Memoria sobre la primera fase de HackingForce

Grupo 4 – PL4, FCR GIITIN01 (20-21)

UO283319 – Juan Francisco Mier Montoto
UO282574 - Miguel del Riego Lázaro
UO281892 - Ignacio Valdés Azorín

Índice

- Fase 1.1 – Ceremonia de iniciación
  - Explicación del código
  - Ejemplos de entradas válidas e inválidas
- Fase 1.2 – Supervivencia en código máquina
  - Introducción general
  - Pregunta 1
  - Pregunta 2
  - Pregunta 3
  - Pregunta 4
- Reparto de trabajo

Fase 1.1 – Ceremonia de iniciación

Explicación del código

El código está compuesto de tres partes fundamentales:

- El receptor de errores, que recibe la cadena de error correspondiente después de que se haya incumplido una de las condiciones, la muestra por pantalla y sale del programa con ERRORLEVEL establecido a 1.
- Los métodos principales del programa, que desarrollan las tareas especificadas, incluyendo las funciones de Assembly.
- El main, que se encarga de dirigir el programa y mostrar el mensaje de confirmación en caso de que no se salga antes del problema.

Ejemplos de entradas válidas e inválidas

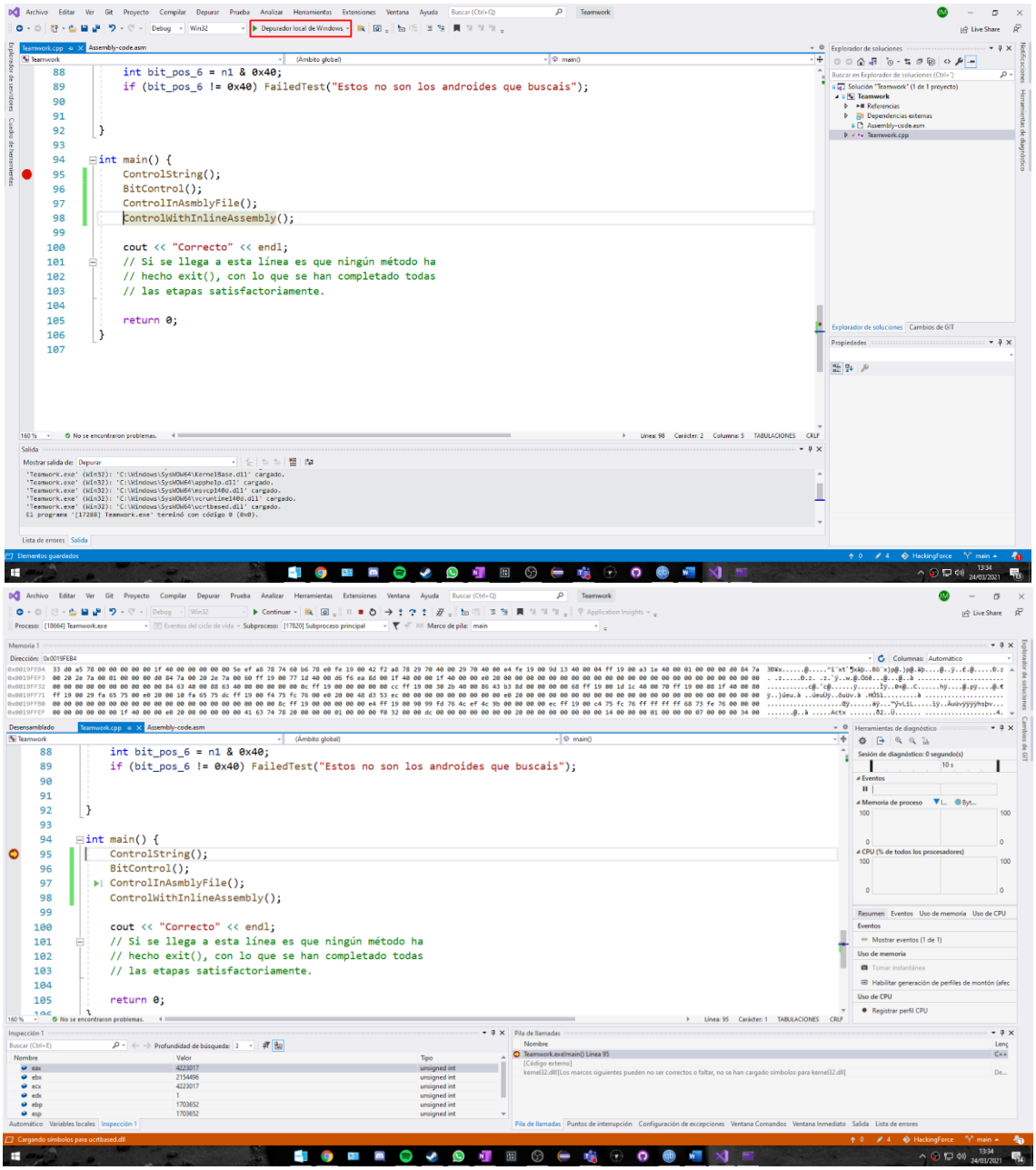
Para cada método, hay una combinación infinita de entradas válidas e inválidas. Estos son algunos ejemplos de ambos casos:

- **ControlString():**
  - En el caso de la primera cadena, solo es válida la entrada "otiPqhXcD". Cualquier otra entrada será inválida y devolverá la cadena "Sayonara, baby" al receptor de errores.
  - En el caso de la segunda cadena, para que sea válida debe tener una longitud de al menos 6 y que el carácter de índice 3 sea igual al de índice 5, como por ejemplo "BBBaBaBB" o "f8ds0sa2d02". Cualquier otra entrada que no cumpla estas condiciones será inválida, como por ejemplo "abcd" o "123456789".
- **BitControl():**
  - En la primera condición, los bits entre los índices 2 y 4 (ambos incluidos) del segundo número introducido deben de ser iguales a la representación en binario de 1, es decir, 001. Algunos ejemplos de entradas válidas del segundo número son "740" o "4".
  - Para la segunda etapa, se aplican máscaras a ambos números y se combinan los resultados, de manera que el nuevo número tiene los 23 bits de mayor peso del primer número y los 9 bits restantes de menor peso del segundo. Además, el bit 6 del primer número debe de ser 1. Utilizando respectivamente los ejemplos de entradas válidas anteriores, para 740 una entrada válida sería "22080" y para 4 una entrada válida sería "3648".
- **ControlInAsmblyFile():**
  - Para el primer número, el valor de los 8 bits de menor peso interpretados en binario debe de ser 238, como por ejemplo 240 o 61695.
  - Para los otros dos números, sus bits de la posición 8 deben coincidir, como por ejemplo 256 y 16777472 o 768 y 260.
- **ControlWithInlineAssembly():**
  - Los 16 bits de mayor peso del número introducido deben de ser iguales que los 16 bits restantes, como por ejemplo el 0 o el 143198345.

Fase 1.2 – Salvando al mundo

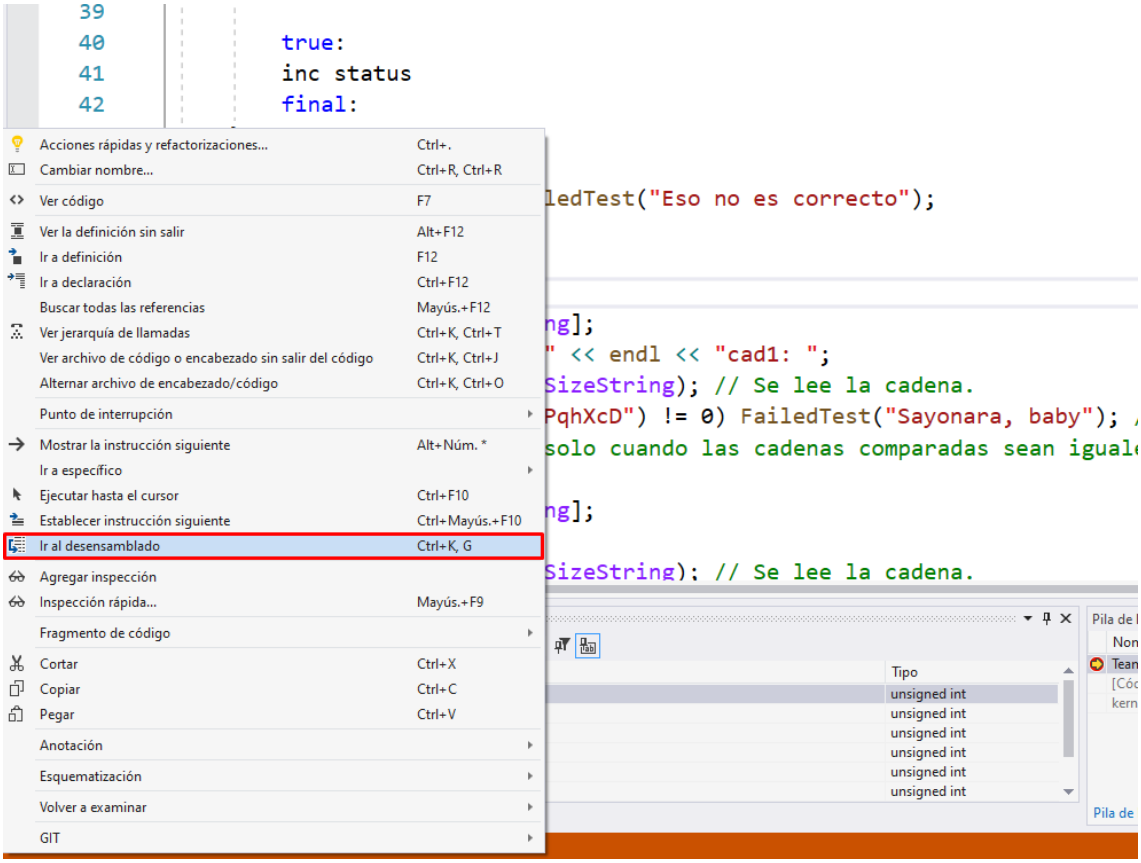
Introducción general

En este apartado se responderán a las preguntas propuestas en el enunciado del trabajo, incluyendo capturas de pantalla de los resultados. Las respuestas se consiguen mediante el depurador de Visual Studio en combinación con otras herramientas como la Memoria en ventana y el modo desensamblado. Para entrar en el modo depuración, primero se inserta un punto de interrupción y luego se extrae la información necesaria en el modo “Debug”.

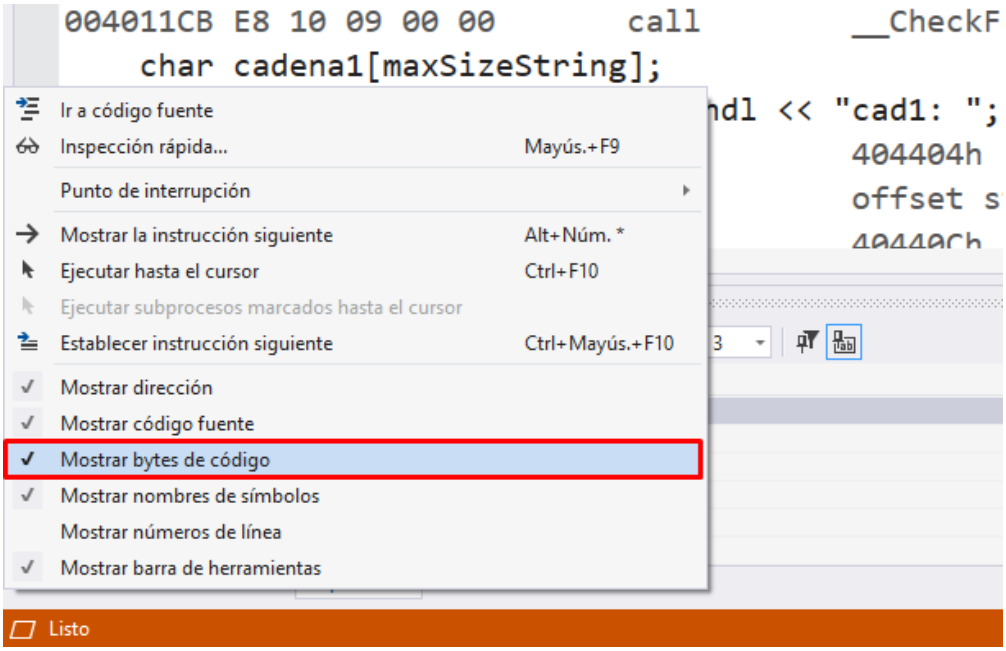


Pregunta 1.

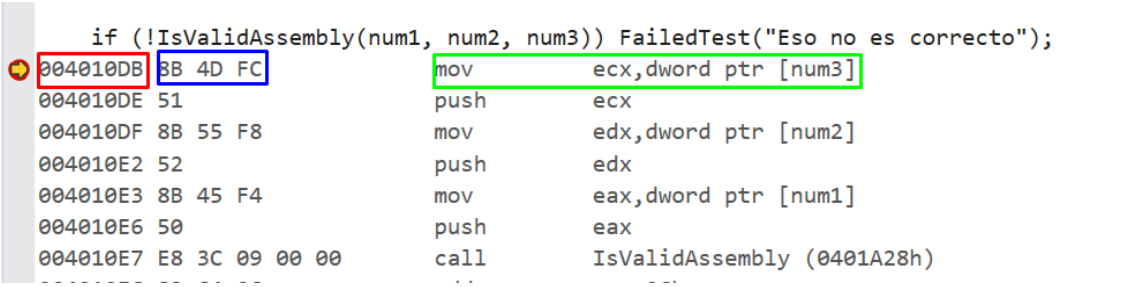
Para acceder al desensamblado, se introduce un punto de interrupción y se hace clic derecho sobre la instrucción que queremos observar en desensamblado. En el menú desplegable, se selecciona “Ir al desensamblado”:



Una vez en el desensamblado del código, para poder ver el código máquina debemos activar la siguiente opción, accesible igual que la anterior mediante el clic derecho:

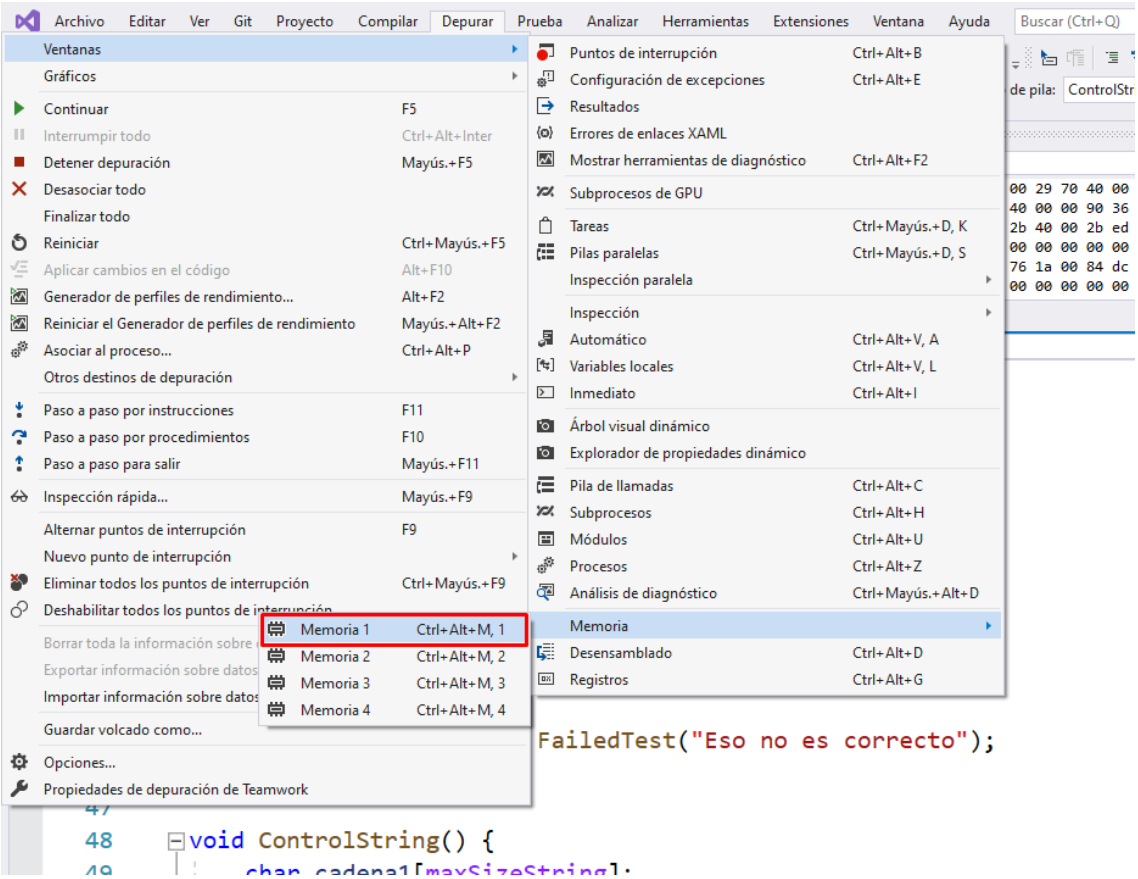


La primera dirección de memoria del paso de parámetros a la función en Assembly “IsValidAssembly(int, int, int)” es “004010DB”, resaltado en la imagen en rojo. El mnemónico de esta instrucción es “mov ecx, […] num3”, la parte resaltada en verde. Por último, en azul, se encuentra el código máquina en hexadecimal. Los pasos de parámetros continúan hasta la dirección “004010E7”, donde se llama al procedimiento.

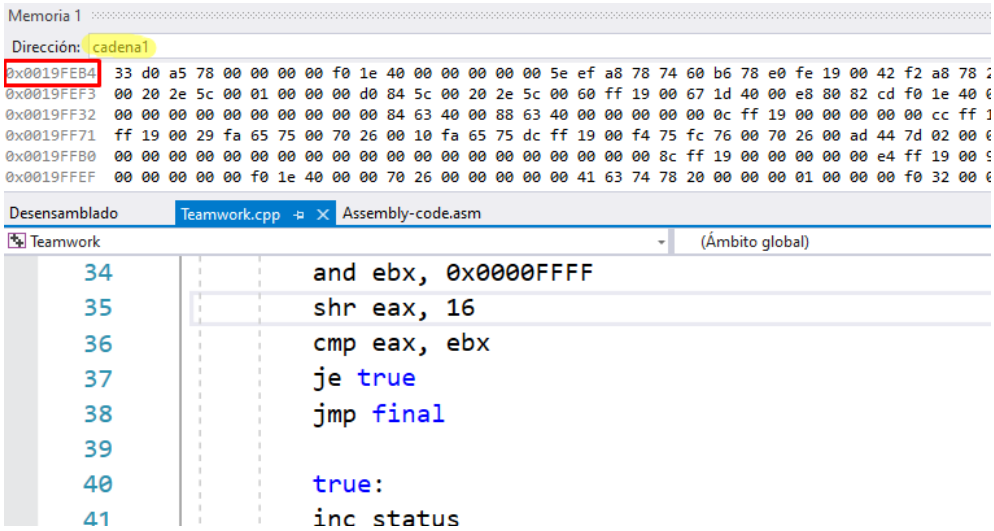


Pregunta 2.

Para obtener la dirección de memoria en la que se encuentra la primera cadena de “ControlString()”, tenemos que acceder a la ventana de “Memoria 1” mientras se ejecuta la depuración:

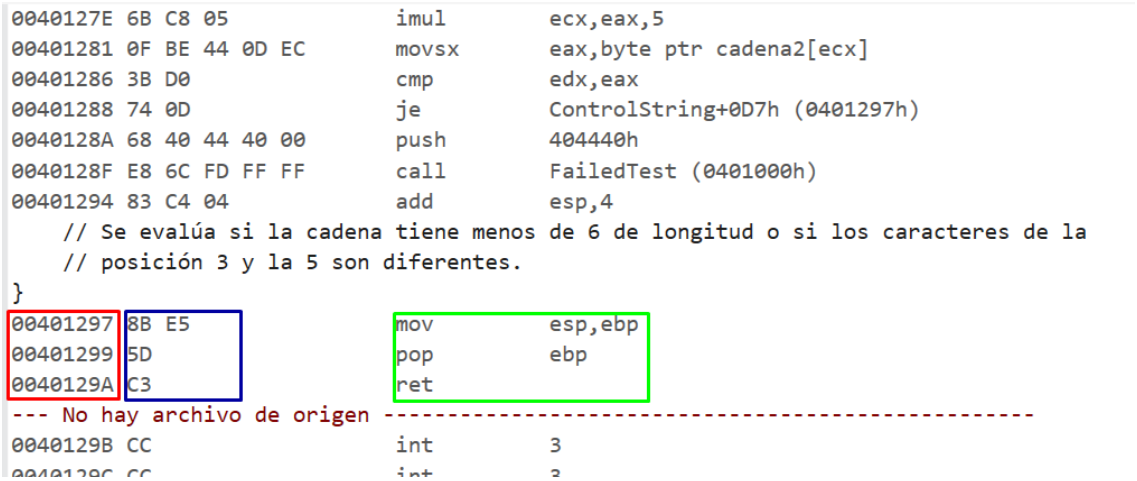


Ahora, solo hay que introducir el nombre de la cadena, en este caso “cadena1”. Así se puede observar que dicha dirección es “0x0019FEB4”.



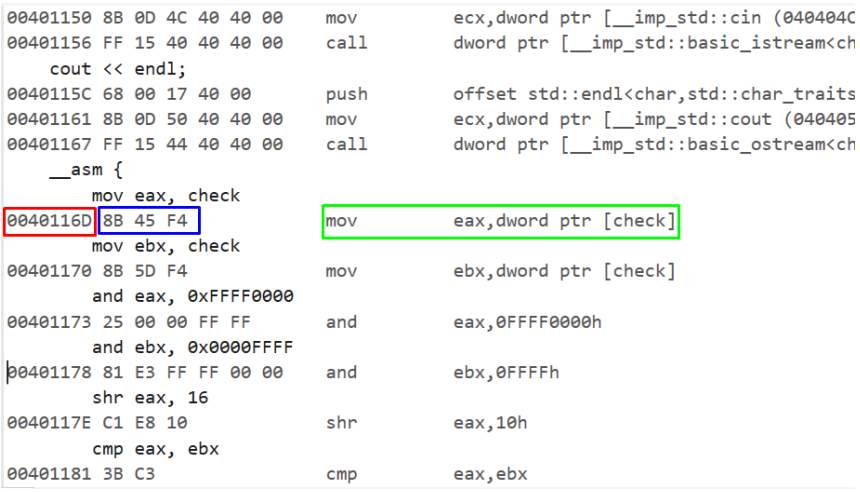
Pregunta 3.

Para obtener la dirección en memoria, el código fuente y el mnemónico del epílogo, se sigue el mismo procedimiento que en la primera pregunta. En el caso de la primera instrucción, su dirección en memoria será “00401297”, su código fuente en hexadecimal es “8BE5” y el mnemónico correspondiente es “mov esp, ebp”.



Pregunta 4.

Igual que en la primera pregunta, al entrar en el modo desensamblado podemos comprobar la dirección de memoria de la primera instrucción, “0040116D”, su código máquina en hexadecimal, “8B45F4”, y por último su mnemónico, “mov eax, […] check”.



Reparto del trabajo

El trabajo se ha repartido de esta manera:

- Miguel del Riego: BitControl()
- Juan Mier: ControlString(), ControlInAsmblyFile(), ControlWithInlineAssembly(), FailedTest() y main(), memoria.
- Ignacio Valdés: no ha participado.