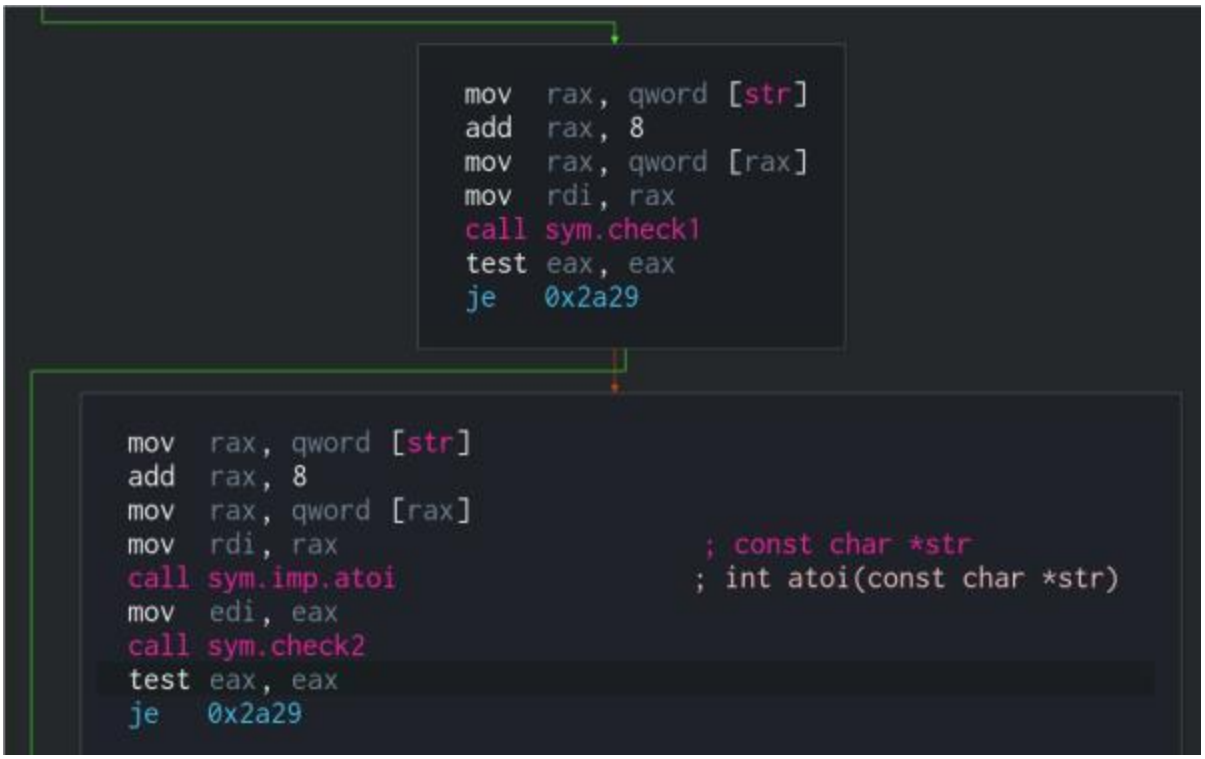
Por último, una vez el participante haya introducido los argumentos y cumplido los condicionantes, se le mostrará el flag por pantalla. Comprobamos que es un ejecutable ELF mediante el comando file

```
└─$ file hazlo
hazlo: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86
-64.so.2, BuildID[sha1]=80b80cc6088c03e5f18f2b7432c7a9091d14f1a5, for GNU/Linux 3.2.0, with debug_info, not strippe
d
```

Abrimos el ejecutable con la herramienta “cutter”, una de las posibles elecciones que tenemos para realizar la ingeniería inversa. Al analizar la función main comprobamos que requiere una serie de parámetros, concretamente 2: si argc >= 4 o argc <= 2 salir de la función, es decir, se requiere que argc sea igual a 3: el nombre del ejecutable más 2 parámetros de entrada.

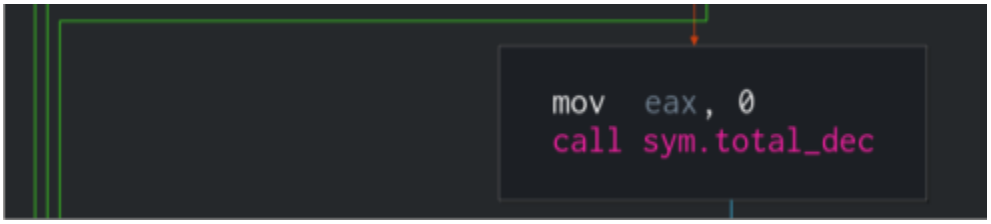


Al seguir mirando en la función main, se puede ver que se ejecuta la función check1 sobre el primer parámetro y en función del resultado se llama a la función atoi sobre dicho parámetro, lo que convierte una cadena en entero. De esto deducimos que la función check1 comprueba si el primer parámetro introducido por el usuario es un entero. Después se llama a la función check2 con el resultado de convertir la cadena en entero.

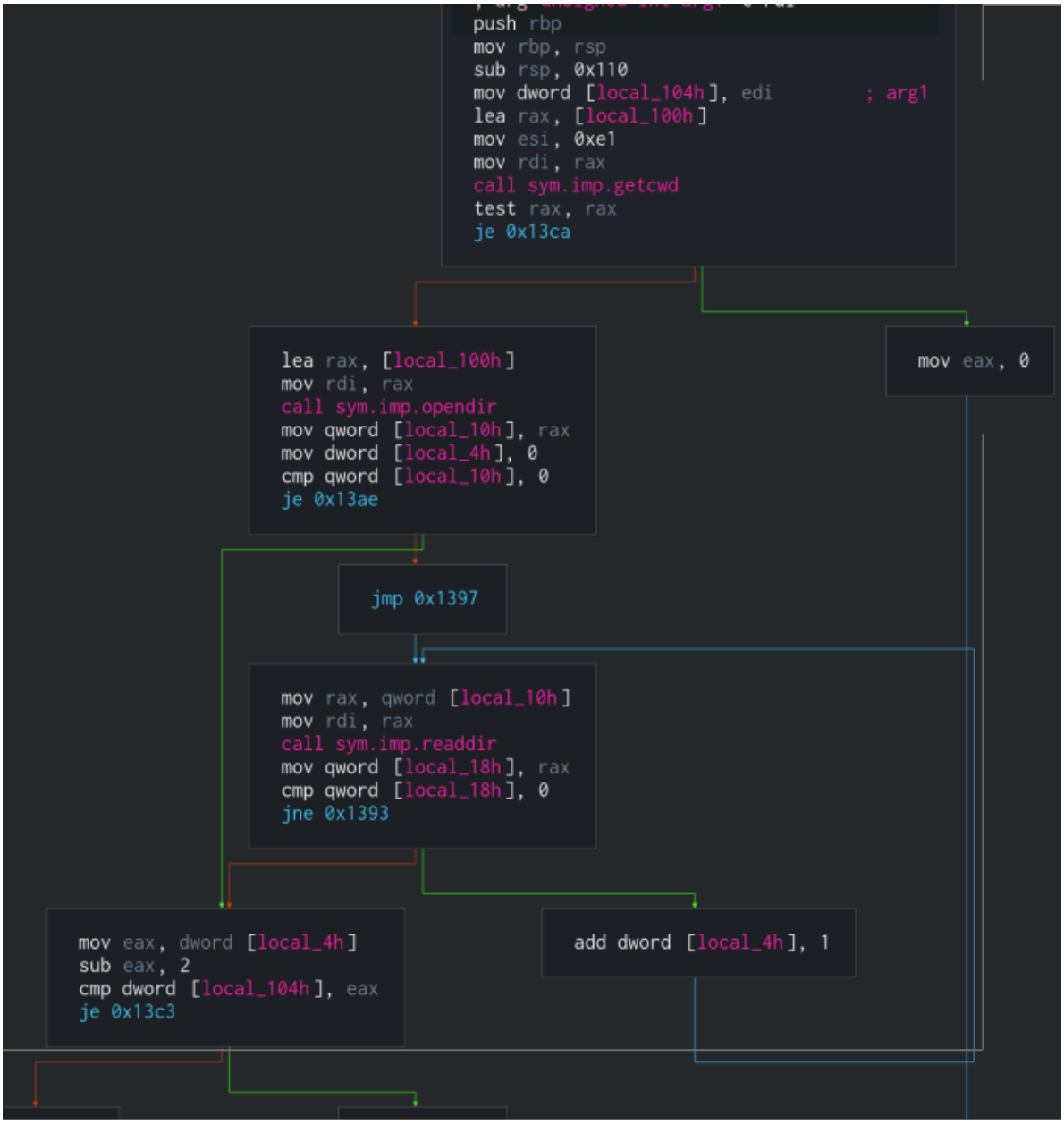


A continuación se llama a la función check3 a la que le pasa el valor del segundo parámetro introducido por el usuario.

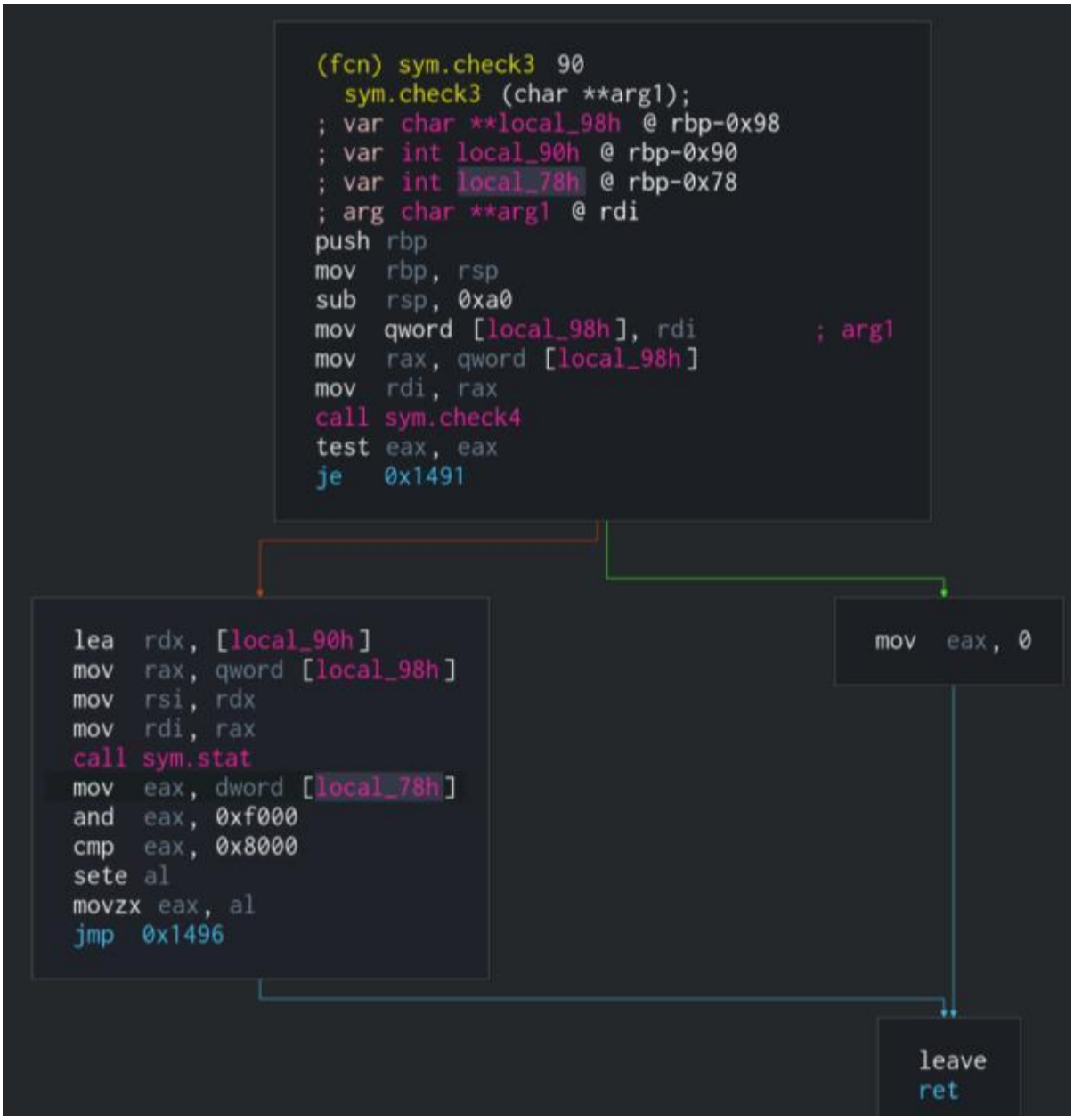
Por último, según los resultados obtenidos de las funciones anteriores, llama a la función total_dec.



A continuación, comprobamos qué es lo que hace la función check2 con el parámetro numérico introducido por el usuario. Se comprueba que lo primero que hace es llamar a la función getcwd para obtener el path del ejecutable, luego llama a la función opendir del path obtenido y a continuación, en un bucle, va accediendo a cada uno de los elementos del directorio (ficheros y directorios que contiene el directorio del ejecutable) mediante la función readdir, y por cada uno suma 1 a la variable local_4h. Al acabar de contar los elementos del directorio realiza una comparación entre local_4h (contador) y local_104h, variable en la cual se ha cargado al inicio de la función el valor del parámetro de la función que coincide con el entero del primer parámetro insertado por el usuario. Es decir, que el primer parámetro introducido por el usuario debe coincidir con el número de elementos (ficheros y directorios) del directorio donde se encuentra el ejecutable.

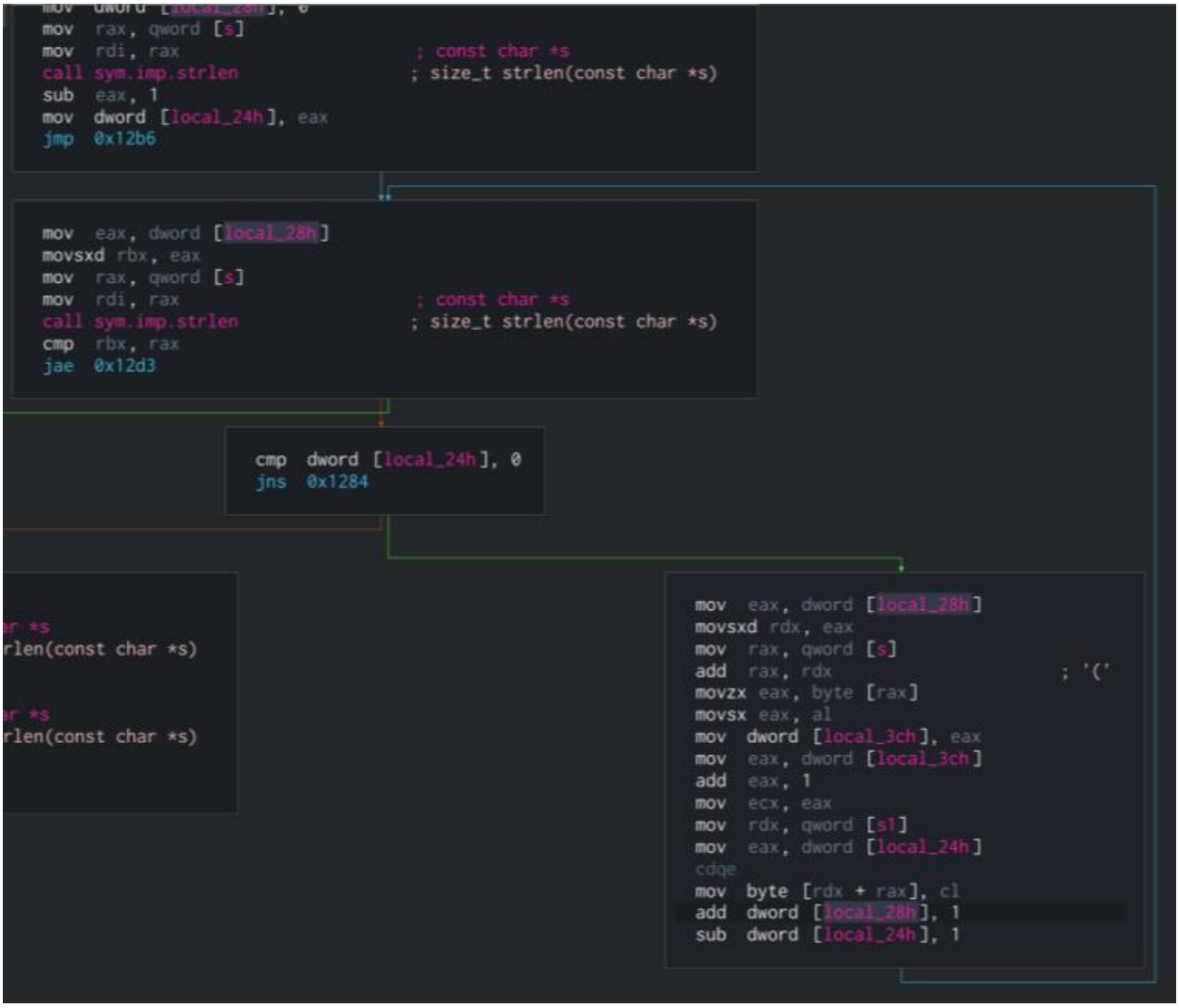


A continuación, analizamos la función check3. Lo primero que hace esta función es llamar a la función check4 para luego, en función del resultado obtenido, llamar a la función stat a la cual le pasa por parámetro el parámetro de la función check3 y la variable local_90 en la que inicializa las propiedades del PATH pasado por parámetro. De esto deducimos que el segundo parámetro proporcionado por el usuario debe ser un PATH a un fichero o un directorio. A continuación, comparará el resultado obtenido con el valor 0x8000. Si buscamos en internet se corresponde con #define S_IFREG 0x8000 /* regular file */ por lo que deducimos que el segundo parámetro introducido por el usuario debe ser el PATH de un fichero regular existente (Estructura stat).

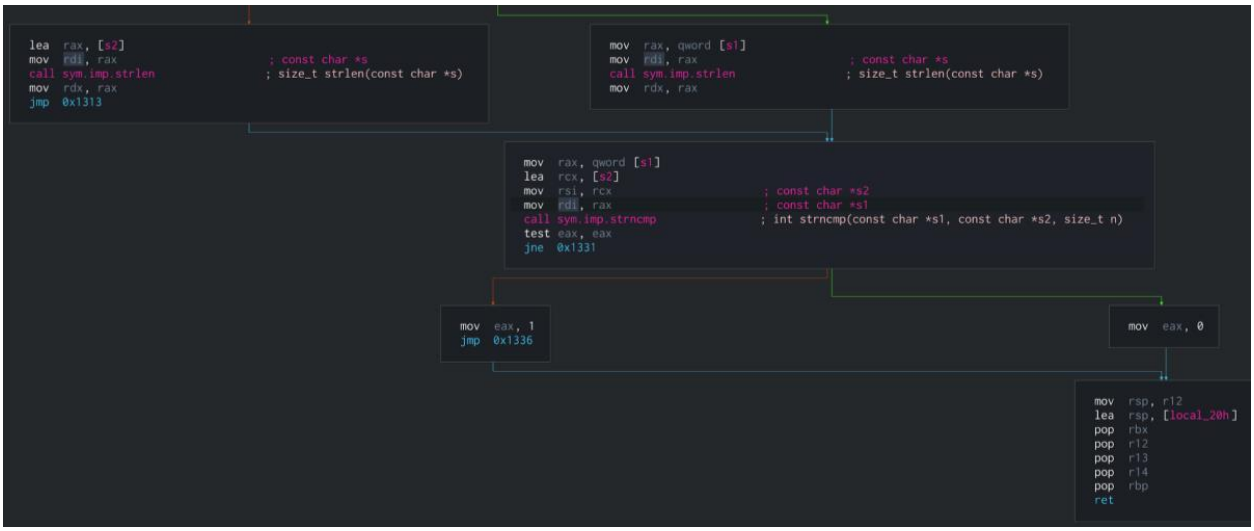


A continuación, comprobamos qué es lo que hace la función check4 que se llama nada más entrar en la función check3. Lo primero que comprobamos es que el valor pasado por parámetro se guarda en la variable s. Más adelante se inicializan las variables local_28h a 0 y local_24h al tamaño del parámetro pasado a la función - 1. Estas variables se emplean como condiciones del bucle que sigue, de tal manera que la variable local_28h va aumentando en una unidad y la variable local_24h va disminuyendo en una unidad en cada vuelta, y la condición de parada es que la primera variable llegue a ser mayor o igual que el tamaño del PATH introducido y que la segunda variable llegue a ser menor o igual a 0. Dentro del bucle, cada elemento de s (PATH introducido) apuntado por el índice local_28h es asignado a la variable local_3ch, a la cual se le suma un 1 y cuyo resultado es

asignado a su vez a la variable s1 apuntada por el índice local_24h, que se recorre en sentido contrario al primero. Es decir, realiza el volteo del parámetro 2 del usuario (PATH de fichero) y a cada carácter le suma 1 convirtiéndolo en el carácter siguiente en la tabla ASCII.



Siguiendo con el análisis de la función check4 comprobamos que se realiza una comparación mediante la función strncmp entre la cadena obtenida en la primera parte (s1) y la cadena s2, que apunta a la constante del principio de la función mbosfuf0tj0ufsdft0zn0. Si son iguales se devuelve 0 mientras que si no lo son se devuelve 1. Para calcular el valor con el cual se compara el parámetro 2 insertado por el usuario, realizamos el procedimiento inverso del bucle sobre la cadena con la cual se compara el reverse del PATH.



Así pues, vamos restando una posición en la tabla ASCII de cada uno de los caracteres de la constante mbosfuf0tj0ufsdft0zn0 y obtenemos la cadena lanrete/si/terces/ym/ y a continuación, le damos la vuelta obteniendo la cadena /my/secret/is/eternal.

```
mov  rax, rsp
mov  r12, rax
movabs rax, 0x6074706d2f757975      ; 'uyu/mpt`'
movabs rdx, 0x7368627430647567      ; 'gud0tbhs'
mov  qword [s2], rax
mov  qword [local_58h], rdx
movabs rax, 0x7075304566746462      ; 'bdtfE0up'
mov  qword [local_50h], rax
mov  word [local_48h], 0x7370       ; 'ps'
mov  byte [local_46h], 0
mov  rax, qword [s]
```

Por último, comprobamos qué es lo que pasa si ejecutamos el binario y le pasamos los parámetros correctos. Primero comprobamos el número de entradas (ficheros y directorios) del directorio del binario mediante el comando ls, y a continuación nos aseguramos de que existe el fichero /my/secret/is/eternal en el directorio correcto y si no es así lo creamos mediante el comando mkdir. Comprobado esto lanzamos el ejecutable y obtenemos el flag.

g00d_j0b_reversers