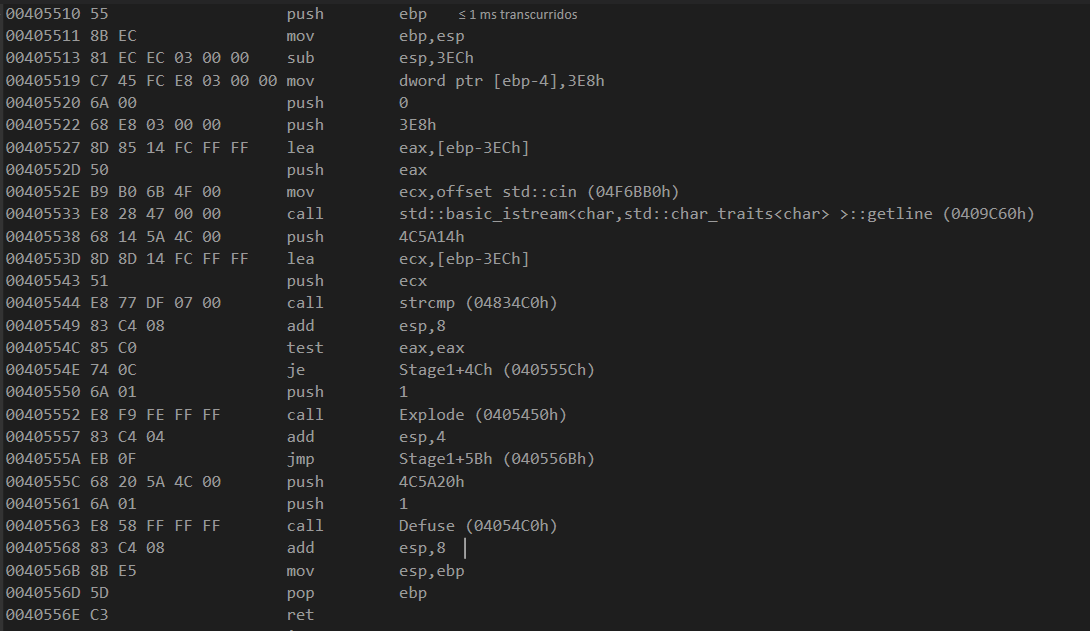
**BOMBA BINARIA**

**MEMORIA**

**-Stage 1:**

En primer lugar, para averiguar la contraseña de esta etapa, nos introducimos mediante F11 en el código de la instrucción:



Podemos observar que se hace un **push** del registro **ebp** y se copia en él el contenido de la pila (registro **esp**). Posteriormente se moverá a **ebp-4** el número 3E8h y a **eax** el contenido de **ebp-3EC**h. Es aquí cuando el programa nos pedirá introducir la contraseña de esta fase:



Una vez hecho esto, el programa apilará la contraseña a través de la siguiente instrucción:

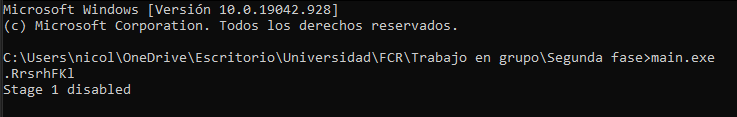


Será entonces cuando se llamará a la función **strcmp()** y se comparará el registro **ecx** (en el cual se almacenó la contraseña que introdujimos) junto con el contenido que hay en la dirección de memoria **0x004C5A14**, apilada anteriormente:



He aquí donde encontramos la **primera de las contraseñas**, que se corresponde con:

**.RrsrhFKl**. Se coge únicamente hasta antes del primer byte a 0, lo que se corresponde con los bytes en hexadecimal: **2E 52 72 73 72 68 46 4B 6C**:

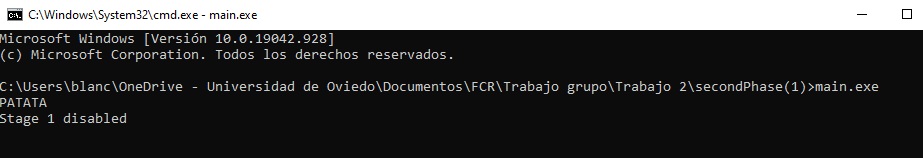


**¿Cómo hacer para que cualquier contraseña sea válida?**

La respuesta es sencilla, hemos de volver a fijarnos en el código inicial y hacer un mínimo cambio. Si bien nos fijamos en dicho código, justo después de hacer la comparación del registro **ecx** y el contenido de la memoria en la que está guardada la contraseña, en el caso de que sean o no sean iguales se hace un **je** (jump if equals). Pues bien, la manera más cómoda para responder a esta pregunta es modificar mediante el uso de **HxD** dicho salto para que en lugar de hacer un **je** haga simplemente un **jmp** a la instrucción posterior al **Explode()**. El código de dicha instrucción se corresponde con **EB 0F**, según la arquitectura x86 de Intel. Entonces hacemos el cambio e introduciendo cualquier combinación de 32 bits habríamos desactivado esta etapa:

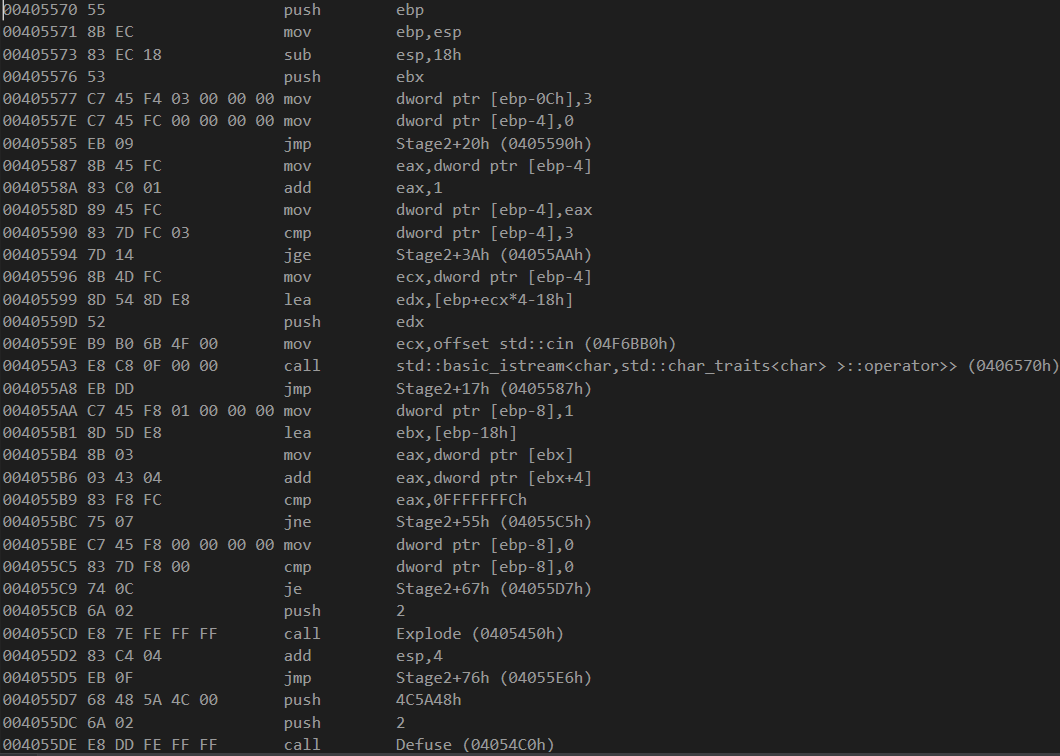






**-Stage 2:**

Pasamos ahora a la segunda fase de la desactivación de la bomba, la fase 2. De nuevo, nos volvemos a introducir en el código de la función pulsando F11:





Vamos a destacar las **operaciones más importantes:**

Justo después de mover a **ebp** el contenido de la pila (**esp**) se mueve a **ebx** el número exacto de números o contraseñas que pedirá el programa, que más adelante se comparará con el número de iteraciones:

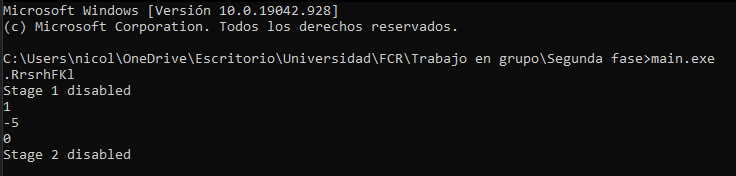


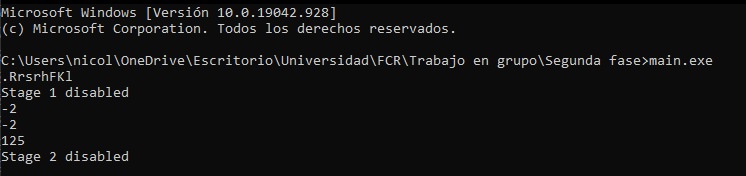
Una vez hecho esto, en el **jmp** el programa salta a la instrucción alojada en **0405590**h, donde se comparará el registro **ebx** y el número de iteraciones realizadas. En el caso de que no sean iguales, el programa continuará hasta pedir un número:



Entonces, saltará de nuevo hacia atrás a la etiqueta **04055C5**h. Este bucle se repetirá un total de 3 veces. A partir de aquí, el programa saltará a la etiqueta **04055AA**h, donde comparará lo que haya en el registro **eax** y el número en complemento a 2 **0FFFFFFFC**h. En el caso de que no sean iguales, la bomba explotará. En caso contrario se desactivará.

Pues bien, para la deducción de esta contraseña numérica, nos hemos ido fijando en lo que sucedía y donde se almacenaban los números que introducíamos. LLegamos a la conclusión de que la suma entre el primer y el segundo número tenía que ser igual a -4, por lo que en realidad hay un gran número de combinaciones posibles, siendo el tercer número irrelevante. En nuestro caso, en primer lugar introdujimos el **1**, después el **-5** y por último el **0**. De esta forma, la etapa es desactivada:





**¿Cómo hacer para que cualquier combinación numérica sea válida?**

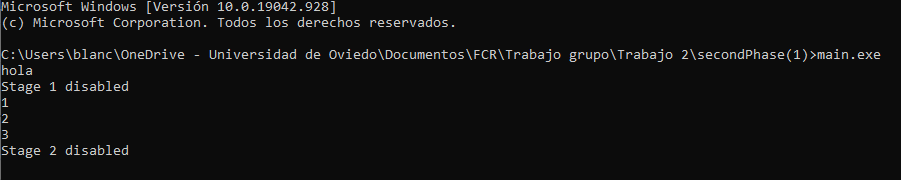
La respuesta es sencilla y muy parecida por no decir idéntica a la de la primera fase.

Hemos de revisar de nuevo el código y hacer una mínima modificación. Si nos fijamos, justo después de hacer la comparación entre el contenido del registro **eax** y el número **-4** en complemento a dos, hay un **jne** (jump not equals). He aquí donde tenemos que modificar el código fuente. Para ello, sustituimos ese **jne** por un **jmp** que salte hasta la función **Defuse**().

Para ello tomamos los datos de codificación de la arquitectura x86 de Intel y codificamos el jump para que salte un total de 25 bytes (**EB 19**h):

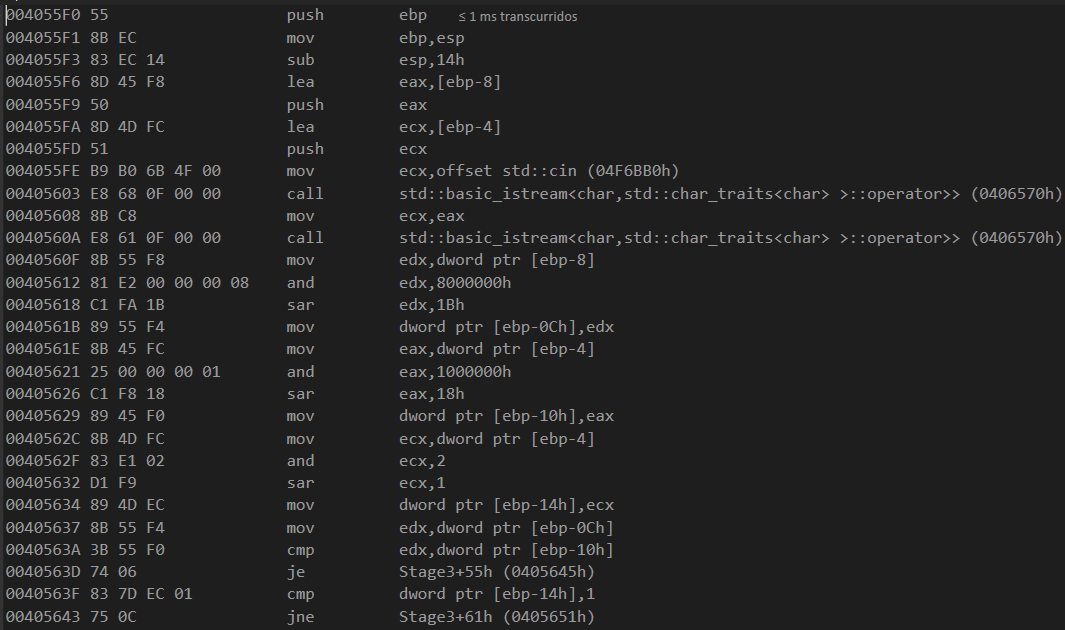


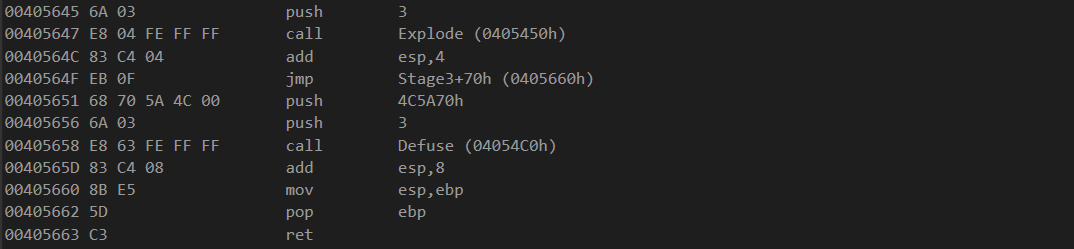


****

**-Stage 3:**

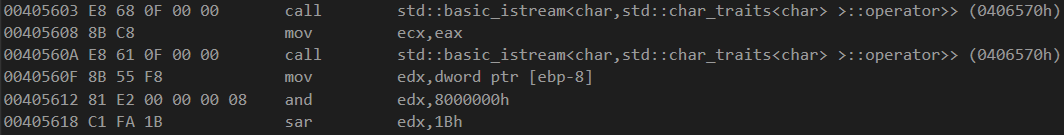
Y por último nos centramos en la última fase de la desactivación de la bomba. Para esta última etapa pulsamos de nuevo y por última vez F11:





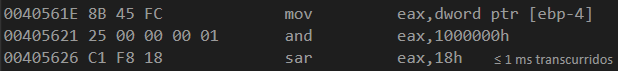
Vamos a comentar los **aspectos más destacables**:

Una vez se ha llamado a la función que permite introducir una clave por teclado, se mueve al registro **ecx**, el contenido del registro **eax**. Posterior a esto, se vuelve a pedir otro número por teclado, el cual se almacenará en **edx**. A continuación se realiza una operación **and** entre el registro **edx** y el número **80000000**h, y en el siguiente paso se realiza una operación **sar**, que mueve el resultado de la operación **and** un total de 27 bits hacia la derecha, dejando un 00000001h en dicho registro, siempre que hayamos introducido como segundo número uno negativo:





A partir de aquí se vuelve a hacer la operación **and** pero con el registro **eax** (el primer número que introdujimos) y el número **10000000**h, para después, mediante la operación **sar** se mueva el bit resultante 24 posiciones hacia la derecha (**18**h) . De esta forma, conseguiremos, introduciendo como primer número un 0 o una potencia de 2, que el registro **eax** quede a 0:





LLegados a este punto, se volverá a hacer una operación **and** entre el registro **ecx** y el número 2, y posteriormente se realizará la operación **sar**, que moverá el byte resultante 1 posición, lo que nos deja un 1 a la derecha:

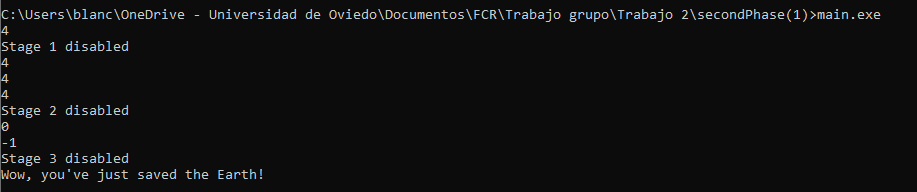


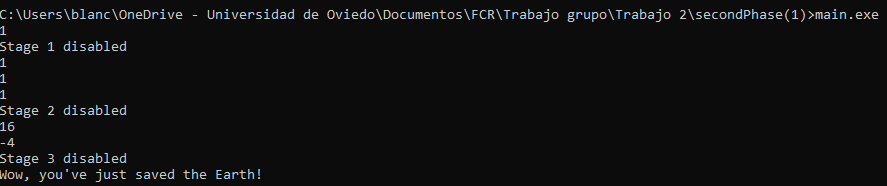


En este punto, se comparará lo que haya en el registro **ecx** y **edx**. En el caso de que sean iguales, el código nos llevaría a la explosión de la bomba. Si los registros contienen números distintos, el programa comparará el registro **[ebp - 14h]** y el número **1**h. Si son iguales la bomba explotará, si son distintos, se llamará a la función **Defuse**() y habremos desactivado la bomba con éxito.

Al igual que en la segunda etapa, en esta fase también hay un gran número de combinaciones posibles, pero todas tienen el mismo patrón:

El primer número introducido ha de ser o bien 0 o una potencia de 2, y el segundo número ha de ser siempre negativo. En nuestro caso, nosotros hemos probado dos **combinaciones funcionales**: **0** y **-1**, **16** y **-4**:



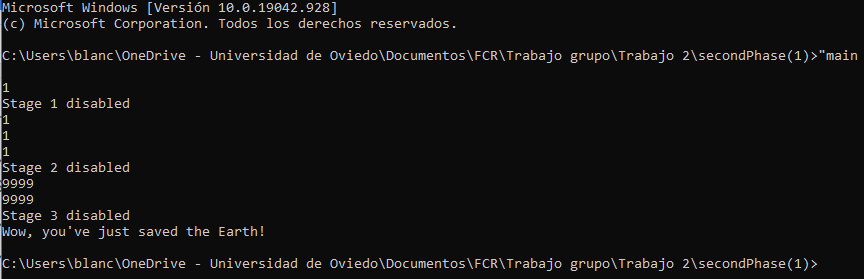


**¿Cómo hacer para que cualquier combinación numérica sea válida?**

La respuesta es idéntica a la anterior. Tenemos que evitar todo el código que haya por debajo de la instrucción **cmp edx, dword ptr [ebp - 10h]**. Para ello, tenemos que modificar el código fuente, cambiando el **je** (jump equals) siguiente por un simple **jmp**, que salte hasta justo antes de la operación **Defuse()**, es decir, un total de 25 bytes (**19**h). Para ello, abrimos de nuevo el **HxD** y buscamos la instrucción **74 06** h, que se corresponde con el **je**, y la sustituímos por **EB 19**h (codificación del **jmp +19**h):



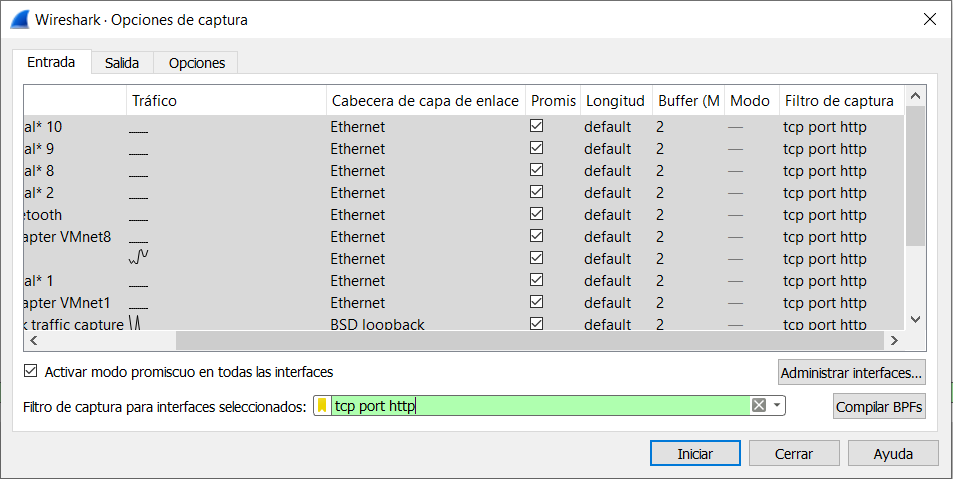




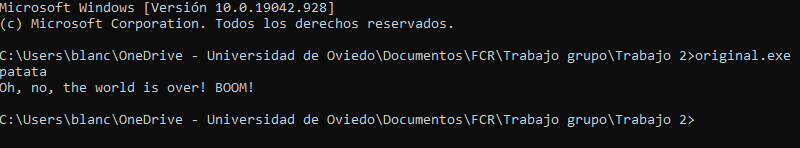
¡Y ya hemos salvado el mundo!. Ahora solo falta averiguar el nombre de quienes han intentado destruirlo…

**-Nombre de los hackers:**

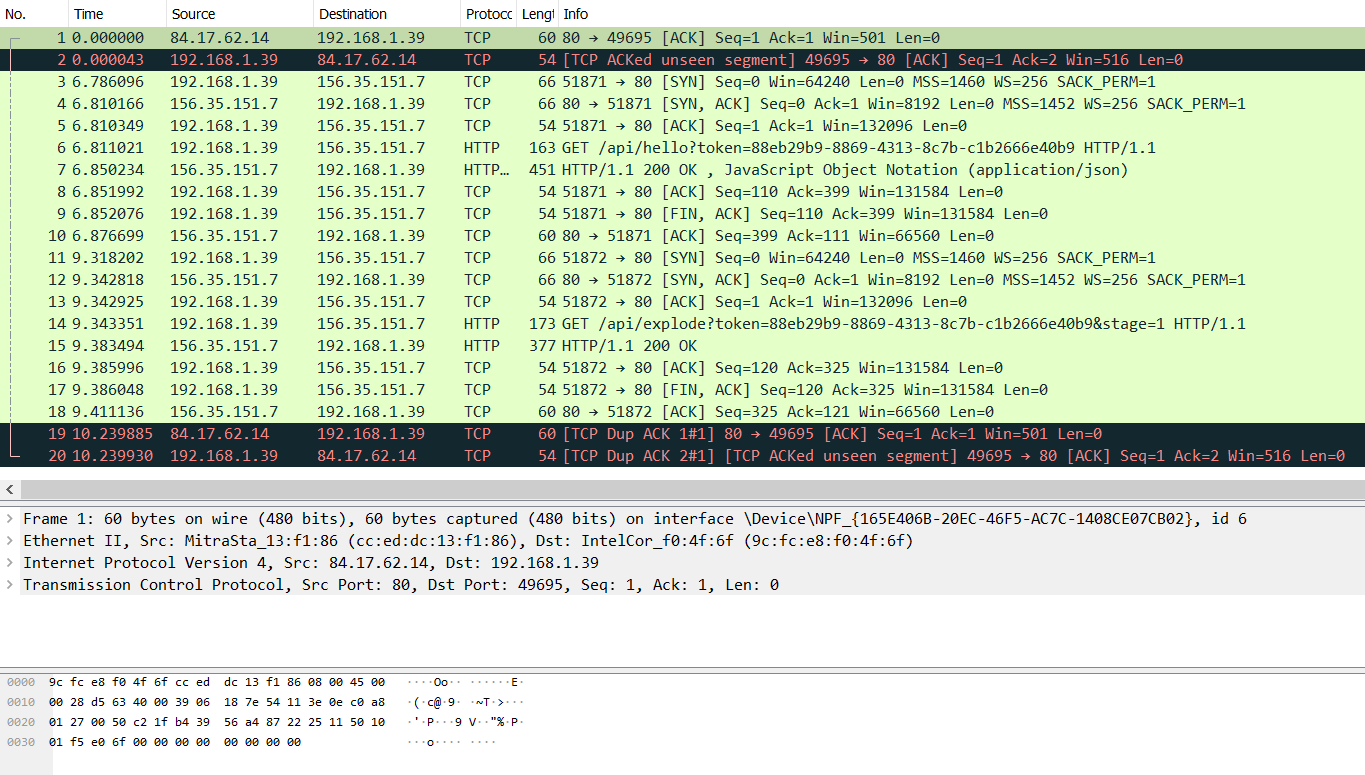
En primer lugar abrimos el Wireshark y seleccionamos el filtro más adecuado para averiguar con quién o qué se conecta nuestro ordenador al ejecutar el .exe . Aplicaremos el filtro **tcp port http** (visto en clase) ya que es el que nos permite observar las comunicaciones con un servidor web:



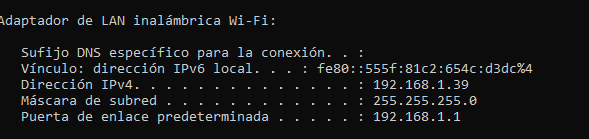
Una vez aplicamos el filtro, abrimos el ejecutable y probamos una contraseña cualquiera:



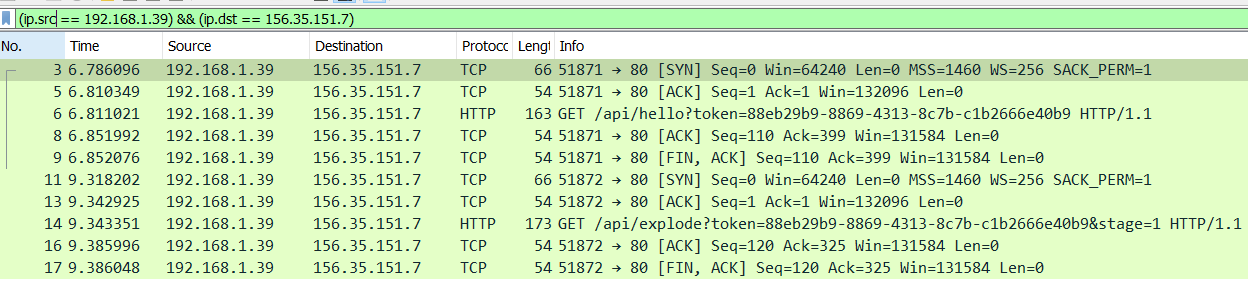
Es ahora cuando Wireshark ha capturado todas las comunicaciones realizadas:



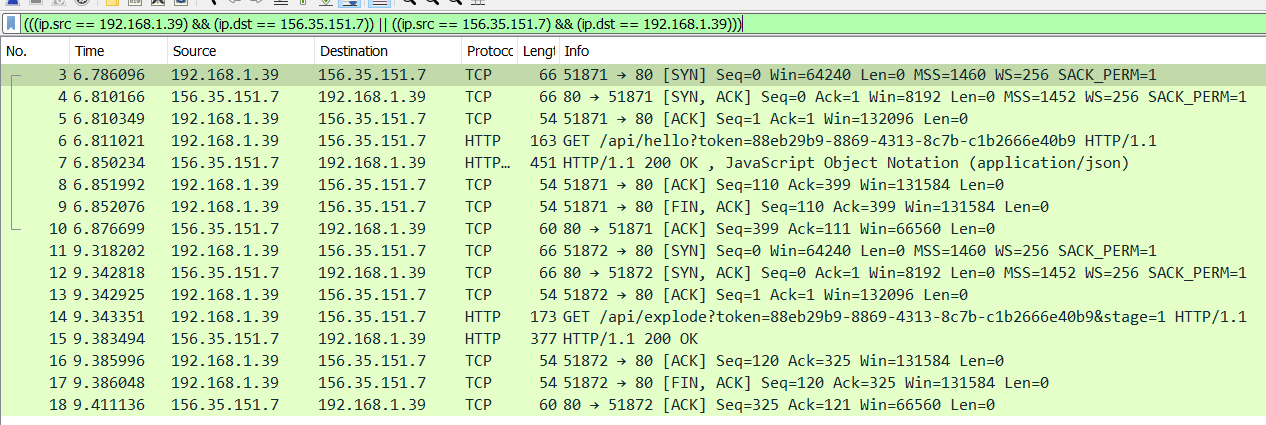
Llegados a este punto, ¿quién es nuestro ordenador y quién es el servidor de los hackers?. Esta pregunta tiene fácil solución. Para resolverla, hemos de abrir nuestro cmd e introducir el comando “**ipconfig**”:



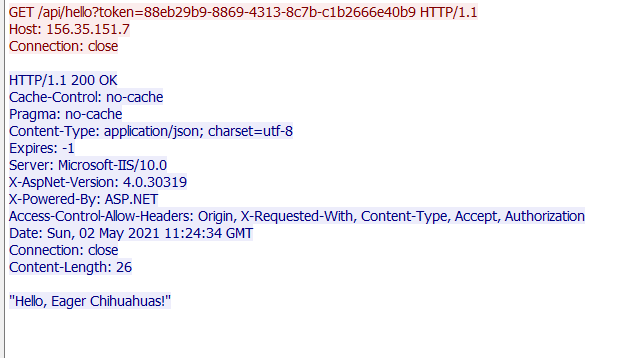
De aquí averiguamos que la dirección ip de nuestro dispositivo es **192.168.1.39**, por lo que ya tenemos parte de uno de los filtros a aplicar. Como bien pudimos observar y anotar en nuestras clases de laboratorio, averiguamos que la dirección ip del servidor de Oviedo al que se comunica la bomba es **156.35.151.7** (también es deducible puesto que la información que se muestra en uno de los mensajes es un hello?). Hemos completado el primer filtro completo:



¿Pero qué sucede con la comunicación Servidor-nuestro Pc?. Como no lo sabemos, aplicaremos también el filtro inverso:

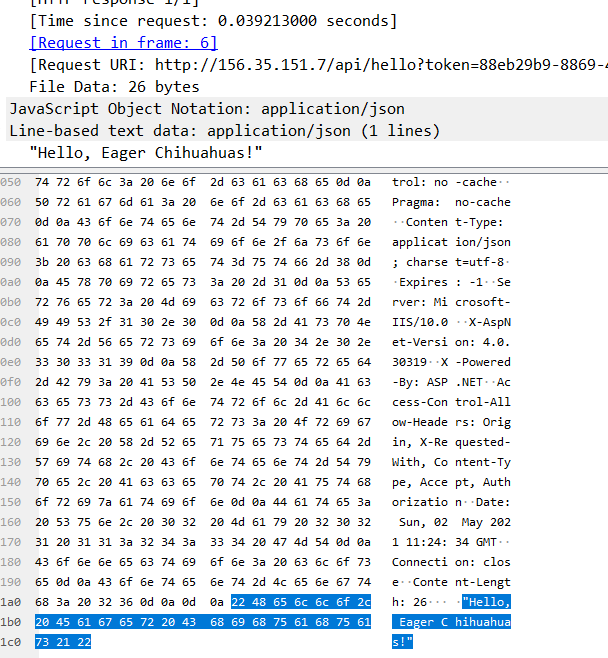


Es ahora cuando ya tenemos todas las comunicaciones entre Pc-Oviedo/Oviedo-Pc. Si vamos observando la información de cada una de las llamadas, hay una cuya info nos muestra un **hello?**,enviado por nosotros, como si quisiéramos entablar una conversación. Es entonces cuando marcamos la opción de seguir flujo tcp, y llegamos a esta conclusión:



El servidor ha respondido a los hackers por su nombre, **Eager Chihuahuas**.

También se puede comprobar observando la respuesta que nos da el servidor una vez hemos hecho la llamada comprobando el apartado **Hypertext Transfer Protocol**:



**-Reparto del trabajo:**

Nicolás Montiel Melendi: **18h**

Hugo Besteiro Gutiérrez: **7h**

Eduardo Blanco Bielsa: **30h**