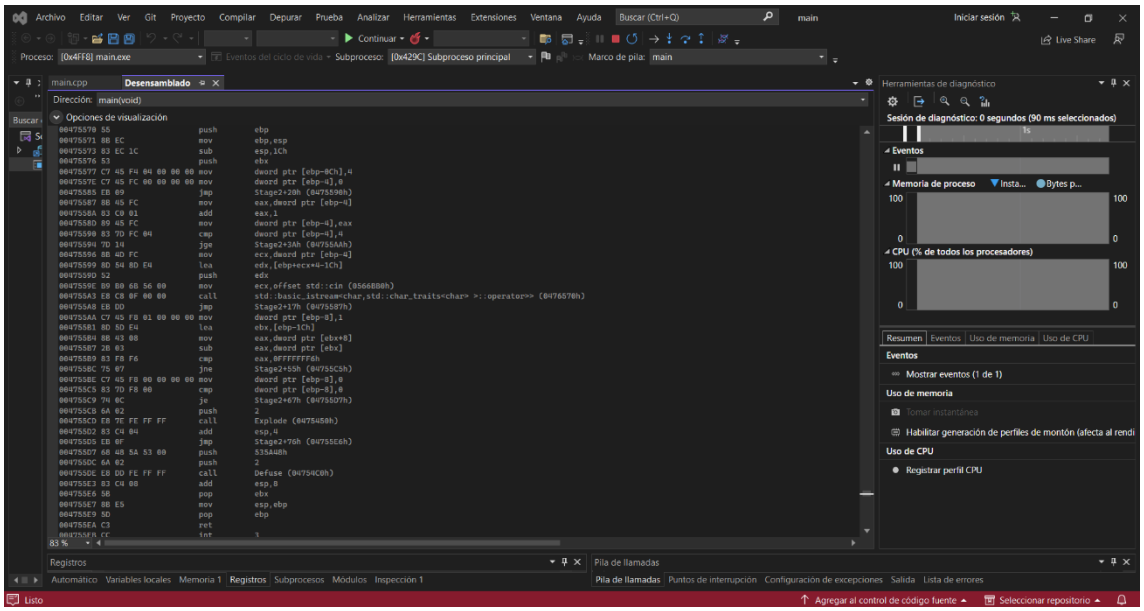
FASE 2 – SALVANDO AL MUNDO

2.Bomba:

Una vez desactivada la primera bomba, nos metemos con F11 dentro de la segunda bomba, encontrando que todo el código de la stage2 se encuentra ahí dentro sin hacer llamadas a ningún método externo como en el caso anterior.

El código es el siguiente:



Lo primero que encontramos es un bucle, en donde en *dword ptr [ebp-0Ch],4* inicializamos el número de interacciones, y en *dword ptr [ebp-4],0* inicializamos la "i".

Es un bucle for que en cada iteración pide por consola un valor, y una vez que pide 4 a través de "jge" salta a parte de verificar lo que introdujiste.

Código del for:

```
00475570 55                 push        ebp     ≤ 1 ms transcurridos
00475571 8B EC              mov         ebp,esp
00475573 83 EC 1C           sub         esp,1Ch
00475576 53                 push        ebx
00475577 C7 45 F4 04 00 00 00 mov       dword ptr [ebp-0Ch],4
0047557E C7 45 FC 00 00 00 00 mov       dword ptr [ebp-4],0
00475585 EB 09              jmp         Stage2+20h (0475590h)
00475587 8B 45 FC           mov         eax,dword ptr [ebp-4]
0047558A 83 C0 01           add         eax,1
0047558D 89 45 FC           mov         dword ptr [ebp-4],eax
00475590 83 7D FC 04        cmp         dword ptr [ebp-4],4
00475594 7D 14              jge         Stage2+3Ah (04755AAh)
00475596 8B 4D FC           mov         ecx,dword ptr [ebp-4]
00475599 8D 54 8D E4        lea         edx,[ebp+ecx*4-1Ch]
0047559D 52                 push        edx
0047559E B9 B0 6B 56 00     mov         ecx,offset std::cin (0566BB0h)
004755A3 E8 C8 0F 00 00     call        std::basic_istream<char,std::char_traits<char> >::operator>> (0476570h)
004755A8 EB DD              jmp         Stage2+17h (0475587h)
```

Una vez el programa a pedido los datos, el código continuo de la siguiente manera:

Guarda en *dword ptr [ebp-8]* un 1 que usará mas adelante, y en guarda en *ebx* lo que hay en memoria en *ebp-1Ch*. Esto es el primer valor que metimos, puesto a que en la primera prueba guardamos un 4 y en esa unidad de memoria "*ebp-1Ch"* se encontraba dicho valor.

Luego guarda en "eax" lo que hay en *dword ptr [ebx+8],* es decir guarda en "eax" el tercer número que metimos por consola. El siguiente paso es restar lo que hay en "eax" (tercer numero introducido) lo que hay en *dword ptr [ebx]* (primer numero

introducido). (Además si buscamos en memoria “[ebx]”, aparecen los datos que introducimos y además el 4 de las iteraciones del bucle y el 1 guardado anteriormente).

Una vez hecha la resta el programa hace una mascara del resultado con 0FFFFFFFFF6h, es decir, mira si el resultado ha sido -10.

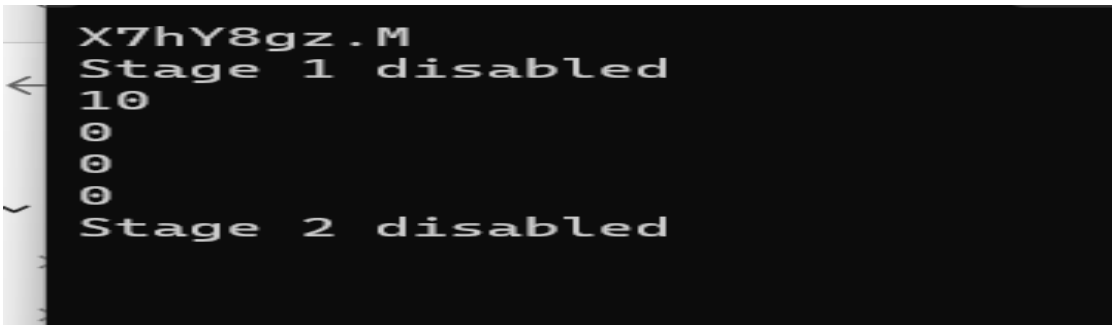
Si la condición se cumple salta dos posiciones hacia abajo, y compara si el 1 que estaba guardado anteriormente es igual a 0 (que es matemáticamente imposible) en el código “cmp *dword ptr [ebx-8],0”*, y al no cumplirse la bomba peta. Si se cumple que el resultado de la resta anterior es -10, salta una única posición, en donde se encuentra “mov *dword ptr [ebx-8],0”,* es decir, guarda en esa variable un 0, que cuando se compare con otro 0 SI será igual, entonces hará el salto por encima del explode(), y se desactivará la bomba.

Código de esto último:

```
004755AA C7 45 F8 01 00 00 00 mov       dword ptr [ebp-8],1
004755B1 8D 5D E4             lea       ebx,[ebp-1Ch]
004755B4 8B 43 08             mov       eax,dword ptr [ebx+8]
004755B7 2B 03                sub       eax,dword ptr [ebx]
004755B9 83 F8 F6             cmp       eax,0FFFFFFF6h
004755BC 75 07                jne       Stage2+55h (04755C5h)
004755BE C7 45 F8 00 00 00 00 mov       dword ptr [ebp-8],0
004755C5 83 7D F8 00          cmp       dword ptr [ebp-8],0
004755C9 74 0C                je        Stage2+67h (04755D7h)
004755CB 6A 02                push      2
```

Conclusión, por consola tienes que meter un primer valor (el que sea) tal que al restar al tercer valor que introduzcas, este primero, el resultado aritmético sea “-10”.

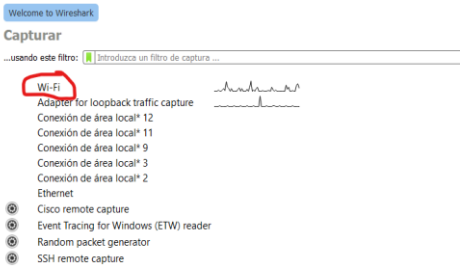
El segundo y cuarto valor dan absolutamente igual. Código de una muestra de meter los valores:

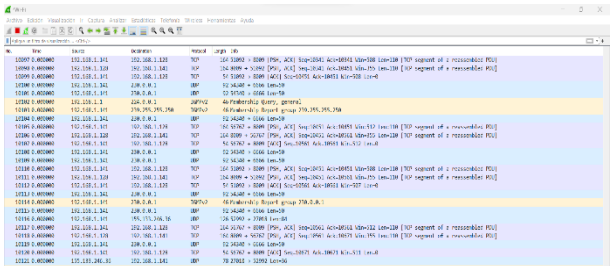


2) Encontrar el nombre del subgrupo criminal:

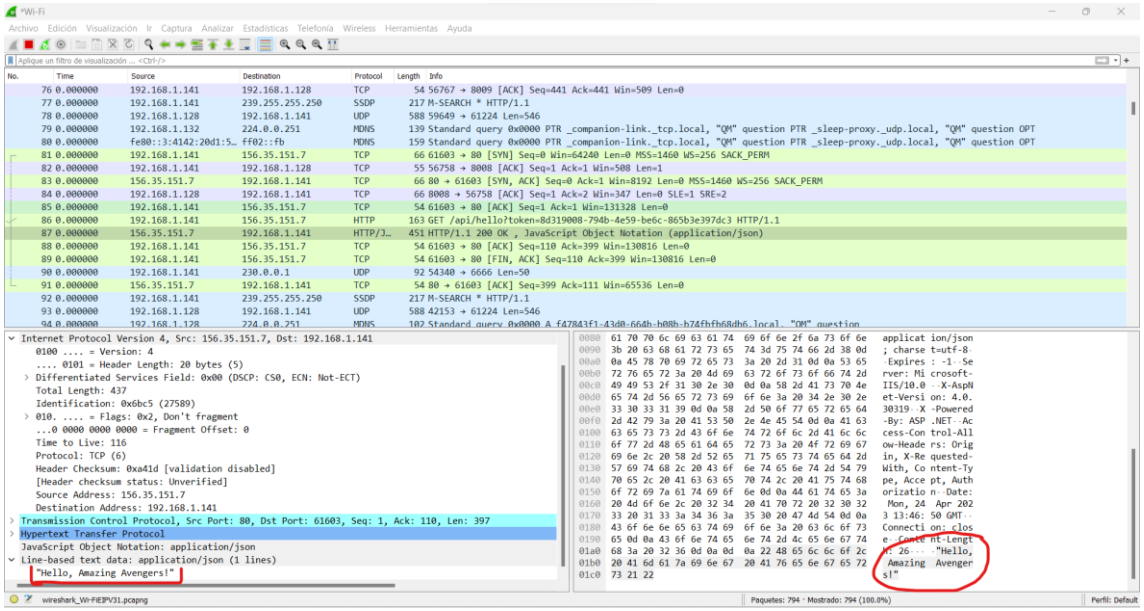
En el mensaje de respuesta a la primera conexión se envía una cadena que identifica el subgrupo criminal que la ha preparado. Creemos que esta información puede ayudarnos a encontrarlos, así que debes obtenerla utilizando un analizador de red.

Para hallar el nombre de este grupo, lo primero es iniciar *Wireshark*. Una vez dentro iniciamos una captura de el apartado “Wi-fi”, en donde empezaremos a ver el tráfico de nuestro rúter como se puede ver a continuación:





Una vez que ya estamos viendo lo que pasa, iniciamos el *main.exe* y al abrirse la consola de comando, ya se habrá enviado la solicitud, y por lo cual nos deberían haber aparecido unos paquetes de tipo http en el *Wireshark*. Entre ellos se encuentra uno con la signatura “hello” en el nombre, y justo debajo otro con “json”. Dentro de este nos permite ver en el propio ASCII y en “Line-based” debajo del “Hypertext transfer Protocol” , una lista de caracteres legible que trae lo siguiente:



Con esto sabemos que el nombre de la organización es “Amazing Avengers”.